Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection

Debidatta Dwibedi      Ishan Misra      Martial Hebert
The Robotics Institute, Carnegie Mellon University
debidatta@cmu.edu, {imisra, hebert}@cs.cmu.edu

Abstract

A major impediment in rapidly deploying object detection models for instance detection is the lack of large annotated datasets. For example, finding a large labeled dataset containing instances in a particular kitchen is unlikely. Each new environment with new instances requires expensive data collection and annotation. In this paper, we propose a simple approach to generate large annotated instance datasets with minimal effort. Our key insight is that ensuring only patch-level realism provides enough training signal for current object detector models. We automatically ‘cut’ object instances and ‘paste’ them on random backgrounds. A naive way to do this results in pixel artifacts which result in poor performance for trained models. We show how to make detectors ignore these artifacts during training and generate data that gives competitive performance on real data. Our method outperforms existing synthesis approaches and when combined with real images improves relative performance by more than 21% on benchmark datasets. In a cross-domain setting, our synthetic data combined with just 10% real data outperforms models trained on all real data.

1. Introduction

Imagine using an object detection system for an environment like your kitchen. Such a system needs to not only recognize different kinds of objects but also distinguish between many different instances of the same object category, e.g., your cup vs. my cup. With the tremendous progress that has been made in visual recognition, as documented on benchmark detection datasets, one may expect to easily take a state-of-the-art system and deploy it for such a setting.

However, one of the biggest drawbacks of using a state-of-the-art detection system is the amount of annotations needed to train it. For a new environment with new objects, we would likely need to curate thousands of diverse images with varied backgrounds and viewpoints, and annotate them with boxes. Traditionally, vision researchers have undertaken such a mammoth task [8, 26] for a few commonly occurring categories like man, cow, sheep etc., but this approach is unlikely to scale to all possible categories, especially the instances in your kitchen. In a personalized setting we need annotations for instances like your cup. We believe that collecting such annotations is a major impediment for rapid deployment of detection systems in robotics or other personalized applications.

Recently, a successful research direction to overcome this annotation barrier, is to use synthetically rendered scenes and objects [22, 34, 47] to train a detection system. This approach requires a lot of effort to make the scenes and objects realistic, ensuring high quality global and local consistency. Moreover, models trained on such synthetic data

Figure 1: We present a simple way to rapidly generate training images for instance detection with minimal human effort. We automatically extract object instance masks and render it on random background images to create realistic training images with bounding box labels. Our results show that such data is competitive with human curated datasets, and contains complementary information.
have trouble generalizing to real data because of the change in image statistics [5, 36]. To address this, an emerging theme of work [16] moves away from graphics based renderings to composing real images. The underlying theme is to ‘paste’ real object masks in real images, thus reducing the dependence on graphics renderings. Concurrent work [12] estimates scene geometry and layout and then synthetically places object masks in the scene to create realistic training images. However, the scene layout estimation step may not generalize to unseen scenes. In our paper, we show a simpler approach that does not require such scene geometry estimation to create training images.

Our key insight is that state-of-the-art detection methods like Faster-RCNN [39] and even older approaches like DPM [9] etc. care more about local region-based features for detection than the global scene layout. As an example, a cup detector mostly cares about the visual appearance of the cup and its blending with the background, and not so much about where the cup occurs in the scene: the table-top or the ground. We believe that while global consistency is important, only ensuring patch-level realism while composing synthetic datasets should go a long way to train these detectors. We use the term patch-level realism to refer to the observation that the bounding box containing the pasted object looks realistic to the human eye.

However, naively placing object masks in scenes creates subtle pixel artifacts in the images. As these minor imperfections in the pixel space feed forward deeper into the layers of a ConvNet [25], they lead to noticeably different features and the training algorithm focuses on these discrepancies to detect objects, often ignoring to model their complex visual appearance. As our results show (Table 1), such models give reduced detection performance.

Since our main goal is to create training data that is useful for training detectors, we resolve these local imperfections and maintain patch level realism. Inspired from methods in data augmentation and denoising auto encoders [51], we generate data that forces the training algorithm to ignore these artifacts and focus only on the object appearance. We show how rendering the same scene with the same object placement and only varying the blending parameter settings (Section 5.2) makes the detector robust to these subtle pixel artifacts and improves training. Although these images do not respect global consistency or even obey scene factors such as lighting etc., training on them leads to high performance detectors with little effort. Our method is also complementary to existing work [12, 34, 47] that ensures global consistency and can be combined with them.

