A Generic Visual Perception Domain Randomisation Framework for Gazebo

João Borrego, Rui Figueiredo, Atabak Dehban, Plinio Moreno, Alexandre Bernardino and José Santos-Victor
Institute for Systems and Robotics, Instituto Superior Técnico, Lisboa, Portugal
Email: {jborrego, ruifigueiredo, adehban, plinio, alex, jasv}@isr.tecnico.ulisboa.pt

***Abstract*—The impressive results of applying deep neural networks in tasks such as object recognition, language translation, and solving digital games are largely attributed to the availability of massive amounts of high quality labelled data. However, despite numerous promising steps in incorporating these techniques in robotic tasks, the cost of gathering data with a real robot has halted the proliferation of deep learning in robotics. In this work, a plugin for the Gazebo simulator is presented, which allows rapid generation of synthetic data. By introducing variations in *simulator-specific* or *irrelevant aspects of a task*, one can train a model which exhibits some degree of robustness against those aspects, and ultimately narrow the reality gap between simulated and real-world data. To show a use-case of the developed software, we build a new dataset for detection and localisation of three object classes: box, cylinder and sphere. Our results in the object detection and localisation task demonstrate that with small datasets generated only in simulation, one can achieve comparable performance to that achieved when training on real-world images.**

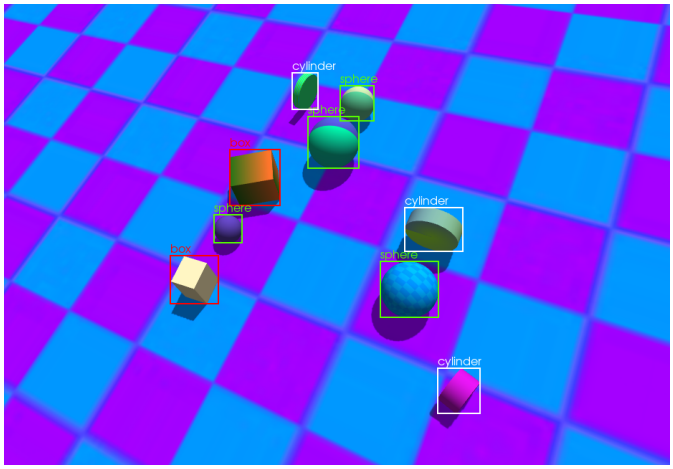

Fig. 1. An example of a synthetically generated scene comprising multiple parametric shapes, overlaid with automatically generated ground truth annotations

I. Introduction

Modern robots can be programmed manually to perform rather complex manipulation tasks, leading them to thrive in rigidly constrained environments such as factory plants. Yet, autonomously interacting with novel objects remains to this day a challenging task due to perceptual limitations in visual and tactile sensors and environmental constraints such as self-occlusion [1]. Moreover, in order for robots to transition from simulated into real-world applications it is important that they become robust to external disturbances.

Classical machine-learning for computer vision approaches usually involve extracting some high-level feature representation from images in order to try to solve the foregoing challenges. However, the emergence of deep neural networks and more specifically convolutional neural networks has caused a paradigm shift, achieving excellent results using raw pixel data directly, thus avoiding feature engineering. The main issue with such an approach is usually associated with the large amount of data required to train these networks. Furthermore, manually annotating results is a slow and tedious process. An alternative is to resort to virtual simulations in order to automate data acquisition and labelling. Simulating a robotic experiment is generally much faster than real-life execution, thus accelerating the data acquisition process. However, even the most sophisticated existing simulator fails to flawlessly capture reality.

For this reason, learning models that are trained solely on pure synthetic data tend to overfit the simulated domain and generalise poorly to the real world.

In this work we explore the concept of domain randomisation, which addresses the aforementioned issue, with the goal of generating synthetic datasets suitable for training deep neural network object detectors. We explore several domain randomisation techniques employed in recent research which led to promising results with regard to overcoming the reality gap. Rather than attempting to perfectly emulate reality, one can create models that strive to achieve robustness to high variability in the environment. Domain randomisation 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. This idea dates back to at least 1997 [2], with Jakobi's observation that evolved controllers exploit the unrealistic details of flawed simulators. His work on evolutionary robotics studies the hypothesis that controllers can evolve to become more robust by introducing random noise in all the aspects of simulation which do not have a basis in reality, and only slightly randomising the remaining which do.

