1703.06907 - Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World

- **Yunqiu Xu**
- Brief review:
    - Work about "Sim-to-real transfer"
    - Train on similated images and then transfer to real images by randomizing rendering in the simulator
    - Domain randomization: colors, textures, lighting conditions, and camera settings in simulated scenes
    - To some extent this is more like robot controling, and the example is about object localization and grasping
- Other reference and further work:
    - OPENAI - Spam Detection in the Physical World
    - OPENAI - Robots that Learn：One-Shot Imitation Learning
    - OPENAI - Generalizing from Simulation
        - Dynamics Randomization
        - Image-Based Learning

---

1. Introduction

- Why learning in simulation:
    - DRL employs random exploration, which can be dangerous on physical hardware
    - It's impractical to collect millions of real-world samples
- Similar to "imitation learning" and "transfer learning", we can learn policies for complex behaviors in simulation, then transfer policies to adapt real environment
- Challenges: reality gap

- System identification is time-consuming and error-prone
- Unmodeled physical effects of real world
- Low-fidelity simulated sensors are insufficient

2. Method

- Goal: Given some objects of interest $\{s_i\}_i$, train an object detector $d(I_0)$ that maps a single monocular camera frame I_0 to the Cartesian coordinates $\{(x_i, y_i, z_i)\}_i$ of each object
- Approach: train a deep neural network in simulation using domain randomization
- Domain randomization:
  - Goal: provide enough simulated variability at training time such that at test time the model is able to generalize to real-world data
  - If the variability in simulation is significant enough, models trained in simulation will generalize to the real world with no additional training
  - In this work, we only focus on the task of training a neural network to detect the location of an object
- The architecture of model is like VGG-16:

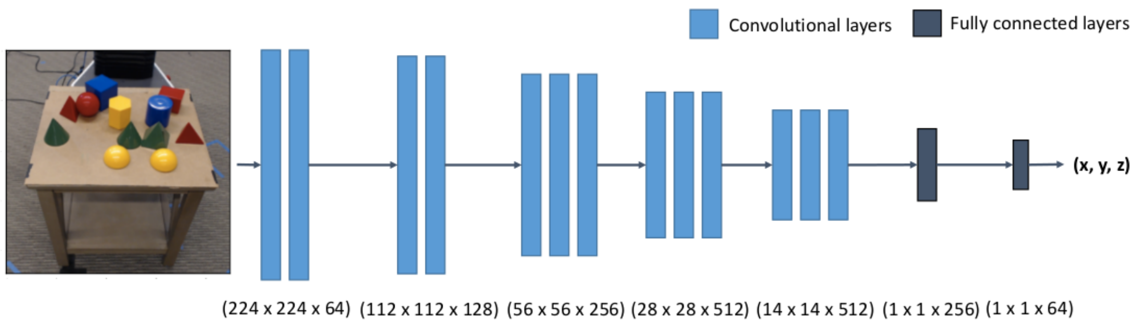
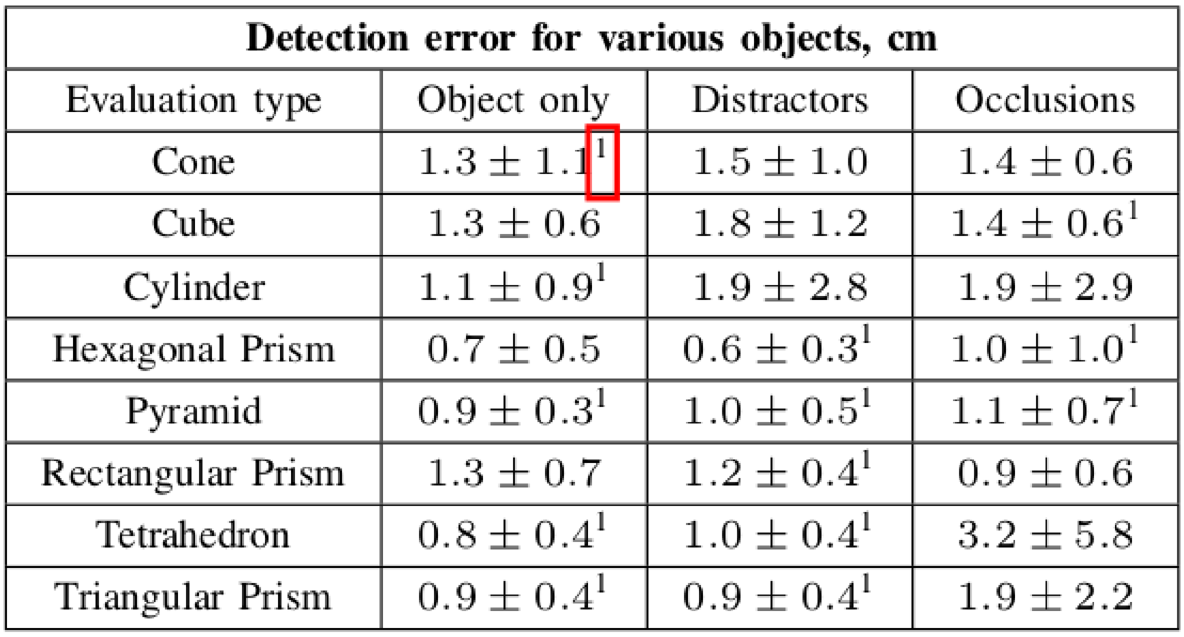


