Fast Single Shot Detection and Pose Estimation

Patrick Poirson[1], Phil Ammirato[1], Cheng-Yang Fu[1], Wei Liu[1], Jana Košecká[2], Alexander C. Berg[1]

[1]UNC Chapel Hill [2]George Mason University

[1]201 S. Columbia St., Chapel Hill, NC 27599 [2]4400 University Dr, Fairfax, VA 22030

[1]{poirson,ammirato,cyfu,aberg}@cs.unc.edu, [2]kosecka@gmu.edu

Abstract

For applications in navigation and robotics, estimating the 3D pose of objects is as important as detection. Many approaches to pose estimation rely on detecting or tracking parts or keypoints [11, 21]. In this paper we build on a recent state-of-the-art convolutional network for sliding-window detection [10] to provide detection and rough pose estimation in a single shot, without intermediate stages of detecting parts or initial bounding boxes. While not the first system to treat pose estimation as a categorization problem, this is the first attempt to combine detection and pose estimation at the same level using a deep learning approach. The key to the architecture is a deep convolutional network where scores for the presence of an object category, the offset for its location, and the approximate pose are all estimated on a regular grid of locations in the image. The resulting system is as accurate as recent work on pose estimation (42.4% 8 View mAVP on Pascal 3D+ [21]) and significantly faster (46 frames per second (FPS) on a TITAN X GPU). This approach to detection and rough pose estimation is fast and accurate enough to be widely applied as a pre-processing step for tasks including high-accuracy pose estimation, object tracking and localization, and vSLAM.

1. Introduction

Detecting and estimating the pose of objects in everyday scenes is a basic capability needed for automatic visual understanding of the world around us, including augmented reality applications, surveillance, navigation, manipulation, and robotics in general. For these applications it is necessary for systems to perceive the pose of objects in addition to the presence and location of an object. A wide range of recent work on object detection in computer vision offers one way to attack this problem. It is now possible, and relatively straightforward, to build detectors for some object *categories*, for example opened laptops. While classical approaches to estimating the pose of objects worked on *instances* of specific objects, often by finding keypoints
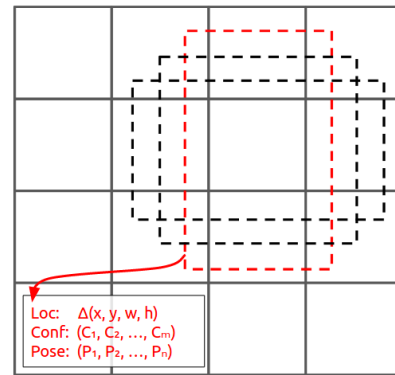
Figure 1. **Default Box Predictions.** At each of a fixed set of locations, indicated by solid boxes, predictions are made for a collection of "default" boxes of different aspect ratios. In the SSD detector, for each default box a score for each object category (Conf) is predicted, as is an offset in the positioning of the box (Loc). This work adds a prediction for the pose of an object in the default box, represented by one of a fixed set of possible poses, $P_1 \ldots P_n$.

whether through detection or tracking, object category detection offers another possible approach. Detecting categories of objects instead of tracking a specific object provides robustness, allows success with previously unseen objects, and can act as initialization or re-initialization for tracking-based approaches to pose estimation. Recently some approaches for object category detection have been extended to estimate the pose of objects, usually in a two stage approach, first detecting objects and then classifying the bounding box around a detected object into a set of coarse pose categories.

In this paper we push further along the direction of category-level object detection and pose estimation by integrating a pose estimation computation directly into a new, state-of-the-art, object detector [10]. This detector uses a deep convolutional network to predict both the presence of a category and adjustments to a bounding box at each of a large set of possible bounding box locations in an image (depicted as the dashed boxes in Fig. 1). We extend

these predictions to include object pose (the red dashed box in Fig. 1). To our knowledge this is the first time a deep-network based approach has been used to perform object detection and pose estimation at the same level. Previous work combining the two built on the deformable parts (DPM) approach [21, 11] adding pose estimation as part of a DPM-based structured prediction model.