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 to become robust. This is the main argument for this method. It has been established that this approach enables achieving valuable generalisation capabilities.

In contrast with previous approaches which were mainly applied to end-to-end visuomotor control during grasping [3], we analyse state-of-the-art object detection deep learning methods performance when trained mostly on synthetic data. Being capable of performing object detection is closely related to the ability to reason about and interact with dynamic environments (for example [4]).

We provide a set of tools to interact with Gazebo[1] [5], a simulation environment well-known in robotic research, in order to allow a programmatic generation of scenarios employing the principle of domain randomisation for posterior training of deep neural networks. Furthermore, we present one such example application and study the performance of a state-of-the-art neural network in a simple three-class object detection task, when trained on the resulting simulated dataset.

The remainder of this paper goes as follows: in section II we clarify the main concepts and overview the related work on deep object detection and domain randomisation. In section III we introduce a novel domain randomisation plugin for Gazebo that allows generating large datasets suitable for training deep object detectors. In section IV we perform a set of experiments that validate the usability of the proposed domain randomisation plugin. Finally, in section V we wrap up with some conclusions.

II. Related Work

In order to fully understand the context in which we can apply domain randomisation to improve robotic perception we provide an overview of recent relevant research. We begin by analysing the rise of convolutional neural networks, a deep learning framework widely used in image processing tasks, with an emphasis on object detection architectures. Then, we present the main advances in the field of domain randomisation for the generation of large synthetic datasets.

A. Deep Neural Network Object Detection Architectures

As of today, Convolutional Neural Networks (CNNs) are the state-of-the-art in visual recognition. Several architectures for CNNs have been proposed recently in part due to ImageNet's Large Scale Visual Recognition Challenge (ILSVRC) [6], which evaluates object detection and localisation performance. Analogously, the PASCAL VOC challenge [7] focuses on the same problems, at the outset.

1) R-CNN and Fast R-CNN: A three-step pipeline was proposed in 2013 [8] in an attempt to improve the state of the art results in PASCAL VOC 2012, achieving a performance gain of over 30% in mean Average Precision (mAP) when compared to the previous best result. Their proposed method entitled Regional CNN (R-CNN) first extracts region proposals from the image, and then feeds each region to a CNN with a similar architecture to that of [9]. The output of the CNN is then evaluated by a Support Vector Machine (SVM) classifier. Finally, the bounding boxes are tightened by resorting to a linear regression model. The output of their network is a set of bounding boxes surrounding the objects of interest and the respective classification. The region proposals are obtained through selective search [10]. This method cannot run in real time, because three different models have to be used sequentially: (i) the CNN to generate image features, (ii) the SVM classifier and (iii) the regression model to tighten the bounding boxes. Moreover, each region proposal requires a forward pass of the neural network. Fast R-CNN [11] addresses the computational complexity issues, using Region of Interest Pooling (RoIPool), which leverages the fact that there is generally an overlap between proposed interest regions. The high-level idea is to have several regions of interest sharing a single forward pass of the network. Specifically, for each region proposal we keep a section of the corresponding feature map and scale it to a pre-defined size, with a max pool operation. Essentially this allows us to obtain fixed-size feature maps for variable-size input rectangular sections. Thus, if an image section includes several region proposals we can execute the forward pass of the network using a single feature map, which dramatically speeds up training times. The second major improvement consists of integrating the three previously separated models into a single network. A Softmax layer replaces the SVM classifier altogether and the bounding box coordinates are calculated in parallel by a dedicated linear regression layer.

