Mind the Gap! Bridging the Reality Gap in Visual Perception and Robotic Grasping with Domain Randomisation

João Borrego
Instituto Superior Técnico, Lisboa, Portugal
Email: jborrego@isr.tecnico.ulisboa.pt

Abstract—In this work, we focus on Domain Randomisation (DR), which has rapidly gained popularity among the research community when the task at hand involves knowledge transfer from simulated to the real-world domain. This is common in the field of robotics, where the cost of performing experiments with real robots is too unwieldy for acquiring the massive amount of data required for deep learning methods. We propose to study the impact of introducing random variations in both visual and physical properties in a virtual robotics environment, by tackling two different problems namely, 1) object detection and 2) autonomous grasping. Concerning 1), to the best of our knowledge, our research was the first to extend the application of DR to a multi-class shape detection scenario. We introduce a novel DR framework for generating synthetic data in a widely popular open-source robotics simulator (Gazebo). We then train a state-of-the-art object detector with several simulated-image datasets and conclude that DR can lead to as much as 26% improvement in mAP over a fine-tuning baseline. Regarding 2) we created a pipeline for simulating grasp trials, with support for both simple grippers and multi-fingered robotic hands. We extend the aforementioned framework to perform randomisation of physical properties and ultimately analyse the effect of applying DR to a virtual grasping scenario.

Index Terms—Domain Randomisation; Reality Gap; Simulation; Object Detection; Grasping; Robotics; Synthetic Data

I. INTRODUCTION

Currently, robots thrive in rigidly constrained environments, such as factory plants, where human-robot interaction is kept to a bare minimum. Our long-term goal as researchers on robotic systems is to aid in the transition of this field into our daily lives, namely to our households. Ultimately, we strive to achieve cooperative, effective and meaningful interactions. For this to occur we are met with numerous challenges and unsolved research problems. In this work, we focus on tackling robotic visual perception and autonomous grasping, which are crucial skills for achieving any kind of significant synergistic relationship with humans.

Regarding computer vision, the emergence of deep neural networks, and specifically Convolutional Neural Networks has led to outstanding results in many challenging tasks. The major drawback of employing deep learning is its requirement of substantial amounts of data. In robotics, acquiring massive datasets of physical experiments is generally impracticable, since physical trials are greatly both resource and time-consuming as well as overall cumbersome to carry out. A common alternative to real-robot experiments is to resort to computer simulation of robotic environments. However, even the most sophisticated simulators fail to flawlessly capture reality. Thus, machine learning algorithms trained solely on synthetic data tend to overfit the virtual domain and therefore generalise poorly when applied to a real-world scenario. Our work studies Domain Randomisation (DR) [1], which attempts to bridge this gap between simulation and reality and ultimately aims to produce synthetic data which can be directly used for training machine learning algorithms for real-robot task execution.

Rather than attempting to perfectly emulate reality, we can create models that strive to achieve robustness to high variability in the environment. DR is a simple yet powerful technique for generating training data for machine-learning algorithms. At its core, it consists in synthetically generating or enhancing data in order to introduce random variance in the environment properties that are not essential to the learning task. It is expected that given enough variability in the simulation, the transition between simulated and real domains is perceived by the model as a mere disturbance, to which it has become robust.

This work focuses on exploring several DR methods employed in recent research which led to promising results with regard to overcoming the reality gap. Concretely, we study two different scenarios.

We create tools for generating datasets employing DR to visual properties of a tabletop scene, integrated into a well-known open-source robotics simulator. We claim that we can use such datasets to train detectors that can overcome the reality gap by fine-tuning on a limited amount of real-world domain-specific images. Subsequently, we move to a robotic grasp setting and aim to understand the impact of applying the principles of DR to the physical properties of the simulated environment. We conjecture that by applying randomness to physical properties such as friction coefficients, and grasp
targets mass, will ultimately lead to more robust systems.

The research questions we propose to examine were split in two subjects, namely (i) object detection, and (ii) simulated grasping, and can be summarised as follows:

(i) **RQ1**: Can we successfully train state-of-the-art object detectors with synthetic datasets created exploiting DR in a multi-class scenario?

(ii) **RQ5**: Can DR improve synthetic grasping dataset generation, employing a physics-based grasp metric in simulation?

Our contributions can be summarised as follows.

(i) **1.** We developed GAP, an open-source tool for employing DR integrated with a prevalent open robotics simulation environment (Gazebo) for generating synthetic tabletop scenarios populated by parametric objects with random visual properties;

(ii) **2.** Using GAP as the main tool for data generation, we demonstrate the efficiency of DR in a real-world shape detection task from visual perception.

