Synthesizing Scenes for Instance Detection

Technical Report Number
CMU-RI-TR-17-21

a dissertation presented
by
Debidatta Dwibedi
to
The Robotics Institute

in partial fulfillment of the requirements
for the degree of
Master of Science
in the subject of
Robotics

Carnegie Mellon University
Pittsburgh, Pennsylvania
May 2017

Thesis advisor: Professor Martial Hebert Debidatta Dwibedi

Synthesizing Scenes for Instance Detection

Abstract

Object detection models have made significant progress in recent years. A major impediment in rapidly deploying these 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. The brute force data collection approach would require a lot of manual effort for each new environment with new instances. In this thesis, we explore three methods to tackle the above problem. First, we present how we can use object tracking in videos to propagate bounding box annotations from one frame to the subsequent frames. Next, we show how 3D reconstruction can be used to produce annotations for object detection and pose estimation. Finally, we present a novel approach for generating synthetic scenes with annotations for instance detection. Our key insight is that ensuring only patch-level realism provides enough training signal for current object detector models. 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 to real data. Our results show that we outperform existing synthesis approaches and that the complementary information contained in our synthetic data when combined with real data improves performance by more than 10 AP points on benchmark datasets.

Contents

Listing of figures

Dedicated to my parents and sister.

The famous pipe. How people reproached me for it! And yet, could you stuff my pipe? No, it's just a representation, is it not? So if I had written on my picture 'This is a pipe', I'd have been lying!

René Magritte

1 Introduction

Researchers in computer vision have long sought to build a model of the world from visual input alone. A small but significant sub-task in that massive undertaking is to be able to represent objects. Over the years many diverse ideas have been suggested regarding what might be the correct way to model objects in computer vision. Considered in the context of research in object detection, the innocuous line, *"Ceci n'est pas une pipe"*(This is not a pipe), scribbled by the surrealist artist René Magritte below the picture of a pipe in the painting titled *The Treachery of Images* takes new meaning. He was referring to the fact that what he had drawn was merely the picture of a pipe which was different from the actual pipe itself. Human perception is good enough to recognize the concept of a pipe from an artistic representation of it and recognize yet unseen pipes. The same is expected from computer vision systems today. The ideal scenario would be able to do so with minimal supervision.

Imagine using an object detection system for an environment like your kitchen. Such a system needs to recognize different kinds of objects, and also be able to 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` , 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 the real world settings of robotics or other personalized applications.

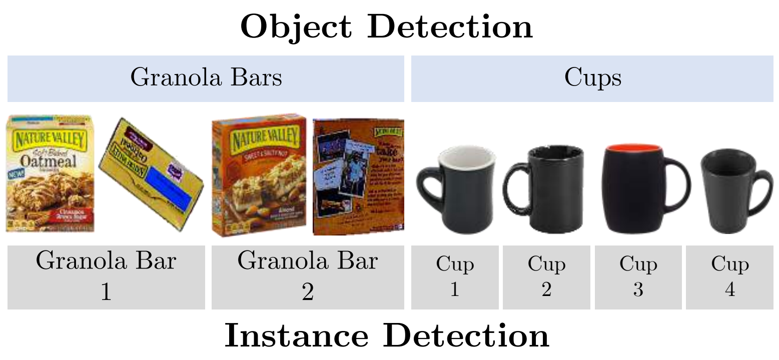

Figure 1.1: Object vs Instance Detection. Instance detection involves fine-grained recognition within the same 'object category'. In this example, instance recognition must distinguish amongst 6 classes: 2 types of granola bars and 4 types of coffee cups. Object detection would distinguish only amongst 2 classes: coffee cups and granola bars. Instance detection must model minute differences between instances of the same generic object category.

1.1 Instance Detection

Instance detection requires accurate localization of a particular object, a particular brand of cereal, a particular cup . In contrast, generic object detection detects an entire generic category like a cereal box or a cup (see Figure 1.1). In fact, in the instance detection scenario correctly localizing a cereal box of some other brand is counted as a mistake. Instance detection occurs commonly in robotics, AR/VR where one is interested in distinguishing instances, *your* cup vs. *my* cup, rather than the entire category. Thus, instance detection can also be viewed as fine-grained recognition.

1.2 Traditional Dataset Collection

Building detection datasets involves a data curation step and an annotation step. Data curation involves collecting internet images for generic detection datasets [8,26]. This fails for instance datasets as finding many internet images of the instances of interest is not easy. For instance detection [45] data curation involves placing the instances in varied background and manually collecting the images. Manually collecting these images requires one to pay attention to ensure diversity in images by placing the object in different backgrounds and collecting different viewpoints. The annotation step is generally crowd sourced. Depending on the type of data, human annotations can be augmented with object tracking or 3D sensor information[24,13,1,46,52].

Unfortunately, both these steps are not suitable for *rapidly* gathering instance annotations. Firstly, as we show in our experiments, even if we limit ourselves to the same *type* of scene, , kitchens, the curation step can lack diversity and create biases that do not hold true in the test setting. Secondly, as the number of images and instances increase, manual annotation requires additional time and expense.

1.3 Organization

In this thesis, we tackle the problem of the human effort involved in collecting and annotating images. In Chapter 2, we describe how we can leverage videos to reduce the number of bounding boxes to be labeled. In Chapter 3, we explore how 3D reconstruction can be used to produce both bounding box annotations and pose labels. In Chapter 4, we go a step further and show how human labeling can be eliminated by synthesizing scenes by cropping objects from videos. We showcase how these methods of data collection stack up against data annotated by humans.

We must perceive in order to move, but we must also move in order to perceive

J.J. Gibson

2 Leveraging Videos for Object Detection

Video collection is effortless given the advent of the high quality portable cameras on mobile phones. Another advantage of using videos for dataset creation is the ease with which multiple views of the object can be captured. It is necessary that multiple viewpoints of the object be collected for training object detectors because we want to capture the diversity in appearance of instances due to viewpoint variations. Since it is easy for a person to collect different viewpoints of the object by recording a video, we intend to collect training data by leveraging the temporal coherence present in videos.

2.1 Current Approach

Manual annotation is usually done by Mechanical Turk workers online. A dataset consisting of training images is hosted online. The workers access the images by the Mechanical Turk interface and annotate the images. The collected data serves as the ground truth for training machine learning algorithms.

Instead of using images, we propose to use videos. The current approach does not take into account the temporal coherence present in videos. Annotations made in one frame can be used to provide information regarding the previous and consecutive neighbouring frames.

