Article

The International Journal of
Robotics Research
2020, Vol. 39(2-3) 183–201

DOI: 10.1177/0278364919859066
journals.sagepub.com/home/ijr
SAGE

Learning robust, real-time, reactive robotic grasping

Douglas Morrison, Peter Corke and Jürgen Leitner

Abstract

We present a novel approach to perform object-independent grasp synthesis from depth images via deep neural networks. Our generative grasping convolutional neural network (GG-CNN) predicts a pixel-wise grasp quality that can be deployed in closed-loop grasping scenarios. GG-CNN overcomes shortcomings in existing techniques, namely discrete sampling of grasp candidates and long computation times. The network is orders of magnitude smaller than other state-of-the-art approaches while achieving better performance, particularly in clutter. We run a suite of real-world tests, during which we achieve an 84% grasp success rate on a set of previously unseen objects with adversarial geometry and 94% on household items. The lightweight nature enables closed-loop control of up to 50 Hz, with which we observed 88% grasp success on a set of household objects that are moved during the grasp attempt. We further propose a method combining our GG-CNN with a multi-view approach, which improves overall grasp success rate in clutter by 10%. Code is provided at https://github.com/dougsm/ggcnn

Keywords

Grasping, vision, learning

1. Introduction

To be successful in real-world, unstructured environments, robotic manipulation systems must overcome a number of challenges. They must be able to generalize to an infinite number of objects that may be encountered, be able to act in dynamic environments, and accommodate changes in the workspace, perception errors and noise, inaccuracies in control, or perturbations to the robot itself.

Grasping is a canonical problem in robotics and has been investigated for decades, yielding a multitude of different techniques (Bicchi and Kumar, 2000; Bohg et al., 2014; Sahbani et al., 2012; Shimoga, 1996). Most recently, deep learning techniques have enabled some of the biggest advancements in grasp detection for previously unseen items. These approaches allow learning of features that correspond to good quality grasps that exceed the capabilities of human-designed features (Johns et al., 2016; Lenz et al., 2015; Mahler et al., 2017; Pinto and Gupta, 2016). The biggest advantage of these deep-learnt approaches is their ability to generalize to previously unseen objects.

However, these approaches typically use adapted versions of convolutional neural network (CNN) architectures designed for object recognition (Johns et al., 2016; Kumra and Kanan, 2017; Pinto and Gupta, 2016; Redmon and Angelova, 2015), and in most cases sample and rank

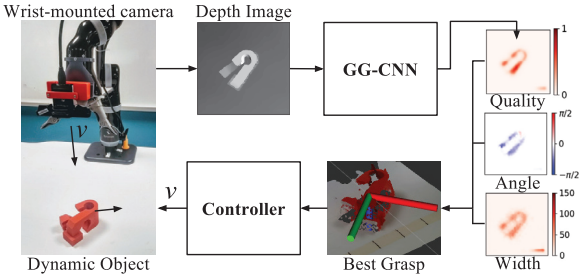

Fig 1. A robust, real-time, reactive grasping system using a wrist-mounted RGBD camera is proposed. Our GG-CNN generates pixel-wise grasp quality from a provided depth image. Based on the position of the best grasp pose, parameterized by position, angle, and gripping width, a velocity command (v) is issued to the robot. This closed-loop system is capable of successfully grasping static and dynamic objects, reacting to control errors, and integrating multiple observations during reaching.

Australian Centre for Robotic Vision (ACRV), Queensland University of Technology (QUT), Brisbane, Australia

Corresponding author:
Douglas Morrison, Australian Centre for Robotic Vision (ACRV), Queensland University of Technology, 2 George Street, Brisbane, Queensland 4001, Australia.
Email: doug.morrison@roboticvision.org

thousands of grasp candidates individually (Lenz et al., 2015; Mahler et al., 2017; Pinto and Gupta, 2016), resulting in long computation times of the order of a second (Mahler et al., 2017) to tens of seconds (Lenz et al., 2015). As a result, these techniques are limited to use in perfectly static environments, and rely on precise camera calibration and precise robot control to grasp objects successfully.

We propose a different approach to selecting grasp points for previously unseen items. Our generative grasping convolutional neural network (GG-CNN) directly generates a dense prediction of antipodal grasp poses and a quality measure for every pixel in an input depth image. It does this in real time, sufficiently fast to enable closed-loop control of grasping in dynamic environments (Figure 1). We use the term "generative" to differentiate our dense grasp generation method from methods that sample and classify grasp candidates.

This paper extends previous work of the authors (Morrison et al., 2018a), which describes the advantages of our real-time grasp generation approach in the context of dynamic environments. The advantages of GG-CNN over other state-of-the-art grasp synthesis CNNs are twofold. First, we do not rely on sampling of grasp candidates, but rather directly generate grasp poses on a pixelwise basis, analogous to advances in object detection where fully convolutional networks are commonly used to perform pixelwise semantic segmentation rather than relying on sliding windows or bounding boxes (Long et al., 2015). Second, our GG-CNN has orders of magnitude fewer parameters than other grasp synthesis networks, allowing our grasp detection pipeline to execute in only 19 ms on a GPU-equipped desktop computer: fast enough for closed-loop grasping.

We evaluate the performance of our system in different scenarios by performing robotic grasping trials, with both static and moving objects. Using a Kinova Mico robot in dynamic grasping trials, where objects are moved during the grasp attempt, we achieve 83% grasping success rate on a set of eight 3D-printed objects with adversarial geometry (Mahler et al., 2017) and 88% on a set of 12 household items chosen from standardized object sets, as well as 81% grasp success rate when grasping from dynamic clutter. Techniques with slow computation times are unable to react to such changes in the environment. We illustrate the advantages of using a closed-loop grasping method by showing an improved grasping accuracy when artificial inaccuracies are added to the robot's control to simulate kinematic errors.

In addition, we highlight the flexibility and generalizability of our approach by directly transferring the GG-CNN onto a new robotic platform, a Franka Emika Panda arm. By comparison, an end-to-end controller cannot be transferred to a new robot or environment without retraining.

Using this platform, we present a novel method for improving grasp success rates in clutter by combining grasp predictions from multiple viewpoints, which is enabled by our GG-CNN's real-time performance and pixelwise grasp prediction. By considering multiple viewpoints, we overcome issues related to occlusion in cluttered environments, potentially discovering high-quality grasps that are not visible from the initial viewpoint, and thus improving the quality of our grasp estimates. In our trials, this multi-viewpoint method increases the grasp success rate by up to 10% compared with predicting grasp poses from a single viewpoint.

We provide a detailed evaluation of the design and training of the GG-CNN, as well as the common failure modes caused by perceptual errors and the network itself. In particular, we investigate the effect of different network parameters, data augmentation and training datasets. This leads to an improved model (GG-CNN2), trained on the recently released Jacquard dataset (Depierre et al., 2018).

2. Related work

2.1. Grasping unknown objects

Grasp synthesis refers to the formulation of a stable robotic grasp for a given object, which is a topic which has been widely researched resulting in a plethora of techniques. Broadly, these can be classified into analytic methods and empirical methods (Bohg et al., 2014; Sahbani et al., 2012). Analytic methods use mathematical and physical models of geometry, kinematics, and dynamics to calculate grasps that are stable (Bicchi and Kumar, 2000; Prattichizzo and Trinkle, 2008), but tend to not transfer well to the real world due to the difficultly in modeling physical interactions between a manipulator and an object (Bicchi and Kumar, 2000; Rubert et al., 2017; Sahbani et al., 2012).

In contrast, empirical methods focus on using models and experience-based approaches. Some techniques work with known items, associating good grasp points with an offline database of object models or shapes (Detry et al., 2009; Goldfeder et al., 2007; Miller et al., 2003), or familiar items, based on object classes (Saxena et al., 2008) or object parts (El-Khoury and Sahbani, 2008), but are unable to generalize to new objects. During the 2017 Amazon Robotics Challenge, 50% of the objects were replaced with novel ones just half an hour before each team's challenge run to test these generalization capabilities (Morrison et al., 2018b; Zeng et al., 2018b).

For grasping unknown objects, large advancements have been seen recently with a proliferation of vision-based deep-learning techniques (Lenz et al., 2015; Mahler et al., 2017; Pinto and Gupta, 2016; Redmon and Angelova, 2015; Wang et al., 2016). Many of these techniques share a common pipeline: classifying grasp candidates sampled from an image or point cloud, then ranking them individually using CNNs. Once the best grasp candidate is determined, a robot executes the grasp open-loop (without any feedback) which requires precise calibration between the camera and the robot, precise control of the robot and a completely static environment.

Execution time is the primary reason that grasps are executed open-loop. For example, in the 2017 Amazon Robotics Challenge the winning robot performed open-loop grasping based on a slow-to-compute heuristic measure, making a closed-loop approach impractical (Morrison et al., 2018b). In many cases, deep-learning approaches use large neural networks with millions of parameters (Johns et al., 2016; Mahler et al., 2017; Pinto and Gupta, 2016) and process grasp candidates using a sliding window at discrete intervals of offset and rotation (Lenz et al., 2015; Pinto and Gupta, 2016), which is computationally expensive and results in grasp planning times of the order of a second (Mahler et al., 2017) to tens of seconds (Lenz et al., 2015).

Some approaches reduce execution time by pre-processing and pruning the grasp candidates (Lenz et al., 2015; Wang et al., 2016) or predicting the quality of a discrete set of grasp candidates simultaneously (Johns et al., 2016; Pinto and Gupta, 2016), trading off execution time against the number of grasps which are sampled, but ignoring some potential grasps as a result.

Instead of sampling grasp candidates, both Kumra and Kanan (2017) and Redmon and Angelova (2015) used a deep CNN to regress a single best grasp pose for an input image. However, these regression methods are liable to output the average of the possible grasps for an object, which itself may not be a valid grasp (Redmon and Angelova, 2015). Similar to our method, Zeng et al. (2018b) used large CNNs to detect both pixelwise suction and antipodal grasp affordances in RGB-D images.

We address the issues of execution time and grasp sampling by directly generating grasp poses for every pixel simultaneously, using a comparatively small neural network.

2.2. Closed-loop grasping

Closed-loop control of a robot to a desired pose using visual feedback is commonly referred to as visual servoing (Hutchinson et al., 1996). The advantages of visual servoing and other closed-loop methods are that they are able to adapt to dynamic environments and do not necessarily require fully accurate camera calibration or position control (Kappler et al., 2018). A number of works apply visual servoing directly to grasping applications, with a survey given by Kragic and Christensen (2002). However, the nature of visual servoing methods mean that they typically rely on hand-crafted image features for object detection (Kober et al., 2012; Vahrenkamp et al., 2008) or object pose estimation (Horaud et al., 1998), so do not perform any online grasp synthesis but instead converge to a pre-determined goal pose and are not applicable to unknown objects. Of increasing interest is the question of how to integrate learning methods with reactive grasping approaches.