(iii) **3.** We performed a novel ablation study regarding the impact of the degree of randomisation of object visual properties, specifically by varying the complexity of the object textures.

(iv) **4.** We develop a framework for full dynamic simulation of grasp trials within Gazebo, which can be used to autonomously evaluate grasp candidates for any robotic manipulator or target object.

(v) **5.** We extend our proposed DR tool to include randomisation of physical properties of entities in a grasping scenario.

(vi) **6.** We create a pipeline for evaluating externally provided grasp candidates [3] with a parallel-plate gripper in a simulated environment, integrated with our physics DR tool, for generating synthetic grasping datasets. Finally, we perform simulated grasp trials with and without physics DR in Gazebo.

The remainder of this document is structured as follows. Section II presents an overview of relevant literature. In Section III we investigate how we can apply Domain Randomisation to visual properties of a scene. Section IV transitions into a grasping scenario, and instead aims to determine the benefits of applying DR to the physical properties in a robotic manipulation scenario. Then, in Section V we analyse our contributions and present the discussion of our findings. Finally, We summarise our work w.r.t to the research questions in section VI.

## II. RELATED WORK

### A. Autonomous grasping

The classical approach to grasp planning consists of using analytic methods. The latter, also referred to as geometric, usually involve a formal description of the grasping action, and solving the task as a constrained optimisation problem over analytic grasp metrics, such as force-closure [4], Ferrari-Canny [5] or caging [6].

Contrastingly, empirical or data-driven methods try to overcome the challenges of incomplete and noisy information concerning the object and surrounding environment by directly estimating the quality of a grasp from a convenient representation obtained from perceptual data [7]. The latter include monocular 3-colour channel (RGB) frames [8, 9, 10, 11], 2.5D RGB-Depth images [3, 12], 3D point cloud scene reconstructions [13] and even contact sensor input [14, 15].

Lenz et al. [16] formulates grasping as a detection task for real RGB-D images, where a grasp candidate is represented by an oriented bounding box. It greatly helped establish Cornell grasping dataset [17], which has since been used as a baseline by several 4-degrees of freedom (DOF) \(\{x, y, z, \theta\}\) grasp detection systems [14, 3].

Alternative approaches attempt to directly estimate the probability of grasp success for a given candidate pose from visual data [8, 3, 10, 13].

### B. Domain randomisation

Recent literature has established that DR enables achieving valuable generalisation capabilities. The term came into prominence with the work of Tobin et al. [1], and has since been applied in several virtual environments to a range of visual and physical properties.

Regarding visual properties, DR has been employed in autonomous pick-and-place tasks [1, 9]. Related work found that randomising object textures, scene lighting conditions, camera pose and increasing scene clutter with parametric distractor objects during synthetic training stages resulted in improved performance on the real robot. W.r.t employed texture patterns, mostly flat colours are used [1], along with more complex textures such as material patterns [18], Perlin noise [9] and generic images [19].

Concerning the randomisation of physical properties, recent research has applied DR in simulated grasp trials [20] and in-hand manipulation [21]. These works have highlighted that one can improve performance by varying the size of objects and robot, their respective masses, surface friction coefficients and robot controller gains. Finally, DR has been used in the procedural generation of object shapes [22, 20].

## III. DOMAIN RANDOMISATION FOR DETECTION

### A. Parametric object detection

The main objective of this section is to explore the application of DR to visual properties of a scene. We chose the task of vision-based object detection, as opposed to instance segmentation as recent methods for object detection have been
shown to achieve real-time performance [23, 24]. In robotics, it is generally preferable to produce rough estimates at a high rate than very accurate predictions with a substantial delay.

1) Offline pattern generation library: We developed a texture generation script, which we have made available as a standalone C++ library pattern-generation-lib. This tool allows us to generate four types of textures, namely single flat colour, gradient between two colours, checkerboard and Perlin noise [25]. Textures are therefore created prior to any simulation occurs.

2) GAP: A collection of Gazebo plugins: We chose Gazebo [26] as the simulation environment for generating synthetic data for our experiments, due to its popularity, community and integration with ROS. We introduce GAP, a collection of tools for applying DR within Gazebo simulations. Our software is open source and available as gap.

GAP was initially built with the purpose of generating synthetic tabletop scenes, with a ground plane and randomly placed parametric objects. Our goal was to produce a DR-powered dataset for a simple shape detection task. In order to achieve it we designed a tool that can:

- Spawn, move and remove objects, either with parametric primitive shapes such as cubes and spheres or a provided 3D mesh;
- Randomly generate textures and apply them to any object, resorting to our offline pattern generation library;
- Spawn light sources with configurable properties and direction;
- Capture images with a virtual camera;
- Obtain 2D bounding boxes from the 3D scene in order to label data for an object detection task.