By integrating pose estimation directly into a very fast object detection pipeline, instead of adding it as a secondary classification stage, the resulting detection and pose estimation system is very fast, up to 46 frames per second (FPS) on a Titan X GPU, at least 7x faster than Faster R-CNN [12] and close to 100 times faster than R-CNN [7]. Those subroutines are used as pre-processing in almost all other recent category detection and pose estimation pipelines, so the entire compute time is even longer. This speed allows the new system to be run as a pre-processing step for other computations, e.g. initializing a more computationally expensive high-accuracy pose estimation system based on dense correspondence, or adding robustness to a real-time object tracking system, or as part of a visual simultaneous localization and mapping (visual SLAM) tool. Employing coarse pose estimation as a pre-processing step makes sense, and the relatively low compute requirements of our approach make this much more tenable than previous work.

We present the new detection and pose estimation pipeline in Sec. 3, and evaluate it on the standard Pascal 3D+ dataset [21] in Sec. 4.1. The Pascal 3D+ data comes from the Pascal VOC detection dataset, which was collected from web images. In order to further validate our approach, we also evaluate on a new collection of images of everyday environments taken with a low-quality RGB camera, Sec. 4.2. There we compare models trained on Pascal 3D+, models trained on Pascal 3D+ and fine tuned on the new data, and models trained only on the new dataset.

2. Previous Work

Pose estimation is a well-studied problem in computer vision. Initially, research on the topic focused primarily on instance-level pose estimation [20, 1, 13, 3]. In instance-level pose estimation, the model is trained on each specific object instance. Given enough training images of a single instance, the problem essentially reduces to constructing a 3D model of the instance and then finding the pose of the 3D model that best matches the test instance.

More recently, however, the field has shifted to working on category-level pose estimation [15, 8, 6]. This presents many new challenges; most notably, the model must learn how the pose changes with the appearance of the object while simultaneously handling the intra-category variation of instances within a category.

One of the earlier attempts at category-level detection and pose estimation was done by Pepik *et al*. [11]; they for-

mulate the joint object detection and pose estimation problem as a structured prediction problem. They propose a structural SVM, which predicts both the bounding box and pose jointly. This will be referred to as DPM-VOC+VP. Similarly, Xiang *et al*. [21] extend the original DPM method such that each mixture component represents a different azimuth section. This will be referred to as VDPM.

While these earlier methods sought to perform joint detection and pose estimation, many of the more recent deep learning approaches to pose estimation have separated the problem into two stages (Fig. 2 (a)). Stage one requires using an off the shelf detector: [18] and [8] use RCNN [7], but more recent detectors such as Faster RCNN [12] could naturally be substituted (although the change in object proposal might affect the accuracy). Stage two requires cropping the image for the detected regions, which are individually processed by a separate network (e.g., Alexnet [9] or VGG [17]) fine-tuned for pose estimation.

Tulisani *et al*. [19] treat pose estimation as a classification problem and use a Convolutional Neural Network (CNN) for predicting the pose. Their pose estimation network is initialized with VGG [17] and finetuned for pose estimation using ground truth annotations from Pascal 3D+. In order to perform object detection and pose estimation they first use R-CNN [7] to detect objects and then use the detected regions as input to their pose estimation network. Their results are the current state of the art for detection and pose estimation on Pascal 3D+. In Sec. 4, we compare our detection and pose estimation model to theirs and demonstrate similar accuracy. Additionally, in table 5 we show a timing comparison between our proposed model and theirs. This table shows that even if their method used the fastest region based detector (Faster RCNN) our method still provides a 46x increase in speed with similar accuracy.

Similarly, Su *et al*. [18] use the same formulation as [19] for pose estimation. However, with their rendering pipeline they are able to generate more than 2 million synthetic images with ground truth pose annotations. They use these synthetic images to train a pose estimation network, which surpasses [19] in viewpoint estimation accuracy. Finally, they too perform two-stage detection and pose estimation with R-CNN detections. In Sec. 4, we compare our detection and pose estimation model to [18] and show significant increases in accuracy.

