Physics-aware Self-supervised Training of CNNs for Object Detection

Chaitanya Mitash, Kun Wang, Kostas E. Bekris and Abdeslam Boularias

Abstract—Impressive progress has been achieved recently in object detection with the use of deep learning. Nevertheless, such tools typically require large amounts of training data and significant manual effort for labeling objects. This limits their applicability in realistic domains, where it is necessary to scale solutions to a large number of objects and a variety of conditions. This work proposes a training process for Convolutional Neural Networks (CNNs), which minimizes manual effort and takes advantage of physical reasoning through the use of a physics-engine. The application involves detection of objects placed in cluttered scenes and tight environments, such as shelves. In particular, given access to 3D object models, the labeling takes place automatically over synthetic images, which correspond to physically realistic object poses. To further improve object detection, the network self-trains over real images that contain the target objects. In these situations, successful detections can be used to provide additional real-world examples. Then, the trained network can be successfully applied in complicated scenes, which involve clutter and object occlusions, without manual labeling. The proposed training process is evaluated by showing that it can match the accuracy of a training process based on manual labeling and can improve over time through self-training resulting in a lifelong learning solution.

I. INTRODUCTION

Object detection is frequently the initial step of robot manipulation. Recently, deep learning methods, such as those employing Convolutional Neural Networks (CNNs), have become popular for object detection after outperforming alternatives in benchmarks for object recognition tasks [1] [2]. These desirable results are typically obtained by training CNNs using datasets that involve a very large number of labeled images (e.g., ImageNet [3] and MS COCO [4]). Creating such large datasets requires intensive human labor. Furthermore, as these datasets are general-purpose, one needs to create new datasets for specific object categories. For instance, datasets that have been generated for object detection in tabletop environments may not be as useful for detecting objects inside shelves, where shadows and lighting conditions are significantly different.

The authors are with the Computer Science Department of Rutgers University in Piscataway, New Jersey, 08854, USA. Email: {cm1074,kw423,kb572,ab1544}@rutgers.edu

The recent Amazon Picking Challenge (APC) has reinforced this realization and has led into the development of datasets specifically for the detection of objects inside shelving units [5]. When state-of-the-art deep learning methods are used for such applications, where just a small number of known objects is involved, a significant effort is still required to add labeled examples to account for shape and texture variations of the given objects. The challenge is further complicated by the presence of clutter and occlusions among objects. This added complexity increases the number of examples needed for training CNNs to achieve a detection accuracy that is good enough for grasping and manipulation.

An increasingly popular approach to avoid manual labeling is to use synthetic datasets generated by rendering 3D CAD models of objects with different viewpoints. Synthetic datasets have been used to train CNNs for object detection [6] and view point estimation [7]. One major challenge in using synthetic data is the difference between virtual training examples and real testing data.

Fig. 1: This work proposes the use of a physics engine to generate realistic training data for object detection. (top) Poses of objects returned by the physics engine and a rendered synthetic image. (bottom) Examples of object detections on a real image.
There is considerable interest in studying the impact of texture, lighting, and shape to address this disparity [8]. One issue with synthetic images generated from rendering engines is that they display objects in poses that are not necessarily physically realistic. Moreover, occlusions are usually treated in a rather naive manner, i.e., by applying cropping, or pasting rectangular patches, which again results in unrealistic scenes [6] [7] [9].

This work proposes an automated system for generating and labeling datasets for learning to detect objects in real images. The objective of the proposed system is to minimize manual labor and incorporate physical reasoning in the data generation process. In particular, a distinct feature of the proposed system is the use of a physics engine, which simulates physical parameters, such as gravity, interactions between different objects and with their environment, to automatically generate a large number of realistic scenes based on given models of the objects. A rendering engine is also used to simulate varying lighting conditions. The accompanying experiments are making use of Blender [10] for rendering and Bullet [11] for physics simulation.