2.2 Proposed Approach

In order to take advantage of the temporal coherence present in videos we propose the following approach. To collect the video, the object is kept in view of the camera as a person walks around it to collect data from all viewpoints. The object is not kept in isolation but along with other objects. An advantage of using videos however is that since the consecutive frames are correlated, there is a possibility of reusing the labels and bounding boxes marked in one frame in the subsequent frames by tracking the object in the bounding box. Since we do not have a perfect model of the object right now the tracking might not track the object throughout the entire video. However, we can still track the bounding box for some frames. When the tracking fails, the annotator can redraw the new bounding

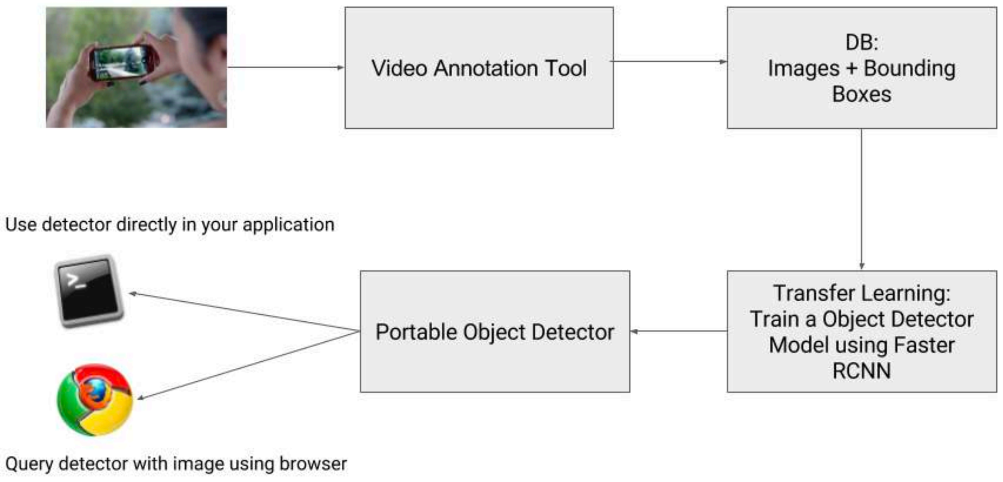
Video Annotation Tool
DB: Images + Bounding Boxes
Use detector directly in your application
Portable Object Detector
Transfer Learning: Train a Object Detector Model using Faster RCNN
Query detector with image using browser

Figure 2.1: We modified the classic object detection pipeline to include a video annotation tool The tool is useful to propagate labels across multiple frames by using object tracking. This reduces the effort involved in labeling images.

box from where tracking can begin again. From the point of view of providing training data to an object detector, the fact that the frames are correlated poses a problem. If the frames are near duplicates then our machine learning algorithm won't benefit much from having similar looking images in the training set. We decide to use a perceptual hash algorithm to check for similarity of images and discard near duplicates from the training set.

Our suggested pipeline can be seen in Fig. 2.1 We modified a publicly available video annotation tool called VATIC to track objects in bounding box in consecutive frames. It automatically annotates the images where the tracking is successful while adjusting the location and size of the bounding box. A human is still required in the loop because the tracker might still be confident in its tracking and label some frames in a wrong manner. For tracking, we have the option to use either:

1. Linear interpolation between manually annotated frames
2. Tracking-Learning-Detection which also models the present appearance of the object being

tracked

2.3 Results

We wanted to gauge how effective this technique might be for the task of object detection. To do so, we build a keyboard detector by collecting 5 videos of keyboards. In our experiments, we found that a person labeled approximately 100 different frames out of 4500 frames. After discarding the duplicates, we end up with approximately 1000 distinct frames. The pipeline of training an object detector in this fashion will not require any coding and can be done using a web interface.

We trained a keyboard detector from the 1000 images collected from 5 videos of the keyboard in different settings. The choice of detector is Faster RCNN[39] which is the state of the art approach in object detection right now. The choice of detector does not affect the data collection method. One can see the results of object detection in 3 test clips in which the keyboard is placed in the office under different lighting conditions here.

2.4 Discussion

The approach in this section modified an existing method of data collection to reduce annotation effort by about 10 times. However, these experiments laid the foundation of our work on using structure from motion and synthesizing images for object detection which forgoes human annotation effort altogether.

Reconstruction helps recognition

Jitendra Malik

3

Generating Pose and Bounding Box Annotations from 3D Reconstruction

Like the previous chapter, we explore how we can use videos to reduce the manual effort involved in annotation. However, instead of requiring any human annotation we try to avoid human annotations altogether.

3.1 Current Approach

In PASCAL3D[55], the authors suggest the use of a GUI to find a 3D model from a database that matches the object in the image and use a series of operations like zoom, rotate etc. to align the 3D model to the image. This alignment allows one to annotate objects in images with 3D pose. More recently, researchers have produced pose annotations by rendering the 3D model using Render-for-CNN[47]. The disadvantages of the approaches are as follows:

1. Too much manual effort in using the GUI
2. No real images in the Render-for-CNN dataset
3. High-quality 3D object models might not be available for rendering

The above approaches are suitable for scenarios where one doesn't have access to objects while capturing the images. However, for our use case we can assume we have access to the instances.

3.2 Proposed Approach

In our approach we propose to use Structure from From(SFM)[54] to produce datasets with pose and bounding box annotations.

Our approach(See Fig 3.1) is as follows:

1. Record the object from all viewpoints
2. Initially features in match sequential frames
3. Merge matched frames by performing bundle adjustment
4. We get 3D model of object and all the cameras' poses

5. Remove the 3D points of the support surface by fitting a 3D plane
6. Remove outliers by finding connected components and retaining largest one
7. For each image, we re-project the 3D points to find bounding box as we know projection matrix for each camera

Our method is most similar to that described in[42]. However, in our case we are creating datasets for instance detection and do not need to perform the pairwise alignment of 3D models described in the above paper.

3.3 Results

In Fig 3.2, we show some examples of pose and bounding box annotations generated by our approach based on SFM. This approach provides significant advantages over traditional approaches because it removes the requirement of humans labeling bounding boxes or aligning 3D models. However, there is a step which involves estimating the support surface. This is based on a heuristic that the biggest plane in the 3D reconstructed point cloud will belong to the support surface. We propose to use a deep learning based segmentation approach that estimates the salient object more robustly.

The proposed approach is not restricted to videos collected by walking around objects. To ease the process of dataset curation, objects can be placed on a turntable and multiple views of the object can be captured by rotating the turntable. Visual SFM is agnostic to the fact how the multi-view images are collected.