Data generated using our approach is surprisingly effective at training detection models. Our results suggest that curated instance recognition datasets suffer from poor coverage of the visual appearances of the objects. With our method, we are able to generate many such images with different viewpoints/scales, and get a good coverage of the visual appearance of the object with minimal effort. Thus, our performance gain is particularly noticeable when the test scenes are different from the training scenes, and thus the objects occur in different viewpoints/scales.

2. Related Work

Instance detection is a well studied problem in computer vision. [55] provides a comprehensive overview of the popular methods in this field. Early approaches, such as [6], heavily depend on extracting local features such as SIFT [30], SURF [3], MSER [32] and matching them to retrieve instances [29, 48]. These approaches do not work well for objects which are not ‘feature-rich’, where shape-based methods [10, 19, 21] are more successful.

Modern detection methods [14, 15, 39] based on learned ConvNet features [23, 25, 44] generalize across feature rich and feature poor objects [43]. With the availability of powerful commodity GPUs, and fast detection al-
The use of these powerful methods for object and instance detection requires large amounts of annotated data. This requirement is both impractical and expensive for rapidly deploying detection systems. Synthesizing data is one way to address this issue. [36, 47] use rendered images of objects to do both object detection and pose estimation. They render 3D models of objects from different viewpoints and place them against randomly sampled backgrounds. [34] also highlight the importance of using photorealistic models in training CNNs.

There is a wide spectrum of work where rendered datasets are used for computer vision tasks. At one end, we have datasets with images of single objects on random backgrounds [34–36, 47]. On the other end, there are datasets where the entire scene is rendered [11, 17, 40]. On that spectrum our work lies in between as we do not render the whole world but use real images of both objects and backgrounds to compose new scenes. In this sense, our work closely related to contemporary work from [16] which generates synthetic data for localizing text in scenes.

Sedaghat et al. [42] show how an annotated dataset can be created for the task of object pose estimation by taking videos by walking around the object. [18] uses synthetic data from [4] for multi-view instance recognition. [31] use real and synthetic images for 2D-3D alignment.

Similarly, [5, 50] render 3D humans in scenes and use this data for pose estimation. Tasks requiring dense annotation, such as segmentation, tracking etc. have also shown to benefit by using such approaches [11, 17, 40, 41]. [46] shows a novel approach for collecting data of objects in a closed domain setting. [1, 13, 24] annotate 3D points belonging to an object in the point cloud reconstruction of a scene and propagate the label to all frames where the object is visible. As synthetic data can be significantly different from real images, [49] shows a domain adaptation approach to overcome this issue. In contrast, our work composes training scenes using real object images as well as real background images.

The existing approaches to synthesizing datasets focus largely on ensuring global consistency and realism [5, 12, 22, 50]. While global consistency is important, we believe that local features matter more for training detection systems. Our approach ensures that when we train our detection model it is invariant to local discrepancies.
4. Approach Overview

We propose a simple approach to rapidly collect data for instance detection. Our results show that our approach is competitive with the manual curation process, while requiring little time and no human annotation.

Ideally, we want to capture all of the visual diversity of an instance. Figures 1 and 3 show how a single instance appears different when seen from different views, scales, orientation and lighting conditions. Thus, distinguishing between such instances requires the dataset to have good coverage of viewpoints and scales of the object. Also, as the number of classes increases rapidly with newer instances, the long-tail distribution of data affects instance recognition problems. With synthetic data, we can ensure that the data has good coverage of both instances and viewpoints. Figure 2 shows the main steps of our method:

1. Collect object instance images: Our approach is agnostic to the way the data is collected. We assume that we have access to object images which cover diverse viewpoints and have a modest background.
2. Collect scene images: These images will serve as background images in our training dataset. If the test scenes are known beforehand (like in the case of a smart-home or a warehouse) one can collect images from those scenes. As we do not compute any scene statistics like geometry or layout, our approach can readily deal with new scenes.
3. Predict foreground mask for the object: We predict a foreground mask which separates the instance pixels from the background pixels. This gives us the object mask which can be placed in the scenes.
4. Paste object instances in scenes: Paste the extracted objects on a randomly chosen background image. We ensure invariance to local artifacts while placing the objects so that the training algorithm does not focus on subpixel discrepancies at the boundaries. We add various modes of blending and synthesize the exact same scene with different blending to make the algorithm robust to these artifacts. We also add data augmentation to ensure a diverse viewpoint scale coverage.