Using the images of the automatically generated scenes, a CNN object detector is trained. The object detector is improved during an automated, self-supervised learning process, by continuously taking new real images, detecting and labeling objects in the images, and adding the images with labeled objects to the training dataset. The new examples provide more diversity in terms of texture, lighting and object poses from real world scenes compared to synthetic images. The bounding boxes of detected objects in the new examples are made more accurate by automatically segmenting the objects using depth information and resizing the bounding boxes, such that the objects do not collide with each other. This self-supervised learning process is possible because of a key observation: if the system can detect and segment out particular objects with high accuracy, then the bounding boxes of other objects in the scene can be made more accurate by resizing them so that they do not collide. For instance, detecting a table with high accuracy helps detecting the bounding boxes of objects on top of it, even when these objects are little known to the system.

The CNN trained on the physics-aware dataset outperforms the one generated with a popular rendering engine. Moreover, by self-training the detector using results from its own detection, it is possible to autonomously improve its performance.

To demonstrate the utility of the proposed training process in the context of a manipulation problem, such as the one in the Amazon Picking Challenge, the simulator is first provided 3D models of the objects and the shelf bin used for the APC. The simulator generates a large, synthetic, labeled dataset autonomously. It does so by randomly choosing a set of objects and initial poses for them above a resting surface. Then, it simulates physics until the objects rest and extracts the final image of the cluttered shelf unit using realistic poses for the objects. The next step is to train, detect, and fine-tune a real-time CNN-based detector [12]. For evaluation purposes, the performance of the proposed training process is compared against alternative processes, such as collecting a lot of examples of real-images and manually labeling them. This work also studies the impact of several simulation parameters on the precision of the object detection process.

II. Related Work

Object Detection: Deep learning has become the method of choice in computer vision since its success in the ImageNet contest [2]. This was then adapted for object detection by combining the region proposals with features calculated using CNNs [13], [12], [14], [1]. We use the Faster R-CNN method [12] to evaluate our system, as it runs in real-time and offers state-of-the-art results in object detection.

Synthetic Data: Popular 2D [4] [15] and 3D [16] datasets have been used as input for training the CNNs but they are limited to specific object categories. Thus, in setups like the APC, to avoid manual labeling, one option is to render 3D CAD models and generate synthetic images. Recently, many researchers have taken synthetic images as input for training CNNs. Some have proposed synthetic data to study the impact of texture, background and lighting [6] [9]. Others used 3D models to generate images for viewpoint estimation [7]. All these works render objects and create scenes in the image space without considering physics. In terms of applications, there is a large variety, ranging from pedestrian detection [17] to urban scene understanding [18] and indoor scene detection [19]. This work considers a 3D warehouse picking setup.

Cluttered Scenes: To mimic occlusions, some previous works used cropped images from a dataset [7], [9]. These methods require considerable efforts, as one needs to verify that the resulting images are realistic. We avoid this problem by using a physics-based simulation.

III. System Design

Figure 2 illustrates the pipeline of the automated process of collecting as well as labeling data, and then
Given 3D models of objects, randomly select some for each scene. Simulate gravity/collisions; wait for stabilization. Extract bounding boxes from the final poses of objects. Train Faster R-CNN detector[12].

(a) Offline training in simulation using images generated with a physics engine.

Take real RGB-D images of the objects in clutter. Detect objects using Faster R-CNN. Improve detection using depth data. Add images to training set. Re-train Faster R-CNN[12].

(b) Lifelong self-improvement using real images.

Fig. 2: Overview of the proposed system.

Training a CNN to detect objects. This process is divided into two parts:

(a) An offline process, which utilizes 3D models of objects and a physics-engine to generate physically-realistic images of multiple objects in a predefined environment; and

(b) an online lifelong process, where the robot collects real-world images of objects, self-labels them using an accompanying method, and adds them to the training data set.

The offline simulation pipeline is shown in Figure 2(a). It starts with loading 3D models of objects and an environment (Section III-A). This is followed by setting up the camera location and its intrinsic parameters, as well as lighting conditions in the simulator, which can have a significant impact in object detection (Section III-B). The objects are initially inserted in random free-flying positions in space, typically above the resting surface. The objects may be colliding in this configuration. Then, the physics engine is used to simulate their motion caused by gravity as well as collisions (Section III-C). The resulting poses after this process are physically realistic in the sense that no penetration exists among the objects and the environment. The resulting 3D scenes are rendered into images, which can be automatically labeled and then used to train the CNN (Section III-D).