B. Proof–of–Concept trials

1) Scene generation: With our novel DR tool integrated with the Gazebo simulated environment, we constructed a tabletop scene generation pipeline. We started by producing 5k textures for each of the four supported pattern types using our offline pattern generation tool. Then, we wrote a client script to spawn simple parametric shapes, namely spheres, cubes and cylinders placed in a grid, with random material properties, sampled uniformly from the aforementioned set of 20k textures. For each scene, we acquire an RGB frame and automatic annotations including object bounding boxes and class labels.

Specifically, each synthetic tabletop-like scene incorporates a ground plane, a single light source and the set of parametric objects. Each object is constrained to occupy a cell in a grid, although they can rotate freely provided they are supported by the ground plane. The objects can be partially occluded in the resulting image frame, but must not collide.

The scene generation procedure consists of 4 steps:

1) Spawn camera and a random number \( N \in [5, 10] \) of parametric shapes. Each object is put in one of 25 cells in a 5 by 5 m grid.
2) Request to move camera and light source to random positions, ensuring all objects are in the camera’s Field of View (FoV).
3) Obtain object bounding boxes, by sampling 3D object surface points and project them onto the 2D image plane.
4) Save image of the scene, as well as object bounding box and class annotations. The latter are compatible with PASCAL VOC 2012 [27] annotation format.

We provide examples of synthetic images and respective labels created with our novel framework in Figure 2.

With the first version of our pipeline setup, we managed to create a dataset of 9.000 synthetic scenes.

C. Preliminary results for Faster R-CNN and SSD

We trained two state–of–the–art object detection networks: Faster-RCNN and Single Shot Detector (SSD). We resorted to the open-source TensorFlow [4] GPU-accelerated implementation which is available on tensorflow/models.

The output of the networks includes both a detection bounding box and its class label per detection. The networks were pre-trained on Common Objects In Context (COCO) dataset [2] and fine-tuned – i.e. trained only the last fully-connected layer of the networks for over 5,000 epochs. We did this using our datasets, with a varying number of frames from our small in-house real-world dataset and our novel preliminary synthetic dataset. The former consisted of 242 images acquired in our lab with a Kinect v2 RGB-D sensor, half of which was always used as the test set. The test set contains object instances not seen in the real-world training set. The acquired synthetic RGB frames had a Full-HD resolution (1920 × 1080) and were encoded in JPEG lossy format, to match the real dataset. An improved version of this dataset was used on later experiments, as reported on Section III-D (for examples check Figure 3).

Specifically, we fine-tune each network with either only synthetic data, only real data or using both training sets. All images are reshaped to a 1080 × 1080 and downscaled to 300 × 300 pixels. We applied standard data augmentation techniques namely random horizontal flips to the preprocessed image.

Finally, we evaluate the performance in the real-world test set with the standard per-class Average Precision (AP) and mean Average Precision (mAP) metrics as defined in PASCAL VOC [27]. The experimental results of these preliminary trials on the real-world test set are shown in Table I, for an Intersection over Union (IoU) of 0.5.

From Table I we observe the following (i) Faster R-CNN performs around 20% better than SSD in average, (ii) the real-world-only training set performs better than the other
training sets in all but one case, yet achieves the best mAP score, (iii) the most difficult object to detect and localise is the cylinder class.

<table>
<thead>
<tr>
<th>Network</th>
<th>S.</th>
<th>R.</th>
<th>AP Box</th>
<th>AP Cyl.</th>
<th>AP Sph.</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster R-CNN</td>
<td>✓</td>
<td>✓</td>
<td>0.89</td>
<td>0.83</td>
<td>0.91</td>
<td>0.88</td>
</tr>
<tr>
<td>Faster R-CNN</td>
<td>✓</td>
<td>X</td>
<td>0.80</td>
<td>0.75</td>
<td>0.90</td>
<td>0.82</td>
</tr>
<tr>
<td>Faster R-CNN</td>
<td>✓</td>
<td>✓</td>
<td>0.82</td>
<td>0.78</td>
<td>0.90</td>
<td>0.83</td>
</tr>
<tr>
<td>SSD</td>
<td>X</td>
<td>✓</td>
<td>0.74</td>
<td>0.60</td>
<td>0.76</td>
<td>0.70</td>
</tr>
<tr>
<td>SSD</td>
<td>✓</td>
<td>X</td>
<td>0.70</td>
<td>0.47</td>
<td>0.77</td>
<td>0.64</td>
</tr>
<tr>
<td>SSD</td>
<td>✓</td>
<td>✓</td>
<td>0.67</td>
<td>0.40</td>
<td>0.80</td>
<td>0.62</td>
</tr>
</tbody>
</table>

At the time we suggested the number of simulated images was not large enough to consider different lighting conditions, and thus training with the randomised dataset alone is not able to reach the desired performance on real-world data. Later we performed tests that would suggest otherwise, in the sense that we could achieve better performance using an even smaller amount of synthetic images. We observe that the networks fine-tuned only with synthetic data achieved lower performance, but were still able to perform decently in a real-world dataset. We published the description of our tool alongside these results in [28], in April 2018.