5. Approach Details and Analysis

We now present additional details of our approach and provide empirical analysis of our design choices.

5.1. Collecting images

We first describe how we collect object/background images, and extract object masks without human effort.

Images of objects from different viewpoints: We choose the objects present in Big Berkeley Instance Recognition Dataset (BigBIRD) to conduct our experiments. Each object has 600 images, captured by five cameras with different viewpoints. Each image also has a corresponding depth image captured by an IR camera.

Background images of indoor scenes: We place the extracted objects from the BigBIRD images on randomly sampled background images from the UW Scenes dataset [24]. There are 1548 images in the backgrounds dataset.

Foreground/Background segmentation: Once we have collected images of the instances, we need to determine the pixels that belong to the instance vs. the background. We automate this by training a model for foreground/background classification. We train a FCN network [28] (based on VGG-16 pre-trained on PASCAL VOC [8] image segmentation) to classify each image pixel into foreground/background. The object masks from the depth sensor are used as ground truth for training this model. We train this model using images of instances which are not present in our final detection evaluation. We use [2] as a post-processing step to clean these results and obtain an object mask. Figure 5 shows some of these results. In practice, we found this combination to generalize to images of unseen objects with modest backgrounds and give good quality object masks from input images. It also generalizes to transparent objects, e.g., coca cola bottle, where the depth sensor does not work well.

5.2. Adding Objects to Images

After automatically extracting the object masks from input images, we paste them on real background images. Naively pasting objects on scenes results in artifacts which the training algorithm focuses on, ignoring the object’s visual appearance. In this section, we present steps to generate data that forces the training algorithm to ignore these artifacts and focus only on the object appearance. To evaluate these steps empirically, we train a detection model on our synthesized images and evaluate it on a benchmark instance detection dataset (real images).

Detection Model: We use the Faster R-CNN [39] method and initialize the model from a VGG-16 [44] model pretrained on object detection on the MSCOCO [26] dataset.

Benchmarking Dataset: After training the detection model on our synthetic images, we use the GMU Kitchen dataset [13] for evaluation. There are 9 scenes in this dataset. Three dataset splits with 6 scenes for training and 3 for testing have been provided in [13] to conduct experiments on the GMU Kitchen dataset. We follow these splits for train/test and report the average over them. No images or statistics from this dataset are used for either dataset synthesis or training the detector. We report Mean Average Precision (mAP) at IOU of 0.5 [8] in all our experiments.

5.2.1 Blending

Directly pasting objects on background images creates boundary artifacts. Figure 6 shows some examples of such
Table 1: We analyze the effect of various factors in synthesizing data by generating data with different settings and training a detector [39]. We evaluate the trained model on the GMU Dataset [13]. As we describe in Section 5, these factors greatly improve the quality of the synthesized data.

| Blending (Sec 5.2.1) | 2D Rot. | 3D Rot. | Trunc. | Occl. | cola | mate | coffee | mate | honey | bunches | sauce | mahatma | rice | nature | v1 | nature | v2 | palmolive | orange | pop | secret | pringles | red | bull | mAP |
|-----------------------|---------|---------|--------|-------|------|------|--------|------|-------|---------|-------|---------|-----|--------|---|--------|----|-----------|-------|-----|--------|----------|-----|------|-----|
| No blending           | ✓       | ✓       | ✓      | ✓     | 65.7 | 91.1 | 83.2   | 59.8 | 57.7  | 92.1    | 84.4  | 61.4    | 59.0| 38.7   | 31.9| 65.9   |
| Gaussian Blurring     | ✓       | ✓       | ✓      | ✓     | 65.3 | 88.1 | 80.8   | 67.5 | 63.7  | 90.8    | 79.4  | 57.9    | 58.9| 65.7   | 40.2| 68.9   |
| Poisson [37]          | ✓       | ✓       | ✓      | ✓     | 62.9 | 82.9 | 63.9   | 59.4 | 20.7  | 84.6    | 67.9  | 60.9    | 73.5| 41.0   | 25.1| 58.4   |
| All Blend             | ✓       | ✓       | ✓      | ✓     | 76.0 | 90.3 | 79.9   | 65.4 | 67.3  | 93.4    | 86.6  | 64.5    | 73.2| 60.4   | 39.8| 72.4   |
| All Blend + same image| ✓       | ✓       | ✓      | ✓     | 78.4 | 92.7 | 81.8   | 66.2 | 69.8  | 93.0    | 82.9  | 65.7    | 76.0| 62.9   | 41.2| 73.7   |
| Data Aug. (See 5.2.2) |         |         |        |       |      |      |        |     |       |         |      |         |     |        |    |         |    |           |    |
| No 2D Rotation        | ✓       | ✓       | ✓      |       | 63.3 | 90.4 | 81.4   | 63.7 | 54.2  | 91.8    | 82.3  | 59.2    | 71.3| 68.2   | 41.4| 69.7   |
| No 3D Rotation        | ✓       | ✓       | ✓      | ✓     | 73.1 | 90.6 | 83.2   | 63.3 | 55.8  | 93.4    | 82.1  | 65.9    | 64.5| 45.7   | 33.6| 68.3   |
| No Trunc.             | ✓       | ✓       | ✓      | ✓     | 73.4 | 92.1 | 77.8   | 59.9 | 64.6  | 92.4    | 84.6  | 62.0    | 74.2| 67.4   | 41.7| 71.8   |
| No Occlusion          | ✓       | ✓       | ✓      | ✓     | 63.1 | 84.9 | 74.4   | 64.5 | 50.8  | 76.9    | 67.6  | 55.7    | 69.0| 58.7   | 28.1| 63.1   |
| All                   | ✓       | ✓       | ✓      | ✓     | 78.4 | 92.7 | 81.8   | 66.2 | 69.8  | 93.0    | 82.9  | 65.7    | 76.0| 62.9   | 41.2| 73.7   |
| All + Distractor      | ✓       | ✓       | ✓      | ✓     | 81.0 | 93.3 | 85.6   | 55.6 | 73.8  | 94.9    | 87.1  | 68.7    | 79.5| 77.1   | 42.0| 76.2   |