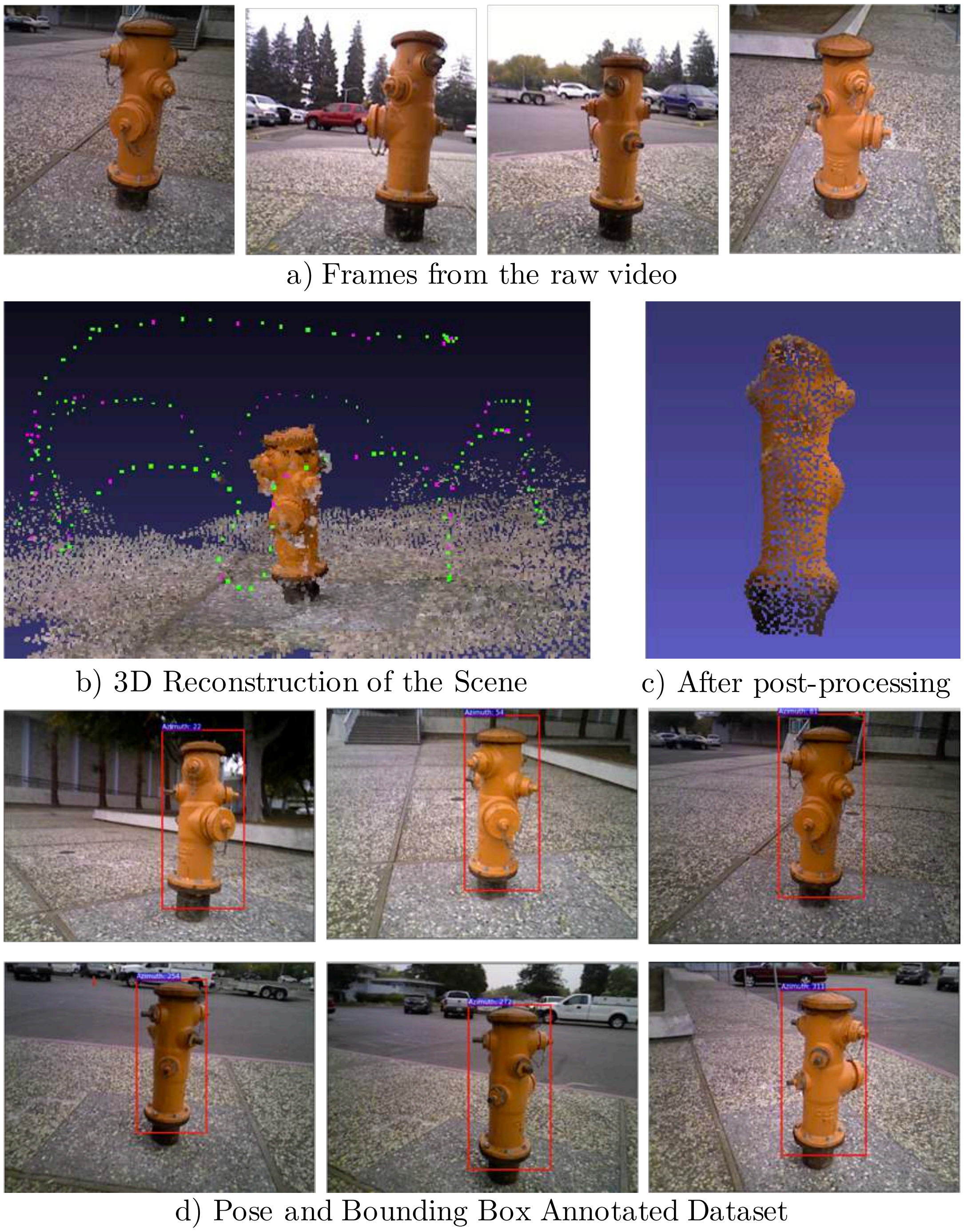
a) Frames from the raw video
b) 3D Reconstruction of the Scene
c) After post-processing
Azimuth: 22
Azimuth: 54
Azimuth: 81
Azimuth: 254
Azimuth: 272
Azimuth: 311
d) Pose and Bounding Box Annotated Dataset

Figure 3.1: Pipeline for generating annotations for pose and bounding box from videos using Structure from Motion.

