Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping

Konstantinos Bousmalis*,[1], Alex Irpan*,[1], Paul Wohlhart*,[2], Yunfei Bai[2], Matthew Kelcey[1], Mrinal Kalakrishnan[2], Laura Downs[1], Julian Ibarz[1], Peter Pastor[2], Kurt Konolige[2], Sergey Levine[1], Vincent Vanhoucke[1]

***Abstract*— Instrumenting and collecting annotated visual grasping datasets to train modern machine learning algorithms can be extremely time-consuming and expensive. An appealing alternative is to use off-the-shelf simulators to render synthetic data for which ground-truth annotations are generated automatically. Unfortunately, models trained purely on simulated data often fail to generalize to the real world. We study how randomized simulated environments and domain adaptation methods can be extended to train a grasping system to grasp novel objects from raw monocular RGB images. We extensively evaluate our approaches with a total of more than 25,000 physical test grasps, studying a range of simulation conditions and domain adaptation methods, including a novel extension of pixel-level domain adaptation that we term the GraspGAN. We show that, by using synthetic data and domain adaptation, we are able to reduce the number of real-world samples needed to achieve a given level of performance by up to 50 times, using only randomly generated simulated objects. We also show that by using only unlabeled real-world data and our GraspGAN methodology, we obtain real-world grasping performance without any real-world labels that is similar to that achieved with 939,777 labeled real-world samples.**

I. Introduction

Grasping is one of the most fundamental robotic manipulation problems. For virtually any prehensile manipulation behavior, the first step is to grasp the object(s) in question. Grasping has therefore emerged as one of the central areas of study in robotics, with a range of methods and techniques from the earliest years of robotics research to the present day. A central challenge in robotic manipulation is generalization: can a grasping system successfully pick up diverse new objects that were not seen during the design or training of the system? Analytic or model-based grasping methods [1] can achieve excellent generalization to situations that satisfy their assumptions. However, the complexity and unpredictability of unstructured real-world scenes has a tendency to confound these assumptions, and learning-based methods have emerged as a powerful complement [2], [3], [4], [5], [6].

Learning a robotic grasping system has the benefit of generalization to objects with real-world statistics, and can benefit from the advances in computer vision and deep learning. Indeed, many of the grasping systems that have shown the best generalization in recent years incorporate convolutional neural networks into the grasp selection process [2], [5], [4], [7]. However, learning-based approaches also introduce a major challenge: the need for large labeled datasets. These labels might consist of human-provided grasp points [8], or they might be collected autonomously [5], [6]. In both cases, there is considerable cost in both time and money, and recent studies suggest that the performance of grasping systems might be strongly influenced by the amount of data available [6].

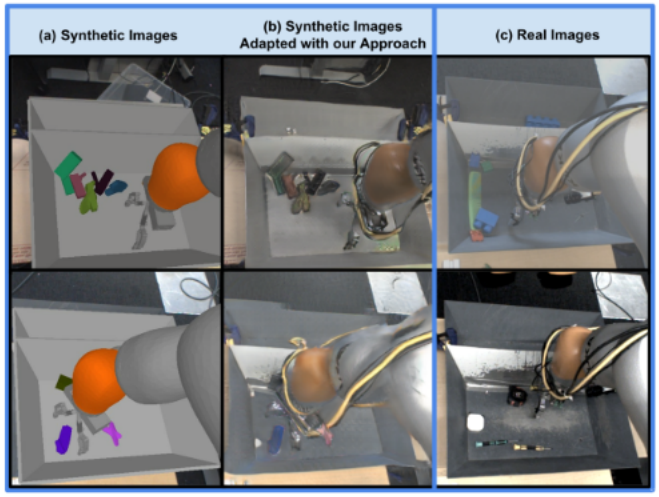


Fig. 1: **Bridging the reality gap:** our proposed pixel-level domain adaptation model takes as input *(a)* synthetic images produced by our simulator and produces *(b)* adapted images that look similar to *(c)* real-world ones produced by the camera over the physical robot's shoulder. We then train a deep vision-based grasping network with adapted and real images, which we further refine with feature-level adaptation.

A natural avenue to overcome these data requirements is to look back at the success of analytic, model-based grasping methods [1], which incorporate our prior knowledge of physics and geometry. We can incorporate this prior knowledge into a learning-based grasping system in two ways. First, we could modify the design of the system to use a model-based grasping method, for example as a scoring function for learning-based grasping [7]. Second, we could use our prior knowledge to construct a simulator, and generate synthetic experience that can be used in much the same way as real experience. The second avenue, which we explore in this work, is particularly appealing because we can use essentially the same learning system. However, incorporating simulated images presents challenges: simulated data differs in systematic ways from real-world data, and simulation must have sufficiently general objects. Addressing these two challenges is the principal subject of our work.

Our work has three main contributions. *(a) Substantial*

*Authors contributed equally, [1]Google Brain, [2]X

improvement in grasping performance from monocular RGB images by incorporating synthetic data: We propose approaches for incorporating synthetic data into end-to-end training of vision-based robotic grasping that we show achieves substantial improvement in performance, particularly in the lower-data and no-data regimes. *(b) Detailed experimentation for simulation-to-real world transfer:* Our experiments involved 25,704 real grasps of 36 diverse test objects and consider a number of dimensions: the nature of the simulated objects, the kind of randomization used in simulation, and the domain adaptation technique used to adapt simulated images to the real world. *(c) The first demonstration of effective simulation-to-real-world transfer for purely monocular vision-based grasping:* To our knowledge, our work is the first to demonstrate successful simulation-to-real-world transfer for grasping, with generalization to previously unseen natural objects, using only monocular RGB images.