The lifelong learning pipeline is illustrated in Figure 2(b), where the CNN is used to detect objects in the real world. The detected bounding boxes are then corrected after segmenting the objects using depth information, and used as labels for retraining the network and improving its detection ratio (Section III-E).

A. 3D Models

The proposed system needs access to 3D models of the objects to be detected. For evaluating the training process, 3D models of 15 objects used in the Amazon Picking Challenge (APC) 2016 were generated. For those objects with simple geometries, it was possible to simply attach texture to different faces of the mesh in the Blender software [10]. For complex mesh objects, the AutoDesk123D software [20] can use as input, images of the objects from different viewpoints to return a raw mesh model, which can then be further fine-tuned in...
A model of the environment is also needed, which for the accompanying evaluation corresponded to a shelf bin with texture similar to the real bin used in APC (Figure 1). All these models are imported in Blender, where the shelf bin is placed at the origin. Then a random selection process takes place for choosing the number of objects and which ones are to be placed in each scene. The initial positions and orientations of each of these objects are sampled uniformly inside the shelf. To demonstrate the adaptability of the system to different conditions, a tabletop environment was also considered.

B. Camera and lighting

To generate synthetic images, the objects are placed in the Blender software, where a perspective camera is defined. Blender requires the sensor width and focal length of the camera, which are calculated from the intrinsic $k - matrix$ of a Kinect sensor.

$$k = \begin{bmatrix} \alpha_u & 0 & u_o \\ 0 & \alpha_v & v_o \end{bmatrix}$$

where $\alpha_u$ is focal length divided by the width of a pixel in millimeters, $\alpha_v$ is focal length divided by the height of a pixel in millimeters, $(u_o, v_o)$ are the pixel coordinates of the principal point of the camera.

A point light source is used to control lighting within the shelf. The location of the source is sampled in front of the shelf and facing towards it. For each scene, a random value of energy is selected, and a light color from the following real-world light colors is randomly chosen so as to provide some robustness in diverse lighting conditions: 40w Tungsten, 100w Tungsten, Halogen, Carbon Arc, High Noon Sun, and Direct Sunlight [21].

C. Simulation and rendering

Physics-based simulation is achieved with the aid of the Bullet engine [11], which is integrated with Blender for visualization purposes. Bullet allows the simulation of rigid body motions by the objects and their interactions with their surroundings (e.g., an APC shelf) once they have been placed randomly inside the workspace. Once placed, the objects fall due to gravity, bounce, and collide with each other and with the shelf. Any inter-penetrations among objects or with the shelf are appropriately treated by the physics engine. The final poses of the objects, when they all stabilize, resemble real-world poses and do not involve. Gravity, friction coefficients and mass parameters can be set at similar values globally in order to promote fast stabilization of the scene. The final poses of the objects are primarily depending on their initial placement and in this way the physical parameters of the simulation do not require manual tuning. This final frame of the animation is rendered in the camera viewpoint. The result is a 2D RGB image of a physically-realistic pile of objects.

D. Extracting 2D bounding boxes

It is then possible to apply perspective projection on the synthetic images, given the objects’ final poses and the camera parameters, to obtain 2D bounding boxes for each object in an autonomous way. Very frequently occlusions will occur between objects in cluttered scenes. To account for occlusions, each pair of bounding boxes is tested on whether an overlap occurs. The overlapping portion of the bounding box for the object that is further away from the camera is automatically pruned.

E. Self-supervised learning

A Region-based Convolutional Network (R-CNN) [12] is then trained for object detection using the rendered synthetic images and the automatically generated bounding boxes. The resulting R-CNN model achieves a reasonable success rate in detecting objects. But the synthetic images may not provide adequate information about the visual conditions in the real world and some adaptation process may be necessary. To deal with this issue, a two-step online process is used for lifelong self-learning, which allows adaptation to various lighting and clutter conditions.