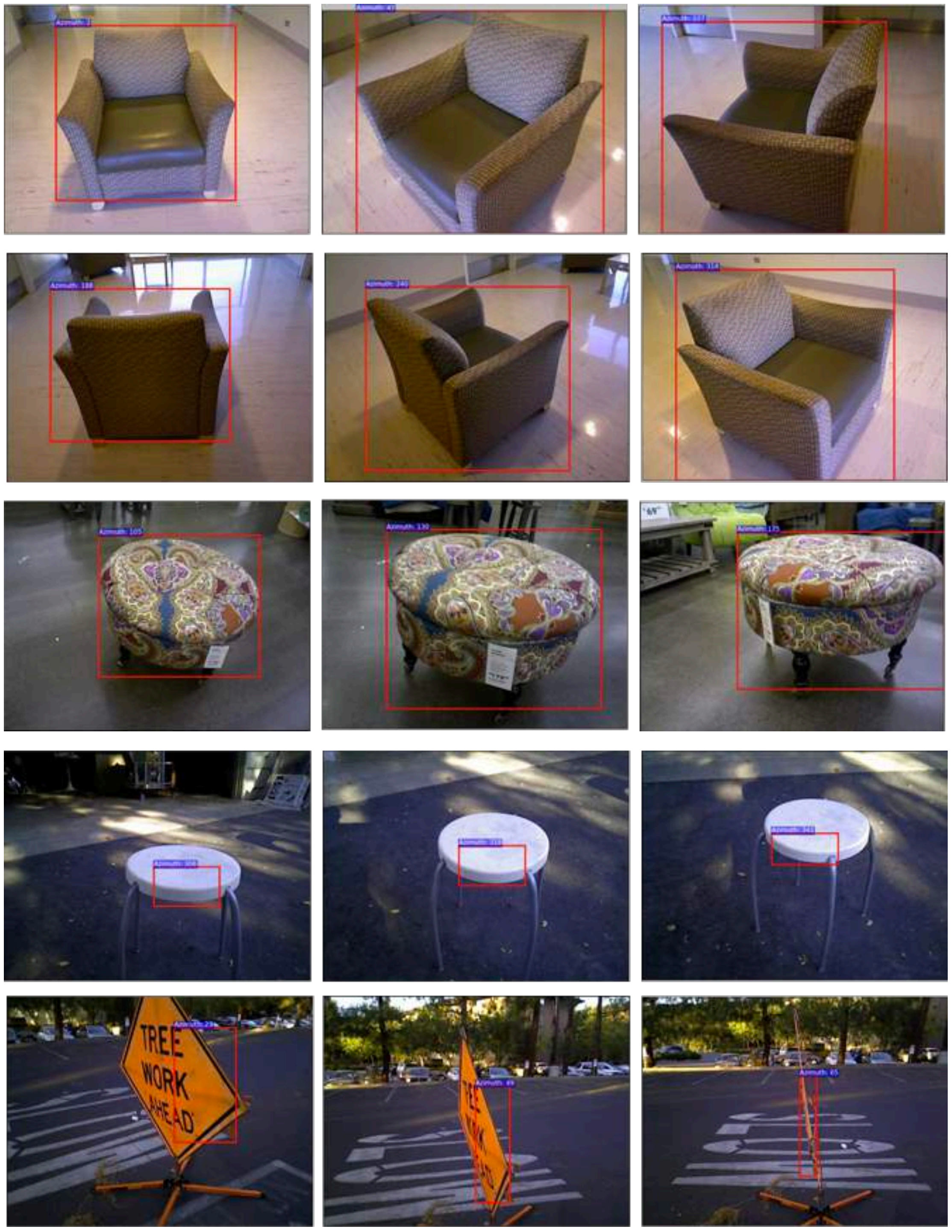

Figure 3.2: Examples of datasets generated by our approach. The bottom rows show failure cases where the bounding box has been wrongly predicted. The errors are introduced due to the support surface estimation step.

I don't want realism. I want magic!

Tennessee Williams

4 Synthesizing Scenes for Instance Detection

4.1 Introduction

In this chapter, we explore a novel approach for dataset generation that aims to reduce the annotation effort required. Recently, a successful research direction to overcome this barrier is to use synthetically rendered scenes and objects[47,34,22] 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 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. 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] 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.

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 4.1), 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 local consistency parameter settings (Section 4.4.2) 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 , training on them leads to high performance detectors with little effort. Our method is also complementary to existing work[34,47,12] 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.

4.2 Related Work

Instance detection is a well studied problem in computer vision.[57] 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[21,10,19] are more successful.

Modern detection methods[14,39,15] based on learned ConvNet features[23,44,25] generalize across fea-

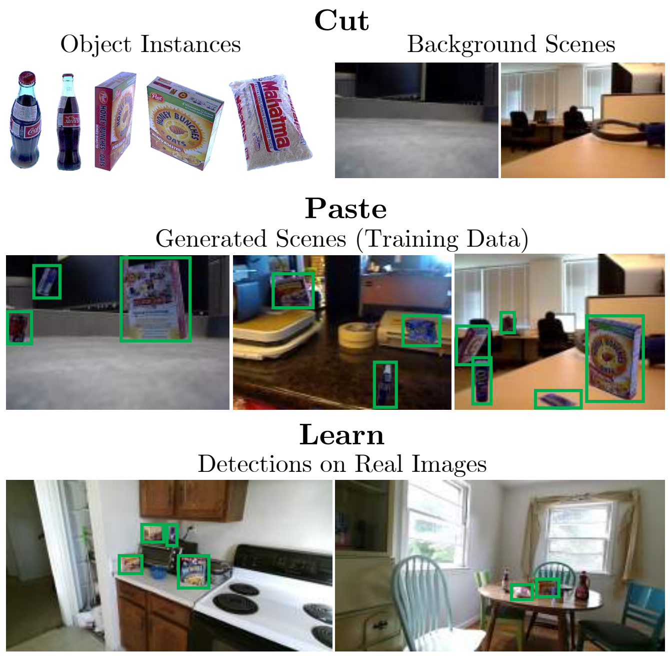

Figure 4.1: We present a straightforward way to rapidly generate training images for instance detection with minimal human effort. We automatically extract object instance masks and render it on random background images to create realistic training images with bounding box labels. Our results show that such data is competitive with human curated datasets, and contains complementary information.

ture rich and feature poor objects[43]. With the availability of powerful commodity GPUs, and fast detection algorithms[27,38], these methods are suitable for real-time object detection required in robotics. More recently, deep learning based approaches in computer vision are being adopted for the task of pose estimation of specific objects[33,56,53]. Improving instance detection and pose estimation in warehouses will be signifcantly useful for the perception pipeline in systems trying to solve the Amazon Picking Challenge[7].

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. Sythesizing data is one way to address this issue. Recently, researchers[47,36] showed how to 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 photo-realsitic 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[47,36,34,35]. On the other end, there are datasets where the entire scene is rendered[11,40,17]. 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 [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,[50,5] render 3D humans in scenes and use this data for pose estimation. Tasks requiring dense annotation, such as segmentation, tracking have also shown benefit by using such approaches[17,11,41,40]. [46] show a novel approach for collecting data of objects in a closed domain setting.[24,13,1] 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] show a domain adaptation approach to overcome this issue. In contrast, our work composes training images using real object and background images.

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

4.3 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 the variations that cause an instance to appear different. Figures 4.1, 1.1 show how a single instance of an object appears drastically 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 4.2 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.