D. Domain randomisation vs fine-tuning

1) Single Shot Detector: Even though our preliminary results established that Faster R-CNN obtained better performance in the object detection task, we elected SSD as the base detector. The main reason for this is that SSD is one of the few detectors that can be applied in real-time while showing a decent accuracy. Furthermore, since SSD is considerably faster to train, we were allowed to perform a thorough ablation study with respect to the individual contribution of texture types for DR, which is further detailed in Section III-D5.

In any case, we believe that SSD has no property which makes it more or less susceptible to benefit from DR. Thus, we expect our results to directly generalise to other deep learning based detectors.

We provide a brief description of the inner workings of SSD in the full thesis report. However, readers should refer to the original publication [24] for a comprehensive study of the detector.

Unlike the original SSD architecture, we used MobileNet [29] as the base Convolutional Neural Network (CNN) for feature extraction in all experiments. MobileNet changes the connections in a conventional CNN to drastically reduce its number of parameters, without having a significant toll on performance, relative to a comparable architecture.

2) Improved scene generation: In order to lower the execution time of synthetic data generation, we modified the scene creation script. Originally, each captured image required parametric objects to be generated from a SDF (Simulation Description Files) formatted string, which was altered during run-time in order to allow for different object dimensions and visuals. Furthermore, objects were created and destroyed in between scenes. We altered our pipeline so we could instead modify their visual properties directly through the Gazebo Application Programming Interface (API), reusing existing objects in simulation. In this way, it is possible to alter an object pose and change its visual appearance. What is more, by changing the objects scale vector we can effectively morph the object shape. We accomplished this by creating a novel plugin that interacts exclusively with the rendering engine and the objects visual representation. This approach is much more efficient than spawning and removing objects each iteration.

We start by spawning the maximum number of objects of each type in the scene, below the ground plane, where they cannot be seen by the virtual camera sensor. Then, for each scene, the client application performs requests to change the objects’ visuals poses, scale and materials. We found this optimisation to be quite effective and obtained almost doubled performance, generating 9,000 synthetic images in little over 1h30min. We believe this contribution to be significant, particularly considering we did not have to write rendering-specific code to optimise the pipeline.

3) Real images dataset: We created an improved dataset of images acquired in our lab with a Kinect v2 RGB-D sensor, part of which were used in our preliminary trials with both SSD and Faster R-CNN. These scenes include household objects (such as cups, mugs, balls and cardboard boxes) placed arbitrarily on the ground, or stacked on top of each other, with varying degrees of occlusion. Each image was manually annotated with object class labels and bounding boxes. The refined dataset consists of 250 real images, 49 of which contain

*SDF Specification http://sdformat.org/, as of December 17, 2018*
objects unseen in training, for the sole purpose of reporting final performance. From the remaining 201, 175 are used for training and 26 for validation.

In this dataset, there was no consideration to explicitly keep the percentage of different classes balanced (Table II). Thus, we have also reported precision-recall curves for each class in the full thesis.

![Real dataset](http://vislab.isr.ist.utl.pt/datasets/#shapes2018, as of December 17, 2018)

![Real dataset](http://vislab.isr.ist.utl.pt/datasets/#shapes2018, as of December 17, 2018)

### Table II
**NUMBER OF INSTANCES AND PERCENTAGE OF DIFFERENT CLASSES IN THE REAL DATASET.**

<table>
<thead>
<tr>
<th>Partition</th>
<th># Box</th>
<th># Cyl.</th>
<th># Sph.</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train set</td>
<td>502 (63%)</td>
<td>209 (26%)</td>
<td>86 (11%)</td>
<td>797</td>
</tr>
<tr>
<td>Test set</td>
<td>106 (40%)</td>
<td>104 (40%)</td>
<td>53 (20%)</td>
<td>263</td>
</tr>
</tbody>
</table>

A set of training images from our small in-house dataset is shown in Figure 3. Finally, the dataset has been made available online³.