II. Related Work

Robotic grasping is one of the most widely explored areas of manipulation. While a complete survey of grasping is outside the scope of this work, we refer the reader to standard surveys on the subject for a more complete treatment [2]. Grasping methods can be broadly categorized into two groups: geometric methods and data-driven methods. Geometric methods employ analytic grasp metrics, such as force closure [9] or caging [10]. These methods often include appealing guarantees on performance, but typically at the expense of relatively restrictive assumptions. Practical applications of such approaches typically violate one or more of their assumptions. For this reason, data-driven grasping algorithms have risen in popularity in recent years. Instead of relying exclusively on an analytic understanding of the physics of an object, data-driven methods seek to directly predict either human-specified grasp positions [8] or empirically estimated grasp outcomes [5], [6]. A number of methods combine both ideas, for example using analytic metrics to label training data [3], [7].

Simulation-to-real-world transfer in robotics is an important goal, as simulation can be a source of practically infinite cheap data with flawless annotations. For this reason, a number of recent works have considered simulation-to-real world transfer in the context of robotic manipulation. Saxena *et al.* [11] used rendered objects to learn a vision-based grasping model. Gulatieri *et al.* and Viereck *et al.* [4], [12] have considered simulation-to-real world transfer using depth images. Depth images can abstract away many of the challenging appearance properties of real-world objects. However, not all situations are suitable for depth cameras, and coupled with the low cost of simple RGB cameras, there is considerable value in studying grasping systems that solely use monocular RGB images.

A number of recent works have also examined using randomized simulated environments [13], [14] for simulation-to-real world transfer for grasping and grasping-like manipulation tasks, extending on prior work on randomization for robotic mobility [15]. These works apply randomization in the form of random textures, lighting, and camera position to their simulator. However, unlike our work, these prior methods considered grasping in relatively simple visual environments, consisting of cubes or other basic geometric shapes, and have not yet been demonstrated on grasping diverse, novel real-world objects of the kind considered in our evaluation.

Domain adaptation is a process that allows a machine learning model trained with samples from a source domain to generalize to a target domain. In our case the source domain is the simulation, whereas the target is the real world. There has recently been a significant amount of work on domain adaptation, particularly for computer vision [16], [17]. Prior work can be grouped into two main types: feature-level and pixel-level adaptation. *Feature-level domain adaptation* focuses on learning domain-invariant features, either by learning a transformation of fixed, pre-computed features between source and target domains [18], [19], [20], [21] or by learning a domain-invariant feature extractor, often represented by a convolutional neural network (CNN) [22], [23], [24]. Prior work has shown the latter is empirically preferable on a number of classification tasks [22], [24]. Domain-invariance can be enforced by optimizing domain-level similarity metrics like maximum mean discrepancy [24], or the response of an adversarially trained domain discriminator [22]. *Pixel-level domain adaptation* focuses on re-stylizing images from the source domain to make them look like images from the target domain [25], [26], [27], [28]. To our knowledge, all such methods are based on image-conditioned generative adversarial networks (GANs) [29]. In this work, we compare a number of different domain adaptation regimes. We also present a new method that combines both feature-level and pixel-level domain adaptation for simulation-to-real world transfer for vision-based grasping.

III. Background

Our goal in this work is to show the effect of using simulation and domain adaptation in conjunction with a tested data-driven, monocular vision-based grasping approach. To this effect, we use such an approach, as recently proposed by Levine *et al.* [6]. In this section we will concisely discuss this approach, and the two main domain adaptation techniques [22], [26], [27] our method is based on.

A. Deep Vision-Based Robotic Grasping

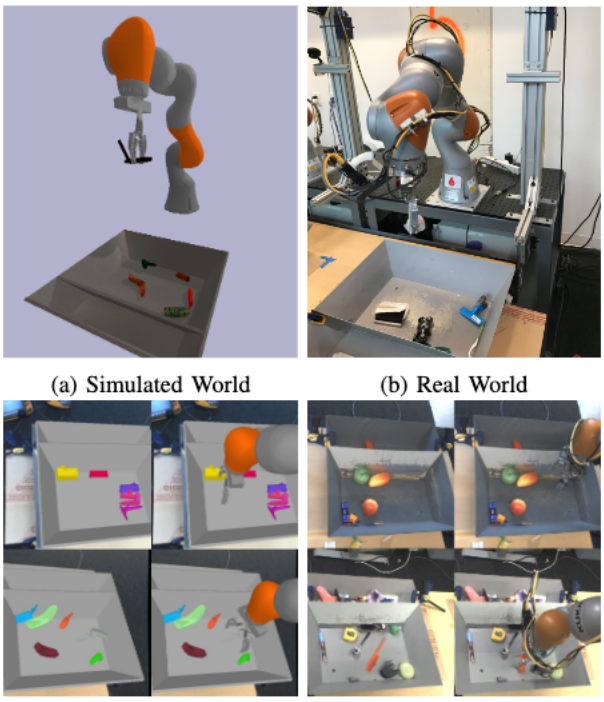
The grasping approach [6] we use in this work consists of two components. The first is **a grasp prediction convolutional neural network** (CNN) $C(\mathbf{x}_i, \mathbf{v}_i)$ that accepts a tuple of visual inputs $\mathbf{x}_i = \{\mathbf{x}_{i_0}, \mathbf{x}_{i_c}\}$ and a motion command $\mathbf{v}_i$, and outputs the predicted probability that executing $\mathbf{v}_i$ will result in a successful grasp. $\mathbf{x}_{i_0}$ is an image recorded before the robot becomes visible and starts the grasp attempt, and $\mathbf{x}_{i_c}$ is an image recorded at the current timestep. $\mathbf{v}_i$ is specified in the frame of the base of the robot and corresponds to a relative change of the end-effector's current position and rotation about the vertical axis. We consider only top-down