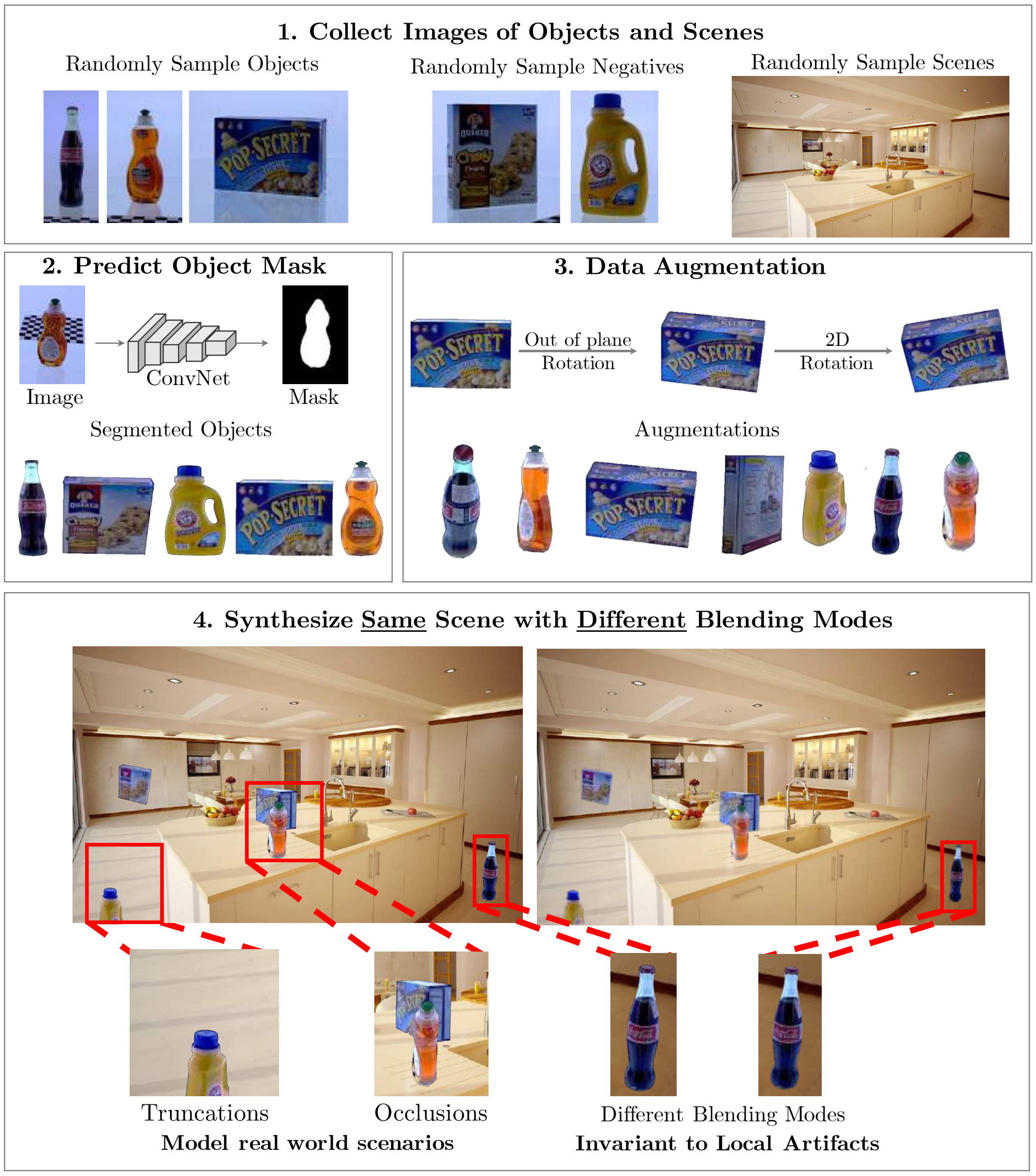

Figure 4.2: We present a simple approach to rapidly synthesize datasets for instance detection without human annotation. We start with a set of images of the instances and background scenes. We then automatically extract the object mask and segment the object. We paste the objects on the scenes with data augmentation removing pixel artifacts created by pasting. Training on this synthesized data provides competitive results to training on real data. We also show how synthetic data provides complementary information to real data.

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.

4.4 Approach Details and Analysis

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

4.4.1 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)[45] 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.

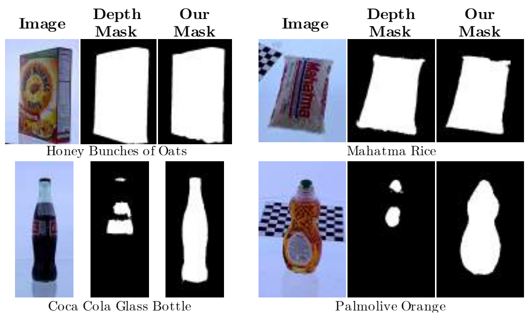

Figure 4.3: Given an image of a new unseen object instance, we use a ConvNet to predict foreground / background pixels. Using these predictions we automatically obtain an object mask. This method generalizes to transparent surfaces where traditional methods relying on depth sensors for segmentation fail (second row).

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[44] 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 4.3 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, , `coca cola` bottle, where the depth sensor does not work well.

4.4.2 Adding Objects to Images

After automatically extracting the object masks from input images, we paste them on real background images. Naïvely 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 pre-trained 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

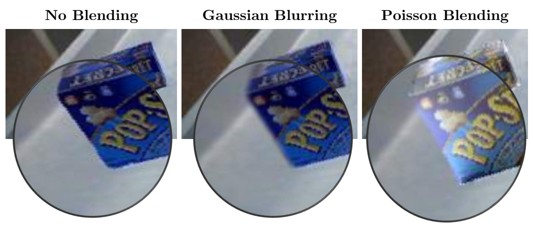

Figure 4.4: Different blending modes used while generating datasets. These modes help the model in ignoring artifacts arising from pasting objects on background scenes. More details in Section 4.4.2

IOU of 0.5 [?] in all our experiments.

Blending

Directly pasting objects on background images creates boundary artifacts. Figure 4.4 shows some examples of such artifacts. Although these artifacts seem subtle, when such images are used to train detection algorithms, they give poor performance as seen in Table 4.1. As current detection methods[39] strongly depend on local region-based features, boundary artifacts substantially degrade their performance.

The blending step 'smoothens' out the boundary artifacts between the pasted object and the background. Figure 4.4 shows some examples of blending. Each of these modes add different image variations, , Poisson blending[37] smooths edges and adds lighting variations. Although these blending methods do not yield visually 'perfect' results, they improve performance of the trained detectors. Table 4.1 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 img’ in Table 4.1. 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.

Data Augmentation

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

1. 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 4.1 shows a gain of 3 AP points by adding this augmentation.

2. 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 4.1 shows a gain of more than 4 AP points because of this augmentation. In Section 4.5.2 and Figure 4.6, 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.

3. Occlusion and Truncation

   Occlusion and truncation naturally appear in images. They refer to partially visible objects (see Figure 4.2 for examples). 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 4.1 shows, adding truncation/occlusion improves the result by as much as 10 AP points.