Closed-loop approaches have been previously applied to other aspects of robotic manipulation. For example, Leitner et al. (2014) presented a method employing trained visual detectors for closed-loop, real-time reaching and obstacle avoidance on the iCub robot. More recently CNN-based controllers for grasping have been proposed to combine deep learning with closed-loop grasping (Kalashnikov et al., 2018; Levine et al., 2016; Viereck et al., 2017). Rather than explicitly performing grasp synthesis, both systems learn controllers which map potential control commands to the expected quality of, or distance to, a grasp after execution of the control, requiring many potential commands to be sampled at each time step. In these cases, the control executes at no more than approximately 5 Hz. While all are closed-loop controllers, grasping in dynamic scenes is only presented in Viereck et al. (2017) and we reproduce these experiments. Levine et al. (2016) learned the grasping task end-to-end, i.e. directly generating motor torques from input images, making the network hard to transfer to other robots or scenes without large amounts of extra training data.

Other grasp regression methods (Kumra and Kanan, 2017; Redmon and Angelova, 2015) have reported real-time performance, but have not been validated by robotic experiments.

2.3. Benchmarking for robotic grasping

Directly comparing results between robotic grasping experiments is difficult due to the wide range of grasp detection techniques used, the lack of standardization between object sets, and the limitations of different physical hardware, e.g. robot arms, grippers, or cameras. Many papers report grasp success rates on sets of "household" objects, which vary significantly in the number and types of objects used.

The ACRV Picking Benchmark (APB) (Leitner et al., 2017) and the YCB Object Set (Calli et al., 2015) both define item sets and manipulation tasks. However, neither has been widely adopted for robotic grasping experiments. This is in part due to the task-centric nature of these object sets, focusing on tasks such as warehouse order fulfilment (APB) or table setting and block stacking (YCB). In addition, many of the items from these two sets are impractically small, large, or heavy for many common research robots and grippers.

For this paper, we propose a set of 20 reproducible items for testing, comprising 8 3D-printed adversarial objects from Mahler et al. (2017) and 12 items from the APB and YCB object sets, which we believe provide a wide enough range of sizes, shapes, and difficulties to effectively compare results with other similar work while not excluding use by any common robots, grippers, or cameras (more details are given in Section 5.2).

3. Grasp point definition

Like much of the related literature (Johns et al., 2016; Lenz et al., 2015; Mahler et al., 2017; Pinto and Gupta, 2016; Viereck et al., 2017), we consider the problem of detecting and executing antipodal grasps on unknown objects,

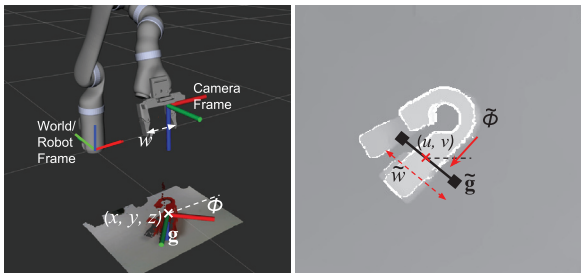

Fig 2. Left: A grasp $\mathbf{g}$ is defined by its Cartesian position (x, y, z), rotation around the z-axis ϕ, and gripper width w required for a successful grasp. Right: In the depth image the grasp pose $\tilde{\mathbf{g}}$ is defined by its center pixel (u, v), its rotation $\tilde{\phi}$ around the image axis, and perceived width $\tilde{w}$.

perpendicular to a planar surface, given a depth image of the scene (Figure 2).

Let $\mathbf{g} = (q, \mathbf{p}, \phi, w)$ define a grasp, executed perpendicular to the x–y plane. The grasp is parameterized by a scalar quality measure q, representing the chances of grasp success, the gripper's center position $\mathbf{p} = (x, y, z)$ in world coordinates, the gripper's rotation ϕ around the z-axis and the required gripper width w.

Unlike many other grasp detection networks which assume a fixed, static gripper width (Johns et al., 2016; Mahler et al., 2017; Viereck et al., 2017), our grasp representation explicitly predicts the required gripper width for each grasp. This addition enables better prediction and better performance over the more commonly used position and rotation only representation, especially in the case of cluttered environments or complex objects where choosing an incorrect grasp width may result in collisions with the surrounding scene. An additional advantage is that the approach is not locked to a specific gripper.

We want to detect grasps given a 2.5D depth image $\mathbf{I} = \mathbb{R}^{H \times W}$ with height H and width W, taken from a camera with known intrinsic parameters. Throughout this paper, we use a tilde ($\sim$) to denote values in the image space. In the image $\mathbf{I}$ a grasp is described by

$$\tilde{\mathbf{g}} = (q, \tilde{\mathbf{s}}, \tilde{\phi}, \tilde{w})$$

where $\tilde{\mathbf{s}} = (u, v)$ is the center point in image coordinates (pixels), $\tilde{\phi}$ is the rotation in the camera's reference frame, and $\tilde{w}$ is the grasp width in image coordinates. A grasp in the image space $\tilde{\mathbf{g}}$ can be converted into a grasp in world coordinates $\mathbf{g}$ by applying a sequence of known transforms,

$$\mathbf{g} = t_{RC}(t_{CI}(\tilde{\mathbf{g}})) \tag{1}$$

where t_{RC} transforms from the camera frame to the world/robot frame and t_{CI} transforms from 2D image coordinates into the 3D camera frame, based on the camera intrinsic parameters and known calibration between the robot and camera. The depth of the grasp is offset by a fixed amount, such that the center of the gripper's fingertips are aligned with the measured depth at the center of the grasp. We do not consider multiple different depths like in Satish et al. (2019).

We refer to the set of grasps in the image space as the *grasp map*, which we denote

$$\tilde{\mathbf{G}} = (\tilde{\mathbf{Q}}, \tilde{\mathbf{\Phi}}, \tilde{\mathbf{W}}) \in \mathbb{R}^{3 \times H \times W}$$

where $\tilde{\mathbf{\Phi}}$, $\tilde{\mathbf{W}}$, and $\tilde{\mathbf{Q}}$ are each $\in \mathbb{R}^{H \times W}$ and contain values of $\tilde{\phi}$, $\tilde{w}$, and q, respectively, at each pixel.

Instead of sampling the input image to create grasp candidates, we wish to directly calculate a grasp $\tilde{\mathbf{g}}$ for each pixel in the depth image $\mathbf{I}$. To do this, we define a function M from a depth image to the *grasp map* in the image coordinates: $M(\mathbf{I}) = \tilde{\mathbf{G}}$. From $\tilde{\mathbf{G}}$ we can calculate the best visible grasp in the image space $\tilde{\mathbf{g}}^* = \max_{\tilde{\mathbf{Q}}} \tilde{\mathbf{G}}$, and calculate the equivalent best grasp in world coordinates $\mathbf{g}^*$ via Equation (1).

4. GG-CNN

We propose the use of a fully - convolutional neural network to approximate the complex function $M : \mathbf{I} \rightarrow \tilde{\mathbf{G}}$. Here M_θ denotes a neural network with θ being the weights of the network.

In this section, we present two fully - convolutional neural network for real-time grasp generation. Fully convolutional networks have been shown to perform well at grasp affordance detection (Zeng et al., 2018a,b), computer vision tasks requiring transfer between image domains, such as image segmentation (Badrinarayanan et al., 2015; Long et al., 2015) and contour detection (Yang et al., 2016).

Each network directly approximates the *grasp map* $\tilde{\mathbf{G}}_\theta$ from an input depth image $\mathbf{I}$. The network is parameterized by its weights θ and computes the function $M_\theta(\mathbf{I}) = (\tilde{\mathbf{Q}}_\theta, \tilde{\mathbf{\Phi}}_\theta, \tilde{\mathbf{W}}_\theta) \approx M(\mathbf{I})$, where $\mathbf{I}$, $\tilde{\mathbf{Q}}_\theta$, $\tilde{\mathbf{\Phi}}_\theta$, and $\tilde{\mathbf{W}}_\theta$ are represented as 300×300 pixel images.

4.1. Grasp map representation

Here $\tilde{\mathbf{G}}$ estimates the parameters of a set of grasps, each executed at the world point $\mathbf{p}$, corresponding to each pixel $\mathbf{s}$. The representation of the *grasp map* $\tilde{\mathbf{G}}$ is as follows.

We use $\tilde{\mathbf{Q}}$ to denote an image that describes the quality of a grasp executed at each point (u, v). The value is a scalar in the range $[0, 1]$ where a value closer to 1 indicates higher grasp quality, i.e. higher chance of grasp success.

Here $\tilde{\mathbf{\Phi}}$ is an image that describes the angle of a grasp to be executed at each point. Because the antipodal grasp is symmetrical around $\pm \frac{\pi}{2}$ radians, the angles are given in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

Instead of learning the grasp angle directly, in our implementation we encode the angle as two components of a unit vector, $\sin(2\tilde{\mathbf{\Phi}})$ and $\cos(2\tilde{\mathbf{\Phi}})$. This removes any discontinuities that would occur in the data where the angle

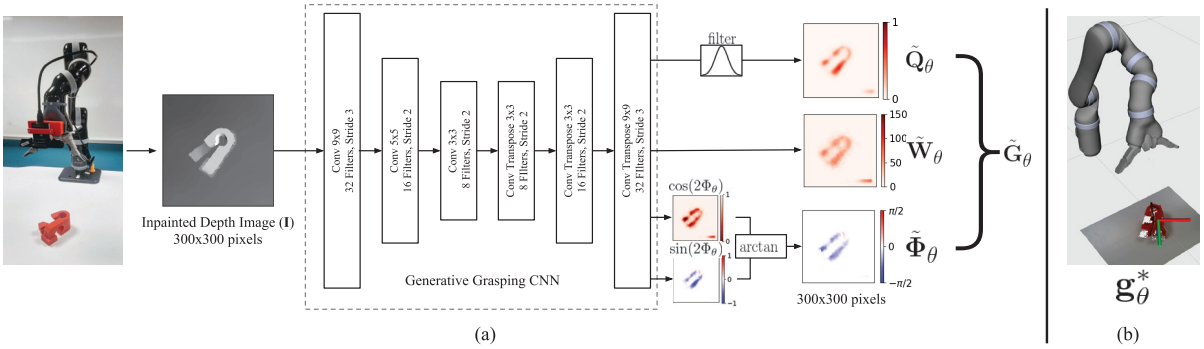

Fig 3. (a) The GG-CNN takes an inpainted depth image (**I**), and directly generates a grasp pose for every pixel (the *grasp map* $\tilde{\mathbf{G}}_\theta$), comprising the grasp quality $\tilde{\mathbf{Q}}_\theta$, grasp width $\tilde{\mathbf{W}}_\theta$, and grasp angle $\tilde{\mathbf{\Phi}}_\theta$. (b) From the combined network output, we can compute the best grasp point to reach for, $\mathbf{g}^*_\theta$.

wraps around $\pm\frac{\pi}{2}$ and provides unique values within $\tilde{\mathbf{\Phi}} \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ that are symmetrical at $\pm\frac{\pi}{2}$, making the distribution easier for the network to learn (Hara et al., 2017). The final grasp angle is computed during post-processing by $\tilde{\mathbf{\Phi}} = \arctan\left(\frac{\sin(2\tilde{\mathbf{\Phi}})}{\cos(2\tilde{\mathbf{\Phi}})}\right)/2$.

