Synthesizing Training Data for Object Detection in Indoor Scenes

Georgios Georgakis*, Arsalan Mousavian*, Alexander C. Berg†, Jana Košecká*
*George Mason University
{ggeorgak,amousavi,kosecka}@gmu.edu
†University of North Carolina at Chapel Hill
aberg@cs.unc.edu

***Abstract*—Detection of objects in cluttered indoor environments is one of the key enabling functionalities for service robots. The best performing object detection approaches in computer vision exploit deep Convolutional Neural Networks (CNN) to simultaneously detect and categorize the objects of interest in cluttered scenes. Training of such models typically requires large amounts of annotated training data which is time consuming and costly to obtain. In this work we explore the ability of using synthetically generated composite images for training state of the art object detectors. We superimpose 2D images of textured object models into images of real environments at variety of locations and scales. Our experiments evaluate different superimposition strategies ranging from purely image-based blending all the way to depth and semantics informed positioning of the object models to real scenes. We demonstrate the effectiveness of these object detector training strategies on publicly available datasets of GMU-Kitchens [5] and Washington RGB-D Scenes v2 [11], and show how object detectors can be trained with limited amounts of annotated real scenes with objects present. This charts new opportunities for training detectors for new objects by exploiting existing object model repositories in either a purely automatic fashion or with only a very small number of human-annotated examples.**

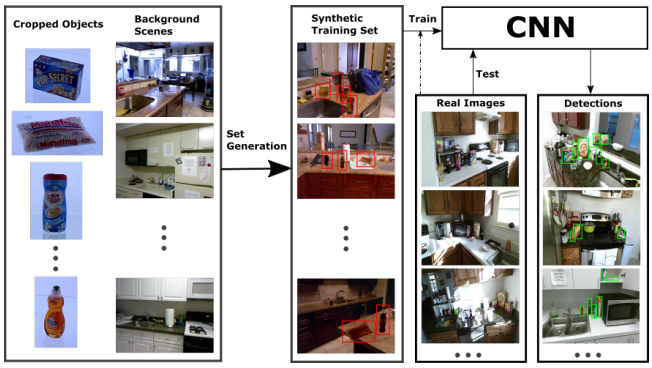

Fig. 1. Given cropped object images and background scenes we propose an automated approach for generating synthetic training sets that can be used to train current state of the art object detectors, which can then be applied to real test images. The generation procedure takes advantage of scene understanding methods in order to place the objects in meaningful positions in the images. We also explore using a combination of synthetic and real images for training and demonstrate higher detection accuracy compared to training with only real data.

I. Introduction

The capability of detecting and searching for common household objects in indoor environments is the key component of the 'fetch-and-delivery' task commonly considered as one of the main functionalities of service robots. Existing approaches for object detection are dominated by machine learning techniques focusing on learning suitable representations of object instances. This is especially the case when the objects of interest are to be localized in environments with large amounts of clutter, variations in lighting, and a range of poses. While the problem of detecting object instances in simpler table top settings has been tackled previously using local features, these methods are often not effective in the presence of large amounts of clutter or when the scale of the objects is small.

Current leading object detectors exploit convolutional neural networks (CNNs) and are either trained end-to-end [12] for sliding-window detection or follow the region proposal approach which is jointly fine-tuned for accurate detection and classification [6] [17]. In both approaches, training and evaluation of object detectors requires labeling of a *large number of training images with objects in various backgrounds and poses*, providing the bounding boxes or the entire object/background segmentation.

Often in robotics, object detection is a prerequisite for tasks such as pose estimation, grasping, and manipulation. Notable efforts have been made to collect 3D models for object instances with and without textures, assuming that objects of interest are in proximity, typically on a table top. Existing approaches either use 3D CAD models [13] or texture mapped models of object instances obtained using traditional reconstruction pipelines [20, 23].

In this work we explore the feasibility of using such existing datasets of standalone objects on uniform backgrounds for training object detectors [12, 17] that can be applied in real-world cluttered scenes. We create "synthetic" training images by superimposing the objects into images of real scenes. We investigate effects of different superimposition strategies ranging from purely image-based blending all the way to using depth and semantics to inform positioning of the objects. Toward this end we exploit the geometry and the a semantic segmentation of a scene obtained using the state of the art method of [14] to restrict the locations and size of the superimposed object model. We demonstrate that, in the context of robotics applications in indoor environments, these

positioning strategies improve the final performance of the detector. This is in contrast with previous approaches [16, 22] which used large synthetic datasets with mostly randomized placement. In summary, our contributions are the following:

1) We propose an automated approach to generate synthetic training data for the task of object detection, which takes into consideration the geometry and semantic information of the scene.
2) Based on our results and observations, we offer insights regarding the generation design choices, that could potentially affect the way training sets for object detection are generated in the future.
3) We provide an extensive evaluation of current state-of-the-art object detectors and demonstrate their behavior under different training regimes.