pinch grasps, and the motion command has, thus, 5 dimensions: 3 for position, and 2 for a sine-cosine encoding of the rotation. The second component of the method is **a simple, manually designed servoing function** that uses the grasp probabilities predicted by C to choose the motor command $\mathbf{v}_i$ that will continuously control the robot. We can train the grasp prediction network C using standard supervised learning objectives, and so it can be optimized independently from the servoing mechanism. In this work, we focus on extending the first component to include simulated data in the training set for the grasp prediction network C, leaving the other parts of the system unchanged.

The datasets for training the grasp prediction CNN C are collections of visual episodes of robotic arms attempting to grasp various objects. Each grasp attempt episode consists of T time steps which result in T distinct training samples. Each sample i includes $\mathbf{x}_i, \mathbf{v}_i$, and the success label y_i of the entire grasp sequence. The visual inputs are 640×512 images that are randomly cropped to a 472×472 region during training to encourage translation invariance.

The central aim of our work is to compare different training regimes that combine both simulated and real-world data for training C. Although we do consider training entirely with simulated data, as we discuss in Section IV-A, most of the training regimes we consider combine medium amounts of real-world data with large amounts of simulated data. To that end, we use the self-supervised real-world grasping dataset collected by Levine *et al.* [6] using 6 physical Kuka IIWA arms. The goal of the robots was to grasp any object within a specified goal region. Grasping was performed using a compliant two-finger gripper picking objects out of a metal bin, with a monocular RGB camera mounted behind the arm. The full dataset includes about 1 million grasp attempts on approximately $1,100$ different objects, resulting in about 9.4 million real-world images. About half of the dataset was collected using random grasps, and the rest using iteratively retrained versions of C. Aside from the variety of objects, each robot differed slightly in terms of wear-and-tear, as well as the camera pose. The outcome of the grasp attempt was determined automatically. The particular objects in front of each robot were regularly rotated to increase the diversity of the dataset. Some examples of grasping images from the camera's viewpoint are shown in Figure 2d.

When trained on the entire real dataset, the best CNN used in the approach outlined above achieved successful grasps 67.65% of the time. Levine *et al.* [6] reported an additional increase to 77.18% from also including 2.7 million images from a different robot. We excluded this additional dataset for the sake of a more controlled comparison, so as to avoid additional confounding factors due to domain shift within the real-world data. Starting from the Kuka dataset, our experiments study the effect of adding simulated data and of reducing the number of real world data points by taking subsets of varying size (down to only $93,841$ real world images, which is 1% of the original set).



(a) Simulated World (b) Real World

(c) Simulated Samples (d) Real Samples

Fig. 2: **Top Row:** The setup we used for collecting the *(a)* simulated and *(b)* real-world datasets. **Bottom Row:** Images used during training of *(c)* simulated grasping experience with procedurally generated objects; and of *(d)* real-world experience with a varied collection of everyday physical objects. In both cases, we see the pairs of image inputs for our grasp success prediction model C: the images at $t = 0$ and the images at the current timestamp.

B. Domain Adaptation

As part of our proposed approach we use two domain adaptation techniques: domain-adversarial training and pixel-level domain adaptation. Ganin *et al.* [22] introduced domain–adversarial neural networks (DANNs), an architecture trained to extract domain-invariant yet expressive features. DANNs were primarily tested in the unsupervised domain adaptation scenario, in the absence of any labeled target domain samples, although they also showed promising results in the semi-supervised regime [24]. Their model's first few layers are shared by two modules: the first predicts task-specific labels when provided with source data while the second is a separate *domain classifier* trained to predict the domain $\hat{d}$ of its inputs. The DANN loss is the cross-entropy loss for the domain prediction task: $\mathscr{L}_{\text{DANN}} = \sum_{i=0}^{N_s+N_t} \left\{ d_i \log \hat{d}_i + (1-d_i) \log(1-\hat{d}_i) \right\}$, where $d_i \in \{0,1\}$ is the ground truth domain label for sample i, and N_s, N_t are the number of source and target samples.

The shared layers are trained to maximize $\mathscr{L}_{\text{DANN}}$, while the domain classifier is trained adversarially to minimize it. This minimax optimization is implemented by a gradient reversal layer (GRL). The GRL has the same output as the identity function, but negates the gradient during backprop. This lets us compute the gradient for both the domain clas-

sifier and the shared feature extractor in a single backward pass. The task loss of interest is simultaneously optimized with respect to the shared layers, which grounds the shared features to be relevant to the task.

While DANN makes the features extracted from both domains similar, the goal in *pixel-level* domain adaptation [26], [27], [28], [25] is to learn a generator function G that maps images from a source to a target domain at the input level. This approach decouples the process of domain adaptation from the process of task-specific predictions, by adapting the images from the source domain to make them appear as if they were sampled from the target domain. Once the images are adapted, they can replace the source dataset and the relevant task model can be trained as if no domain adaptation were required. Although all these methods are similar in spirit, we use ideas primarily from PixelDA [26] and SimGAN [27], as they are more suitable for our task. These models are particularly effective if the goal is to maintain the semantic map of original and adapted synthetic images, as the transformations are primarily low-level: the methods make the assumption that the differences between the domains are primarily low-level (due to noise, resolution, illumination, color) rather than high-level (types of objects, geometric variations, etc).