$\tilde{\mathbf{W}}$ is an image that describes the required gripper width to execute a grasp at each point. To allow for depth invariance, values are in the range of $[0, 150]$ pixels, which can be converted into a physical measurement using the depth camera parameters and measured depth.

4.2. Grasp detection pipeline

Our grasp detection pipeline comprises four stages: image processing, generation of pixel-wise grasp quality through our GG-CNN, filtering, and computation of the best grasp pose (Figure 3).

The depth image is first cropped to a square, and scaled to 300×300 pixels to suit the input of the GG-CNN. We inpaint invalid depth values using OpenCV (Bradski, 2000). A trained GG-CNN is then used to produce a *grasp map* given the processed depth image.

Johns et al. (2016) showed that filtering their *grasp function* with a Gaussian kernel led to more robust grasps by removing maxima that were close to regions of poor quality grasps. We do the same, and find that it produces grasp poses which are not only more spatially diverse but are more stable between consecutive depth frames, shown in Figure 4. We filter our network output $\tilde{\mathbf{Q}}$ with a Gaussian kernel with standard deviation of 5 pixels. In Section 5.9 we experimentally show the advantages of adding such an output filter, especially in closed-loop grasping trials, where frame-to-frame stability of the *grasp map* is important.

Finally, the best grasp pose in the image space $\tilde{\mathbf{g}}^*_\theta$ is computed by identifying the maximum pixel $\tilde{\mathbf{s}}^*$ in $\tilde{\mathbf{Q}}_\theta$, and the rotation and width are computed from $\tilde{\mathbf{\Phi}}_\theta$ and $\tilde{\mathbf{W}}_\theta$, respectively. The grasp in Cartesian coordinates $\mathbf{g}^*_\theta$ is computed via Equation (1) (Figure 3 b).

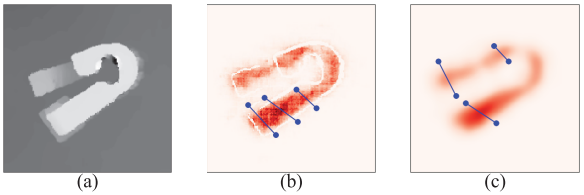

Fig 4. The effect of applying a Gaussian filter to $\tilde{\mathbf{Q}}_\theta$. (a) The input depth image. (b) The unfiltered network output $\tilde{\mathbf{Q}}_\theta$, in which the grasps corresponding to the top three local maxima are shown bunched close together. These local maxima are likely to change between frames. (c) In the filtered $\tilde{\mathbf{Q}}_\theta$, the local maxima are more likely to represent distinct, good quality grasp points that are more stable between consecutive frames.

4.3. Grasp datasets

In order to train the GG-CNN to perform dense, pixel-wise grasp prediction we require a densely labeled grasping dataset. Two such datasets exist, the smaller, hand-labeled Cornell Grasping Dataset (Lenz et al., 2015) and the much larger, synthetic Jacquard dataset (Depierre et al., 2018). Both represent antipodal grasps in RGB and depth images as rectangles using pixel coordinates (Yun Jiang et al., 2011).

The Cornell Grasping Dataset contains 885 RGB-D images of real objects, with 5,110 human-labeled positive and 2,909 negative grasps. While this is a relatively small grasping dataset compared with some other, synthetic datasets (Mahler et al., 2017, 2016), the data best suits our pixelwise grasp representation as multiple labeled grasps are provided per image. This is a more realistic estimate of the full pixel-wise *grasp map*, than using a single image to represent one grasp, such as in Mahler et al. (2017).

The Jacquard dataset, collected in simulation, removes the limitations of human labeling. As a result, the dataset is much larger than the Cornell Grasping Dataset, and contains 54,000 rendered images labeled with over 1 million

positive grasps on a much wider variety of objects. In addition, the dataset provides access to the simulation environment in which it was created, allowing for predicted grasps for each dataset entry to be trialled in a consistent environment through simulated grasp trials (SGTs). We use these SGTs as a benchmark for grasp network performance in order to study the effects of both the dataset size and network design.

4.4. Training data generation

To convert from the rectangle representation used by both datasets to our image-based representation $\tilde{\mathbf{G}}$, we use the center third of each grasping rectangle as an image mask which corresponds to the position of the center of the gripper. We use this image mask to update sections of our training images, as described in the following and shown in Figure 5.

For the Cornell Grasping Dataset, we consider only the positive labeled grasps for training our network and assume any other area is not a valid grasp. In addition, to counteract its small size and provide robustness to the real-world conditions, we augment the dataset with random crops, zooms, and rotations to create a set of 8,840 depth images and associated *grasp map* images $\tilde{\mathbf{G}}_T$, effectively incorporating 51,100 grasp examples.

4.4.1. Grasp quality. We treat each ground-truth positive grasp from the datasets as a binary label and set the corresponding area of $\tilde{\mathbf{Q}}_T$ to a value of 1, with all other pixels being 0.

4.4.2. Angle. We compute the angle of each grasping rectangle in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$, and set the corresponding area of $\tilde{\mathbf{\Phi}}_T$. As described in Section 4.1, the angle is further decomposed into two components of a unit vector to make the distribution simpler to learn by the network.

4.4.3. Width. Similarly, we compute the width in pixels (maximum of 150) of each grasping rectangle representing the width of the gripper and set the corresponding portion of $\tilde{\mathbf{W}}_T$. During training, we normalize the values to the range $[0, 1]$. The physical gripper width can be calculated using the parameters of the camera and the measured depth.

4.4.4. Depth input. The depth images from the training set are inpainted using OpenCV (Bradski, 2000) to remove any invalid values. We subtract the mean of each depth image, centering its value around 0 to provide depth invariance.

4.5. Network architectures and design

When designing the networks we performed a number of parameter sweeps, varying the number of convolutional filters, filter sizes and dilation parameters in order to find the network design that offers the best performance while still being small enough to allow for real-time inference. The execution time was recorded on a PC running Ubuntu 16.04 with a 3.6 GHz Intel Core i7-7700 CPU and NVIDIA GeForce GTX 1070 graphics card. Code for the GG-CNN and robotic experiments can be found at **https://github.com/dougsm/ggcnn**.

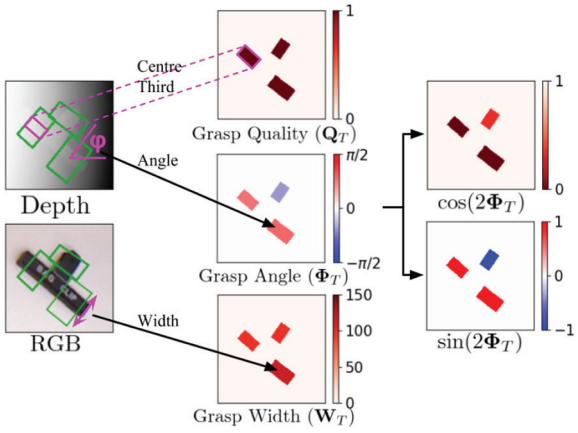

Fig 5. Generation of training data for our GG-CNN. Left: The cropped and rotated depth images from an antipodal grasping dataset, with the ground-truth positive grasp rectangles shown in green. NB: The RGB image is added for illustration only and is not used by our system. Right: The grasp quality ($\tilde{\mathbf{Q}}_T$), grasp angle ($\tilde{\mathbf{\Phi}}_T$), and grasp width ($\tilde{\mathbf{W}}_T$) images to train our network. The angle is further decomposed into $\cos(2\tilde{\mathbf{\Phi}}_T)$ and $\sin(2\tilde{\mathbf{\Phi}}_T)$ for training as described in Section 4.3 .

We use two methods of testing the performance of the trained networks against a training dataset. First, the commonly used metric which counts a success if the best predicted grasping rectangle aligns within 30° of and has an intersection-over-union (IoU) of greater than 25% with a ground-truth grasp rectangle (which we refer to as the *IoU metric*) (Depierre et al., 2018; Kumra and Kanan, 2017; Lenz et al., 2015). Second, we use SGTs in the cloud-based physics simulator provided by the Jacquard dataset. The simulator executes a predicted grasp and returns a success if the object was successfully lifted by the simulated robot. The IoU metric allows for quick offline evaluation, whereas SGTs provide a higher-fidelity evaluation metric but are much slower to compute and require the use of a remote server for evaluation.

In addition to network performance, we take into account the number of parameters in each network, and the average inference time of the network, whereby it may be necessary to ignore a certain configuration that performed well due to an excessive execution time which would not allow for real-time performance.

GG-CNN is a fully convolutional hourglass topology, shown in Figure 3(a), originally from Morrison et al. (2018a). The GG-CNN was designed using the Cornell Grasping Dataset. To determine the best network

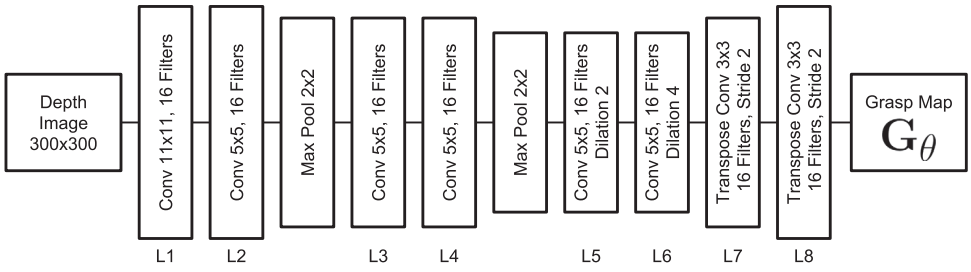

Fig 6. Network architecture for GG-CNN2.

configuration, we compare relative performance between our trained networks by evaluating each on detecting ground-truth grasps on a 20% evaluation dataset containing 1,710 augmented images from the Cornell Grasping Dataset. We trained 95 networks with similar architectures but different combinations of convolutional filters and stride sizes for 100 epochs each. Using the IoU metric, our highest performing network has a detection accuracy of 78% when considering the global maximum over $\tilde{\mathbf{Q}}_\theta$ and 88% when considering the top two local maxima on our augmented dataset.

The GG-CNN contains 62,000 parameters takes on average 2.1 ms to compute a forward pass for a single depth image. Computation of the entire grasping pipeline (Section 4.2), including image acquisition, pre-processing, and overhead in transferring data to and from the GPU, takes 19 ms, with the code predominantly written in Python. We use GG-CNN as a baseline to design an improved network for real-time grasp prediction, which we call GG-CNN2.

GG-CNN2 is a fully convolutional network based on the semantic segmentation architecture from Yu and Koltun (2016), which uses dilated convolutional layers to provide improved performance in semantic segmentation tasks (Figure 6). The GG-CNN2 uses the same input and outputs as the GG-CNN. To accelerate evaluation, we use the IoU metric for local testing of many different network configurations, and use the SGT simulator for only a smaller subset of well-performing networks. We vary three parameters: filter sizes, number of filters, and size of dilated convolutions. In addition, we look at the effects of dataset augmentation and filtering of the network output. In Section 8 we give a more in-depth analysis of the network design and training for GG-CNN2. In each case, we train each candidate network using 95% of the Jacquard dataset, and test performance on a held-out verification set comprising 5% of the dataset (approximately 2,500 images). On the Jacquard dataset, highest performing network achieves 84% using the IoU metric and 85% success rate in SGTs. The final network contains 66,000 parameters, has an average inference time of 3 ms, with the entire grasping pipeline taking on average 20 ms.