The first step for self-learning is to expose the network to real images of objects. In this step, each of the object is demonstrated in a simple setup, where detection success based on synthetic data is high. The bounding box returned from this detection is then fine-tuned using depth information from an RGB-D sensor. Given depth data it is possible to eliminate the part of bounding box that corresponds to background and include the part that corresponds to the object. To achieve this, the process calculates the depth histogram inside the bounding box to identify the most likely set of points belonging to the object by using the most frequently appearing depth value. Given this initial set of points, the object’s label propagates to its neighbors, which lie within a threshold distance and adds them to the set. These auto-corrected bounding boxes and the demonstrated scenes are added to the training dataset and the R-CNN is re-trained. This step helps the detector collect information regarding the real texture of the object under natural light conditions and helps to avoid overfitting to the textures used in the 3D models.

1 The models generated in this manner outperformed models released by a team participating in the APC (Team PFN) in 13 out of 15 cases. These models will be provided to the community as part of this work.
At this point, the detector has already overcome the dataset bias and is tuned to perform equivalently to a detector trained with real scenes and manual labels. Then, it enters into an automated, lifelong learning phase where it keeps on adding more examples from the environment as it performs detection on given examples of objects in clutter. In particular, given a threshold confidence value for the detection of an object by the network (0.8 in the presented experiments), and without any other external information, the process tries to detect all objects in the scene and adds the labels of detected objects with confidence greater than the threshold to its dataset. Further improvement can be achieved by reasoning about bounding boxes that overlap. Overlapping regions between pairs of bounding boxes are classified again using the same network to confirm the presence of one object or the other and in this way providing tighter detection (Figure 3). Collecting more examples of real world scenes with clutter allows the CNN to improve and adapt its performance in varying conditions.

IV. Evaluation Process

In order to test the performance of CNNs trained under alternative conditions, a test set was generated containing real images that correspond to 150 different scenes. Each scene contains five different objects placed in an Amazon Picking Challenge shelf. These five objects were randomly selected from a list of 15 objects total, meaning that each object appears on average in 50 scenes. For each scene, test RGB-D images were collected under four different lighting conditions: a) background light only, b) background light + white LED bulb light, c) background light + yellow light bulb, d) background light + white LED bulb light + yellow light bulb. Therefore, the test set contains a total of 600 manually labeled real images, providing 200 diverse data points per object on average.

The performance of the proposed PHYSIM training process for the CNN-based detector, which makes use of the physics-based simulator, was compared against the following popular alternatives:

1) **MANUAL**: This is ubiquitous but labor-intensive method of capturing images of real scenes and manually labeling bounding boxes indicating the presence of an object. For this purpose, a training dataset of 451 real scenes was created, each with 5 randomly selected objects, and which included examples with all four lighting conditions. This provides approximately 125 labeled examples for each object in different poses and with different levels of occlusions. This method forms the baseline as it tends to be a robust approach for dataset generation and does not suffer from dataset bias. It is also the standard approach used for training CNNs for object detection.

2) **SYNTH**: This process involves the collection of a few real images for each object in different realistic poses and then the use of the popular GrabCut algorithm [22] to generate fine masks for these objects on the background. These masks, taken from real images, are then used to artificially generate a large number of images of clutter. Specifically, we sample random positions on a background image, which corresponds to an image of the APC shelf, and paste the masks of multiple parts of different objects.

3) **SIM**: This alternative uses 3D models of the objects to render their images from several viewpoints sampled on a spherical surface centered at the object. The rendering process took place under varying lighting conditions to provide again a diverse set of examples. The background image corresponded to the APC shelf, on top of which randomly selected objects were pasted at sampled locations. This process allows to simulate occlusions and mask subtraction provides the accurate bounding boxes in these cases. Similarly to PHYSIM, this alternative does not require manual labeling effort. It is also similar to SYNTH in the way the clutter images are artificially generated as mosaics of object parts (masks), the difference being that SYNTH uses object parts taken from real images. Unlike in PHYSIM, objects in images generated by SIM or SYNTH are not guaranteed to have physically realistic poses.