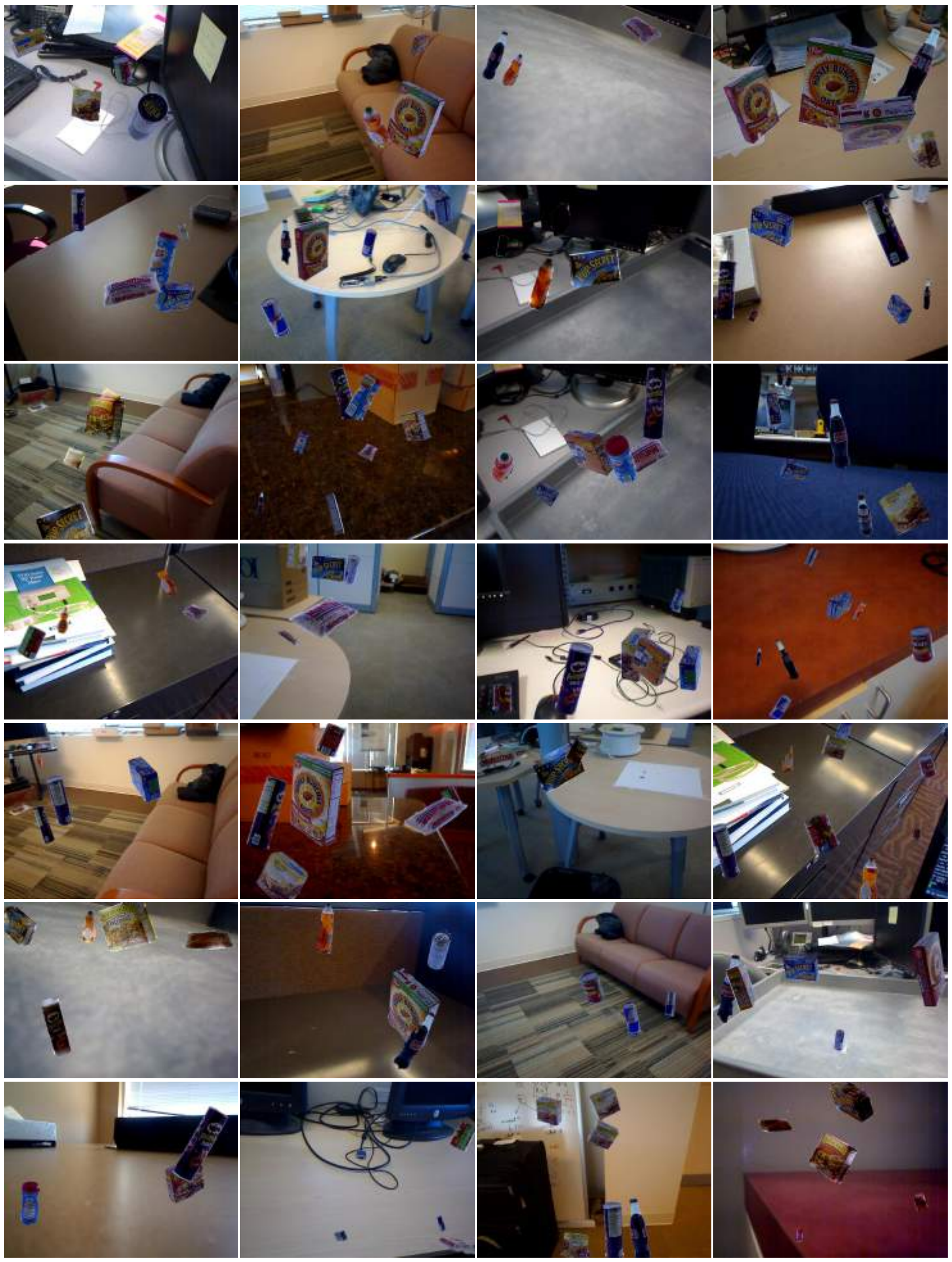

Figure 4.5: A few randomly chosen samples from our synthesized images. We describe the details of our approach in Section 4.4.

4. 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. Adding such data improves performance by 3 AP points.

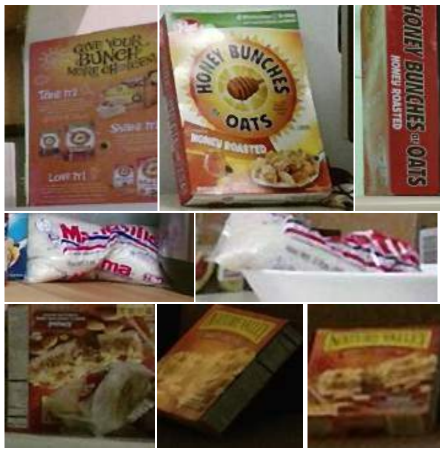

Figure 4.6: Missed detections on the Active Vision Dataset [1] for a model trained on the hand-annotated GMU Dataset [13]. The model consistently fails to detect certain viewpoints as the training data has poor viewpoint coverage and has biases different from the test set. Each row shows a single instance.

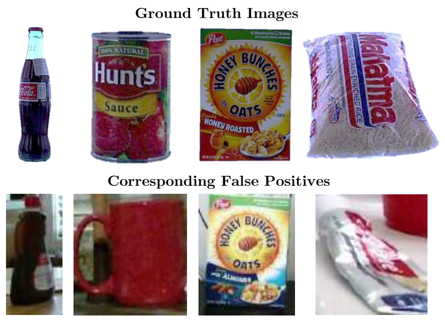

Figure 4.7: Examples of false positives from the UNC dataset by the detector trained on the hand-annotated bounding boxes from the GMU dataset. Object detectors trained on hand annotated scenes also need new negatives to be able to perform well in newer scenes.

Table 4.1: 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 4.4, these factors greatly improve the quality of the synthesized data.

	2D Rot.	3D Rot.	Trunc.	Occl.	coca cola	coffee mate	honey bunches	hunt's sauce	mahatma rice	nature v1	nature v2	palmolive orange	pop secret	pringles bbq	red bull	mAP
Blending (Sec 4.4.2)																
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 img.	✓	✓	✓	✓	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. (Sec 4.4.2)																
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

4.5 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 4.4 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 4.4.1) 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 4.4, 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 4.4. 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 overlap of 0.75 between objects. For truncations, we allow at least 0.25 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 4.4.2. Figure 4.5 shows samples of generated images. We use this synthetic data for all our experiments.

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.

4.5.1 Training and Evaluation on the GMU Dataset

Similar to Section 4.4, 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, , 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 local consistency.