Both GG-CNN and GG-CNN2 are significantly smaller and faster to compute than the CNNs used for grasp candidate classification in other works that contain hundreds of thousands (He et al., 2016; Levine et al., 2016) or millions (Johns et al., 2016; Mahler et al., 2017; Pinto and Gupta, 2016; Redmon and Angelova, 2015) of parameters, with a comparison given in Table 1.

Table 1. Comparison of network sizes used for grasp prediction, and the execution time of the complete grasping pipeline. Execution times for GG-CNN are measured on a PC running Ubuntu 16.04 with a 3.6 GHz Intel Core i7-7700 CPU and NVIDIA GeForce GTX 1070 graphics card.

	Parameters (approx.)	Time
Lenz et al. (2015)		13.5 s
Pinto and Gupta (2016)	60 million	
Johns et al. (2016)	60 million	
Mahler et al. (2017)	18 million	0.8 s
Levine et al. (2016)	1 million	0.2–0.5 s
GG-CNN	62,000	19 ms
GG-CNN2	66,000	20 ms

4.6. Testing and results

We compare the effects of both training data and network design by comparing the best-performing GG-CNN and GG-CNN2 architectures under a number of conditions. We use the Jacquard dataset as our testing environment, owing to its larger size, more varied objects, and the ability to use SGTs that allow for testing different networks on identical simulated grasping scenarios. In the cases where the network is trained and tested on the Jacquard dataset, we perform five-fold cross-validation where the network is trained on 95% of the Jacquard dataset and evaluated against the remaining 5% (approximately 2,500 images) that were unseen during training and quote the average result. The results are summarized in Table 2. Both GG-CNNs outperform the original results from Depierre et al. (2018), which train an AlexNet network (approximately 60 million parameters) on the Jacquard dataset and evaluate under the same conditions, despite GG-CNN being orders of magnitude smaller.

4.6.1. GG-CNN versus GG-CNN2. The performance of the GG-CNN2 is consistently higher than that of the GG-CNN when evaluated using SGTs, regardless of the

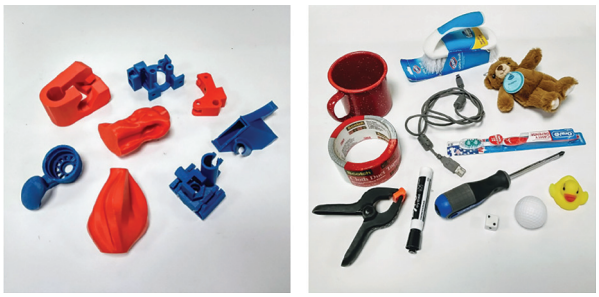

Fig 7. The objects used for grasping experiments. Left: The 8 adversarial objects from Mahler et al. (2017). Right: The 12 household objects selected from Calli et al. (2015) and Leitner et al. (2017).

training set used. For the IoU metric, the only exception is GG-CNN outperforms GG-CNN2 when trained on the Cornell Grasping Dataset and evaluated on the Jacquard dataset. The impact of the networks' design choices are discussed further in Section 8.

4.6.2. Effect of dataset. We train both GG-CNN and the GG-CNN2 on the Cornell and Jacquard datasets in order to observe the effect that the training dataset plays. In both cases, the networks trained on the Jacquard dataset exhibit significantly better performance in terms of both IoU and success rate using SGTs, which is unsurprising given that the Jacquard dataset is significantly larger and more diverse.

4.6.3. Effect of data augmentation. We find that augmenting the training data is a critical step in being able to transfer the GG-CNN to a physical robot. Since all images captured in both the Cornell and Jacquard datasets are from a single viewpoint, we apply two types of augmentation to the data to allow for a robust transfer to the robot. We apply both random rotations, and random crops and zooms of the dataset images to achieve viewpoint and depth invariance. We find that without this augmentation of the dataset, the trained network is completely unable to function on data collected from a real camera placed at an arbitrary position, and does not generate any valid grasps. In particular, this highlights the importance of data augmentation for training a robotic grasping system, rather than optimizing for image-based datasets which, in this case, produces brittle results that do not transfer to a real robotic system.

On the other hand, the network trained without any data augmentation achieves a higher performance in terms of IoU (84% versus 82%) and on simulated grasps (85% versus 79%) on the Jacquard dataset, indicating that it overfits to the dataset (Table 2).

4.6.4. Verification on robot. To validate the performance of both networks on a real robot, we perform 10 open-loop grasp attempts on the adversarial and household objects sets (Figure 7) using the GG-CNN2 and show slightly improved results compared to the GG-CNN.

5. Reactive closed-loop grasping

The primary advantage of our real-time grasp generation approach is the ability to use it in a reactive, closed-loop fashion. In this section we present several grasping experiments in both static and dynamic environments. First, we compare with existing work on robotic grasping, by grasping singulated, static objects from our two object sets. Second, we evaluate grasping of objects which are moved during the grasp attempt, to show the ability of our system to perform dynamic grasping. Third, our system's ability to generalize to dynamic cluttered scenes is presented by reproducing the experimental setup from Viereck et al. (2017). We further show the advantage of our closed-loop grasping method over open-loop grasping by performing grasps in the presence of simulated kinematic errors of our robot's control.

In order to compare our results with other published techniques, we aim to reproduce similar experiments where possible. We also aim to present experiments that are reproducible in themselves by using our setups, our defined set of objects (Section 5.2), and defined dynamic motions.

Table 2. Results of training the two architectures with different training data, evaluated on the Jacquard dataset and a robot using both household (HH) and adversarial (Adv.) objects.

			Jacquard		
Network	**Trained on**	**Data augmentation**	**IoU**	**SGT**	**Robot**
AlexNet (Depierre et al., 2018)	Jacquard	No	74%	72%	78%
GG-CNN	Cornell	Yes	73%	69%	91% (HH) / 83% (Adv.)
GG-CNN	Jacquard	No	78%	79%	—
GG-CNN	Jacquard	Yes	77%	74%	—
GG-CNN2	Cornell	Yes	65%	70%	—
GG-CNN2	Jacquard	No	**84%**	**85%**	0%
GG-CNN2	Jacquard	Yes	82%	79%	**94%** (HH) / **84%** (Adv.)

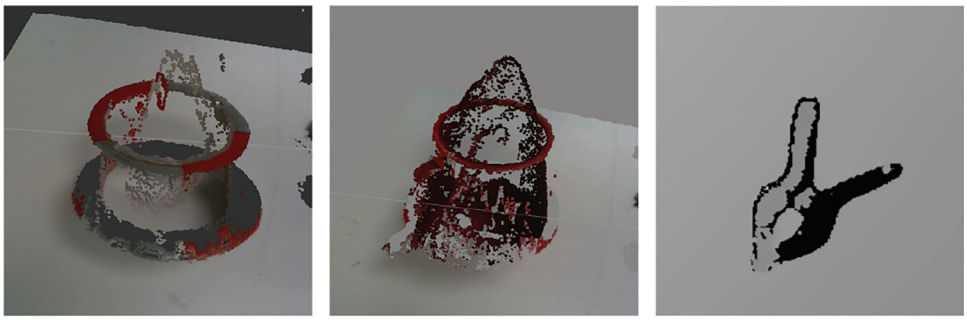

Fig 8. Examples of perceptually challenging objects from our object set as seen by an Intel Realsense SR300 camera. (a) and (b) The reflective edge of the tape and mug cause large amounts of sensor noise and false depth readings. (c) Many parts of the black clamp are not visible to the depth camera (black areas of the image indicate missing depth points).

5.1. Setup and limitations

These experiments were performed using the GG-CNN network, trained on the Cornell Grasping Dataset (Morrison et al., 2018a). We use a Kinova Mico six-degree-of-freedom (6DOF) robot fitted with a Kinova 2-fingered gripper (KG-2) and an Intel RealSense SR300 RGB-D camera. The camera is mounted to the wrist of the robot (eye-in-hand setup), approximately 80 mm above the closed fingertips and inclined at 14° towards the gripper (Figure 3 (a)).

The RealSense camera has a specified minimum range of 200 mm. In reality, we find that the RealSense camera is unable to produce accurate depth measurements from a distance closer than 150 mm, as the separation between the camera's infra-red projector and camera causes shadowing in the depth image by the object. As such, when performing closed-loop grasping trials (Section 5.4.2), we stop updating the target grasp pose at this point, which equates to the finger tips being approximately 70 mm from the object. In addition, we find that the RealSense camera is unable to provide valid depth data on many black or reflective objects, however we still include these in our object set (Figure 8).

The Kinova KG-2 gripper has a maximum opening width of 175 mm, which could easily envelop all of the test items. To encourage more precise grasps, we limit the maximum gripper width to approximately 70 mm. The fingers of the gripper have some built-in compliance and naturally splay slightly at the tips. This means that objects with a height less than 15 mm (especially those that are cylindrical, like a thin pen) cannot be grasped.

5.2 Test objects

There is no set of test objects that are commonly and repeatedly used for robotic grasping experiments. Many researchers choose to use random "household" objects that are not easily reproducible. We propose here two sets of reproducible benchmark objects (Figure 7) on which we test the grasp success rate of our approach.

5.2.1 Adversarial set. The first set consists of eight 3D-printed objects with adversarial geometry, which were used by Mahler et al. (2017) to verify the performance of their grasp quality CNN. The objects all have complex geometry, meaning there is a high chance of a collision with the object in the case of an inaccurate grasp, as well as many curved and inclined surfaces which are difficult or impossible to grasp. The object models are available online as part of the released datasets for Dex-Net 2.0[1] (Mahler et al., 2017).

5.2.2 Household set. This set of items contains 12 household items of varying sizes, shapes, and difficulty with minimal redundancy (i.e. minimal objects with similar shapes). The objects were chosen from two standard robotic grasping datasets: the APB (Leitner et al., 2017) and the YCB Object Set (Calli et al., 2015). Both provide item specifications and online purchase links.[2] Half of the item classes (mug, screwdriver, marker pen, die, ball, and clamp) appear in both datasets. We have made every effort to produce a balanced object set containing objects that are deformable (bear and cable), perceptually challenging (black clamp and screwdriver handle, thin reflective edges on the mug and duct tape, and clear packaging on the toothbrush as shown in Figure 8), and objects which are small and require precision (golf ball, duck, and die).

While both the APB and YCB object sets contain a large number of objects, many are physically impossible for our robot to grasp due to being too small and thin (e.g. screws, washers, envelope), too large (e.g. large boxes, saucepan, soccer ball) or too heavy (e.g. power drill, saucepan). While manipulating these objects is an open problem in robotics, we do not consider them for our experiments in

order to compare our results with other papers that use similar object classes to ours (Johns et al., 2016; Lenz et al., 2015; Levine et al., 2016; Mahler et al., 2017; Pinto and Gupta, 2016).

5.3. Object placement

To remove bias related to object pose, objects are shaken in a cardboard box and emptied into the robot's workspace for each grasp attempt, for both isolated and cluttered scenarios. The workspace is an approximately 250×300 mm^2 area in the robot's field of view in which the robot's kinematics allow it to execute a vertical grasp.