II. Related Work

We first briefly review related works in object detection to motivate our choice of detectors, then discuss previous attempts to use synthetic data as well as different datasets and evaluation methodologies.

a) Object Detection: Traditional methods for object detection in cluttered scenes follow the sliding window based pipeline with hand designed flat feature representations (e.g. HOG) along with discriminative classifiers, such as linear or latent SVMs. Examples include DPMs [4] which exploit efficient methods for feature computation and classifier evaluation. These models have been used successfully in robotics for detection in the table top setting [10]. Other effectively used strategies for object detection used local features and correspondences between a model reference image and the scene. These approaches [3, 25] worked well with textured household objects, taking advantage of the discriminative nature of the local descriptors. In an attempt to reduce the search space of the sliding window techniques, alternative approaches concentrated on generating category-independent object proposals [27, 2] using bottom up segmenation techniques followed by classification using traditional features. The flat hand-designed feature representations have been recently superceded by approaches based on convolutional neural networks, which learn features with increased amount of invariance by repeated layering of convolutional and pooling layers. While these methods have been intially introduced for image classification task [9], initial extentions of these methods to object detection include [7] [19]. The R-CNN approach [7] relied on finding object proposals and extracting features from each crop using a pre-trained network, making the proposal generating module independent from the classification module. Recent state of the art object detectors such as Faster R-CNN [17] and SSD [12] are trained jointly in a so called end-to-end fashion to both find object proposals and also classify them.

b) Synthetic Data: There are several previous attempts to use synthetic data for training CNNs. The work of [16] used existing 3D CAD models, both with and without texture, to generate 2D images by varying the projections and orientations of the objects. The approach was evaluated on 20 categories in PASCAL VOC2007 dataset. While that work used deep CNN's, they were earlier models [7] where the proposal generation module was independent from fine-tuning the CNN classifier, hence making the dependence on the context and background less prominent than in current models. In the work of [22] the authors used the rendered models and their 2D projections on varying backgrounds to train a deep CNN for pose estimation. In these representative works, objects typically appeared on simpler backgrounds and were combined with the object detection strategies that rely on the proposal generation stage. Our work differs in that we perform informed compositing on the background scenes, instead of randomly generating object-centric synthetic images that lack object localization information. This allows us to train the CNN object detectors to produce higher quality object proposals, rather than relying on unsupervised bottom-up techniques. In [18], a Grand Theft Auto video game engine was used to collect scenes with realistic appearance for the problem of semantic segmentation. Authors showed that using these high realism renderings can significantly reduce the effort for annotation. They used a combination of synthetic data and real images to train models for semantic segmentation. Perhaps the closest work to ours is [8], which also generates a synthetic training set by taking advantage of scene segmentation to create synthetic training examples, however the task is that of text localization instead of object detection.

III. Approach

A. Synthetic Set Generation

CNN-based object detectors require large amounts of annotated data for training, due to the large number of parameters that need to be learned. For object instance detection the training data should also cover the variations in the object's viewpoint and other nuisance parameters such as lighting, occlusion and clutter. Manually collecting and annotating scenes with the aforementioned properties is time-consuming and costly. Another factor in annotation is the sometimes low generalization capability of trained models across different environments and backgrounds. The work of [21] addressed this problem by building a map of an environment including objects of interest and using Amazon Mechanical Turk for annotation and subsequent training of object detectors in each particular environment. While the authors demonstrated this approach on commonly encountered categories ($\approx$ 20) of household objects, this approach is not scalable to the large number of small object instances one may be interested in detecting.

Our approach focuses on object instances and their superimposition into real scenes at different positions, scales, while reducing the difference in lighting conditions and exploiting proper context. To this end, we choose to use cropped images from existing object recognition datasets such as BigBird [20] rather than using 3D CAD models [16, 22]. This allows us to have real colors and textures for our training instances as opposed to rendering them with randomly chosen or artificial