2) Faster R-CNN: The progress of Fast R-CNN exposed the region proposal procedure as the bottleneck of the object detection pipeline. A Region Proposal Network (RPN) is introduced in [12] and is a fully Convolutional Neural Network (i.e. every layer is convolutional) that simultaneously predicts object's bounding boxes as well as objectness score. The latter term refers to a metric for evaluating the likelihood of the presence of an object of any class in a given image window. Since the calculation of region proposals depends on features of the image calculated during the forward pass of the CNN, the authors merge RPN with Fast R-CNN into a single network. This further optimises runtime while achieving state of the art performance in the PASCAL VOC 2007, 2012 and Microsoft's COCO [13] datasets. However, the method is still too computationally intensive to be used in real-time applications, running at roughly 7 frames per second in a high end graphics card (Nvidia® GTX Titan X).

3) Single Shot Detector: An alternative approach is that of [14], which eliminates the need for interest region proposal entirely. Their fully convolutional network known as Single Shot Detector (SSD) predicts category scores and bounding box offsets for a fixed set of default output bounding boxes. SSD outperformed Faster R-CNN and previous state of the art single shot detector YOLO [15], with 74.3% mAP on PASCAL VOC07. Furthermore, it achieved real-time perfor-

[1] http://gazebosim.org/, as of April 11, 2018

mance[2], processing images of 512×512 resolution at 59 FPS on an Nvidia® GTX Titan X GPU [14].

B. Domain Randomisation

Domain randomisation was employed in the creation of simulated tabletop scenarios to train real-world object detectors with position estimation. The work of [16] aims to predict the 3D position of a single object from single low-fidelity 2D RGB images with a network trained in a simulated environment. This prediction must be sufficiently accurate to perform a grasp action on an object in a real-world setting. This work's main contribution is proving that indeed a model trained solely on simulated data generated using domain randomisation was able to generalise to the real-world setting without additional training. Moreover, it shows that this technique can be useful for robotic tasks that require precision. Their method estimates an object's 3D position with an accuracy of 1.5 cm, succeeding in 38 out of 40 trials, and was the first successful transfer from a fully simulated domain to reality in the context of robotic control. Furthermore, it provides an example of properties that can be easily randomised in simulation, namely object position, colour and texture as well as lighting conditions. Their model uses CNNs trained using hundreds of thousands of 2D low resolution renders of randomly generated scenes. Specifically, the authors employ a modified VGG-16 architecture [17] and use the MuJoCo[3] [18] physics simulation environment. When we started developing our software tool, MuJoCo was proprietary software, and was therefore not considered as an option to perform the test. Instead we created an open-source tool for domain randomisation in Gazebo which is a better fit software in robotics, due to its widespread utilisation and integration with ROS [4].

In [19] the concept of domain randomisation was applied to object synthesis instead of scene randomisation. This work's goal was to learn a mapping from a set of observations (depth images) of a scene and output a feasible grasp. The grasp is defined as the centre of the gripper and its orientation, which results in 6-D grasp space, or 4-D if it is an upright grasp. In a similar setting to their previous work [16], the authors proposed a data generation pipeline for training deep learning methods in the task of object grasping. This method includes procedurally generating novel objects. Millions of unique objects are created by the concatenation of small convex blocks segmented from 3D objects present in the popular ShapeNet dataset [20]. Specifically, over 40,000 3D object meshes were decomposed into more than 400,000 approximately convex parts using V-HACD[5]. The latter is an open-source library for decomposing 3D surfaces into near convex sub-units.

Each of these parts is used as a primitive building block for generating objects. A random number of primitive meshes is chosen and combined such that there is an intersection with at least another mesh. The position and scale of each primitive mesh is randomised and the final object scale is also recalculated in order to fit the distribution of object sizes present in the original real-world objects dataset.

Their method was evaluated in simulation with a set of previously unseen realistic objects achieving a high ($\geq$90%) success rate. It demonstrates the use of domain randomisation tools in order to generate a large dataset of random unrealistic objects and generalise to unseen realistic objects. However it should be mentioned that their experiments are all performed in a simulated environment. The knowledge transfer between domains in this case refers to the transition from procedurally generated unrealistic object meshes to 3D models of real-life objects.

Recent research [3] has explored a similar tabletop scenario with the objective of performing a simple pick and place routine with an end-to-end control approach, using domain randomisation during the data generation. The authors introduced an end-to-end (image to velocity) controller that is trained on purely synthetic data from a simulator. The task at hand is a multi-stage action that involves the detection of a cube, picking it up and dropping it in a basket. To fulfil this task, a reactive neural network controller is trained to continuously output motor velocities given a live image feed.