5.4 Grasp execution

We evaluate the performance of our system using two grasping methods. First, an open-loop grasping method similar to Lenz et al. (2015), Pinto and Gupta (2016), and Mahler et al. (2017), where the best grasp pose is calculated from a single viewpoint and executed by the robot open-loop. Second, we implement a closed-loop visual servoing controller that we use for evaluating our system in dynamic environments.

5.4.1. Open-loop grasp method. To perform open-loop grasps, the camera is positioned approximately 350 mm above and parallel to the surface of the table. An item is placed in the field of view of the camera. A depth image is captured and the pose of the best grasp is computed using the grasp detection pipeline. The robot moves to a pre-grasp position, with the gripper tips aligned with and approximately 170 mm above the computed grasp. From here, the robot moves straight down until the grasp pose is met or a collision is detected via force feedback in the robot. The gripper is closed and lifted, and the grasp is recorded as a success if the object is successfully lifted to the starting position.

5.4.2. Closed-loop grasp method. To perform closed-loop grasping, we implement a position-based visual servoing (PBVS) controller (Kragic and Christensen, 2002). The camera is initially positioned approximately 400 mm above the surface of the table, and an object is placed in the field of view. Depth images are generated at a rate of 30 Hz and processed by the grasp detection pipeline to generate grasp poses in real time.

The system is initialized to track the global maxima of $\tilde{\mathbf{Q}}_\theta$ at the beginning of each grasp attempt. However, there may be multiple high-quality grasps (local maxima) detected in an image. To avoid switching rapidly between them, we implement a simple method to track the position of grasps between consecutive frames by selecting the grasp which is closest (in image coordinates) to the tracked grasp from the previous frame. Because the camera's frame rate is fast compared with the movement of the robot, the difference between frames due to the camera motion is typically minimal. Similarly, while this method does not take into account angular difference, in practice we do not notice any large discontinuities in angle between frames that adversely effect performance.

We represent the poses of the grasp $\mathbf{T}_{\mathbf{g}^*_\theta}$ and the gripper fingers $\mathbf{T}_f$ as 6D vectors comprising the Cartesian position and roll, pitch, and yaw Euler angles $(x, y, z, \alpha, \beta, \gamma)$, and generate a 6D velocity signal for the end-effector:

$$\mathbf{v} = \lambda(\mathbf{T}_{\mathbf{g}^*_\theta} - \mathbf{T}_f)$$

where λ is a 6D scale for the velocity, which causes the gripper to converge to the grasp pose. Simultaneously, we control the gripper fingers to the computed gripper width via velocity control. Control is stopped when the grasp pose is reached or a collision is detected. The gripper is closed and lifted and the grasp is recorded as a success if the object is successfully lifted to the starting position.

5.5. Static grasping

To evaluate the performance of our GG-CNN under static conditions, we performed grasping trials using both the open- and closed-loop methods on both sets of test objects, using the setup shown in Figure 9(a). We perform 10 trials on each object. For the adversarial object set, the grasp success rates were 84% (67/80) and 81% (65/80) for the open- and closed-loop methods, respectively. For the household object set, the open-loop method achieved 92% (110/120) and the closed-loop 91% (109/120).

We note that the results may not be directly comparable owing to the different objects and experimental protocol used. We however aim to show that we achieve comparable performance to other works using much larger neural networks with longer computation times (Table 3). A noteworthy difference in method is Levine et al. (2016), which does not require precise camera calibration, but rather learns the spatial relationship between the robot and the objects using vision. However, this approach incurs a significant training overhead.

Viereck et al. (2017) demonstrated a visuomotor controller for robotic grasping in clutter that is able to react to disturbances to the objects being grasped. As this work is closely related to our own, we have made an effort to recreate their experiments using objects as close as possible to their set of 10 (Figure 10) to perform a comparison. Even though our GG-CNN has not been trained on cluttered environments, we show here its ability to perform grasping in the presence of clutter. We recreate grasping experiments from Viereck et al. (2017) as follows.

5.5.1 Isolated objects. We performed four grasps on each of the 10 test objects (Figure 10) in isolation, and achieved a grasp success rate of 100%, compared with 98% (39/40) in Viereck et al. (2017).

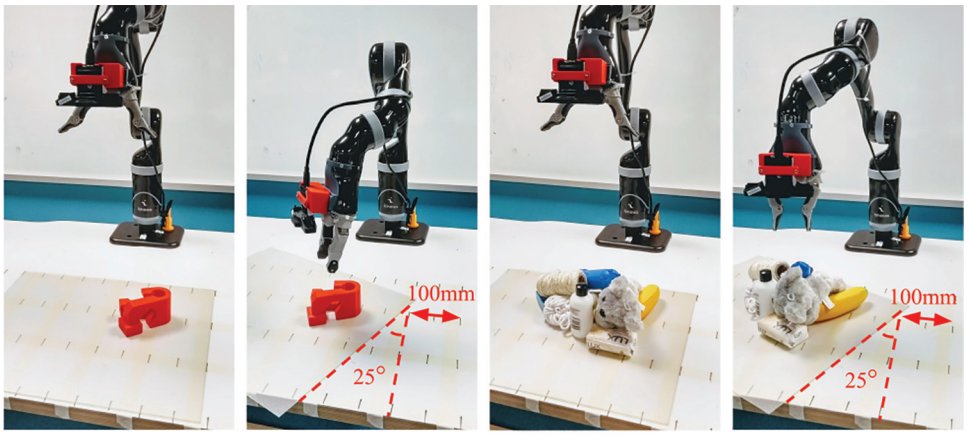

Fig 9. Grasping experiments. (a) Setup for static grasping, and initial setup for dynamic grasping. (b) During a dynamic grasp attempt, the object is translated at least 100 mm and rotated at least 25°, measured by the grid on the table. (c) Setup for static grasping in clutter, and initial setup for dynamic grasping in clutter. (d) During a dynamic grasp attempt, the cluttered objects are translated at least 100 mm and rotated at least 25°, measured by the grid on the table.

Table 3. Results for our approach compared with other methods. Including open- and closed-loop experiments on static and dynamic objects, with 95% confidence intervals provided.

	Grasp success rate (%)		
Object set	Household	Advers.	Viereck
Static objects			
Lenz et al. (2015)	89		
Pinto and Gupta (2016)	73		
Johns et al. (2016)	80		
Viereck et al. (2017)			
Mahler et al. (2017)	80	**93***	
Viereck et al. (2017)			98 +
GG-CNN (Ours)	**92±5**	84±8	**100** +
Dynamic objects			
Viereck et al. (2017)			77
GG-CNN (Ours)	88±6	83±8	**81±8**

∗In contrast to our approach, Mahler et al. (2017) trained their network on the adversarial objects.
+ Performance reported for isolated items.

5.5.2 Cluttered objects. The 10 test objects are shaken in a box and emptied in a pile below the robot (Figure 9(c)). The robot attempts multiple grasps, and any objects that are grasped are removed. This continues until all objects are grasped, three consecutive grasps are failures or all objects are outside the workspace of the robot. We run this experiment 10 times.

Despite our GG-CNN not being trained on cluttered scenes, we achieved a grasp success rate of 87% (83/96) compared with 89% (66/74) in Viereck et al. (2017). Our most common failure cause was collision of the gripper with two objects that had fallen up against each other. A total of 8 out of the 13 failed grasps were from two runs where objects had fallen into an ungraspable position and failed repeatedly. A total of 8 out of the 10 runs finished with 0 or 1 grasp failures.

5.6 Dynamic grasping

To perform grasps on dynamic objects we take further inspiration from Viereck et al. (2017), where items are moved once by hand randomly during each grasp attempt. To assist reproducibility, we define this movement to consist of a translation of at least 100 mm and a rotation of at least 25° after the grasp attempt has begun, as shown in Figure 9(a) and (b), which we measure using a grid on the table.

We perform 10 grasp attempts on each adversarial and household object using our closed-loop method, and achieve grasp success rates of 83% (66/80) for the adversarial objects and 88% (106/120) for the household objects. These results are not significantly different to our results on static objects, and are within the 95% confidence bounds of our results on static objects, showing our method's ability to maintain a high level of accuracy when grasping dynamic objects.

We do not compare directly to an open-loop method as the object movement moves the object sufficiently far from the original position that no successful grasps would be possible.

5.7. Dynamic grasping in clutter

For dynamic scenes, we combine the procedures from above: 10 objects are shaken in a box and emptied in a pile and we add a random movement of the objects during the grasp. Again, at least 100 mm and 25° for each attempt (Figure 9(d)).

Fig 10. The objects used in this paper (left) to reproduce the dynamic grasping in clutter experiment by Viereck et al. (2017) (right). We have attempted to closely recreate the object set.

In 10 runs of the experiment, we performed 94 grasp attempts of which 76 were successful (81%), compared with 77% (58/75) in Viereck et al. (2017). Like the static case, 8 of the 18 failed grasps were from two runs where the arrangement of the objects resulted in repeated failed attempts. In the other eight runs, all available objects (i.e. those that did not fall/roll out of the workspace) were successfully grasped with two or fewer failed grasps.

Despite not being trained on cluttered scenes, this shows our approach's ability to perform grasping in clutter and its ability to react to dynamic scenes, showing only a 5% decrease in performance for the dynamic case compared with 12% in Viereck et al. (2017).

For the same experiments, Viereck et al. (2017) shows that an open-loop baseline approach on the same objects that is able to achieve 95% grasp success rate for the static cluttered scenes achieves only 23% grasp success rate for dynamic scenes as it is able to react to the change in item location.

5.8. Robustness to control errors

The control of a robot may not always be precise. For example, when performing grasping trials with a Baxter Research Robot, Lenz et al. (2015) found that positioning errors of up to 20 mm were typical. A major advantage of using a closed-loop controller for grasping is the ability to perform accurate grasps despite inaccurate control. We show this by simulating kinematic inaccuracies by introducing a cross-correlation between Cartesian (x, y, and z) velocities:

$$\mathbf{v}_c = \mathbf{v} \cdot \begin{pmatrix} 1+c_{xx} & c_{xy} & c_{xz} \\ c_{yx} & 1+c_{yy} & c_{yz} \\ c_{zx} & c_{zy} & 1+c_{zz} \end{pmatrix}$$

where each $c \sim \mathcal{N}(0, \sigma^2)$ is sampled at the beginning of each grasp attempt. While a real kinematic error (e.g. a link length being incorrectly configured) would result in a more nonlinear response, simulated control errors provide a good approximation which is independent of the robot's kinematic model, so has a deterministic effect with respect to end-effector positioning and is more easily replicated on a different robotic system.