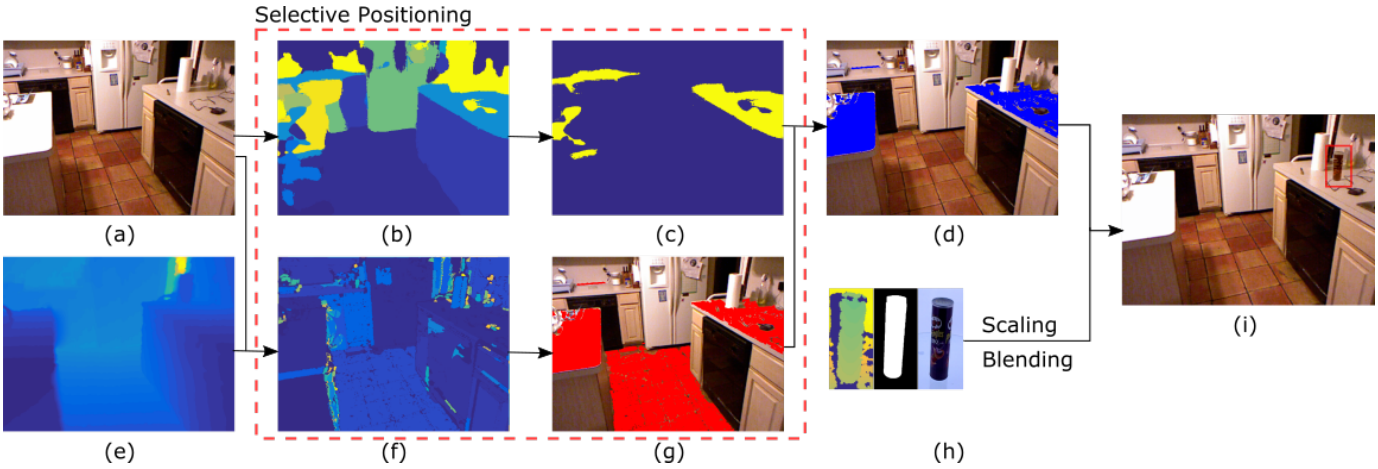

Fig. 2. Overview of the procedure for blending an object in a background scene. We take advantage of estimated support surfaces (g) and predictions for counters and tables (c) in order to find meaningful regions (d) to position the objects. The semantic segmentation of the scene [14], and the plane extraction are shown in (b) and (f) respectively. (h) presents an example of an object's RGB, depth, and mask images, while (i) shows the final blending result. RGB and depth images of the background scene are in (a) and (e) respectively. Best viewed in color.

samples. The BigBird dataset captures 120 azimuth angles from 5 different elevations for a total of 600 views per object. It contains a total of 125 object instances with a variety of textures and shapes. In our experiments we use the 11 object instances that can be found in the GMU-Kitchens dataset.

The process of generating a composite image with superimposed objects can be summarized in the following steps. First, we choose a background scene and estimate the positions of any support surfaces. This is further augmented by semantic segmentation of the scene, used to verify the support surfaces found by plane fitting. The objects of interest are placed on support surfaces, ensuring their location in areas with appropriate context and backgrounds. The next step is to randomly choose an object and its pose, followed by choosing a position in the image. The object scale is then determined by the depth value of the chosen position and finally the object is blended into the scene. An example of this process is shown in Figure 2. We next describe these steps in more detail.

Selective Positioning: In natural images, small handheld objects are usually found on supporting surfaces such as counters, tables, and desks. These planar surfaces are extracted using the method described in [26], which applies RANSAC to fit planes on regions after an initial over-segmentation of the image, and has been shown to perform very well in indoor environments. Given the extracted planar surfaces's orientations, we select the planes with large extent, which are aligned with the gravity direction as candidate support surfaces. To ensure that the candidate support surfaces belong to a desired semantic category, a support surface is considered valid if it overlaps in the image space with semantic categories of counters, tables and desks obtained by semantic segmentation of the RGB-D image.

Semantic Segmentation: To determine the semantic categories in the scene, we use the semantic segmentation CNN of [14], which is pre-trained on MS-COCO and Pascal-VOC datasets, and fine-tuned on NYUDepth v2 dataset for the task of semantic segmentation for 40 semantic categories. The model is jointly trained for semantic segmentation and depth estimation, which allows the scene geometry to be exploited for better discrimination between some of the categories. We do not rely solely on the semantic segmentation for positioning the object, since it rarely covers the entire support surface, as can be seen in Figure 2(c). The combination of the support surfaces detection and semantic segmentation produces more accurate regions for placing the objects.

Having found all the regions in the image for object placement, a region that belongs to a valid support surface is randomly chosen to position the object. To avoid having objects floating above a surface, the chosen region needs to have some overlap with the aforementioned semantic segmentation categories. Finally, occlusion levels are regulated by allowing a maximum of 50% overlap between positioned objects in the image.

Selective Scaling and Blending: The size of the object is determined by using the depth of the selected position and scaling the width w and height h accordingly:

$$\hat{w} = \frac{w z_t}{z} \qquad \hat{h} = \frac{h z_t}{z}$$

where z_t is the median depth of the object's training images, z is the depth at the selected position in the background image, and $\hat{w}$, $\hat{h}$ are the scaled width and height respectively.

The last step in our process is to blend the object with the background image in order to mitigate the effects of changes in illumination and contrast. We use the Fast Seamless Cloning [24] and its provided implementation, with a minor modification. Instead of blending a rectangular patch of the object, we provide a masked object to the fast seamless cloning algorithm which produces a cleaner result. Figure 3 illustrates examples of scenes with multiple blended objects.

Fig. 3. Examples of blending object instances from the BigBird dataset into scenes from the NYUDepth v2 dataset. The blended objects are marked with a red bounding box. Best viewed in color.

B. Object Detectors

For our experiments we employ two state-of-the-art object detectors, the Faster R-CNN [17] and the Single-Shot Multi-box Detector (SSD) [12]. Both Faster R-CNN and SSD are trained end-to-end but the architectures of them are different. Faster R-CNN consists of two modules. The first module is Region Proposal Network (RPN) which is a fully convolutional network that outputs object proposals and also an objectness score for each proposal which reflects the probability of having an object inside the object proposal region. The second detection network module resizes the feature maps, corresponding to each object proposal to a fixed size, classifies it to an object category and refines the location and the height and width of the bounding box associated with each proposal. The advantage of Faster R-CNN is the modularity of the model; one module that finds object proposals and the second module which classifies each of the proposals. The downside of Faster-RCNN is that it uses the same feature map to find objects of different sizes which causes problems for small objects. SSD tackles this problem by creating feature maps of different resolutions. Each cell of the coarser feature maps captures larger area of the image for detecting large objects whereas the finer feature maps are detecting smaller objects. Both of these detectors have difficulties with the smaller objects, hence the image resolution they use can affect the results.