More formally, let $\mathbf{X}^s = \{\mathbf{x}_i^s, \mathbf{y}_i^s\}_{i=0}^{N^s}$ represent a dataset of N^s samples from the source domain and let $\mathbf{X}^t = \{\mathbf{x}_i^t, \mathbf{y}_i^t\}_{i=0}^{N^t}$ represent a dataset of N^t samples from the target domain. The generator function $G(\mathbf{x}^s; \theta_G) \rightarrow \mathbf{x}^f$, parameterized by θ_G, maps a source image $\mathbf{x}^s \in \mathbf{X}^s$ to an adapted, or fake, image $\mathbf{x}^f$. This function is learned with the help of an adversary, a discriminator function $D(\mathbf{x}; \theta_D)$ that outputs the likelihood d that a given image $\mathbf{x}$ is a real-world sample. Both G and D are trained using the standard adversarial objective [29]. Given the learned generator function G, it is possible to create a new dataset $\mathbf{X}^f = \{G(\mathbf{x}^s), \mathbf{y}^s\}$. Finally, given an adapted dataset $\mathbf{X}^f$, the task-specific model can be trained as if the training and test data were from the same distribution.

PixelDA was evaluated in simulation-to-real-world transfer. However, the 3D models used by the renderer in [26] were very high-fidelity scans of the objects in the real-world dataset. In this work we examine for the first time how such a technique can be applied in situations where *(a)* no 3D models for the objects in the real-world are available and *(b)* the system is supposed to generalize to yet another set of previously unseen objects in the actual real-world grasping task. Furthermore, we use images of 472×472, more than double the resolution in [26], [27]. This makes learning the generative model G a much harder task and requires significant changes compared to previous work: the architecture of both G and D, the GAN training objective, and the losses that aid with training the generator (content-similarity and task losses) are different from the original implementations, resulting in a novel model evaluated under these new conditions.

IV. Our Approach

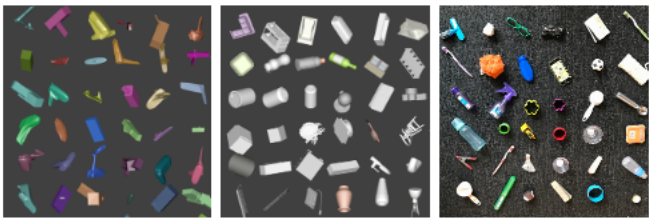
One of the aims of our work is to study how final grasping performance is affected by the 3D object models our simulated experience is based on, the scene appearance and dynamics in simulation, and the way simulated and real experience is integrated for maximal transfer. In this section we outline, for each of these three factors, our proposals for effective simulation-to-real-world transfer for our task.

A. Grasping in Simulation

A major difficulty in constructing simulators for robotic learning is to ensure diversity sufficient for effective generalization to real-world settings. In order to evaluate simulation-to-real world transfer, we used one dataset of real-world grasp attempts (see Sect. III-A), and multiple such datasets in simulation. For the latter, we built a basic virtual environment based on the Bullet physics simulator and the simple renderer that is shipped with it [30]. The environment emulates the Kuka hardware setup by simulating the physics of grasping and by rendering what a camera mounted looking over the Kuka shoulder would perceive: the arm, the bin that contains the object, and the objects to grasp in scenes similar to the ones the robot encounters in the real world.

A central question here is regarding the realism of the 3D models used for the objects to grasp. To answer it, we evaluate two different sources of objects in our experiments: *(a)* procedurally generated random geometric shapes and *(b)* realistic objects obtained from the publicly-available ShapeNet [31] 3D model repository. We procedurally generated $1,000$ objects by attaching rectangular prisms at random locations and orientations, as seen in Fig. 3a. We then converted the set of prisms to a mesh using an off-the-shelf renderer, Blender, and applied a random level of smoothing. Each object was given UV texture coordinates and random colors. For our Shapenet-based datasets, we used the ShapeNetCore.v2 [31] collection of realistic object models, shown in Figure 3b. This particular collection contains $51,300$ models in 55 categories of household objects, furniture, and vehicles. We rescaled each object to a random graspable size with a maximum extent between 12cm and 23cm (real-world objects ranged from 4cm to 20cm in length along the longest axis) and gave it a random mass between 10g and 500g, based on the approximate volume of the object.

Once the models were imported into our simulator, we collected our simulation datasets via a similar process to the one in the real world, with a few differences. As mentioned above, the real-world dataset was collected by using progressively better grasp prediction networks. These networks were swapped for better versions manually and rather infrequently [6]. In contrast to the 6 physical Kuka IIWA robots that were used to collect data in the real world, we used 1,000 to 2,000 simulated arms at any given time to collect our synthetic data, and the models that were used to collect the datasets were being updated continuously by an automated process. This resulted in datasets that were



(a) Procedural (b) ShapeNet [31] (c) Real

Fig. 3: Comparison of *(a)* some of our 1,000 procedural, *(b)* some of the 51,300 ShapeNet objects, both used for data collection in simulation, and the *(c)* 36 objects we used only for evaluating grasping in the real-world, that were not seen during training. The variety of shapes, sizes, and material properties makes the test set very challenging.

collected by grasp prediction networks of varying performance, which added diversity to the collected samples. After training our grasping approach in our simulated environment, the simulated robots were successful on 70%-90% of the simulated grasp attempts. Note that all of the grasp success prediction models used in our experiments were trained from scratch using these simulated grasp datasets.