Figure 5: Given an image of a new unseen object instance, we use a ConvNet to predict foreground/background pixels. Using these predictions we automatically obtain an object mask. This method generalizes to transparent surfaces where traditional methods relying on depth sensors for segmentation fail (second row).

artifacts. Although these artifacts seem subtle, when such images are used to train detection algorithms, they give poor performance as seen in Table 1. As current detection methods [39] strongly depend on local region-based features, boundary artifacts substantially degrade their performance.

The blending step ‘smoothes’ out the boundary artifacts between the pasted object and the background. Figure 6 shows some examples of blending. Each of these modes add different image variations, e.g., Poisson blending [37] smoothes edges and adds lighting variations. Although these blending methods do not yield visually ‘perfect’ results, they improve performance of the trained detectors. Table 1 lists these blending methods and shows the improvement in performance after training on blended images.

To make the training algorithm further ignore the effects of blending, we synthesize the exact same scene with the same object placement, and only vary the type of blending used. We denote this by ‘All Blend + same image’ in Table 1. Training on multiple such images where only the blending factor changes makes the training algorithm invariant to these blending factors and improves performance by 8 AP points over not using any form of blending.

5.2.2 Data Augmentation

While pasting the objects on background, we also add the following modes of data augmentation:

2D Rotation: The objects are rotated at uniformly sampled
random angles in between 30 to $-30$ degrees to account for camera/object rotation changes. Table 1 shows a gain of 3 AP points by adding this augmentation.

**3D Rotation:** As we can control this step, we have many images containing atypical 3D rotations of the instances which is hard to find in real data. Table 1 shows a gain of more than 4 AP points because of this augmentation. In Section 6.2 and Figure 7, we show examples of how a model trained on human collected data consistently fails to detect instances from certain viewpoints because the training data has poor viewpoint coverage and different biases from the test set. This result shows the value of being able to synthesize data with diverse viewpoints.

**Occlusion and Truncation:** Occlusion and truncation naturally appear in images. They refer to partially visible objects (such as those in Figure 2). We place objects at the boundaries of the images to model truncation, ensuring at least 0.25 of the object box is in the image. To add occlusion, we paste the objects with partial overlap with each other (max IOU of 0.75). Like other modes of augmentation, we can easily vary the amount of truncation/occlusion. As Table 1 shows, adding truncation/occlusion improves the result by as much as 10 AP points.

**Distractor Objects:** We add distractor objects in the scenes. This models real-world scenarios with multiple distractor objects. We use additional objects from the BigBIRD dataset as distractors. Presence of synthetic distractors also encourages the learning algorithm to not only latch on to boundary artifacts when detecting objects but also improves performance by 3 AP points.

### 6. Experiments

We now compare the effectiveness of our synthesized data against human annotated data on two benchmark datasets. We first describe our common experimental setup.