IV. Experiments

In order to evaluate the object detectors trained on composited images, we have conducted three sets of experiments on two publicly available datasets, the GMU-Kitchen Scenes [5] and the Washington RGB-D Scenes v2 dataset [11]. In the first experiment, composite training images are generated by choosing different compositing strategies to determine the effect of positioning, scaling, and blending on the performance. The object detectors are trained on composited images and evaluated on real scenes as well as composited scenes. In the second set of experiments we examine the effect of varying proportion of synthetic/composited images and real training images. Finally we use synthetic data for both training and testing in order to show the reduction of over-fitting during training when the proposed approach of data generation is employed.

Fig. 4. Comparison between masks from BigBird (top row), and masks after refinement with Graph-cut (bottom row).

	coca cola	coffee mate	honey bunches	hunts sauce	mahatma rice	nature valley 1	nature valley 2	palmolive orange	pop secret	pringles bbq	red bull	mAP
Real to Real												
1. **3-Fold**	46.7 / 78.9	73.7 / 92.0	81.8 / 91.9	64.5 / 81.8	62.9 / 74.7	83.9 / 93.4	70.3 / 85.9	69.8 / 76.6	76.1 / 90.7	64.8 / 86.4	27.7 / 54.6	65.6 / 82.5
Synthetic to Real												
2. **RP-SI-RS**	9.1 / 37.2	15.9 / 68.3	49.4 / **72.5**	9.8 / 26.1	**13.5** / 32.3	61.6 / 70.2	42.2 / 57.2	15.5 / 29.0	36.9 / 46.9	1.0 / 2.9	0.8 / 20.4	23.2 / 42.1
3. **RP-BL-RS**	9.1 / **62.4**	15.7 / 69.3	41.4 / 58.2	**10.7** / 32.4	9.9 / 4.3	41.2 / 51.7	39.0 / 47.1	**28.7** / **39.3**	36.2 / 32.2	11.5 / **59.2**	0.7 / **30.0**	22.2 / 44.2
4. **SP-SI-SS**	10.5 / 45.2	17.5 / **71.7**	47.2 / 66.6	0.1 / 26.0	9.1 / **45.5**	44.9 / 80.5	36.5 / **78.4**	24.0 / 37.8	9.1 / 46.1	5.5 / 27.1	**10.3** / 9.7	19.5 / 48.6
5. **SP-BL-SS**	**18.3** / 55.5	**22.1** / 67.9	**58.9** / 71.2	9.5 / **34.6**	11.1 / 30.6	**75.7** / **82.9**	**65.5** / 66.2	23.8 / 33.1	**59.4** / **54.3**	**14.6** / 54.8	9.1 / 17.7	**33.5** / **51.7**
Synthetic+Real to Real												
6. **1% real**	39.4 / 65.1	71.8 / 85.8	80.4 / 85.7	50.2 / 62.3	45.2 / 51.6	82.6 / 90.4	74.9 / 85.6	57.8 / 54.3	78.1 / 79.4	54.2 / 70.6	28.5 / 32.2	60.3 / 69.3
7. **10% real**	59.4 / 70.5	83.8 / 91.5	83.7 / 89.6	66.2 / 82.2	60.7 / 62.8	87.3 / 94.6	**79.8** / 87.4	72.6 / 66.3	83.4 / 89.5	77.6 / 87.4	33.0 / 49.5	71.6 / 79.2
8. **50% real**	**64.6** / 79.3	84.2 / 92.5	**87.6** / 91.1	70.4 / 77.3	67.1 / **86.2**	**89.2** / 95.4	79.7 / 87.9	**75.4** / 77.8	80.1 / 91.6	79.3 / 90.1	**37.6** / 52.2	**74.1** / 83.8
9. **100% real**	59.0 / **82.6**	**84.5** / **92.9**	85.1 / **91.4**	**74.2** / **85.5**	**67.5** / 81.9	87.4 / **95.5**	78.9 / **88.6**	71.3 / **78.5**	**85.2** / **93.6**	**79.9** / **90.2**	**37.6** / **54.1**	73.7 / **85.0**
Synthetic to Synthetic												
10. **RP-SI-RS**	**90.8** / **99.6**	**90.9** / **100**	**90.8** / **99.7**	**90.8** / **99.6**	**90.9** / **99.6**	**90.9** / **99.8**	**90.8** / **99.7**	**90.7** / **98.9**	**90.9** / **99.7**	**90.8** / **99.4**	**90.6** / **98.7**	**90.8** / **99.5**
11. **SP-BL-SS**	84.3 / 79.2	86.7 / 84.4	88.1 / 94.8	81.7 / 79.3	88.9 / 94.6	83.5 / 92.6	80.8 / 89.5	83.1 / 79.9	84.5 / 93.1	86.4 / 89.1	74.0 / 65.8	83.8 / 85.7