B. Virtual Scene Randomization

Another important question is whether randomizing the visual appearance and dynamics in the scene affects grasp performance and in what way. One of the first kind of diversities we considered was the addition of ε cm, where $\varepsilon \sim \mathcal{N}(0,1)$, to the horizontal components of the motor command. This improved real grasp success in early experiments, so we added this kind of randomization for all simulated samples. Adding this noise to real data did not help. To further study the effects of virtual scene randomization, we built datasets with four different kinds of scene randomization: *(a) No randomization*: Similar to real-world data collection, we only varied camera pose, bin location, and used 6 different real-world images as backgrounds; *(b) Visual Randomization*: We varied tray texture, object texture and color, robot arm color, lighting direction and brightness; *(c) Dynamics Randomization*: We varied object mass, and object lateral/rolling/spinning friction coefficients; and *(d) All*: both visual and dynamics randomization.

C. Domain Adaptation for Vision-Based Grasping

As mentioned in Sect. II, there are two primary types of methods used for domain adaptation: feature-level, and pixel-level. Here we propose a feature-level adaptation method and a novel pixel-level one, which we call GraspGAN. Given original synthetic images, GraspGAN produces adapted images that look more realistic. We subsequently use the trained generator from GraspGAN as a fixed module that adapts our synthetic visual input, while performing feature-level domain adaptation on extracted features that account for both the transferred images and synthetic motor command input.

For our feature-level adaptation technique we use a DANN loss on the last convolutional layer of our grasp success prediction model C, as shown in Fig. 4c. In preliminary experiments we found that using the DANN loss on this layer yielded superior performance compared to applying it at the activations of other layers. We used the domain classifier proposed in [22]. One of the early research questions we faced was what the interaction of batch normalization (BN) [32] with the DANN loss would be, as this has not been examined in previous work. We use BN in every layer of C and in a naïve implementation of training models with data from two domains, a setting we call **naïve mixing**, batch statistics are calculated without taking the domain labels of each sample into account. However, the two domains are bound to have different statistics, which means that calculating and using them separately for simulated and real-world data while using the same parameters for C might be beneficial. We call this way of training data from two domains **domain-specific batch normalization (DBN) mixing**, and show it is a useful tool for domain adaptation, even when a DANN loss is not used.

In **our pixel-level domain adaptation model, GraspGAN**, shown in Fig. 4, G is a convolutional neural network that follows a U-Net architecture [33], and uses average pooling for downsampling, bilinear upsampling, concatenation and 1×1 convolutions for the U-Net skip connections, and instance normalization [34]. Our discriminator D is a patch-based [35] CNN with 5 convolutional layers, with an effective input size of 70×70. It is fully convolutional on 3 scales (472×472, 236×236, and 118×118) of the two input images, $\mathbf{x}_0^s$ and $\mathbf{x}_c^s$, stacked into a 6 channel input, producing domain estimates for all patches which are then combined to compute the joint discriminator loss. This **novel multi-scale patch-based discriminator** design can learn to assess both global consistency of the generated image, as well as realism of local textures. Stacking the channels of the two input images enables the discriminator to recognize relationships between the two images, so it can encourage the generator to respect them (*e.g.*, paint the tray with the same texture in both images, but insert realistic shadows for the arm). Our task model C is the grasp success prediction CNN from [6].

To train GraspGAN, we employ a least-squares generative adversarial objective (LSGAN) [36] to encourage G to produce realistic images. During training, our generator $G(\mathbf{x}^s;\theta_G) \rightarrow \mathbf{x}^f$ maps synthetic images $\mathbf{x}^s$ to adapted images $\mathbf{x}^f$, by individually passing $\mathbf{x}_0^s$ and $\mathbf{x}_c^s$ through two instances of the generator network displayed in Figure 4. Similar to traditional GAN training, we perform optimization in alternating steps by minimizing the following loss terms w.r.t. the parameters of each sub-network:

$$\min_{\theta_G} \lambda_g \mathscr{L}_{gen}(G,D) + \lambda_{tg} \mathscr{L}_{task}(G,C) + \lambda_c \mathscr{L}_{content}(G) \quad (1)$$

$$\min_{\theta_D,\theta_C} \lambda_d \mathscr{L}_{discr}(G,D) + \lambda_{td} \mathscr{L}_{task}(G,C), \quad (2)$$

where $\mathscr{L}_{gen}$ and $\mathscr{L}_{discr}$ are the LSGAN generator and discriminator losses, $\mathscr{L}_{task}$ is the task loss, $\mathscr{L}_{content}$ is the content-similarity loss, and λ_g, λ_d, λ_{tg}, λ_{td}, λ_c, the respective weights. The LSGAN discriminator loss is the $L2$ distance between its likelihood output $\hat{d}$ and the domain labels $d = 0$