To compare the performance of the detectors, the following evaluation metrics were used:

- **Overlap area**: The detected bounding box $b_d$ is
compared with the ground truth bounding box \( b_g \) in terms of how much they overlap:

\[
A = \frac{\text{area}(b_d \cap b_g)}{\text{area}(b_d \cup b_g)}
\]

- **Success Ratio**: A successful detection corresponds to the case that the area of overlap calculated above is greater than 50%. Then, the mean success ratio is used to compare results among methods:

\[
S_c = \frac{1}{|c|} \sum_{i \in c} f(A_i), \quad f(A_i) = \begin{cases} 
0 & : A_i < 0.5 \\
1 & : A_i \geq 0.5
\end{cases}
\]

where \( c \) is a class of objects, and \(|c|\) is the number of objects of class \( c \) in the images. We also report \( \hat{S} \) the mean success ratio over all classes.

V. RESULTS

Tables I and II provide a comparison of the performance of the R-CNN detector for the four different training procedures, i.e., MANUAL, SYNTH, SIM and the proposed PHYSIM as described in Section IV. Table I summarizes the results for the area of overlap for each object, and Table II reports detection success ratios. In this first experiment, only the offline training part (the simulation part) of PHYSIM was used. In the training procedure MANUAL, we trained the R-CNN with 451 manually labeled images, which required about 10 hours of human labeling effort. In SYNTH, SIM, and PHYSIM, we used 30000 automatically labeled images.

The proposed training process PHYSIM with a mean success rate of 73.3% outperforms the other synthetic datasets. SYNTH has a mean success rate of 71.3%, which is close to the result of PHYSIM, but still requires some manual effort for creating fine masks for each object. As expected, MANUAL achieves the highest success rate, 88.5%, because it uses real images.

The proposed training process PHYSIM with an average overlay of 59.0% also outperforms the other synthetic datasets. MANUAL still achieves the highest performance, 70.5%, because it uses real images. Note that the performance of PHYSIM is closer to that of MANUAL using the overlay measure instead of the success ratio. The reason is that in most failed detections using PHYSIM, the overlay was barely beneath the 50 threshold used to count a detection as a success. Reducing this threshold would increase the success rate.
Moreover, evaluating object detection methods is particularly difficult when the objects are partially occluded because there are multiple bounding boxes that can be used as ground truth for evaluating the accuracy of the detection.

The result from the training phase using synthetic only data is still below the performance of the baseline alternative MANUAL, which includes manually labeled real images. This can be attributed to the problem of dataset bias, i.e. the unavoidable difference between synthetic and real images. This bias is the only reason why SYNTH outperforms SIM, because the only difference between the two methods is that one uses real images and the other uses rendered images.

Consequently, synthetic images alone are not sufficient. This issue is addressed by the second step in the proposed pipeline, i.e., lifelong online self-training. Tables III and IV show the results using additional training examples that were automatically labeled, without any human intervention, using PHYSIM. The proposed

<table>
<thead>
<tr>
<th>method</th>
<th>crayola expo</th>
<th>folgers</th>
<th>tape</th>
<th>bottle</th>
<th>dvd</th>
<th>glucose</th>
<th>book</th>
<th>bulb</th>
<th>tissue</th>
<th>pencils</th>
<th>soap</th>
<th>brush</th>
<th>glue</th>
<th>ball</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>MANUAL</td>
<td>67.4</td>
<td>66.1</td>
<td>77.6</td>
<td>73.0</td>
<td>70.0</td>
<td>59.7</td>
<td>74.5</td>
<td>68.0</td>
<td>75.4</td>
<td>72.2</td>
<td>61.7</td>
<td>74.3</td>
<td>67.2</td>
<td>74.7</td>
<td>76.6</td>
</tr>
<tr>
<td>SYNTH</td>
<td>65.1</td>
<td>52.1</td>
<td>66.0</td>
<td>55.3</td>
<td>57.8</td>
<td>40.8</td>
<td>61.2</td>
<td>59.3</td>
<td>60.9</td>
<td>61.3</td>
<td>52.8</td>
<td>57.0</td>
<td>58.7</td>
<td>64.0</td>
<td>70.6</td>
</tr>
<tr>
<td>SIM</td>
<td>48.6</td>
<td>40.3</td>
<td>42.3</td>
<td>38.9</td>
<td>30.9</td>
<td>30.3</td>
<td>45.3</td>
<td>43.6</td>
<td>47.1</td>
<td>55.2</td>
<td>33.9</td>
<td>45.9</td>
<td>43.2</td>
<td>23.7</td>
<td>46.8</td>
</tr>
<tr>
<td>PHYSIM</td>
<td>60.4</td>
<td>59.3</td>
<td>64.4</td>
<td>62.8</td>
<td>55.9</td>
<td>47.5</td>
<td>69.3</td>
<td>51</td>
<td>67.2</td>
<td>70.4</td>
<td>45.3</td>
<td>48.2</td>
<td>62.2</td>
<td>50.9</td>
<td>71.2</td>
</tr>
</tbody>
</table>