TABLE I

AVERAGE PRECISION RESULTS (SSD / FASTER-RCNN) FOR ALL EXPERIMENTS ON THE GMU-KITCHENS DATASET. THE SYNTHETIC+REAL TO REAL EXPERIMENTS WERE PERFORMED USING THE SP-BL-SS SET PLUS THE PERCENTAGE OF REAL DATA SHOWN IN THE TABLE.

	bowl	cap	cereal box	coffee mug	soda can	mAP
Real to Real						
1.**Scenes 1-7**	90.8 / 99.7	90.2 / 95.5	90.9 / 99.6	89.9 / 96.7	89.3 / 96.7	90.2 / 97.9
Synthetic to Real						
2. **RP-SI-RS**	77.2 / 65.2	78.5 / 39.4	**90.9** / 69.2	74.1 / 57.0	70.1 / 29.4	78.2 / 52.0
3. **RP-BL-RS**	71.5 / 82.6	62.9 / 47.3	90.3 / 93.0	73.1 / **74.9**	65.2 / 52.7	72.6 / 70.1
4. **SP-SI-SS**	**77.8** / **86.6**	**79.8** / 62.4	90.8 / 93.6	73.4 / 68.6	**75.5** / 55.5	**79.5** / 73.4
5. **SP-BL-SS**	71.9 / 82.3	75.9 / **70.9**	90.7 / **96.8**	**74.3** / 74.2	75.0 / **66.3**	77.5 / **78.1**
Synthetic+Real to Real						
6. **1% real**	87.7 / 98.3	88.3 / 92.7	90.8 / 98.6	88.1 / 96.6	89.5 / 94.9	88.9 / 96.2
7. **10% real**	90.8 / **99.6**	89.5 / 96.0	90.8 / 99.6	90.4 / 96.9	**90.8** / 97.1	90.5 / 97.8
8. **50% real**	**90.9** / 99.5	**90.6** / 96.5	**90.9** / **99.9**	90.3 / **97.3**	90.6 / 97.8	90.7 / **98.2**
9. **100% real**	**90.9** / 99.4	90.5 / **97.0**	**90.9** / 99.3	**90.8** / 97.2	**90.8** / **98.1**	**90.8** / **98.2**
Synthetic to Synthetic						
10. **RP-SI-RS**	90.5 / **98.5**	**90.9** / **99.5**	**90.9** / **99.5**	**90.2** / **96.9**	**90.0** / 92.6	**90.5** / **97.4**
11. **SP-BL-SS**	**90.7** / 97.4	90.7 / 97.5	90.4 / 97.3	89.5 / 95.1	89.2 / **93.5**	90.1 / 96.2

TABLE II

AVERAGE PRECISION RESULTS (SSD / FASTER-RCNN) FOR ALL EXPERIMENTS ON THE WRGB-D DATASET. THE SYNTHETIC+REAL TO REAL EXPERIMENTS WERE PERFORMED USING THE SP-BL-SS SET PLUS THE PERCENTAGE OF REAL DATA SHOWN IN THE TABLE.

A. Datasets and Backgrounds

For our experiments, we utilized the following datasets:

a) GMU Kitchen Scenes dataset [5]: The GMU-Kitchens dataset includes 9 RGB-D videos depicting kitchen scenes with 11 object instances from the BigBird dataset. We also used all 71 raw kitchen videos from the NYU Depth Dataset V2 [15] with a total of around 7000 frames as background images. For each background image we generate four synthetic images with different variations in objects that are added to the scene, pose, scale, and the location that the objects are put. The object identities and their poses are randomly sampled from the BigBird dataset, with 360 examples per object with 3 elevations and 120 azimuths.

The images where the support surfaces were not detected are removed from the training set, making our effective set around 5000. Cropped object images from BigBird dataset of the 11 instances contained in GMU-Kitchens were used for superimpositioning. We refine the provided object masks with Graph-cut [1], in order to get cleaner outlines for the objects. This helps with the jagged and incomplete boundaries of certain objects (e.g. coke bottle), which are due to imperfect masks obtained from the depth channel of RGB-D data caused by reflective materials. Figure 4 illustrates a comparison between masks from BigBird and masks refined with Graph-cut. For comparison we also provide the performance of the object detectors (row 1 of Table I) trained and tested on the real data. The train-test split follows the division of the dataset into three different folds, where in each fold six scenes are used for training and three are used for testing, as shown in [5].