![Real dataset](http://vislab.isr.ist.utl.pt/datasets/#shapes2018, as of December 17, 2018)

**Figure 3. Real image training set examples**

4) **Experiment: domain randomisation vs fine-tuning:** We have conducted various experiments and tests to quantify the results of different scenarios. Initially, two sets of 30k synthetic images were generated. The software modifications we mentioned in section III-D2 have greatly facilitated this process. These two sets differed from one another by the degree to which the virtual camera in the scene has changed its location. In the first set, the viewpoint was fixed, whereas in the other set, its location varied largely across the scene. All synthetic images have Full-HD (1920 × 1080) resolution and are encoded in JPEG lossy format, to match the training images from the real images dataset. For training and testing, images are down-scaled to half these dimensions (960 × 540) which is the resolution employed for all test scenarios in our pipeline.

For baseline calculations, we have used SSD, trained on COCO, and fine-tuned it on the train set until the performance on the real-image validation set failed to improve. In other experiments, we have used MobileNet which was trained on ImageNet as the CNN classifier of SSD and first fine-tuned it on synthetic datasets with bigger learning rates and later, in some experiments, fine-tuned again with smaller learning rates on the real dataset.

Finally, smaller synthetic datasets of 6k images were generated, each with a type of texture missing, and an additional baseline for comparison which includes every pattern type. These datasets allowed us to study the contribution of each individual texture in the final performance, as well as evaluate the effectiveness of smaller synthetic datasets.

Networks were trained with mini-batches of size 8, on a machine with two Nvidia Titan Xp Graphical Processing Units (GPUs), for a duration depending on the performance in a real image validation set. We have only used horizontal flips and random crops, with parameters reported in the original SSD paper, as the pre-processing step, since we are interested in studying the effects of synthetic data and not different pre-processings. Finally, in compliance with the findings in [19], all the weights of the network are being updated in our experiments.

We wish to quantify how much an object detector performance would improve due to the usage of synthetic data. To this purpose, initially, we fine-tuned a SSD, pre-trained on COCO dataset with our real image dataset for 16,000 epochs, which was deemed sufficient by evaluating the performance on our validation set. We used a decaying learning rate $\alpha_0 = 0.004$, with a decay factor $k = 0.95$ every $t = 100k$ steps. In the subsequent sections, we refer to this network as the baseline.

Afterwards, we trained SSD with only its classifier pre-trained on ImageNet, using our two synthetic datasets of 30k images each.

These datasets are similar to the one generated during our preliminary trials, yet contain simulated tabletop scenarios with a random number of objects $N \in [2, 7]$, placed randomly on the ground plane in a smaller $3 \times 3$ grid, to avoid overlap.

In the first dataset, the camera pose is randomly generated for each scene, such that it points to the centre of the object grid. This generally results in high variability in the output, which may improve generalisation capabilities of the network at the expense of added difficulty to the learning task, as, for instance, it exhibits higher levels of occlusion. In the second dataset, the camera is fixed overlooking the scene at a downward angle, which is closer to the scenario we considered in the real dataset. Example scenes with viewpoint candidates for each dataset are shown in Figure 4.

![Real dataset](http://vislab.isr.ist.utl.pt/datasets/#shapes2018, as of December 17, 2018)

The scene light source is always allowed to move in a manner akin to the camera in the first dataset. Similar to the baseline, the networks were trained on these datasets for over 90 epochs, based on their performance on the validation set employing an exponentially decaying learning rate, starting at $\alpha_0 = 8 \times 10^{-3}$, and a decay of $k = 0.95$ every $t = 50k$ steps. These networks were then directly applied to the real-world image test set without any fine-tuning on our dataset of real object data, in order to quantify how much knowledge can be directly transferred from synthetic to real domain.

³Real dataset http://vislab.isr.ist.utl.pt/datasets/#shapes2018, as of December 17, 2018
Finally, these two detectors were fine-tuned on the real dataset for over 2200 epochs and with a fixed learning rate of $\alpha = 10^{-3}$. The result of this analysis is depicted in Figure 5 and summarised in Table III. Additionally, Figure 6 shows example output of our best performing network for our test set.

![Moving viewpoint](image1)

![Fixed viewpoint](image2)

---

**Figure 4.** Viewpoint candidates in synthetic scene generation. Left: Viewpoint changes both position and rotation in between scenes. Subfigure represents four possible camera poses. Right: Viewpoint is static.

The network trained on the dataset with no camera pose variation and fine-tuning on real data exhibits the best performance at 0.83 mAP, which corresponds to an improvement of 26% over baseline.

Perhaps unintuitively, variations in camera position have hurt the mAP score. We believe this is due to the absence of noticeable camera variations in our real test set.