We test grasping on both object sets with 10 grasp attempts per object for both the open- and closed-loop methods with $\sigma = 0.0$ (the baseline case), 0.05, 0.1, and 0.15. In the case of our open-loop controller, where we only control velocity for 170 mm in the z direction from the pre-grasp pose (Section 5.4.1), this corresponds to having a robot with an end-effector precision described by a normal distribution with zero mean and standard deviation 0.0, 8.5, 17.0, and 25.5 mm, respectively, by the relationship for scalar multiplication of the normal distribution:

$$\Delta x = \Delta y = \Delta z \cdot \mathcal{N}(0, \sigma^2) = \mathcal{N}(0, \Delta z^2 \sigma^2)$$

$$\Delta z = 170\text{mm}$$

The results are illustrated in Figure 11, and show that the closed-loop method outperforms the open-loop method in the presence of control error. This highlights a major advantage of being able to perform closed-loop grasping, as the open-loop methods are unable to respond, achieving only 38% grasp success rate in the worst case. In comparison, the closed-loop method achieves 68% and 73% grasp success rate in the worst case on the adversarial and household objects, respectively.

The decrease in performance of the closed-loop method is due to the limitation of our camera (Section 5.1), where we are unable to correct for errors in close range, i.e. when the gripper is within 70 mm of the object. Objects which require precise grasps (e.g. adversarial objects and smaller ones, such as the die and ball) are most effected by control error or inaccuracies. Simpler objects which are more easily caged by the gripper, such as the pen, still report good results.

5.9. Effect of output filtering

As mentioned in Section 4.2, we filter the neural network output $\tilde{\mathbf{Q}}_\theta$ with a Gaussian kernel, similar to Johns et al. (2016). To test the effect of this, we perform 10 open- and closed-loop grasps on each of the adversarial objects without any filtering. We do not observe a large performance decrease in the open-loop case, achieving 80% grasp success rate, as the local maxima of the unfiltered $\tilde{\mathbf{Q}}_\theta$ still tend to correspond to good quality grasps. However, we note a

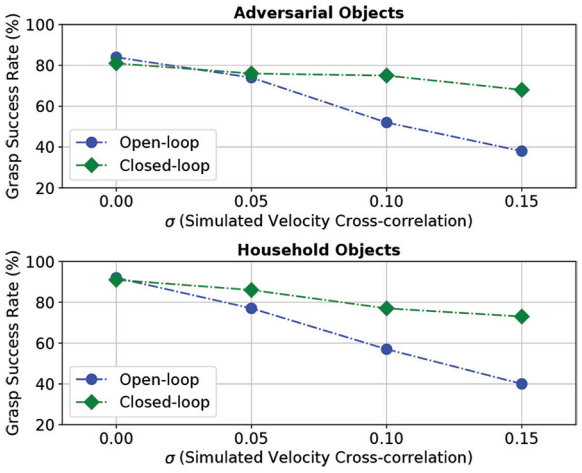

Fig 11. Comparison of grasp success rates for open- and closed-loop control methods with simulated kinematic errors (see Section 5.8 for full details). The closed-loop method outperforms the open-loop method in all cases where kinematic inaccuracies are present. A total of 10 trials were performed on each object in both the adversarial and household object sets.

large drop in performance for the closed-loop grasping, achieving only 70% grasp success without the filter. Removing the filter in the closed-loop case causes the local maxima of $\tilde{\mathbf{Q}}_\theta$ to change more rapidly between frames, resulting in the controller not converging to a single good pose.

6. Multi-viewpoint grasping in clutter

In this section we present a novel approach to performing multi-viewpoint grasp detection from a densely packed pile of cluttered objects. Grasping objects from dense clutter presents a much more challenging environment for robotic grasp detection systems (Mahler and Goldberg, 2017). One major challenge is that many visual occlusions occur. Hence, the viewpoint from which the scene is perceived plays an important role in accurately predicting a grasp pose.

Prior work has proposed methods for overcoming this challenge by performing point cloud fusion from multiple viewpoints, and performing grasp pose detection on the fused point cloud (Arruda et al., 2016; Kahn et al., 2015; ten Pas et al., 2017). For example, ten Pas et al. (2017) collected a fused point cloud of a cluttered environment, achieving a 9% increase in grasp success rate compared with using a point cloud collected from two static cameras. In these cases, the secondary task of fusing a point cloud is used as a data collection mechanism, rather than directly optimizing for grasp quality. In an alternative approach, Gualtieri and Platt (2017) use a set of heuristics to determine the best viewpoint for grasp prediction for specific object classes.

Many existing grasp detection methods perform grasp prediction via a sample-and-rank approach, or generating only a single predicted grasp pose. These do not easily lend themselves to combining grasp estimates from multiple viewpoints or defining a distribution over observed grasp estimates (Gualtieri and Platt, 2017). Adding to this, the expensive computation of most other methods makes them impractical to compute grasp poses from multiple viewpoints in real time.

Our GG-CNN approach, however, overcomes these limitations. By generating a dense, pixel-wise prediction of grasp poses at every viewpoint, we can easily create a distribution of observed grasp estimates. In addition, the real-time performance of our approach means that there is practically no additional computation overhead involved in computing the grasp poses from multiple viewpoints beyond movement of the robot. As such, we combine grasp pose estimates from multiple viewpoints along a trajectory to improve the quality of grasping from clutter.

6.1. Combining multiple grasp predictions

At each observation, the GG-CNN provides 90,000 (300×300) grasp pose predictions, each of which corresponds to a 3D position in space (Section 3). To combine observations from multiple viewpoints, we represent the workspace of the robot as a 2D grip map of cells, M, where each cell represents a small area of space. From each viewpoint, each predicted grasp is recorded in its corresponding grid cell. We are then able to compute the average grasp pose $\bar{\mathbf{g}} = (\mathbf{p}, \bar{\phi}, \bar{w}, \bar{q})$ from N observations for each cell as follows. Here $\mathbf{p}$ is given by the center position of the grid cell. The average grasp quality and average grasp width are simply given by the mean of observations:

$$\bar{q} = \frac{1}{N} \sum_{n=1}^{N} q_n \tag{2}$$

$$\bar{w} = \frac{1}{N} \sum_{n=1}^{N} \tilde{w}_n \tag{3}$$

and the mean angle is the vector mean of the observed angles (Mardia, 2015) weighted by the corresponding grasp quality observations:

$$\bar{\phi} = \arctan \frac{\sum_{n=1}^{N} \sin(\tilde{\phi}_n) q_n}{\sum_{n=1}^{N} \cos(\tilde{\phi}_n) q_n} \tag{4}$$

By using the vector mean, as well as our decomposed representation of the angle in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$ (Section 4.1), we avoid discontinuities in the predicted grasp angle. Some small symmetrical objects, e.g. a small ball, may be graspable at any angle resulting in multiple predicted orientations, however due to the object's symmetry the mean grasp angle is generally also valid.

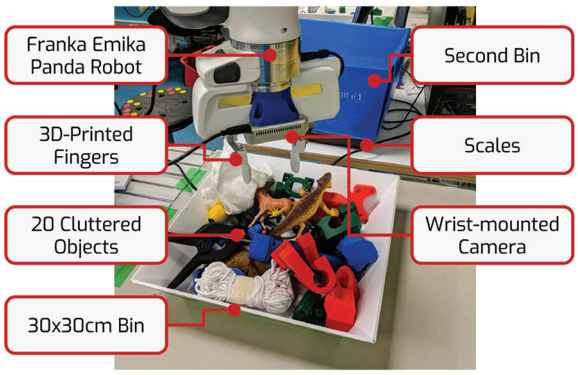

Fig 12. Experimental setup of the Franka Emika Panda robot for grasping from clutter.

Fig 13. Extended set of objects used for cluttered grasping experiment, comprising 20 household and 20 adversarial objects.

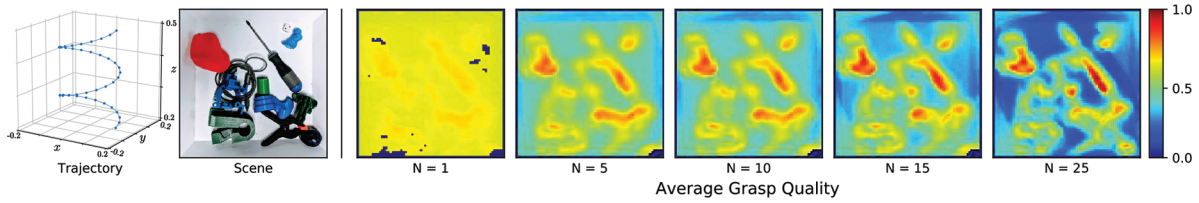

Fig 14. (Left) The robot executes a spiral trajectory while reaching towards a cluttered environment. (Right) As multiple grasp estimates are averaged together, the estimate becomes more refined, resulting in an improved grasp success rate.

6.2. Multi-view grasping experiments

To demonstrate this approach, we use a Franka Emika Panda robot with a wrist-mounted Intel RealSense D435 to perform grasping from dense clutter, shown in Figure 12. The robot is fitted with custom, 3D-printed fingers with silicone tips using the design from Guo et al. (2017). As our approach relies only on depth information, it is not tied to a specific robotic platform and transferring to a new robot and camera is trivial.

We use an extended object set, comprising 20 adversarial objects and 20 household objects in order to create a more varied and complex cluttered scene than in Section 5.7 (Figure 13). For each experimental run, we randomly select 20 objects, comprising 10 adversarial and 10 household objects. As above, the objects are mixed and then emptied into a bin (300×300 mm^2). The robot's task is to empty the bin by attempting to remove the objects one at a time. A *run* ends when all objects have been successfully removed from the bin.

For the multi-viewpoint data collection procedure, we use a scripted spiral trajectory illustrated in Figure 14. We run two experiments capturing 25 and 50 viewpoints evenly spaced along the trajectory. At the end of the trajectory, we choose and execute a grasp in the cell with the highest average grasp quality $\bar{q}$. Each experiment comprises five runs of emptying the bin, and we report the average grasp success rate across all runs. As a baseline for comparison, we also perform seven runs where the best grasp pose is detected from a single viewpoint centered over the bin and executed by the robot open-loop.

Table 4. Results from multi-viewpoint grasping experiments.

Viewpoints:	1	25	50
Total attempts	196	137	125
Failures	62	37	28
Grasp success rate (%)	68	73	**78**
Mean time per grasp (s)	8.8	11.4	11.4

The results from our experiments are shown in Table 4. The number of viewpoints considered plays a very strong role in the quality of the predicted grasp points when grasping from dense clutter. A 10% improvement in the overall grasp success rate was observed when using 50 viewpoints compared to a single, central viewpoint.

As more grasp predictions are combined, the average estimate of the grasp quality becomes more refined, and allows the system to discover high-quality grasps that may not be clearly visible from the initial viewpoint, or to disambiguate potential false positive grasp poses (Figure 14). As shown in the plot, high-quality grasps (such as the screwdriver handle) become well defined when multiple viewpoints are combined, whereas regions of higher clutter where the grasp prediction is more dependent on viewpoint and susceptible to occlusion remain uncertain. The multi-viewpoint technique

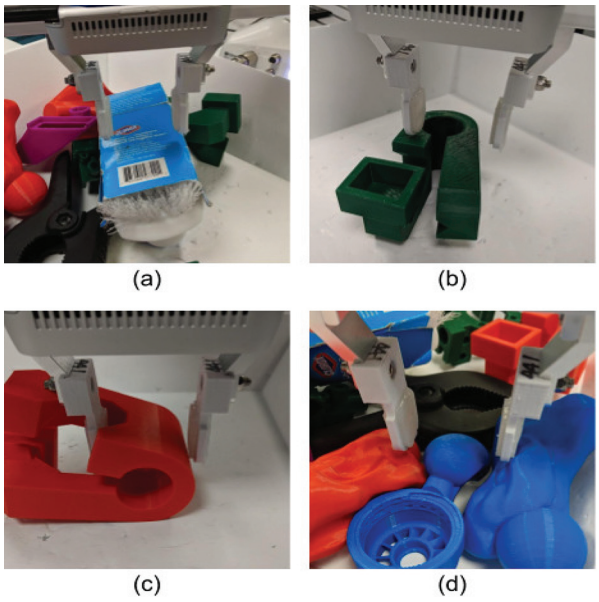

Fig 15. Common failure modes, see Section 7 for details.

does come with a small time penalty (2.6 s on average) owing to the robot's longer trajectory, but is not a result of computational overhead.