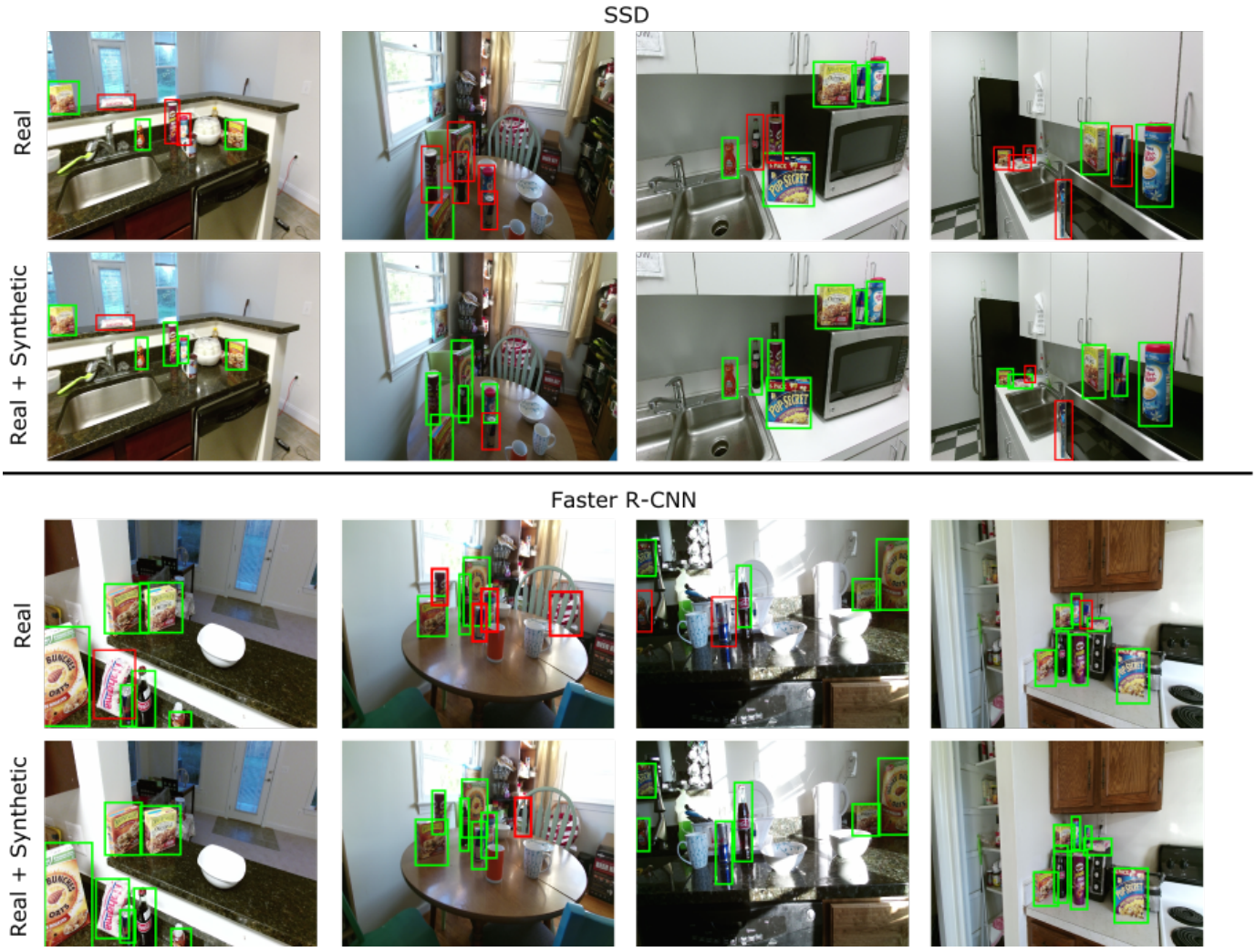

Fig. 5. Detection examples for the SSD and Faster R-CNN object detectors on the GMU-Kitchens dataset. Rows 1 and 3 show results when only the real training data were used, while rows 2 and 4 present results after the detectors were trained with the synthetic set **SP-BL-SS** and 50% of the real training data. The green bounding boxes depict correct detections, while the red represent false classifications and missed detections. Training with a combination of synthetic and real data proves beneficial for the detection task, as the detectors are more robust to small objects and viewpoint variation. Best viewed in color.

b) Washington RGB-D Scenes v2 dataset (WRGB-D) [11]: The WRGB-D dataset includes 14 RGB-D videos of indoor table-top scenes containing instances of objects from five object categories: bowl, cap, cereal box, coffee mug, and soda can. The synthetic training data is generated using the provided background scenes (around 3000 images) and cropped object images for the present object categories in the WRGB-D v1 dataset [10]. For each background image we generate five synthetic images to get a total of around 4600 images. As mentioned earlier, images without a support surface are discarded. The WRGB-D dataset consists of 14 scenes where each scene has around 800 images. The images that belong to seven of these scenes are used for training and the rest are used for testing. Line 1 in Table II shows the performance of the two object detectors when this split of the real training data is used.

B. Synthetic to Real

In this experiment we use the synthetic training sets generated with different combinations of generation parameters for training, and test on real data. The generation parameters that we vary are: Random Positioning (**RP**) / Selective Positioning (**SP**), Simple Superimposition (**SI**) / Blending (**BL**), and Random Scale (**RS**) / Selective Scale (**SS**), where **SP**, **SS**, and **BL** are explained in Section III-A. For **RP** we randomly sample the position for the object in the entire image, for **RS** the scale of the object is randomly sampled from the range of 0.2 to 1 with a step of 0.1, and for **SI** we do not use blending but instead we superimpose the masked object directly on the background.