James et. al. [3] report that without using domain randomisation it was possible to fulfil the task 100% of the time in simulation. However, the performance degrades to 72% in the real-world setting if the model were to be directly applied. This work further analyses relevant research questions, namely: (i) What is the impact on the performance when varying the training data size?, (ii) which properties should be randomised during simulation?, and (iii) which are the additional features that may complement the image information (e.g. joint angles) and improve the performance? To answer these questions, the authors assessed the importance of each randomised properties in the simulated training data, namely removing object clutter, the presence of shadows, randomisation of textures, the change in camera position and joint angle features. The main observations from the experiment results are as follows: (i) It was possible to obtain over 80% success rate in simulation with a relatively small dataset of 100,000 images, but (ii) to obtain similar results in a real-world scenario it was needed roughly four times more data, (iii) training with realistically textured scenes as opposed to randomising textures also results in a drastic performance decrease, and (iv) not moving the camera during simulation yields the worst performance in the real-world scene as the cube is only grasped in 9% of the trials and the task finished in 3%.

III. Random Worlds: A Domain Randomisation Plugin for Gazebo

The main contribution of this work is a set of tools that integrate with Gazebo under the form of a plugin, that allow using domain randomisation techniques in order to generate large synthetic datasets for object detection.

We chose Gazebo as the simulated environment in which to produce the synthetic data. Gazebo is an open-source physics

[2]Note that the term real-time is respective to common image capture rates.
[3]http://www.mujoco.org, as of April 11, 2018
[4]http://www.ros.org/, as of April 11, 2018
[5]https://github.com/kmammou/v-hacd, as of April 11, 2018

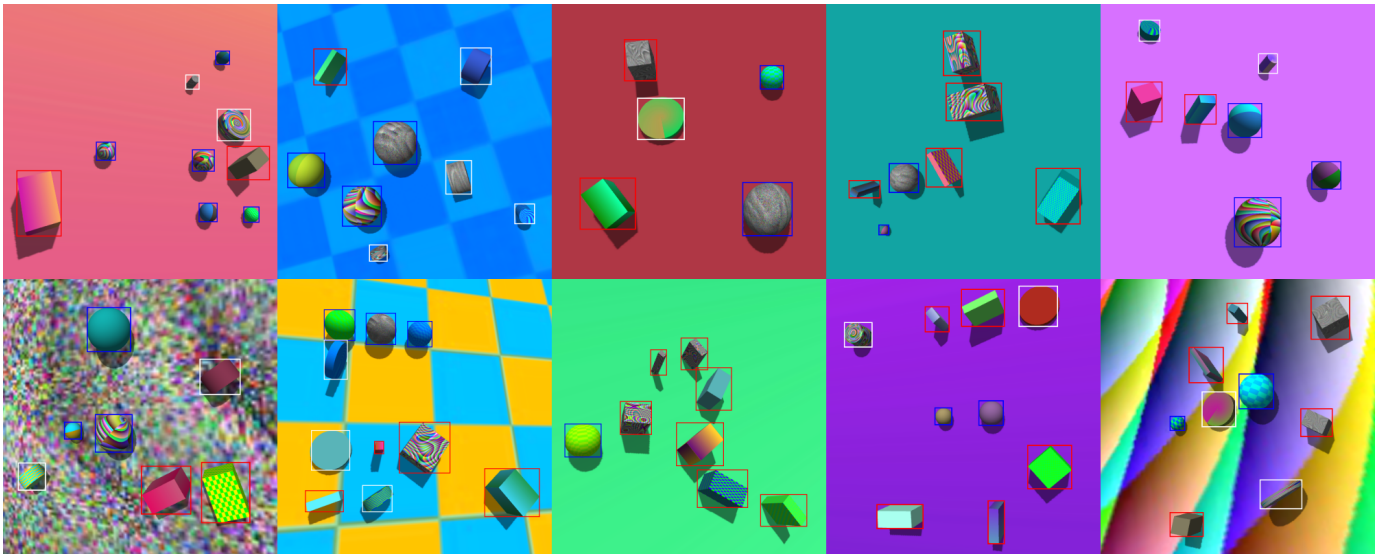

Fig. 2. Synthetically generated scenes comprising basic parametric shapes, with ground truth bounding box overlays created using the proposed Gazebo plugin.

simulator used in robotics, known for its integration with ROS, an increasingly popular robot middleware platform. We claim that the main advantage of using Gazebo as opposed to a dedicated rendering environment is first and foremost the ease of integration of simulated robots in experiments.