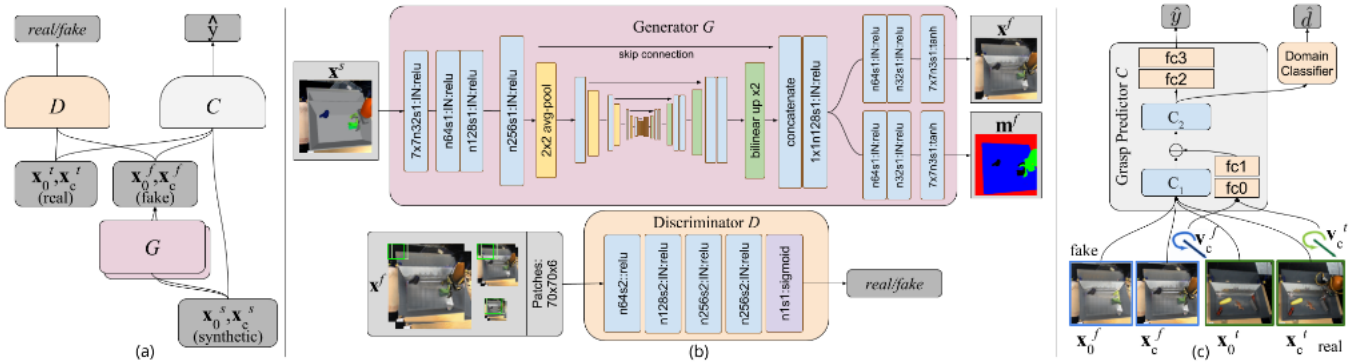


Fig. 4: **Our proposed approach: (a)** Overview of our pixel-level domain adaptation model, GraspGAN. Tuples of images from the simulation $\mathbf{x}^s$ are fed into the generator G to produce realistic versions $\mathbf{x}^f$. The discriminator D gets unlabeled real world images $\mathbf{x}^t$ and $\mathbf{x}^f$ and is trained to distinguish them. Real and adapted images are also fed into the grasp success prediction network C, trained in parallel (motion commands $\mathbf{v}$ are not shown to reduce clutter). G, thus, gets feedback from D and C to make adapted images look real and maintain the semantic information. **(b)** Architectures for G and D. Blue boxes denote convolution/normalization/activation-layers, where *n64s2:IN:relu* means 64 filters, stride 2, instance normalization *IN* and *relu* activation. Unless specified all convolutions are 3×3 in G and 4×4 in D. **(c)** DANN model: C_1 has 7 conv layers and C_2 has 9 conv layers. Further details can be found in [6]. Domain classifier uses GRL and two 100 unit layers.

for fake and $d = 1$ for real images, while for the generator loss the label is flipped, such that there is a high loss if the disciminator predicts $\hat{d} = 0$ for a generated image. The task loss measures how well the network C predicts grasp success on transferred and real examples by calculating the binomial cross-entropy of the labels y_i.

It is of utmost importance that the GraspGAN generator, while making the input image look like an image from the real world scenario, does not change the semantics of the simulated input, for instance by drawing the robot's arm or the objects in different positions. Otherwise, the information we extract from the simulation in order to train the task network would not correspond anymore to the generated image. We thus devise several additional loss terms, accumulated in $\mathscr{L}_{content}$, to help anchor the generated image to the simulated one on a semantic level. The most straightforward restriction is to not allow the generated image to deviate much from the input. To that effect we use the PMSE loss, also used by [26]. We also leverage the fact that we can have semantic information about every pixel in the synthetic images by computing segmentation masks $\mathbf{m}^f$ of the corresponding rendered images for the background, the tray, robot arm, and the objects. We use these masks by training our generator G to also produce $\mathbf{m}^f$ as an additional output for each adapted image, with a standard $L2$ reconstruction loss. Intuitively, it forces the generator to extract semantic information about all the objects in the scene and encode them in the intermediate latent representations. This information is then available during the generation of the output image as well. Finally, we additionally implement a loss term that provides more dense feedback from the task tower than just the single bit of information about grasp success. We encourage the generated image to provide the same semantic information to the task network as the corresponding simulated one by penalizing differences in activations of the final convolutional layer of C for the two images. This is similar in principle to the perceptual loss [37] that uses the activations of an ImageNet-pretrained VGG model as a way to anchor the restylization of an input image. In contrast, here C is trained at the same time, the loss is specific to our goal, and it helps preserve the semantics in ways that are relevant to our prediction task.

V. Evaluation

This section aims to answer the following research questions: *(a)* is the use of simulated data from a low quality simulator aiding in improving grasping performance in the real world? *(b)* is the improvement consistent with varying amounts of real-world labeled samples? *(c)* how realistic do graspable objects in simulation need to be? *(d)* does randomizing the virtual environment affect simulation-to-real world transfer, and what are the randomization attributes that help most? *(e)* does domain adaptation allow for better utilization of simulated grasping experience?

In order to answer these questions, we evaluated a number of different ways for training a grasp success prediction model C with simulated data and domain adaptation[1]. When simulated data was used, the number of simulated samples was always approximately 8 million. We follow the grasp success evaluation protocol described by Levine *et al.* [6]. We used 6 Kuka IIWA robots for our real-world experiments

[1]Visit `https://goo.gl/G1HSws` for our supplementary video.

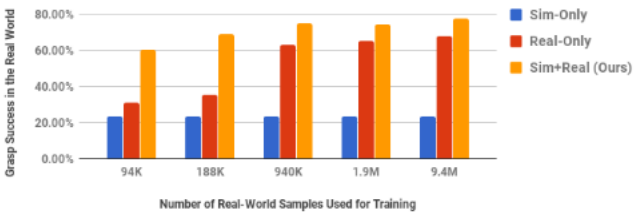


Fig. 5: The effect of using 8 million simulated samples of procedural objects with no randomization and various amounts of real data, for the best technique in each class.

TABLE I: **The effect of our choices for simulated objects and randomization in terms of grasp success.** We compared the performance of models trained jointly on grasps of procedural vs ShapeNet objects with 10% of the real data. Models were trained with DANN and DBN mixing.