The objective of this experiment is to investigate the effect of the generation parameters on the detection accuracy. For example, if a detector is trained on a set generated with selective positioning, without blending, and selective scale,

how does it compare to another detector which is trained on a completely randomly generated set without blending ? If the former demonstrates higher performance than the later, then we can assume that selective positioning and scaling are important and superior to random positioning. For each trained detector, a combination of the generation parameters (e.g. **SP-BL-SS**) is initially chosen, and then the synthetic set is generated using our proposed approach along with its bounding box annotations for each object instance. The detector is trained only on the synthetic data and then applied on the real testing data.

The results are shown on lines 2-5 in Table I for the GMU-Kitchens dataset and in Table II for the WRGB-D dataset. Note that for the GMU-Kitchens dataset, all 9 video scenes were used for testing. We report detection accuracy on four combinations of generation parameters, **RP-SI-RS**, **RP-BL-RS**, **SP-SI-SS**, and **SP-BL-SS**. Other combinations such as **SP-BL-RS** and **RP-BL-SS** have also been tried, however we noticed that applying selective positioning without selective scaling and vice-versa, does not yield any significant improvements.

For both datasets, we first notice that using only synthetic data for training considerably lowers the detection accuracy compared to using real training data. Regarding the GMU-Kitchens dataset, the best performing is **SP-BL-SS** for both detectors, which outperforms the completely randomized set of **RP-SI-RS** by 10.3% and 9.6% for SSD and Faster-RCNN respectively. This suggests that selective positioning and selective scaling are both important factors when generating the training set. The same trend can also be seen in the case of the WRGB-D dataset, where **SP-SI-SS** and **SP-BL-SS** produce the best results for SSD and Faster-RCNN respectively.

C. Synthetic+Real to Real

We are interested to see how effective our generated synthetic training set is when combined with real training data. Towards this end the two detectors are trained using our generated set of **SP-BL-SS** plus a certain percentage of the real training data: 1%, 10%, 50%, and 100%. For the real training sets, besides the case of 100%, the images are chosen randomly.

Results are shown on lines 6-9 in Table I for the GMU-Kitchens dataset and in Table II for the WRGB-D dataset. What is surprising in these results is that when our synthetic training set is combined with only 10% of the real training data, we achieve higher or comparable detection accuracy than when the training set is only comprised with real data (see line 1 in both tables). Especially in the case of SSD in the GMU-Kitchens dataset, we observe an increase of 6%. Only exception is Faster-RCNN on the GMU-Kitchens dataset which achieves a 2.3% lower performance, however, when we use 50% of the real training data we get a better performance of 1.3%. In all cases, when the synthetic set is combined with 50% and 100% of the real data, it outperforms the training with only a real training set.

The results suggest that our synthetic training data can effectively augment existing datasets even when the actual number of real training examples is small. This is particularly useful when only a small subset of a data is annotated. Specifically, in our settings, the 10% of real training data refers to around 400 images in the GMU-Kitchens dataset, and to around 600 in the WRGB-D dataset. Figure 5 presents examples for which the detectors were unable to detect objects when they were trained with only real data, but succeeded when the training set was augmented with our synthetic data.

D. Synthetic to Synthetic

In this experiment, the object detectors are trained and tested on synthetic sets. The objective is to show the reduction of over-fitting on the training data when using our approach to generate the synthetic images, instead of creating them randomly. We used the synthetic sets of **RP-SI-RS** and **SP-BL-SS** and split them in half in order to create the train-test sets.

The results are presented on lines 10 and 11 in Table I for the GMU-Kitchens dataset and in Table II for the WRGB-D dataset. For GMU-Kitchens, we observe that **RP-SI-RS** achieves results of over 90%, and in the case of Faster-RCNN almost 100%, while at the same time it is the least performing synthetic set in the synthetic to real experiment (see line 2 in table I) described in Section IV-B. This is because the detectors over-fit on the synthetic data and cannot generalize to an unseen set of real test data. While the detectors still seem to over-fit on **SP-BL-SS**, the gap between the accuracy on the synthetic testing and real testing data is much smaller, at the order of 17% for SSD, and 33.4% for Faster-RCNN (see line 5 in table I).

On the other hand, for the WRGB-D dataset both synthetic training sets achieve similar results on their synthetic test sets. This is not surprising as the complexity of the scenes is lower in WRGB-D than in the GMU-Kitchens dataset, with less variety of backgrounds, rendering the **SP-BL-SS** set less effective. This is also supported in [5], where the authors demonstrated that a simple proposal generation approach can achieve $\approx$ 90% recall with only 17 proposals per image on average. They also report that the same approach on the GMU-Kitchens dataset achieves only 62% with a much higher number of proposals per image.

V. Discussion and Conclusion

One of the advantages of our method is that it is scalable both to the number of objects of interest and also the set of the possible backgrounds which makes our method suitable for robotics application. For example, the object detectors can be trained with significantly less annotated data using our proposed training data augmentation. We also showed that our method is more effective when the object placements are based on semantic and geometric context of the scene. This is due to the fact that CNNs implicitly consider the surrounding context of the objects and when the superimposition are done in informed way, the performance gain is larger.