At its core, our software consists of three separate Gazebo modules, that allow the user to perform otherwise complex operations such as generating primitive-shaped objects with random properties and texture from within the simulated environment.

Gazebo's off-the-shelf tools for real-time scene manipulation are insufficient to achieve our goal. For instance, none offers the ability to move an object during simulation without GUI interaction. Furthermore, most of the available solutions depend on ROS, which adds an unnecessary layer of abstraction and hinders performance. Thus, this work provides an API which can be used to perform the necessary operations.

S. James et al. [3] demonstrate that the success of their network is due to employing domain randomisation in a few concrete properties, specifically relative camera pose, presence of distractor objects, object texture and scene lighting conditions. Thus, we incorporate the possibility to manipulate these features into a tool that works on top of Gazebo. Furthermore, if we want a fully automated dataset generation we must ensure accurate automatic labelling of training data. This of course depends on the task at hand, which can include object detection or direct grasp quality estimation.

We compiled the most important features for a novel tool to support perceptual domain randomisation-powered dataset generation, namely, the ability to:

- Spawn, move and remove objects, either with a parametric primitive shapes such as cubes and spheres or a provided 3D mesh;
- Randomly generate textures and apply them to any object. More specifically, the ability to produce four different pattern types of textures, namely flat, gradient, checkerboard and Perlin noise [21];
- Manipulate the lighting conditions of the scene;
- Capture images with a virtual camera, possibly with added noise;
- Obtain 2D bounding boxes from the 3D scene in order to label data for an object detection task. This is achieved by sampling points on the object surface and projecting them onto the camera plane.

The variation of the object textures and position has been observed in previous work and is considered to be part of the basic randomisation process.

The spawned objects can be parametric basic shapes, such as spheres, cylinders or cuboids as well as described by their Simulation Description Files (SDF)[6] specification or unique identifier, provided they are included in the Gazebo model path, and their pose can be fully specified.

It was also in our interest to be able to specify custom textures for each object during run-time, as we have seen that most related research randomises colour and texture in scene objects.

Gazebo employs a distributed architecture, providing an abstraction layer between the physics and rendering engines as well as sensor data generation. This in turn led us to divide our plugin in three modules, tasked respectively with changing objects' visual properties, interacting with the main world instance and finally querying camera sensors. We aimed for a modular design, so that each tool could be used independently. Thus, each module provides an interface for receiving requests and replying with feedback messages. The latter can be leveraged by a client application to create a synchronous scene generation pipeline, despite the program's distributed nature.

[6]http://sdformat.org/, as of April 11, 2018

IV. Experiments

We trained state-of-the-art CNNs for three-class object detection task using a synthetic dataset generated with the proposed domain randomisation Gazebo plugin. We have validated these methods by designing an experiment that replicates some of the elements seen in recently published research.

A. Synthetic Data Synthesis

Each generated scene consists of a ground plane, a single light source, a set of models which include cylinders, spheres or cuboids. The position of these models is constrained to a grid, but they can rotate freely, provided they are supported by the ground plane. Partial occlusion is allowed, but the grid approach prevents collisions. An example of generated scenes, overlaid with the respective annotations can be seen in Figure 2. Even though the physics engine is explicitly deactivated, the object pose should be that of a static scene, i.e. objects should be in a stable resting position. The scene generation loop procedure unfolds as follows.