Even if a two-staged pipeline for detection and pose estimation uses a fast detector (e.g., Faster RCNN [12]) in stage one, it will still be slower than performing joint detection and pose estimation, since the detected objects must be cropped and then processed by a separate network. In contrast, the method presented in this paper is a joint detector and pose estimator, which requires just one network and a single evaluation of the image (Fig. 2 (b)).
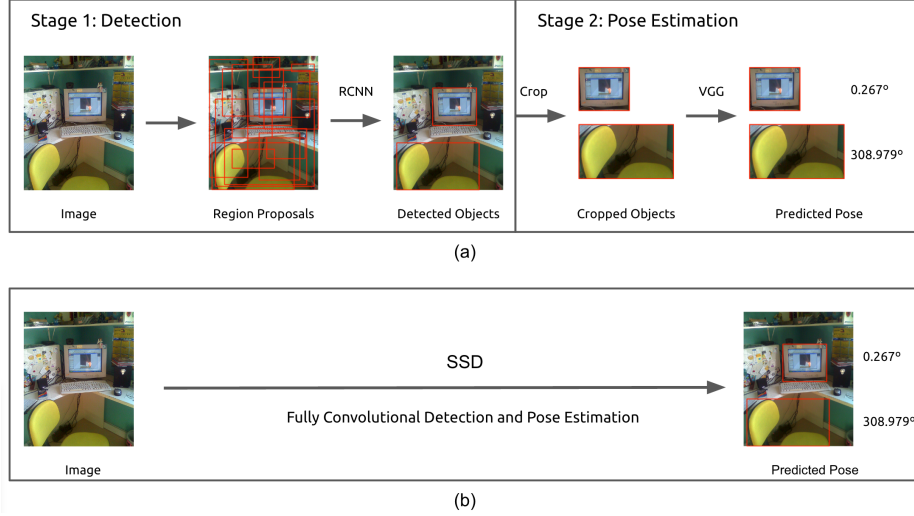
Figure 2. **Two-stage vs. Proposed.** (a) The two-stage approach separates the detection and pose estimation steps. After object detection, the detected objects are cropped and then processed by a separate network for pose estimation. This requires resampling the image at least three times: once for region proposals, once for detection, and once for pose estimation. (b) The proposed method, in contrast, requires no resampling of the image and instead relies on convolutions for detecting the object and its pose in a single forward pass. This offers a large speed up because the image is not resampled, and computation for detection and pose estimation is shared.

3. Model

For an input RGB image, a single evaluation of the model network is performed and produces scores for category, bounding box offset directions, and pose, for a constant number of boxes. These are filtered by non-max suppression to produce the final output. The network is a variant of the single shot detection (SSD) network from [10] with additional outputs for pose. Here we present the network's design choices, structure of the outputs, and training.

An SSD-style detector [10] works by adding a sequence of feature maps of progressively decreasing spatial resolution to an image classification network such as VGG [17]. These feature layers replace the last few layers of the image classification network, and 3x3 and 1x1 convolutional filters are used to transform one feature map to the next along with max-pooling. See Fig. 3 for a depiction of the model.

Predictions for a regularly spaced set of possible detections are computed by applying a collection of 3x3 filters to channels in one of the feature layers. Each 3x3 filter produces one value at each location, where the outputs are either classification scores, localization offsets, and, in our case, discretized pose predictions for the object (if any) in a box. See Fig. 1. Note that different sized detections are produced by different feature layers instead of taking the more traditional approach of resizing the input image or predicting different sized detections from a single feature layer.

We take one of two different approaches for pose predictions, either sharing outputs for pose across all the object categories (*share*) or having separate pose outputs for each object category (*separate*). One output is added for each of

N_θ possible poses. With N_c categories of objects, there are $N_c \times N_\theta$ pose outputs for the separate model and N_θ pose outputs for the share model. While we do add a 3x3 filter for each of the pose outputs, this added cost is relatively small and the original SSD pipeline is quite fast, so the result is still faster than two stage approaches that rely on a (often slower) detector followed by a separate pose classification stage. See Fig. 2 (a).

3.1. Pose Estimation Formulation

There are a number of design choices for a joint detection and pose estimation method. This section details three particular design choices, and Sec. 4.1.1 shows justifications for them through experimental results.