**Synthesized Data:** We analyze our design choices in Section 5 to pick the best performing ones. We use a total of 33 object instances from the BigBIRD Dataset [45] overlapping with the 11 instances from GMU Kitchen Dataset [13] and the 33 instances from Active Vision Dataset [1]. We use a foreground/background ConvNet (Section 5.1) to extract the foreground masks from the images. The foreground/background ConvNet is not trained on instances we
use to evaluate detection. As in Section 5, we use backgrounds from the UW Scenes Dataset [24] We generate a synthetic dataset with approximately 6000 images using all modes of data augmentation from Section 5. We sample scale, rotation, position and the background randomly. Each background appears roughly 4 times in the generated dataset with different objects. To model occlusions we allow a maximum IOU of 0.75 between objects. For truncations, we allow at least 25% of the object box to be in the image. For each scene we have three versions produced with different blending modes as described in Section 5.2.1. Figure 4 shows samples of generated images. We use this synthetic data for all our experiments. The code used for generating scenes is available at: https://goo.gl/imXRt7.

**Model:** We use a Faster R-CNN model [39] based on the VGG-16 [44] pre-trained weights on the MSCOCO [26] detection task. We initialize both the RPN trunk and the object classifier trunk of the network in this way. We fine-tune on different datasets (both real and synthetic) and evaluate the model’s performance. We fine-tune all models for 25K iterations using SGD+momentum with a learning rate of 0.001, momentum 0.9, and reduce the learning rate by a factor of 10 after 15K iterations. We also use weight decay of 0.0005 and dropout of 0.5 on the fully-connected layers. We set the value of all the loss weights (both RPN and classification) as 1.0 in our experiments. We ensure that the model hyperparameters and random seed do not change across datasets/experiments for consistency.

**Evaluation:** We report Average Precision (AP) at IOU of 0.5 in all our experiments for the task of instance localization. Following [1], we consider boxes of size at least 50 × 30 pixels in the images for evaluation.

### 6.1. Training and Evaluation on the GMU Dataset

Similar to Section 5, we use the GMU Kitchen Dataset [13] which contains 9 kitchen scenes with 6,728 images. We evaluate on the 11 objects present in the dataset overlapping with the BigBIRD [45] objects. We additionally report results from [12]. Their method synthesizes images by accounting for global scene structure when placing objects in scenes, e.g., ensure that cups lie on flat surfaces like table tops. In contrast, our method does not use take into account such global structure, but focuses on patch-level realism. We note that their method [12] uses a different background scenes dataset for their synthesis.

Table 2 shows the evaluation results. We see that training on the synthetic data is competitive with training on real images (rows 1 vs 3) and also outperforms the synthetic dataset from [12] (rows 2 vs 3). Combining synthetic data with the real data shows a further improvement for all synthetic image datasets (rows 4, 5). These results show that the data generated by our approach is not only competitive with both real data and existing synthetic data, but also provides complementary information. Figure 9 shows qualitative examples illustrating this point.

### 6.2. Evaluation on the Active Vision Dataset

To test generalization across datasets, we now present experiments where we train on either our synthetic data or the GMU Dataset [13], and evaluate on the Active Vision Dataset [1]. The Active Vision Dataset [1] has 9 scenes and 17,556 images. It has 33 objects in total and 6 objects in overlap with the GMU Kitchen Scenes. We use these 6 objects for our analysis. We do not use this dataset for training.

We train a model trained on all the images from the GMU Dataset (Section 6.1). This model serves as a base-

| Dataset | coca cola coffee mate honey bunches hunt’s sauce mahatma rice nature v1 nature v2 pulmolive orange pop secret pringles bbq red | mAP |
|---------|---------------------------------------------------------------------------------------------------------------------------------|-----|
| Real Images from GMU | 81.9 95.3 92.0 87.3 86.5 96.8 88.9 | 80.5 |
| SP-BL-SS [12] | 55.5 67.9 71.2 34.6 30.6 82.9 66.2 | 33.1 |
| (Ours) Synthetic Images | 81.0 93.3 85.6 55.6 73.8 94.9 87.1 | 68.7 |
| SP-BL-SS + Real Images [12] | 82.6 92.9 91.4 85.5 81.9 95.5 88.6 | 78.5 |
| (Ours) Synthetic + Real Images | 88.5 95.5 94.1 88.1 90.3 97.2 91.8 | 80.1 |
| (Ours) Synthetic + Real Images | 81.0 93.3 85.6 55.6 73.8 94.9 87.1 | 68.7 |