First, we spawn a camera and a random number of parametric objects $N_o \in [5, 10]$ belonging to one of the following three classes: cylinder, sphere, box. The position of each object is randomly chosen from 25 possible cells in a 5 m $\times$ 5 m grid. We perform a request to move the light source and camera to a random position, where the camera is constrained to move on a dome surface, with its centre directly above the world origin, so as to ensure that the objects are in the camera's field of view.

Then, obtain the 2D bounding boxes of each object. In order to achieve this, the proposed plugin is first queried for the 3D points sampled from the primitive shape, which are further projected into the 2D image plane. Finally, the bounding box is computed via min-max over the resulting set of image projected points. At this point we save the scene annotations with the object class and 2D bounding boxes.

B. Results

With the aforementioned set of tools it is already possible to perform some experiments with domain randomisation, namely for object detection and localisation.

In order to demonstrate the applicability of the proposed Gazebo domain randomisation plugin, we designed the following experiment. First we generated over 5000 textures per pattern type. Then, we ran our example client, acquiring exactly 9,000 images of randomly generated scenes with Full-HD resolution (1920×1080). The texture generation took little over 20 minutes, but is heterogeneous with regard to required time, as Perlin noise is more computationally intensive to generate. Each simulated scene can be generated in roughly a second on a standard laptop, with an Intel® i7 4712HQ CPU and an Nvidia® 940m GPU. This means that the full dataset generation process should take less than 3 hours. The images were then cropped to 1080×1080, subsampled to 300×300 and then fed to two separate state of the art networks for training, namely SSD and Faster R-CNN. We resorted to an open-source Tensor-Flow[7] GPU-accelerated implementation of these networks available on GitHub[8]. The networks were pretrained on COCO [13] and fine-tuned (i.e. training only the last layer of the network over 5000 epochs) with our datasets, comprised by renders from Gazebo and real world images. Both networks output a classification label for each detected object instance as well as the respective bounding box. We tested the trained networks with a small dataset of 242 real-world images acquired in our laboratory that contain two overlapping groups of objects.

We performed several experiments summarised in Table I, namely including the real world images in training (121 images for training, 121 for testing and then swapping the real world images) with random horizontal flips. The networks are trained to predict the object class and respective bounding box in the original image. An ensemble of example output is presented in Figure 3. 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, and (iii) the most difficult object to detect and localise is the cylinder class. Since the number of simulated images is not large enough to consider different lighting conditions, training with the randomised dataset alone is not able to reach the desired performance on real-world data.

V. Conclusions

In this work we developed a domain randomisation plugin for Gazebo that allows generating large datasets in simulation, which are suitable for training deep object detection and localisation architectures. The plugin was used to generate a fully annotated dataset in a format compatible with the PASCAL VOC 2012 annotation format, and allows to deploy the classification of new object classes in a straightforward and easy to program way. Our results in the three object detection and localisation task are very promising, because with a small dataset generated only in simulation, we are very close to reach the performance of real-world images.

Our code is open-source and publicly available on GitHub[9]. To our knowledge, such a tool was either non-existent or publicly unavailable at the time of development.

Acknowledgements

This work has been partially supported by the Portuguese Foundation for Science and Technology (FCT) project [UID/EEA/50009/2013]. Rui Figueiredo and Atabak Dehban are funded by FCT PhD grant PD/BD/105779/2014 and PD/BD/105776/2014, respectively.

[7] https://www.tensorflow.org/, as of April 11, 2018
[8] https://github.com/tensorflow/models, as of April 11, 2018
[9] https://github.com/jsbruglie/gap, as of April 11, 2018

Network	Simulated	Real	IoU	mAP	AP Box	AP Cylinder	AP Sphere
Faster R-CNN	0	121	0.5	**0.88**	**0.89**	**0.83**	**0.91**
Faster R-CNN	9000	0	0.5	0.82	0.8	0.75	0.90
Faster R-CNN	9000	121	0.5	0.83	0.82	0.78	0.9
SSD	0	121	0.5	**0.7**	**0.74**	**0.6**	0.76
SSD	9000	0	0.5	0.64	0.7	0.47	0.77
SSD	9000	121	0.5	0.62	0.67	0.4	**0.8**

TABLE I

EXPERIMENT RESULTS. SECOND AND THIRD COLUMN STATE THE NUMBER OF TRAINING EXAMPLES FROM EACH DATASET. IOU IS THE INTERSECTION OVER UNION THRESHOLD. THE OVERALL MAP METRIC RESULT IS PRESENTED AS WELL AS FOR EACH CLASS. NOTE THAT FASTER R-CNN IS CONSIDERABLY SLOWER TO TRAIN.