Qualitatively, we also find that using multiple viewpoints overcomes many of the limitations of our camera hardware. By viewing perceptually challenging objects, such as those in Figure 8, from multiple viewpoints, the effects of the noisy or missing depth data is greatly diminished.

7. Failure modes

The primary failure modes of the system can be broadly categorized into two main categories. The first is failures that are caused by inaccurate visual information. This can often be caused by objects that are not perceived well by the depth camera, such as those that are reflective, transparent, or black (Figure 8). On the other hand, common failures that result from the GG-CNN's prediction are shown in Figure 15. Incorrectly estimating the grasp width most often occurs on wide (Figure 15(a)) and narrow objects (Figure 15(b)), resulting in a collision with the gripper, which may be a result of few examples of such grasps in the training data. In some cases, a predicted grasp may be on a curved surface (Figure 15(c)), causing the object to slip out of the gripper. In clutter, the most common failure mode was a collision with nearby objects (Figure 15(d)) when densely packed together.

8. Network design analysis

Many grasp detection networks are adapted versions of large CNNs designed for computer vision tasks such as image classification (Johns et al., 2016; Kumra and Kanan, 2017; Pinto and Gupta, 2016; Redmon and Angelova, 2015), which have relatively long execution times. In contrast, the GG-CNN and GG-CNN2 network architectures are defined by a small number of parameters that allow for real-time inference. We performed a number of parameter sweeps, varying the number of convolutional filters, filter sizes, and dilation parameters in order to find the network design that offers the best performance while allowing for real-time execution. In this section, we present a subset of results for choosing the GG-CNN2 network parameters by testing against the Jacquard dataset.

The SGT's long computation time makes it impractical for performing large-scale parameter sweeps on the simulator. Initially we use IoU as the performance metric to compare the large number of networks (as seen in Figures 16 and 17). A final subset of high-performing networks were evaluated with the SGT environment (Figure 18).

We first look at the effects of jointly varying the size of the convolutional filter at each layer as well as the number of filters. The results for the first layer are shown in Figure 16, and similar analyses were performed for each of the remaining layers. We observe that, especially in the first layer, the size of the convolutional filter plays a large role in the performance of the network. By comparison, adding extra convolutional filters to the first layer provides very little benefit compared with the increased execution time. Increasing the filter kernel size increases the receptive field of the network, and hence the amount of spatial context used to make predictions, which proves to be an important aspect of the network design. We discuss this further in the following when adding dilated convolutions.

Second, we vary the number of convolutional filters per layer, shown in Figure 17. Adding additional convolutional filters has the largest effect on execution time. We choose the set of parameters, indicated in red, which provide the greatest benefit in IoU score relative to the increased computation time of the network. In each case there is a point beyond which adding extra convolutional filters has very little impact on the performance of the network, while still increasing computation time. This indicates that the extra size of larger neural networks used in other work may be redundant when predicting grasps from depth images.

Above we identified that increasing the network's receptive field is an important aspect. While larger networks benefit from increased receptive field with increased depth (Yu and Koltun, 2016), this comes with significant extra computational overhead. As such, we finally investigate the effect of adding dilated convolutions in the fifth and sixth layer of the network. Dilated convolutions increase the receptive field of the network, allowing the network to aggregate contextual information from multiple scales within the input image while adding very little computational overhead, resulting in improved performance in tasks such as semantic segmentation (Yu and Koltun, 2016). We find that, to a point, increasing the receptive field of the network provides a significant boost to the performance of the network (Figure 18). We observe similar results to those when varying the network kernel sizes (Figure 16),

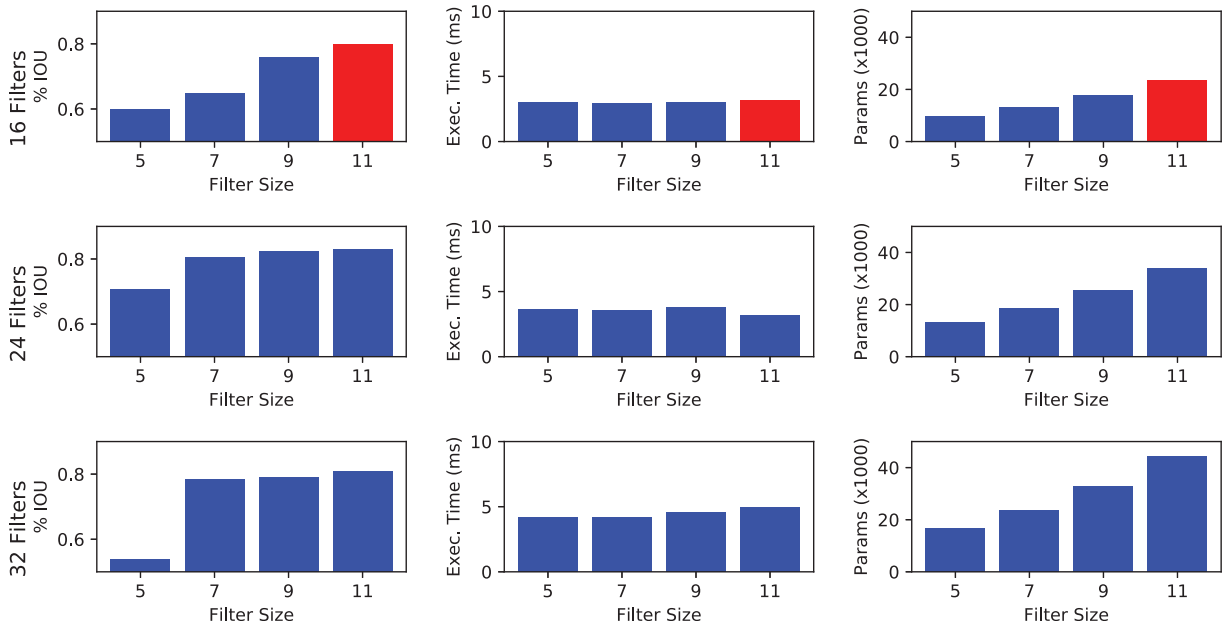

Fig 16. Results of training GG-CNN2 with different filter sizes (only Layer 1 shown here), evaluated using the IoU metric against the Jacquard dataset. Red indicates the chosen parameters.

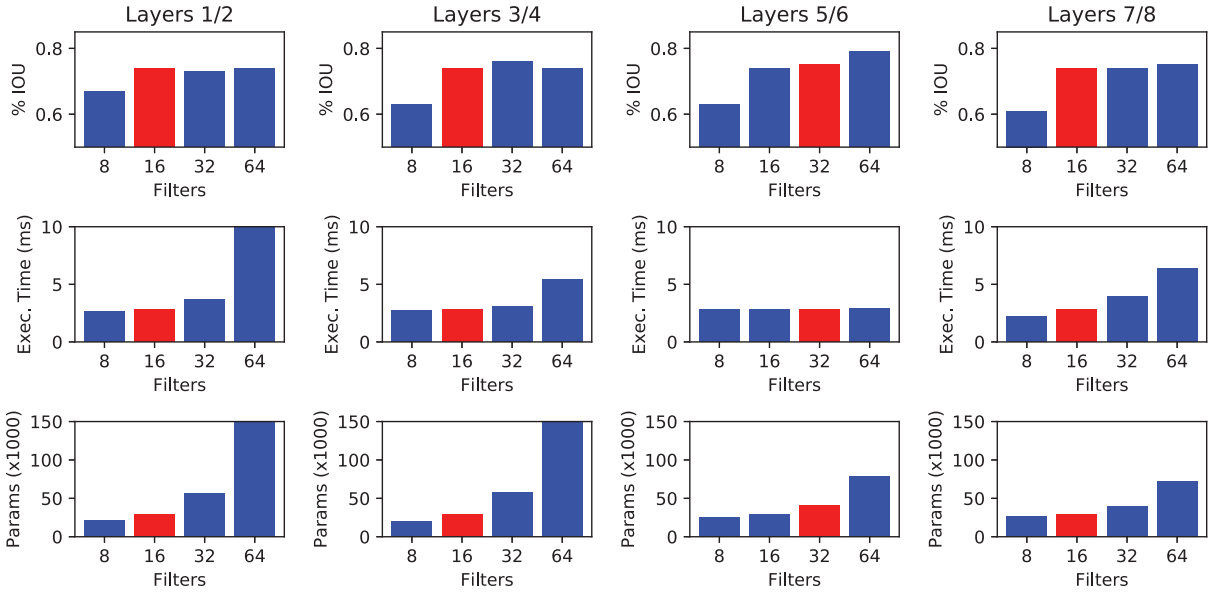

Fig 17. Results of training GG-CNN2 with different numbers of convolutional filters per layer, evaluated using the IoU metric against the Jacquard dataset. Red indicates the chosen parameters.

which also has the effect of increasing the network's receptive field, albeit with a significant increase in the number of network parameters and computation time compared with the dilated convolution approach.

9. Discussion

We have presented a real-time, generative approach to robotic grasping. Our GG-CNN is an object-independent grasp synthesis model that directly generates grasp poses from a depth image of a pixelwise basis. In doing this, we overcome a large limitation of other deep learning techniques for grasp detection, which rely on sampling and classifying individual grasp candidates, resulting in long computation times. Our GG-CNN is orders of magnitude smaller than other recent grasping networks, allowing us to generate grasp poses at a rate of up to 50 Hz, enabling closed-loop control.