Consistently, the networks trained on the fixed viewpoint dataset and fine-tuned on the real dataset out-perform other variations. This trend is only less prominent in the case of the sphere class, where, seemingly, due to the smaller amount of samples of this class in the real dataset (Table II) the training benefits less from fine-tuning on the real dataset. This observation is also visible in Table III. We hypothesise that more real sphere examples could help the detector in improving the AP score for spheres.

5) **Ablation study: individual contribution of texture types:**

We have created smaller synthetic datasets with only 6k images, wherein each of them one specific texture is missing. Once more, we selected MobileNet pre-trained on ImageNet as the classifier CNN. However, the detectors were instead trained on these smaller synthetic datasets and then fine-tuned on the real dataset.

The training of all networks on synthetic datasets lasted for 130 epochs, which was found to be the point where the mAP did not improve over the validation set, with an exponentially decaying learning rate starting at $\alpha_0 = 0.004$, $k = 0.95$ and $t = 50k$ steps. Finally, these networks were fine-tuned with the real-image dataset for 1100 epochs, with a constant learning rate $\alpha = 0.001$. The performance of SSD on the validation set during training is shown in Figure 8. The results of these experiments are reported in Figure 7 and Table IV.

![SSD output for real image test set when trained on FV and fine-tuned on Real datasets.](image3)

**Figure 6.** SSD output for real image test set when trained on FV and fine-tuned on Real datasets.

By comparing figures Figures 7(a) and 7(b) it is clear that all variations have benefited from fine-tuning with the real dataset.

Regarding individual texture patterns, generally speaking, by removing more and more complex textures (flat to be the least complex and Perlin to be the most complex), the performance hurts, and we found Perlin noise to be a vital
texture for object detection, while the flat texture has the least significance. Consistent with this observation, according to Figure 7(b), the small dataset with all the textures cannot always compete with some of the datasets where a texture is missing.

According to Figures 7(a) and 7(b), the detector trained on the small dataset with all texture classes outperformed other variations on the validation set during training. However, presumably due to the smaller number of samples and simultaneously many texture types it overfitted to the objects in the train set and failed to generalise to the test set as easily as the remaining detectors.

The network trained on our smaller dataset of only 6k images without the “flat” texture has even slightly out-performed the network that was trained on 30k synthetic images. This result seems to be consistent for detectors with classifiers fine-tuned with real samples. After a fixed number of images, the mAP performance oscillates for one or two percents.

The code for replicating our experiments is also available as tf-shape-detection\(^8\).

**IV. DOMAIN RANDOMISATION FOR GRASPING**

In this section, we aim to explore how to apply DR in a simulated grasping environment. Particularly, we wish to apply randomisation to the physical properties of the entities at play, such as mass, friction coefficients, robotic joint dynamic behaviour and controller parameters. Ultimately, we hope to create synthetic datasets which can be used for real robotic grasping.

For this, we require a tool that incorporates a grasp proposal module and pairs each of the grasp candidates with a quality metric. Additionally, in order to achieve a robust grasp quality estimation, we wish to obtain some metric through full dynamic simulation of grasp trials, preferably in an open and established robotics simulator. This approach differs from that of Mahler et al [3], which computes metrics from the geometry of the object, such as force-closure and Ferrary-Canny [5].

Instead, we propose to obtain a grasp quality estimation by replicating real robotic trials in a simulated environment, and evaluate whether they are successful or not. In this fashion, by applying DR to the physical properties between different attempts of a given grasp configuration we aim to achieve a robust metric.

We are aware that robotic grasping is a hard task, which is why we must first simplify the problem at hand, imposing reasonable constraints so as to streamline our research. Our principal assumptions involve using

1) A disembodied robotic manipulator with unconstrained movement in 3-D space, thus avoiding the reaching part of the grasping action, much like [30, 31, 32, 10, 33];
2) Simplified rigid body contact and Coulomb friction models;
3) A parallel-plate gripper as opposed to dexterous multi-fingered robot hands.

**A. Dex-Net pipeline for grasp quality estimation**

Rather than designing our end-to-end pipeline for grasping candidate generation and evaluation, we decided to employ that of Dex-Net [3]. This allows us to focus on studying the impact of employing DR and prevents additional biases stemming from novel self-proposed methods. The authors tackle grasping unknown objects with parallel-plate grippers given visual input from an RGB-D camera mounted on the gripper. The problem of grasping is split into two steps, namely (i) grasp candidate proposal and (ii) grasp quality evaluation.