As mentioned before, state-of-the-art object detectors have difficulties with small objects. One remedy for this problem is using higher resolution images so that small objects become larger and therefore detectable by the models. Since Faster R-CNN is designed to work with larger image resolution, it performs better than SSD in our experiments. However, since SSD estimates bounding boxes and classes together, the size of the feature maps grows more than Faster R-CNN which prevents it from working on larger resolution. One interesting observation is that the accuracy of SSD on small objects increases when it is trained with the augmented training set.

While we showed it is possible to train an object detector with fewer annotated images using synthetically generated images, more improvements can be made through domain adaptation. In domain adaptation the objective is to transfer a model that is trained on a source task, which is synthetic images in our case, to the target task, which is the real images, using very few instances of the target task. We believe that our generated training sets can be useful when attempting to adapt the model from synthetic to real data, and plan to investigate further in the future.

In conclusion, we have presented an automated procedure for generating synthetic training data for deep CNN object detectors. The generation procedure takes into consideration the geometry and semantic segmentation of the scene in order to make informative decisions regarding the positions and scales of the objects. We have employed two state-of-the-art object detectors and demonstrated an increase in their performance when they are trained with an augmented training set. In addition, we also investigated the effect of different generation parameters and provided some insight that could prove useful in future attempts of generating synthetic training data for object detection.

ACKNOWLEDGMENTS

We acknowledge support from NSF NRI grant 1527208. Some of the experiments were run on ARGO, a research computing cluster provided by the Office of Research Computing at George Mason University, VA. (URL: http://orc.gmu.edu).

REFERENCES

[1] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, November 2001. ISSN 0162-8828. doi: 10.1109/34.969114. URL http://dx.doi.org/10.1109/34.969114.

[2] Ming-Ming Cheng, Ziming Zhang, Wen-Yan Lin, and Philip H. S. Torr. BING: Binarized normed gradients for objectness estimation at 300fps. In *IEEE CVPR*, 2014.

[3] A. Collet, M. Martinez, and S. Srinivasa. The MOPED framework: Object recognition and pose estimation for manipulation. In *in International Journal of Robotics Research*, 2011.

[4] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[5] Georgios Georgakis, Md Reza, Arsalan Mousavian, Phi-Hung Le, and Jana Kosecka. Multiview rgb-d dataset for object instance detection. In *IEEE International Conference on 3DVision (3DV)*, 2016.

[6] Ross Girshick. Fast R-CNN. In *in IEEE International Conference on Computer Vision (ICCV)*, 2015.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *in IEEE Conference Computer Vision and Pattern Recognition (CVPR)*, 2014.

[8] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[9] A. Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[10] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view RGB-D object dataset. In *in IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[11] Kevin Lai, Liefeng Bo, and Dieter Fox. Unsupervised feature learning for 3d scene labeling. In *IEEE International Conference on on Robotics and Automation*, 2014.

[12] W. Liu, D. Anguelov, C. Szegedy D. Erhan, S. Reed, C. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *in European Conference on Computer Vision (ECCV)*, 2016.

[13] Jeffrey Mahler, Florian T. Pokorny, Brian Hou, Melrose Roderick, Mathieu Aubry Michael Laskey, Kai Kohlhoff, Torsten Kroeger, James Kuffner, and Ken Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *ICRA*, 2016.

[14] Arsalan Mousavian, Hamed Pirsiavash, and Jana Kosecka. Joint semantic segmentation and depth estimation with deep convolutional networks. In *IEEE International Conference on 3DVision (3DV)*, 2016.

[15] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[16] Xingchao Peng, Baochen Sun, Karim Ali, and Kate Saenko. Learning deep object detectors from 3d models. In *ICCV*, 2015.

[17] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[18] Stephan R. Richter, Vibhav Vineet, and Vladlen Koltun Stefan Roth. Playing for data: Ground truth from computer games. In *ECCV*, 2016.

[19] Pierre Sermanet, David Eigen, Michael Mathieu Xiang Zhang, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *CVPR*, 2013.

[20] A. Singh, J. Sha, K. Narayan, T. Achim, and P. Abbeel. A large-scale 3D database of object instances. In *in IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[21] S. Song, L. Zhang, and J. Xiao. Robot in a room: Toward perfect object recognition in closed environments. In *arXiv:1507.02703 [cs.CV] 9 Jul 2015*, July 2015.

[22] Hao Su, Charles R. Qi, Yangyan Li, and Leonidas J. Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[23] Y. Sun, M. Bianchi, J. Bohg, and A. Dollar. http://rhgm.org/activities/workshopicra16/. In *Workshop on Grasping and Manipulation Datasets (ICRA)*, 2016.

[24] Masayuki Tanaka, Ryo Kamio, and Masatoshi Okutomi. Seamless image cloning by a closed form solution of a modified poisson problem. In *SIGGRAPH Asia 2012 Posters*, SA '12, 2012.

[25] J. Tang, S. Miller, A. Singh, and P. Abbeel. A textured object recognition pipeline for color and depth image data. In *in IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[26] Camillo J. Taylor and Anthony Cowley. Parsing indoor scenes using rgb-d imagery. In *Robotics: Science and Systems (RSS)*, July 2012.

[27] 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*, 2013. doi: 10.1007/s11263-013-0620-5. URL http://www.huppelen.nl/publications/selectiveSearchDraft.pdf.