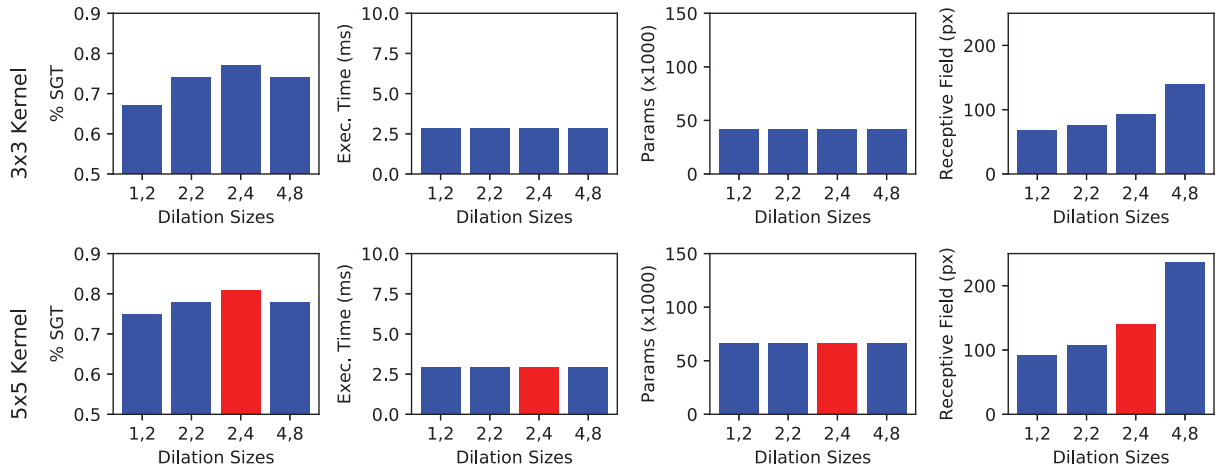

Fig 18. Results of training GG-CNN2 with varying sizes of dilated convolutions in the fifth and sixth convolutional layers, increasing the receptive field of the network, evaluated against the Jacquard dataset by SGTs. Red indicates the chosen parameters.

Performing robotic grasping in a closed-loop manner has a number of advantages over open-loop solutions, especially in the case of the types of dynamic, unstructured environments a robot may encounter in the real world. We have highlighted these advantages by performing grasping trials in a number of different scenarios.

We have performed a number of grasping trials on previously unseen household and adversarial objects that are moved during the grasp attempt, achieving 88% and 83% grasp success rates, respectively. Alternative techniques with longer computation times are unable to react to such dynamic environments and fail to perform successful grasps. Using the same closed-loop approach, we have performed grasps in the presence of simulated kinematic errors of our robot's control, outperforming an open-loop solution, which relies on precise calibration and control in order to succeed.

In addition to real-time performance, the dense, pixel-wise grasp generation performed by our approach enables grasp predictions to be easily combined from multiple viewpoints. We have shown that this is important in complex scenes such as dense clutter, where many visual occlusions are present and high-quality grasps may not be visible from an initial viewpoint. Combining grasp predictions from 50 distinct viewpoints achieves a 10% improvement in success rate when grasping from clutter.

Designing a light-weight network for real-time execution requires a trade-off between performance and execution time. We have presented results from our two architectures, highlighting some of the important design aspects. We have also investigated the effect of using different training datasets, discussing the importance of dataset augmentation and image pre- and post-processing for robustly transferring a learnt grasp generation algorithm to real robots.

Using a robot as an active part of a closed-loop grasping pipeline is an important aspect in designing robust, reliable robotic systems. The ideas presented in this paper can be extended in a number of ways which would improve their performance further. For example, the current system does not pick specific objects, in fact the network is not even aware of objects or affordances. These though can easily be integrated by treating them as masks for our grasp map generated by the network.

In future work we propose to extend our multi-viewpoint picking technique to include an active perception component, allowing the robot to select informative viewpoints which reduce uncertainty in the grasp estimate in a more efficient way. Integrating other sensors such as touch (Calandra et al., 2017) or allowing the robot to execute other manipulation actions such as pushing (Zeng et al., 2018a) are also both powerful ways to improve the robustness of a closed-loop robotic grasping system in which the robot is an active and integral component.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research was supported by the Australian Research Council Centre of Excellence for Robotic Vision (project number CE140100016).

Notes

1. Available at: https://berkeleyautomation.github.io/dex-net/
2. Available at http://www.ycbbenchmarks.com/object-set/ and http://juxi.net/dataset/acrv-picking-benchmark/

ORCID iD

Douglas Morrison https://orcid.org/0000-0002-1700-4775

References

Arruda E, Wyatt J and Kopicki M (2016) Active vision for dexterous grasping of novel objects. In: *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2881–2888.

Badrinarayanan V, Kendall A and Cipolla R (2015) SegNet: A deep convolutional encoder–decoder architecture for image segmentation. arXiv preprint arXiv:1511.00561.

Bicchi A and Kumar V (2000) Robotic grasping and contact: A review. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 348–353.

Bohg J, Morales A, Asfour T and Kragic D (2014) Data-driven grasp synthesis – a survey. *IEEE Transactions on Robotics* 30(2): 289–309.

Bradski G (2000) The OpenCV library. *Dr. Dobb's Journal of Software Tools*.

Calandra R, Owens A, Upadhyaya M, et al. (2017) The feeling of success: Does touch sensing help predict grasp outcomes? In: *Proceedings of the Conference on Robot Learning (CoRL)*.

Calli B, Walsman A, Singh A, Srinivasa S, Abbeel P and Dollar AM (2015) Benchmarking in manipulation research: Using the Yale–CMU–Berkeley object and model set. *IEEE Robotics and Automation Magazine* 22(3): 36–52.

Depierre A, Dellandréa E and Chen L (2018) Jacquard: A large scale dataset for robotic grasp detection. In: *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*.

Detry R, Baseski E, Popovic M, et al. (2009) Learning object-specific grasp affordance densities. In: *Proceedings of the IEEE International Conference on Development and Learning (ICDL)*, pp. 1–7.

El-Khoury S and Sahbani A (2008) Handling objects by their handles. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Goldfeder C, Allen PK, Lackner C and Pelossof R (2007) Grasp planning via decomposition trees. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4679–4684.

Gualtieri M and Platt R (2017) Viewpoint selection for grasp detection. In: *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 258–264.

Guo M, Gealy DV, Liang J, et al. (2017) Design of parallel-jaw gripper tip surfaces for robust grasping. In: *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2831–2838.

Hara K, Vemulapalli R and Chellappa R (2017) Designing deep convolutional neural networks for continuous object orientation estimation. *arXiv preprint arXiv:1702.01499*.

He K, Zhang X, Ren S and Sun J (2016) Deep residual learning for image recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

Horaud R, Dornaika F and Espiau B (1998) Visually guided object grasping. *IEEE Transactions on Robotics and Automation* 14(4): 525–532.

Hutchinson S, Hager GD and Corke PI (1996) A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation* 12(5): 651–670.

Johns E, Leutenegger S and Davison AJ (2016) Deep learning a grasp function for grasping under gripper pose uncertainty. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4461–4468.

Kahn G, Sujan P, Patil S, et al. (2015) Active exploration using trajectory optimization for robotic grasping in the presence of occlusions. In: *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4783–4790.

Kalashnikov D, Irpan A, Pastor P, et al. (2018) QT-OPT: Scalable deep reinforcement learning for vision-based robotic manipulation. In: *Conference on Robot Learning (CoRL)*.

Kappler D, Meier F, Issac J, et al. (2018) Real-time perception meets reactive motion generation. *IEEE Robotics and Automation Letters* 3(3): 1864–1871.

Kober J, Glisson M and Mistry M (2012) Playing catch and juggling with a humanoid robot. In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 875–881.

Kragic D and Christensen HI (2002) *Survey on Visual Servoing for Manipulation*. Technical report, Computational Vision and Active Perception Laboratory, KTH.

Kumra S and Kanan C (2017) Robotic grasp detection using deep convolutional neural networks. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 769–776.

Leitner J, Frank M, Förster A and Schmidhuber J (2014) Reactive reaching and grasping on a humanoid: Towards closing the action-perception loop on the icub. In: *Proceedings of the International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, pp. 102–109.

Leitner J, Tow AW, Dean JE, et al. (2017) The ACRV Picking Benchmark: A robotic shelf picking benchmark to foster reproducible research. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4705–4712.

Lenz I, Lee H and Saxena A (2015) Deep learning for detecting robotic grasps. *The International Journal of Robotics Research* 34(4–5): 705–724.

Levine S, Pastor P, Krizhevsky A and Quillen D (2016) Learning hand–eye coordination for robotic grasping with large-scale data collection. In: *Proceedings of the International Symposium on Experimental Robotics (ISER)*, pp. 173–184.

Long J, Shelhamer E and Darrell T (2015) Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440.

Mahler J and Goldberg K (2017) Learning deep policies for robot bin picking by simulating robust grasping sequences. In: *Proceedings of the Conference on Robot Learning (CoRL)*, pp. 515–524.

Mahler J, Liang J, Niyaz S, et al. (2017) Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. In: *Proceedings of Robotics: Science and Systems (RSS)*.

Mahler J, Pokorny FT, Hou B, et al. (2016) 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: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1957–1964.

Mardia KV (2015) *Statistics of Directional Data*. New York: Academic Press.

Miller AT, Knoop S, Christensen HI and Allen PK (2003) Automatic grasp planning using shape primitives. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1824–1829.

Morrison D, Corke P and Leitner J (2018a) Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach. In: *Proceedings of Robotics: Science and Systems (RSS)*.

Morrison D, Tow AW, McTaggart M, et al. (2018b) CartMan: The low-cost Cartesian Manipulator that won the Amazon Robotics Challenge. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7757–7764.

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

Prattichizzo D and Trinkle JC (2008) Grasping. In: *Springer Handbook of Robotics*. Berlin: Springer, pp. 671–700.

Redmon J and Angelova A (2015) Real-time grasp detection using convolutional neural networks. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1316–1322.

Rubert C, Kappler D, Morales A, Schaal S and Bohg J (2017) On the relevance of grasp metrics for predicting grasp success. In: *Proceedings of the IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, pp. 265–272.

Sahbani A, El-Khoury S and Bidaud P (2012) An overview of 3D object grasp synthesis algorithms. *Robotics and Autonomous Systems* 60(3): 326–336.

Satish V, Mahler J and Goldberg K (2019) On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks. *IEEE Robotics and Automation Letters* 4(2): 1357–1364.

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

Shimoga KB (1996) Robot grasp synthesis algorithms: A survey. *The International Journal of Robotics Research* 15(3): 230–266.

ten Pas A, Gualtieri M, Saenko K and Platt R (2017) Grasp pose detection in point clouds. *The International Journal of Robotics Research* 36(13–14): 1455–1473.

Vahrenkamp N, Wieland S, Azad P, Gonzalez D, Asfour T and Dillmann R (2008) Visual servoing for humanoid grasping and manipulation tasks. In: *Proceedings of the International Conference on Humanoid Robots (Humanoids)*, pp. 406–412.

Viereck U, Pas A, Saenko K and Platt R (2017) Learning a visuomotor controller for real world robotic grasping using simulated depth images. In: *Proceedings of the Conference on Robot Learning (CoRL)*, pp. 291–300.

Wang Z, Li Z, Wang B and Liu H (2016) Robot grasp detection using multimodal deep convolutional neural networks. *Advances in Mechanical Engineering* 8: 9.

Yang J, Price B, Cohen S, Lee H and Yang MH (2016) Object contour detection with a fully convolutional encoder–decoder network. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 193–202.

Yu F and Koltun V (2016) Multi-scale context aggregation by dilated convolutions. In: *International Conference on Learning Representations (ICLR)*.

Yun Jiang, Moseson S and Saxena A (2011) Efficient grasping from RGBD images: Learning using a new rectangle representation. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3304–3311.

Zeng A, Song S, Welker S, Lee J, Rodriguez A and Funkhouser T (2018a) Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Zeng A, Song S, Yu KT, et al. (2018b) Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1–8.