In our work, we wish to use Dex-Net as the main building block. Our goal is to generate a novel grasping dataset, using 3D models scanned from real-world household objects found in KIT object dataset [34]. Grasp candidates for these objects should be computed using the method of [3] so we can establish an adequate baseline. Analogously, we should use this framework to provide baseline grasp quality estimates, computed resorting to geometric metrics.

We employ Dex-Net grasp generation algorithm, which can be described in two main steps (i) use a generic algorithm for sampling grasps; (ii) prune grasps that are too similar, for maximum surface coverage. Dex-Net implements three distinct grasp sampling methods, with increasing computation power requirements and expected output grasp quality. We focus on antipodal grasp sampling, which supplies robust grasp candidates, albeit at the expense of added complexity and longer run times.

Antipodal points are a pair of points on an object surface, which surface normal vectors are colinear but have opposite

---

\(^8\) [jbruglie/tf-shape-detection](https://github.com/jbruglie/tf-shape-detection) Object detection pipeline, on GitHub as of December 17, 2018
direction. Under specific finger contact conditions, antipodal grasps ensure force-closure. For a detailed description of both generic grasp generation and antipodal grasp sampler algorithms, we refer the interested reader to the full thesis report. A set of antipodal grasp candidates is present in Figure 9.

![Figure 9. Antipodal grasp candidates for Baxter gripper, aligned with table normal for 3 objects from KIT dataset. Gripper colour encodes robust force-closure metric as a gradient: green is stable, orange unstable (check Section IV-D).](image)

B. GRASP: Dynamic grasping simulation within Gazebo

We propose a novel framework for Gazebo, for performing dynamic simulation of grasp trial outcome, which we named GRASP, and is open source as grasp9.

So far, the development of this platform has produced

- A model plugin for programmatic control of a floating robotic manipulator, either a single DOF gripper or multi-fingered hand;
- An interface class for simplifying interaction with the former from a client application;
- A tool for efficiently querying Gazebo server for surface contacts;
- A script for obtaining object stable poses via dynamic simulation;
- A pipeline to compute grasp trial outcomes using full dynamic simulation, given a dataset of objects and a set of grasp candidates per object computed externally for the desired robotic manipulator.

Currently, our setup employs DART physics engine, which provides realistic and stable grasping simulation, unlike ODE, the default engine available in Gazebo. For an in-depth analysis of this choice as well as implementation details regarding the features offered by our tool, we once more refer the reader to our full thesis report.

C. Extending GAP with physics-based domain randomisation

We decided to enhance our proposed framework GAP and develop a brand-new module for physics-based DR. Our revised set of tools allows us to:

- Change gravity vector components;
- Alter the scale of existing models;
- Modify individual link properties, namely mass and inertia tensor;
- Change individual joint properties, such as angle lower and upper limits, as well as maximum effort and velocity constraints;
- Adjust individual robot PID controller parameters and target values;
- Vary individual collision surface properties, including $\mu$ Coulomb friction coefficients.

We also provide a client interface that can easily be imported into third-party applications and used as an API.

1) Physics randomiser class: We designed a physics randomiser interface class, as part of our grasping pipeline toolkit. It maps random distributions to physical properties specified in a yaml configuration file, samples values from these distributions, and communicates with the Gazebo server via our custom API in GAP to update the physical properties of the scene. This class allows the automation of the whole scene randomisation process, through a clean interface, and swift testing.

D. DexNet: Domain randomisation vs geometry-based metrics

Our current process for generating a grasping dataset can be summarised as follows

1) Create and pre-process object dataset in Dex-Net;
2) Import objects into Gazebo compatible models;
3) Compute antipodal grasp candidates in Dex-Net and respective baseline quality metrics, from object geometry;
4) Compute realistic object rest poses in Gazebo via physics simulation;
5) Import object rest poses into Dex-Net in order to obtain aligned grasp candidates;
6) Import resulting grasp candidates into Gazebo and perform grasp trials in order to ascertain whether grasp is robust or not, introducing DR by randomising properties between trial repetitions.

The process of running simulated grasp trials essentially involves (i) spawning the object in a pre-computed stable pose; (ii) placing the robotic manipulator in the pre-grasp pose; If a collision is detected at the outset, the grasp is deemed invalid and interrupted; (iii) closing the fingers and attempt to lift the object.

1) Grasp quality estimation: As a baseline metric, we employ Dex-Net’s robust force-closure, which performs Monte Carlo sampling when computing the force-closure metric for a target object, randomising object and gripper relative poses, as well as the friction coefficient between the two. Since our simulation can only provide binary grasp success labels per trial, there is no evident way to directly compare them. Instead, we propose a derived binary metric, by thresholding the robust force-closure metric at several values of $\delta$.