Table 3: Evaluation on the entire Active Vision dataset by varying the amount of real data from the GMU Kitchen Scenes train dataset

| Dataset | coca cola honey bunches hunt’s sauce mahatma rice nature v2 pop secret mAP |
|---------|--------------------------------------------------------------------------------|-----|
| Real Images | 57.7 34.4 48.0 39.9 24.6 46.6 41.9 |
| Synthetic | 63.0 29.3 34.2 20.5 49.0 23.0 36.5 |
| Synthetic + Real Images | 69.9 44.2 51.0 41.8 49.5 50.9 51.1 |
| 10% Real | 15.3 19.1 31.6 11.2 6.1 11.7 15.8 |
| 10% Real + Syn | 66.1 36.5 44.0 26.4 48.9 37.6 43.2 |
| 40% Real | 55.8 31.6 47.3 27.4 24.8 41.9 38.2 |
| 40% Real + Syn | 69.8 41.0 55.7 38.3 52.8 47.0 50.8 |
| 70% Real | 55.3 30.6 47.9 36.4 25.0 41.2 39.4 |
| 70% Real + Syn | 67.5 42.0 50.9 43.0 48.5 51.8 50.6 |
Figure 9: We show qualitative detection results and mark true positives in green, false positives in red and arrows to highlight regions. The top two rows are from the GMU Kitchen Scenes [13] and the bottom two rows from the Active Vision Dataset [1]. (a), (b): Model trained on real data misses objects which are heavily occluded (a) or stops detecting objects as viewpoint changes from a to b. (c), (d): Model trained on synthetic data detects occluded and truncated objects. (e): Combining synthetic data removes false positives due to training only on real data. (g), (h): Combining real data removes false positives due to training only on synthetic data. (f), (h): Viewpoint changes cause false negatives. (Best viewed electronically)

Failure modes of real data: Upon inspecting the errors [20] made by the GMU model, we see that a common error mode of the detector is its failure to recognize certain views in the test-set (see Figure 7). These viewpoints were sparsely present in the human annotated training data. In contrast, our synthetic training data has a diverse viewpoint coverage. The model trained on the synthesized images drastically reduces these errors. Combining the synthesized images with the real images from GMU gives a further improvement of 10 AP points suggesting that synthesized images do provide complementary information.

Varying Real Data: We investigate the effect of varying the number of real images combined with the synthesized data. We randomly sample different amounts of real images from the GMU Dataset and combine them with the synthetic data to train the detector. As a baseline we also train the model on varying fractions of the real data. Table 3 shows that by adding synthetic images to just 10% of the real images we get a boost of 10 AP points over just using real images. This performance is also tantalizingly close to the performance of combining larger fractions of real data. This result reinforces the effectiveness and complementary nature of our approach. In the supplementary material, we present additional such results.

7. Discussion and Future Work

We presented a simple technique to synthesize annotated training images for instance detection. Our key insights were to leverage randomization for blending objects into scenes and to ensure a diverse coverage of instance viewpoints and scales. We showed that patch-based realism is sufficient for training region-proposal based object detectors. Our method performs favorably to existing hand curated datasets and captures complementary information. In a realistic cross-domain setting we show that by combining just 10% of the available real annotations with our synthesized data, our model performs better than using all the real annotations. From a practical standpoint our technique affords the possibility of generating scenes with non-uniform distributions over object viewpoints and scales without additional data collection effort.

We believe our work can be combined with existing approaches [12] that focus on global consistency for placing objects and [22] which model realism. Future work should focus on a combination of such approaches.

Acknowledgements: The authors are grateful to Georgios Georgakis and Phil Ammirato for their help with the datasets and discussions. This work was supported in part by NSF Grant CNS1518865.
S. Song, L. Zhang, and J. Xiao. Robot in a room: Toward
A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel.
Bigbird: A large-scale 3d database of object instances. In
Robotics and Automation (ICRA), 2014 IEEE International
Conference on, pages 509–516. IEEE, 2014.
K. Simonyan and A. Zisserman. Very deep convolutional
networks for large-scale image recognition. arXiv preprint
arXiv:1409.1556, 2014.
A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel.
Bigbird: A large-scale 3d database of object instances. In
Robotics and Automation (ICRA), 2014 IEEE International
Conference on, pages 509–516. IEEE, 2014.
S. Song, L. Zhang, and J. Xiao. Robot in a room: Toward
perfect object recognition in closed environments.
Figure 10: A few randomly chosen samples from our synthesized images.