Fig. 2. The model architecture used in our experiments. Each vertical bar corresponds to a layer of the model. ReLU nonlinearities are used throughout, and max pooling occurs between each of the groupings of convolutional layers. The input is an image from an external webcam downsized to (224×224) and the output of the network predicts the (x, y, z) coordinates of object(s) of interest.

3. Experiment

- Localization accuracy of detectors in real world:

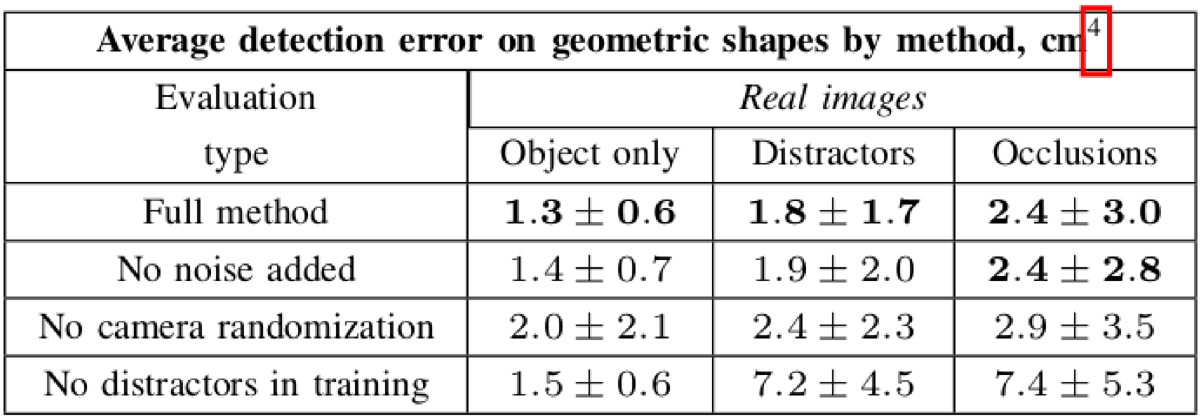
TABLE I



Detection error for various objects, cm			
Evaluation type	Object only	Distractors	Occlusions
Cone	1.3 ± 1.1[1]	1.5 ± 1.0	1.4 ± 0.6
Cube	1.3 ± 0.6	1.8 ± 1.2	1.4 ± 0.6[1]
Cylinder	1.1 ± 0.9[1]	1.9 ± 2.8	1.9 ± 2.9
Hexagonal Prism	0.7 ± 0.5	0.6 ± 0.3[1]	1.0 ± 1.0[1]
Pyramid	0.9 ± 0.3[1]	1.0 ± 0.5[1]	1.1 ± 0.7[1]
Rectangular Prism	1.3 ± 0.7	1.2 ± 0.4[1]	0.9 ± 0.6
Tetrahedron	0.8 ± 0.4[1]	1.0 ± 0.4[1]	3.2 ± 5.8
Triangular Prism	0.9 ± 0.4[1]	0.9 ± 0.4[1]	1.9 ± 2.2

  - Localize objects to within 1.5cm (on average) in the real world
  - Perform well in the presence of clutter and partial occlusions
  - Still over-fitting the simulated training data, but comparable with traditional technique on higher-resolution images
- Ablation study, assess the sensitivity of some elements

TABLE II



Average detection error on geometric shapes by method, cm[4]			
Evaluation type	*Real images*		
	Object only	Distractors	Occlusions
Full method	$\mathbf{1.3 \pm 0.6}$	$\mathbf{1.8 \pm 1.7}$	$\mathbf{2.4 \pm 3.0}$
No noise added	1.4 ± 0.7	1.9 ± 2.0	$\mathbf{2.4 \pm 2.8}$
No camera randomization	2.0 ± 2.1	2.4 ± 2.3	2.9 ± 3.5
No distractors in training	1.5 ± 0.6	7.2 ± 4.5	7.4 ± 5.3

- Incorporating distractors during training: critical
- Randomizing the position of the camera: not critical
- Adding noise: not critical but helpful to avoid local optima