We are not interested in invalid grasp poses, i.e. which result in an instant collision between the gripper and the target object or planar surface. We started by validating our grasp candidates

9/Gazebo plugins for running grasp trials, on GitHub as of December 17, 2018.
proposal integration by acquiring 200 antipodal grasps for each of the 129 objects of KIT dataset. Then, we align these grasps by providing a realistic object rest pose, computed in Gazebo using our framework. We then proceed to simulate grasp trials for each of the aligned grasp candidates, without performing any randomisation whatsoever. Thus, there is no point in performing trial repetitions.

Each grasp candidate is automatically annotated with a binary success label \( s \in \{0, 1\} \), or a flag indicating that the grasp candidate is invalid due to collisions at the outset. Table V presents the averaged results for our grasp trials for KIT dataset, where KIT\(^{-}\) dataset denotes KIT without objects for which only invalid grasps were available.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Successful trials</th>
<th>Failed trials</th>
<th>Invalid trials</th>
</tr>
</thead>
<tbody>
<tr>
<td>KIT</td>
<td>13.4591 %</td>
<td>5.7578 %</td>
<td>80.7836 %</td>
</tr>
<tr>
<td>KIT(^{-})</td>
<td>14.8514 %</td>
<td>6.3529 %</td>
<td>78.7957 %</td>
</tr>
</tbody>
</table>

This high amount of invalid grasps is expected, as our grasp proposal module only enforces constraints on the contact points, and aligns the grasp angle according to the planar surface normal, for a given object stable pose. It does not, however, take into consideration the shape of the gripper. Figure 10 shows a time-lapse of a successful grasp attempt.

![Figure 10. Time-lapse of successful grasp trial on CokePlasticLarge object](image)

For each valid grasp candidate, we count the number of successful attempts. This value is then averaged over the number of trials per grasp candidate (in this case 10). We wish to compare this result with the binary success label for a single grasp trial, without employing DR. In order to do so, we convert the robust force-closure metric to a binary value, by choosing a threshold \( \delta \). The outcome average success rate for trials employing DR is thresholded at 0.5, which corresponds to classifying a grasp as a success when at least 50% of the trials succeeded. We then use the thresholded robust force-closure as our ground truth, and formulate the metric correlation as a binary classification problem. Thus, we can effectively compare the metrics from trials with and without DR.

We evaluate the accuracy at several values of \( \delta \), which corresponds to varying the level at which our ground truth classifies a grasp to be successful. Figure 11 presents the accuracy for the six objects chosen for the grasp trials with and without physics-based DR (orange and blue, respectively).

![Figure 11. Accuracy of our binary success metric, for the grasp trials with and without physics-based DR (orange and blue, respectively).](image)

It seems that for low values of the threshold \( \delta \), i.e. weaker constraint for classifying a grasp as successful, DR causes our grasp quality to slightly differ from the baseline binary metric. Evaluating the concrete detections (number of true positives, true negatives and so on), we identify that DR causes a higher amount of false negative detections, which means that it provides a more conservative grasp quality estimate, as it rejects low confidence grasps when the baseline classifies them as successful. This results in the slightly lower accuracy observed in each subfigure, for smaller \( \delta \). On the other hand, for higher values of \( \delta \), we notice that DR results in a fewer amount of false positives, once more suggesting that the metric is more conservative, and thus, robust.

V. DISCUSSION

This section presents merely a gist of the discussion of our findings; the remaining is argued in the body of the full thesis. Our work reinforced the benefits of employing DR in computer vision, specifically the generalisation capabilities of CNNs when trained with synthetic data, as opposed to large domain-generic datasets. However, we were also able to witness first hand that unnecessary variations merely increase the difficulty of the learning task, and may, in fact, hinder performance.

Concerning autonomous grasping, we conjecture our pipeline can be used for providing conservative and robust...
ground truth labels, but future work is required for adequately supporting this claim. We argue that our experiments have shown that DR is a promising research topic, and the tools we developed and experiments carried out have made some progress in this direction.

VI. CONCLUSION

Regarding our object detection trials, our work was the first to demonstrate the application of DR in a multi-class scenario. We prove that domain-specific synthetic datasets employing DR can lead to substantial improvements over pre-training on datasets such as COCO, which is currently a common approach. Our ablation study also established that employing complex random textures result in better performance gains than simple patterns.

With respect to our research on grasping, we provide a novel framework for Gazebo to simulate dynamic trials and obtain success metrics, with added robustness resultant of the application of DR. Future work should build upon our findings, employing our tools and generalise to a multi-fingered hand scenario.

REFERENCES