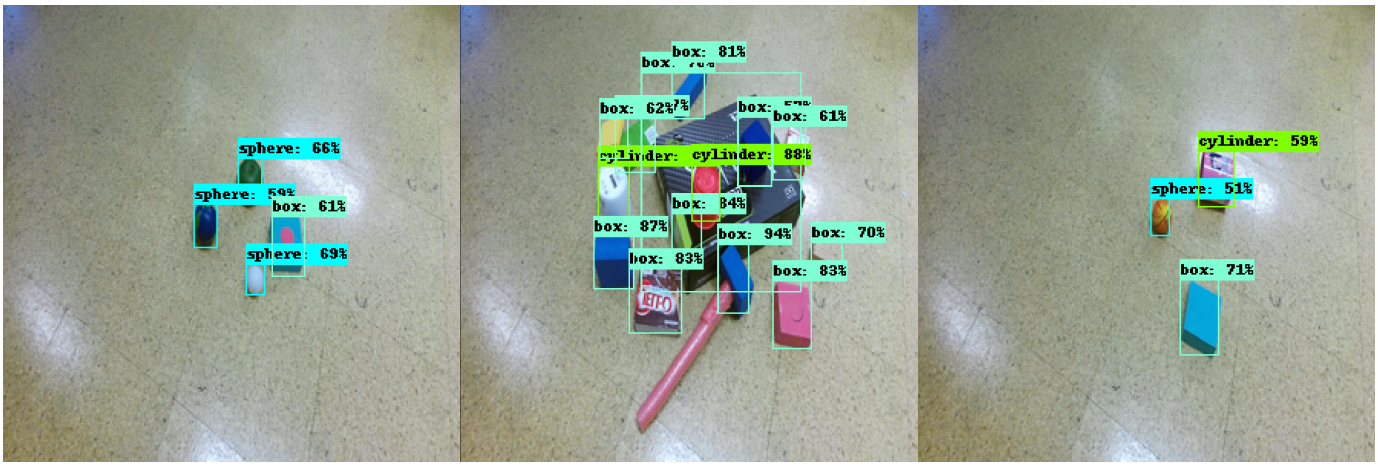

Fig. 3. Example object detections on our real dataset

REFERENCES

[1] R. Johansson and J. Flanagan, "Tactile sensory control of object manipulation in humans," *The Senses: A Comprehensive Reference, Academic Press, New York, NY*, vol. 6, pp. 67–86, 01 2008.

[2] N. Jakobi, "Evolutionary robotics and the radical envelope-of-noise hypothesis," *Adaptive Behavior*, vol. 6, no. 2, pp. 325–368, 1997. [Online]. Available: https://doi.org/10.1177/105971239700600205

[3] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," in *Conference on Robot Learning*, 2017, pp. 334–343.

[4] A. Dehban, L. Jamone, A. R. Kampff, and J. Santos-Victor, "A deep probabilistic framework for heterogeneous self-supervised learning of affordances," in *IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 2017, pp. 476–483.

[5] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 2004, pp. 2149–2154.

[6] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," 2012. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*. Washington, DC, USA: IEEE Computer Society, 2014, pp. 580–587.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[10] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, Sep 2013. [Online]. Available: https://doi.org/10.1007/s11263-013-0620-5

[11] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015.

[12] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[13] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015.

[15] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[16] 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," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 23–30.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[18] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[19] J. Tobin, W. Zaremba, and P. Abbeel, "Domain randomization and generative models for robotic grasping," *CoRR*, vol. abs/1710.06425, 2017.

[20] 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*, vol. abs/1512.03012, 2015.

[21] K. Perlin, "Improving noise," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 681–682, Jul. 2002. [Online]. Available: http://doi.acm.org/10.1145/566654.566636