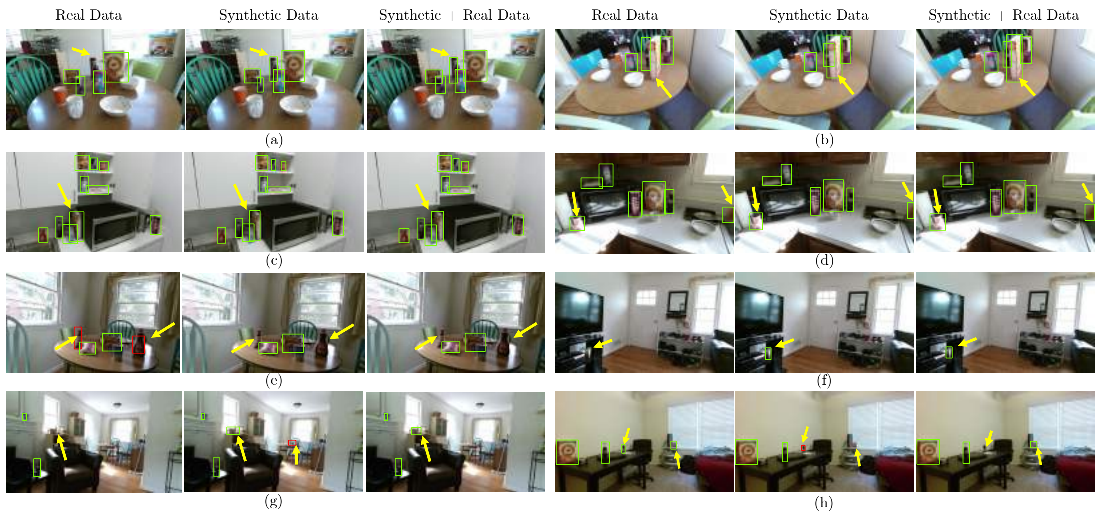

Figure 4.8: We show qualitative detection results and mark true positives in green, false positives in red and arrows to highlight regions. The top two rows are from the GMU Kitchen Scenes [13] and the bottom two rows from the Active Vision Dataset [1]. (a), (b): Model trained on real data misses objects 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), (g): Viewpoint changes cause false negatives. (Best viewed electronically)

We note that their method [12] uses a different background scenes dataset for their synthesis.

Table 4.2 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 our approach is not only competitive with real data and existing synthetic data, but also provides complimentary information. Figure 4.8 shows qualitative examples illustrating this point.

4.5.2 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 use the Active Vision Dataset[1] for evaluation. 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 consider 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 4.5.1). This model serves as a baseline for our model trained on synthetic data. As Table 4.3 shows, a model trained on our synthetic data is still competitive to the model trained on *all* the real data. The performance gap between these datasets is smaller than in Section 4.5.1.

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 4.6). 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 to 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. In Table 4.3 we see that by collecting just 10% images and adding our synthetically generated images, we are able to get more MAP than using the real images in the dataset without the synthetic images. This highlights how useful our approach of dataset generation is in scenarios where there is a dearth of labeled 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.

4.6 Discussion

Our key insight while generating synthetic scenes was to maintain local consistency by using real images of objects and get a diverse coverage of instance viewpoint and scales. Instead of trying to achieve photo-realism we render scenes with varying blending modes to make the model ignore blending artifacts. We showed how our improvements can affect detection performance of current state-of-the-art methods because they rely on region-based features. Our method performs favorably to existing hand curated datasets and captures complementary information. We show that combining just 10% of real annotations with our synthesized data achieves performance that is tantalizingly close to that achieved by using all the real annotations.

From a practical viewpoint, our work can be adapted even more to specific applications. If one is building a detection system(especially if the camera's location is fixed and known), one might not desire to have a model that recognizes the object of interest from all viewpoints. For example, an object detection system running on a CCTV camera that is attached on the ceiling sees more top views of the object or a fixed camera in a warehouse or factory that sees only front/side views of the object. One would want to incorporate this information in the model by training on more pertinent views of the same instance. In the approach we have proposed, one can create datasets with relevant samples of the instance without additional physical effort.

Table 4.2: We compute the performance of training a model on synthetic data and compare it against training on real data. We evaluate on the test split of the GMU Kitchen Dataset [13].

Dataset	coca cola	coffee mate	honey bunches	hunt's sauce	mahatma rice	nature v1	nature v2	palmolive orange	pop secret	pringles bbq	red bull	mAP
Real Images from GMU	81.9	95.3	92.0	87.3	86.5	96.8	88.9	80.5	92.3	88.9	58.6	86.3
SP-BL-SS [12]	55.5	67.9	71.2	34.6	30.6	82.9	66.2	33.1	54.3	54.8	17.7	51.7
(Ours) Synthetic Images	81.0	93.3	85.6	55.6	73.8	94.9	87.1	68.7	79.5	77.1	42.0	76.2
SP-BL-SS + Real Images [12]	82.6	92.9	91.4	85.5	81.9	95.5	88.6	78.5	93.6	90.2	54.1	85.0
(Ours) Synthetic + Real Images	88.5	95.5	94.1	88.1	90.3	97.2	91.8	80.1	94.0	92.2	65.4	88.8

Table 4.3: Results on the entire Active Vision dataset by varying amount of real data from the GMU Kitchen Scenes *train* dataset

Dataset	coca cola	honey bunches	hunt's sauce	mahatma rice	nature v2	red bull	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	48.7	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

5 Conclusion

As modern computer vision approaches for tasks like object detection and pose estimation have improved in performance, the demand for large annotated datasets has also increased. A key reason behind this increase is the growing popularity of data-intensive deep learning based approaches in computer vision. A factor that might prevent the large scale adoption of these approaches for detecting instances might be the dearth of such datasets. In order to make these approaches more accessible, we present three different approaches to reduce the annotation effort involved. We show how classical approaches in computer vision for the problems of object tracking and Structure from Motion can be used to tackle the data problem for deep learning based object detection approaches. Finally, we generate synthetic scenes for the task of instance detection. The key insight we found is that generating scenes with lots of random variations improves performance of object detectors and is much easier than synthesizing photo-realistic scenes. Future work on object pose estimation can also benefit from our approach. By associating pose labels to the object instance images and rendering them, one can get a large collection of detection and pose labels. This holds practical value, as large datasets of particular instances with accurate pose labels are hard to collect. In practice, a combination of all these approaches can be used to generate large annotated datasets without much human intervention.

References

[1] Ammirato, P., Poirson, P., Park, E., Kosecka, J., & Berg, A. C. (2017). A dataset for developing and benchmarking active vision. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*.

[2] Barron, J. T. & Poole, B. (2016). The fast bilateral solver. In *European Conference on Computer Vision* (pp. 617–632).: Springer International Publishing.

[3] Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3), 346–359.

[4] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. (2015). Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*.

[5] Chen, W., Wang, H., Li, Y., Su, H., Wang, Z., Tu, C., Lischinski, D., Cohen-Or, D., & Chen, B. (2016). Synthesizing training images for boosting human 3d pose estimation. In *3D Vision (3DV), 2016 Fourth International Conference on* (pp. 479–488).: IEEE.

[6] Collet, A., Martinez, M., & Srinivasa, S. S. (2011). The moped framework: Object recognition and pose estimation for manipulation. *The International Journal of Robotics Research*, 30(10), 1284–1306.