| method | crayola expo | folgers | tape | bottle | dvd | glucose | book | bulb | tissue | pencils | soap | glue | ball | mean |
|--------|--------------|---------|------|--------|-----|---------|------|------|--------|--------|------|-------|------|------|------|
| MANUAL | 87.5         | 90.7    | 90.9 | 94.0   | 88.0| 76.0    | 94.9 | 84.5 | 98.4   | 82.4   | 74   | 93.1  | 80.2 | 90.5 | 94.6 |
| SYNTH  | 82.5         | 58.7    | 80.8 | 68.0   | 72.9| 42.1    | 78.0 | 72.5 | 78.7   | 68.1   | 61.5 | 70.6  | 64.5 | 79.0 | 92.0 |
| SIM    | 58           | 41.2    | 39.3 | 40     | 20.3| 25      | 49.4 | 52   | 52.6   | 62.2   | 33   | 53.2  | 36.9 | 25.5 | 57.4 |
| PHYSIM | 76           | 75.6    | 80.3 | 83.5   | 67.7| 50.5    | 89.8 | 61.5 | 83.5   | 83.5   | 52.5 | 66.3  | 72.4 | 63.0 | 93.0 |

| Method | crayola expo | folgers | tape | bottle | dvd | glucose | book | bulb | tissue | pencils | soap | glue | ball | mean |
|--------|--------------|---------|------|--------|-----|---------|------|------|--------|--------|------|-------|------|------|------|
| MANUAL | 67.4         | 66.1    | 77.6 | 73.0   | 70.0| 59.7    | 74.5 | 68.0 | 75.4   | 72.2   | 61.7 | 74.3  | 67.2 | 74.7 | 76.6 |
| PHYSIM | 60.4         | 59.3    | 64.4 | 62.8   | 55.9| 47.5    | 69.3 | 51   | 67.2   | 70.4   | 45.3 | 48.2  | 62.2 | 50.9 | 71.2 |

| Method | crayola expo | folgers | tape | bottle | dvd | glucose | book | bulb | tissue | pencils | soap | glue | ball | mean |
|--------|--------------|---------|------|--------|-----|---------|------|------|--------|--------|------|-------|------|------|------|
| MANUAL | 87.5         | 90.6    | 90.9 | 94     | 94.9| 84.5    | 98.4 | 82.4 | 95.1   | 96.5   | 94.6 | 91.7  | 94.6 | 91.7 | 77.8 |
| PHYSIM | 76           | 75.6    | 80.3 | 83.5   | 89.8| 61.5    | 83.5 | 83.5 | 66.3   | 63.0   | 93.0 | 77.8  | 93.0 | 88.7 | 88.7 |

| Method | crayola expo | folgers | tape | bottle | dvd | glucose | book | bulb | tissue | pencils | soap | glue | ball | mean |
|--------|--------------|---------|------|--------|-----|---------|------|------|--------|--------|------|-------|------|------|------|
| MANUAL | 87.5         | 90.6    | 90.9 | 94     | 94.9| 84.5    | 98.4 | 82.4 | 95.1   | 96.5   | 94.6 | 91.7  | 94.6 | 91.7 | 77.8 |
| PHYSIM | 76           | 75.6    | 80.3 | 83.5   | 89.8| 61.5    | 83.5 | 83.5 | 66.3   | 63.0   | 93.0 | 77.8  | 93.0 | 88.7 | 88.7 |