Randomization	None	Visual	Dynamics	Both
Procedural	**71.93%**	**74.88%**	**73.95%**	**72.86%**
ShapeNet	69.61%	68.79%	68.62%	69.84%

and a test set consisting of the objects shown in Fig. 3c, the same used in [6], with 6 different objects in each bin for each robot. These objects were not included in the real-world training set and were not used in any way when creating our simulation datasets. Each robot executes 102 grasps, for a total of 612 test grasps for each evaluation. During execution, each robot picks up objects from one side of the bin and drops them on the other, alternating every 3 grasps. This prevents the model from repeatedly grasping the same object. Optimal models were selected by using the accuracy of C on a held-out validation set of $94,000$ real samples.

The first conclusion from our results is that simulated data from an off-the-shelf simulator always aids in improving vision-based real-world grasping performance. As one can see in Fig. 5, which shows the real grasp success gains by incorporating simulated data from our procedurally-generated objects, using our simulated data significantly and consistently improves real-world performance regardless of the number of real-world samples.

We also observed that we do not need realistic 3D models to obtain these gains. We compared the effect of using random, procedurally-generated shapes and ShapeNet objects in combination with 10% of the real-world data, under all randomization scenarios. As shown in Table I we found that using procedural objects is the better choice in all cases. This finding has interesting implications for simulation to real-world transfer, since content creation is often a major bottleneck in producing generalizable simulated data. Based on these results, we decided to use solely procedural objects for the rest of our experiments.

Table III shows our main results: the grasp success performance for different combinations of simulated data generation and domain adaptation methods, and with different quantities of real-world samples. The different settings are: ***Real-Only***, in which the model is given only real data; *Naïve Mixing (**Naïve Mix**)*: Simulated samples generated with no virtual scene randomization are mixed with real-world samples such that half of each batch consists of simulated images; *DBN Mixing & Randomization (**Rand.**)*: The simulated dataset is generated with visual-only randomization. The simulated samples are mixed with real-world samples as in the naive mixing case, and the models use DBN; *DBN Mixing & DANN (**DANN**)*: Simulated samples are generated with no virtual scene randomization and the model is trained with a domain-adversarial method with DBN; *DBN Mixing, DANN & Randomization (**DANN-R**)*: Simulated samples are generated with visual randomization and the model is trained with a domain-adversarial method with DBN; *GraspGAN, DBN Mixing & DANN (**GraspGAN**)*: The non-randomized simulated data is first refined with a GraspGAN generator, and the refined data is used to train a DANN with DBN mixing. The generator is trained with the same real dataset size used to train the DANN. See Figure 1b for examples.

TABLE II: Real grasp performance when no labeled real examples are available. Method names explained in the text.

Sim-Only	**Rand.**	**GraspGAN**
23.53%	35.95%	**63.40%**

TABLE III: Success of grasping 36 diverse and unseen physical objects of all our methods trained on different amounts of real-world samples and 8 million simulated samples with procedural objects. Method names are explained in the text.

Method	**All 9,402,875**	**20% 1,880,363**	**10% 939,777**	**2% 188,094**	**1% 93,841**
Real-Only	67.65%	64.93%	62.75%	35.46%	31.13%
Naïve Mix.	73.63%	69.61%	65.20%	58.38%	39.86%
Rand.	75.58%	70.16%	73.31%	63.61%	50.99%
DANN	76.26%	68.12%	71.93%	61.93%	59.27%
DANN-R.	72.60%	66.46%	**74.88%**	63.73%	43.81%
GraspGAN	**76.67%**	**74.07%**	70.70%	**68.51%**	**59.95%**

Table III shows that using visual randomization with DBN mixing improved upon naïve mixing with no randomization experiments across the board. The effect of visual, dynamics, and combined randomization for both procedural and ShapeNet objects was evaluated by using 10% of the real data available. Table I shows that using only visual randomization slightly improved grasp performance for procedural objects, but the differences were generally not conclusive.

In terms of domain adaptation techniques, our proposed hybrid approach of combining our GraspGAN and DANN performs the best in most cases, and shows the most gains in the lower real-data regimes. Using DANNs with DBN Mixing performed better than naïve mixing in most cases. However the effect of DANNs on Randomized data was not conclusive, as the equivalent models produced worse results in 3 out of 5 cases. We believe the most interesting results however, are the ones from our experiments with no labeled real data. We compared the best domain adaptation method (GraspGAN), against a model trained on simulated data with and without randomization. We trained a GraspGAN on all 9 million real samples, without using their labels. Our grasping model was then trained only on data refined by G. Results in Table II show that the unsupervised adaptation model outperformed not only sim-only models with and without randomization but also a real-only model with 939,777 labeled real samples.

Although our absolute grasp success numbers are consistent with the ones reported in [6], some previous grasping work reports higher absolute grasp success. However, we note the following: *(a)* our goal in this work is not to show that we can train the best possible grasping system, but that for the same amount of real-world data, the inclusion of synthetic data can be helpful; we have relied on previous work [6] for the grasping approach used; *(b)* our evaluation was conducted on a diverse and challenging

range of objects, including transparent bottles, small round objects, deformable objects, and clutter; and *(c)* the method uses only monocular RGB images from an over-the-shoulder viewpoint, without depth or wrist-mounted cameras. These make our setup considerably harder than most standard ones.