[7] Correll, N., Bekris, K. E., Berenson, D., Brock, O., Causo, A., Hauser, K., Okada, K., Rodriguez, A., Romano, J. M., & Wurman, P. R. (2016). Lessons from the amazon picking challenge. *arXiv preprint arXiv:1601.05484*.

[8] Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1), 98–136.

[9] Felzenszwalb, P., McAllester, D., & Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (pp. 1–8).: IEEE.

[10] Ferrari, V., Tuytelaars, T., & Van Gool, L. (2006). Object detection by contour segment networks. In *European conference on computer vision* (pp. 14–28).: Springer.

[11] Gaidon, A., Wang, Q., Cabon, Y., & Vig, E. (2016). Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4340–4349).

[12] Georgakis, G., Mousavian, A., Berg, A. C., & Kosecka, J. (2017). Synthesizing training data for object detection in indoor scenes. *arXiv preprint arXiv:1702.07836*.

[13] Georgakis, G., Reza, M. A., Mousavian, A., Le, P.-H., & Košecká, J. (2016). Multiview rgb-d dataset for object instance detection. In *3D Vision (3DV), 2016 Fourth International Conference on* (pp. 426–434).: IEEE.

[14] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1440–1448).

[15] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580–587).

[16] Gupta, A., Vedaldi, A., & Zisserman, A. (2016). Synthetic data for text localisation in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2315–2324).

[17] Handa, A., Patraucean, V., Badrinarayanan, V., Stent, S., & Cipolla, R. (2015). Scenenet: Understanding real world indoor scenes with synthetic data. *arXiv preprint arXiv:1511.07041*.

[18] Held, D., Thrun, S., & Savarese, S. (2016). Robust single-view instance recognition. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on* (pp. 2152–2159).: IEEE.

[19] Hinterstoisser, S., Cagniart, C., Ilic, S., Sturm, P., Navab, N., Fua, P., & Lepetit, V. (2012). Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5), 876–888.

[20] Hoiem, D., Chodpathumwan, Y., & Dai, Q. (2012). Diagnosing error in object detectors. In *European conference on computer vision* (pp. 340–353).: Springer.

[21] Hsiao, E., Collet, A., & Hebert, M. (2010). Making specific features less discriminative to improve point-based 3d object recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (pp. 2653–2660).: IEEE.

[22] Karsch, K., Hedau, V., Forsyth, D., & Hoiem, D. (2011). Rendering synthetic objects into legacy photographs. In *ACM Transactions on Graphics (TOG)*, volume 30 (pp. 157).: ACM.

[23] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

[24] Lai, K., Bo, L., Ren, X., & Fox, D. (2011). A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on* (pp. 1817–1824).: IEEE.

[25] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1.

[26] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *ECCV*.

[27] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European Conference on Computer Vision* (pp. 21–37).: Springer.

[28] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3431–3440).

[29] Lowe, D. G. (2001). Local feature view clustering for 3d object recognition. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1 (pp. I–I).: IEEE.

[30] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2), 91–110.

[31] Massa, F., Russell, B. C., & Aubry, M. (2016). Deep exemplar 2d-3d detection by adapting from real to rendered views. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 6024–6033).

[32] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., & Van Gool, L. (2005). A comparison of affine region detectors. *International journal of computer vision*, 65(1-2), 43–72.

[33] Mitash, C., Bekris, K. E., & Boularias, A. (2017). A self-supervised learning system for object detection using physics simulation and multi-view pose estimation. *arXiv preprint arXiv:1703.03347*.

[34] Movshovitz-Attias, Y., Kanade, T., & Sheikh, Y. (2016). How useful is photo-realistic rendering for visual learning? In *Computer Vision–ECCV 2016 Workshops* (pp. 202–217).: Springer International Publishing.

[35] Park, D. & Ramanan, D. (2015). Articulated pose estimation with tiny synthetic videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 58–66).

[36] Peng, X., Sun, B., Ali, K., & Saenko, K. (2015). Learning deep object detectors from 3d models. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1278–1286).

[37] Pérez, P., Gangnet, M., & Blake, A. (2003). Poisson image editing. In *ACM Transactions on Graphics (TOG)*, volume 22 (pp. 313–318).: ACM.

[38] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[39] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91–99).

[40] Richter, S. R., Vineet, V., Roth, S., & Koltun, V. (2016). Playing for data: Ground truth from computer games. In *European Conference on Computer Vision* (pp. 102–118).: Springer.

[41] Ros, G., Sellart, L., Materzynska, J., Vazquez, D., & Lopez, A. M. (2016). The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[42] Sedaghat, N. & Brox, T. (2015). Unsupervised generation of a viewpoint annotated car dataset from videos. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1314–1322).

[43] Sharif Razavian, A., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 806–813).

[44] Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[45] Singh, A., Sha, J., Narayan, K. S., Achim, T., & Abbeel, P. (2014). Bigbird: A large-scale 3d database of object instances. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on* (pp. 509–516).: IEEE.

[46] Song, S., Zhang, L., & Xiao, J. (2015). Robot in a room: Toward perfect object recognition in closed environments. *arXiv preprint arXiv:1507.02703*.

[47] Su, H., Qi, C. R., Li, Y., & Guibas, L. J. (2015). Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2686–2694).

[48] Tolias, G., Sicre, R., & Jégou, H. (2015). Particular object retrieval with integral max-pooling of cnn activations. *arXiv preprint arXiv:1511.05879*.

[49] Tzeng, E., Hoffman, J., Darrell, T., & Saenko, K. (2015). Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 4068–4076).

[50] Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M., Laptev, I., & Schmid, C. (2017). Learning from synthetic humans. *arXiv preprint arXiv:1701.01370*.

[51] Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning* (pp. 1096–1103).: ACM.

[52] Vondrick, C., Patterson, D., & Ramanan, D. (2013). Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, 101(1), 184–204.

[53] Wong, J. M., Kee, V., Le, T., Wagner, S., Mariottini, G.-L., Schneider, A., Hamilton, L., Chipalkatty, R., Hebert, M., Johnson, D., et al. (2017). Segicp: Integrated deep semantic segmentation and pose estimation. *arXiv preprint arXiv:1703.01661*.

[54] Wu, C. et al. (2011). Visualsfm: A visual structure from motion system.

[55] Xiang, Y., Mottaghi, R., & Savarese, S. (2014). Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*.

[56] Zeng, A., Yu, K.-T., Song, S., Suo, D., Walker Jr, E., Rodriguez, A., & Xiao, J. (2016). Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. *arXiv preprint arXiv:1609.09475*.

[57] Zheng, L., Yang, Y., & Tian, Q. (2016). Sift meets cnn: a decade survey of instance retrieval. *arXiv preprint arXiv:1608.01807*.