| Method | crayola expo | folgers | tape | bottle | dvd | glucose | book | bulb | tissue | pencils | soap | glue | ball | mean |
|--------|--------------|---------|------|--------|-----|---------|------|------|--------|--------|------|-------|------|------|------|
| MANUAL | 87.5         | 90.6    | 90.9 | 94     | 94.9| 84.5    | 98.4 | 82.4 | 95.1   | 96.5   | 94.6 | 91.7  | 94.6 | 91.7 | 77.8 |
| PHYSIM | 76           | 75.6    | 80.3 | 83.5   | 89.8| 61.5    | 83.5 | 83.5 | 66.3   | 63.0   | 93.0 | 77.8  | 93.0 | 88.7 | 88.7 |

Fig. 6: The success rate of detecting different classes of objects using different training procedures

TABLE III: Bridging the gap in overlay achieved with manually labeled real scenes and the proposed PHYSIM by using self learning technique

| Method | crayola expo | folgers | tape | bottle | dvd | glucose | book | bulb | tissue | pencils | soap | glue | ball | mean |
|--------|--------------|---------|------|--------|-----|---------|------|------|--------|--------|------|-------|------|------|------|
| MANUAL | 87.5         | 90.6    | 90.9 | 94     | 94.9| 84.5    | 98.4 | 82.4 | 95.1   | 96.5   | 94.6 | 91.7  | 94.6 | 91.7 | 77.8 |
| PHYSIM | 76           | 75.6    | 80.3 | 83.5   | 89.8| 61.5    | 83.5 | 83.5 | 66.3   | 63.0   | 93.0 | 77.8  | 93.0 | 88.7 | 88.7 |

TABLE IV: Bridging the gap in success rate achieved with manually labeled real scenes and the proposed PHYSIM by self-supervised lifelong learning

Moreover, evaluating object detection methods is particularly difficult when the objects are partially occluded because there are multiple bounding boxes that can be used as ground truth for evaluating the accuracy of the detection.

The result from the training phase using synthetic only data is still below the performance of the baseline alternative MANUAL, which includes manually labeled real images. This can be attributed to the problem of dataset bias, i.e. the unavoidable difference between synthetic and real images. This bias is the only reason why SYNTH outperforms SIM, because the only difference between the two methods is that one uses real images and the other uses rendered images.

Consequently, synthetic images alone are not sufficient. This issue is addressed by the second step in the proposed pipeline, i.e., lifelong online self-training. Tables III and IV show the results using additional training examples that were automatically labeled, without any human intervention, using PHYSIM. The proposed
approach PHYSIM is able to close the gap on the performance achieved with MANUAL without the need of a manual labeling process. Three incremental data sets were used to show the improvement in performance:

1) DET1: where 450 real images of single object scenes automatically labeled by PHYSIM where used to re-train the R-CNN.

2) DET2: where 1202 real images of single object scenes automatically labeled by PHYSIM where used to re-train the R-CNN.

3) DET3: where on top of DET2, 451 additional real images of cluttered scenes automatically labeled by PHYSIM where used to re-train the R-CNN.

Note that DET1 ⊂ DET2 ⊂ DET3. As Tables III and IV show, the performance of the R-CNN detector improves with larger training data sets, obtained without any human effort. This gradual, lifelong improvement is illustrated in Figure 7.

VI. CONCLUSION

Object detection is an important part of robotic manipulation in unstructured environments. State-of-the-art performance on solving this problem is achieved by Convolutional Neural Networks, which are typically trained by using a large number of manually labeled images. This empirical study has shown that there is a way to minimize manual effort for labeling and achieve good detection performance.

Instead, a physics engine is used to generate labeled, synthetic training examples for realistic poses of the objects, given access to their 3D models. Synthetic only training may not be sufficient as it may not generalize to the lighting conditions present in a real environment. To avoid overfitting to the synthetic conditions, the proposed training process includes a self-learning process, where successful, high confidence, detections on real images are used to include additional positive examples in the training set. The combination of physical reasoning and self-learning results in a success ratio that is equivalent to that achieved using manual labeling.

REFERENCES