VI. Conclusion

In this paper, we examined how simulated data can be incorporated into a learning-based grasping system to improve performance and reduce data requirements. We study grasping from over-the-shoulder monocular RGB images, a particularly challenging setting where depth information and analytic 3D models are not available. This presents a challenging setting for simulation-to-real-world transfer, since simulated RGB images typically differ much more from real ones compared to simulated depth images. We examine the effects of the nature of the objects in simulation, of randomization, and of domain adaptation. We also introduce a novel extension of pixel-level domain adaptation that makes it suitable for use with high-resolution images used in our grasping system. Our results indicate that including simulated data can drastically improve the vision-based grasping system we use, achieving comparable or better performance with 50 times fewer real-world samples. Our results also suggest that it is not as important to use realistic 3D models for simulated training. Finally, our experiments indicate that our method can provide plausible transformations of synthetic images, and that including domain adaptation substantially improves performance in most cases.

Although our work demonstrates very large improvements in the grasp success rate when training on smaller amounts of real world data, there are a number of limitations. Both of the adaptation methods we consider focus on invariance, either transforming simulated images to look like real images, or regularizing features to be invariant across domains. These features incorporate both appearance and action, due to the structure of our network, but no explicit reasoning about physical discrepancies between the simulation and the real world is done. We did consider randomization of dynamics properties, and show it is indeed important. Several recent works have looked at adapting to physical discrepancies explicitly [38], [39], [40], and incorporating these ideas into grasping is an exciting avenue for future work. Our approach for simulation to real world transfer only considers monocular RGB images, though extending this method to stereo and depth images would be straightforward. Finally, the success rate reported in our experiments still has room for improvement, and we expect further research in this area will lead to even better results. The key insight from our work comes from the comparison of the different methods: we are not aiming to propose a novel grasping system, but rather to study how incorporating simulated data can improve an existing one.

Acknowledgments

The authors thank John-Michael Burke for overseeing the robot operations. The authors also thank Erwin Coumans, Ethan Holly, Dmitry Kalashnikov, Deirdre Quillen, and Ian Wilkes for contributions to the development of our grasping system and supporting infrastructure.

References

[1] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[2] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis-a survey," *IEEE Transactions on Robotics*, 2014.

[3] D. Kappler, J. Bohg, and S. Schaal, "Leveraging Big Data for Grasp Planning," in *ICRA*, 2015.

[4] U. Viereck, A. t. Pas, K. Saenko, and R. Platt, "Learning a visuomotor controller for real world robotic grasping using easily simulated depth images," *arxiv:1706.04652*, 2017.

[5] P. L. and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *ICRA*, 2016.

[6] 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*, 2016.

[7] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics," in *RSS*, 2017.

[8] I. Lenz, H. Lee, and A. Saxena, "Deep Learning for Detecting Robotic Grasps," *IJRR*, 2015.

[9] A. Bicchi, "On the Closure Properties of Robotic Grasping," *IJRR*, 1995.

[10] A. Rodriguez, M. T. Mason, and S. Ferry, "From Caging to Grasping," *IJRR*, 2012.

[11] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic Grasping of Novel Objects using Vision," *IJRR*, 2008.

[12] M. Gualtieri, A. ten Pas, K. Saenko, and R. Platt, "High precision grasp pose detection in dense clutter," in *IROS*, 2016, pp. 598–605.

[13] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," *arxiv:1703.06907*, 2017.

[14] S. James, A. J. Davison, and E. Johns, "Transferring End-to-End Visuomotor Control from Simulation to Real World for a Multi-Stage Task," *arxiv:1707.02267*, 2017.

[15] F. Sadeghi and S. Levine, "CAD2RL: Real single-image flight without a single real image." *arxiv:1611.04201*, 2016.

[16] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa, "Visual domain adaptation: A survey of recent advances," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 53–69, 2015.

[17] G. Csurka, "Domain adaptation for visual applications: A comprehensive survey," *arxiv:1702.05374*, 2017.

[18] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *AAAI*, 2016.

[19] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *CVPR*, 2012.

[20] R. Caseiro, J. F. Henriques, P. Martins, and J. Batista, "Beyond the shortest path: Unsupervised Domain Adaptation by Sampling Subspaces Along the Spline Flow," in *CVPR*, 2015.

[21] R. Gopalan, R. Li, and R. Chellappa, "Domain Adaptation for Object Recognition: An Unsupervised Approach," in *ICCV*, 2011.

[22] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *JMLR*, 2016.

[23] M. Long and J. Wang, "Learning transferable features with deep adaptation networks," *ICML*, 2015.

[24] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, "Domain separation networks," in *NIPS*, 2016.

[25] Y. Taigman, A. Polyak, and L. Wolf, "Unsupervised cross-domain image generation," in *ICLR*, 2017.

[26] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial neural networks," in *CVPR*, 2017.

[27] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *CVPR*, 2017.

[28] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," in *ICCV*, 2017.

[29] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014.

[30] E. Coumans and Y. Bai, "pybullet, a python module for physics simulation in robotics, games and machine learning," http://pybullet.org/, 2016–2017.

[31] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," *CoRR*, 2015.

[32] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Network Training by Reducing Internal Covariate Shift," in *ICML*, 2015.

[33] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *MICCAI*, 2015.

[34] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arxiv:1607.08022*, 2016.

[35] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," *arxiv:1611.07004*, 2016.

[36] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, "Least Squares Generative Adversarial Networks," *arxiv:1611.04076*, 2016.

[37] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," in *ECCV*. Springer, 2016.

[38] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba, "Transfer from simulation to real world through learning deep inverse dynamics model," *arXiv:1610.03518*, 2016.

[39] A. Rajeswaran, S. Ghotra, S. Levine, and B. Ravindran, "Epopt: Learning robust neural network policies using model ensembles," *arxiv:1610.01283*, 2016.

[40] W. Yu, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," *arXiv:1702.02453*, 2017.