One important choice is in how the pose estimation task is formulated. A possibility is to train for continuous pose estimation and formulate the problem as a regression. However, in this work we discretize the pose space into N_θ disjoint bins and formulate the task as a classification problem. Doing so not only makes the task feasible (since both the quantity and consistency of pose labels is not high enough for continuous pose estimation), but also allows us to measure the confidence of our pose prediction. Furthermore, discrete pose estimation still presents a very challenging problem.

Another design choice is whether to predict poses separately for the N_c object classes or to use the same weights to predict poses for all classes. Sec. 4.1.1 assess these options.

The final design choice is the resolution of the input image. Specifically, we consider two resolutions for input:
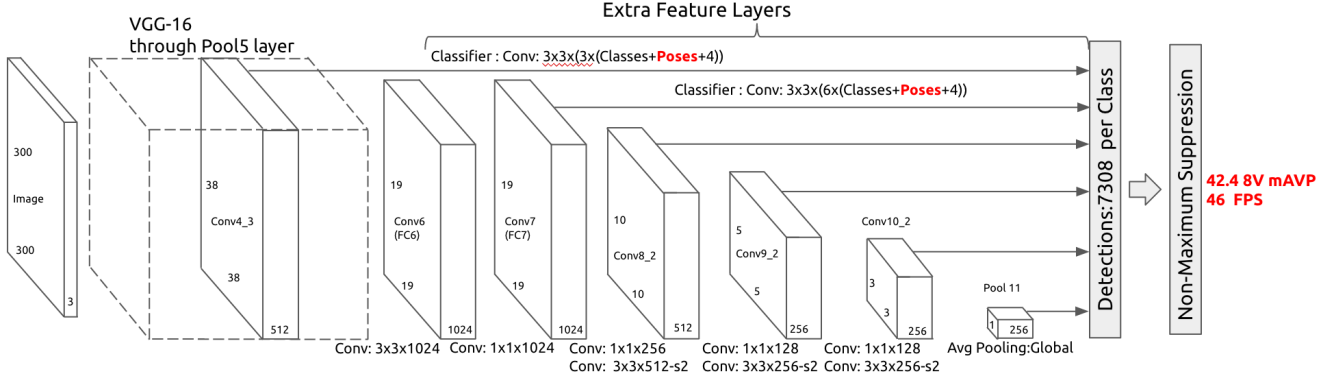
Figure 3. **Our Model Architecture.** Detection and pose estimation network for the "Share 300" model that shares a single pose prediction across all categories at each location and resizes images to 300x300 before using them as input to the network. Feature maps are added in place of the final layers of a VGG-16 network and small convolutional filters produce estimates for class, pose, and bounding box offsets that are processed through non-max suppression to make the final detections and pose estimates. Red indicates additions to the architecture of SSD [10]

300x300 and 500x500. In Sec. 4.1.1, we compare models trained on both resolutions. Using 500x500 images should provide better accuracy because higher resolution images provide finer details that assist in determining the pose. However, processing 500x500 images reduces the speed of computation. Previous deep learning pose estimation formulations [19, 18] crop the detected regions then resize the region to a fixed size of 227x227. This has the advantage of always providing a higher res view of the object. However, with our 300x300 model, an object can take up far less than a 100x100 region in the image since there is no cropping step in our pipeline; therefore, we are fighting against a low resolution. As a result, the higher resolution helps combat this issue.

3.2. Training

The training objective is composed of a weighted sum of three losses: L_{cls}, L_{loc}, and L_{pose} (the class, localization, and pose loss, respectively). In order to compute the loss and backpropagate through our network, we need to first match the ground truth detection and pose annotations to the appropriate default boxes. We follow the approach presented in [10] and match a default box with a ground box if their $IoU > 0.5$. As a result, one ground truth bounding box can be matched to multiple default boxes.

Let N be the total number of default boxes matched to a ground truth box. Normalizing our training objective by N, our total loss is

$$L_{tot} = \frac{1}{N}(L_{cls} + \alpha_1 L_{loc} + \alpha_2 L_{pose}), \qquad (1)$$

where α_1 is set to 1 and α_2 is set to 1.5 through cross validation. Our L_{cls} and L_{pose} losses are both softmax losses, whereas our L_{loc} loss is a Smooth L1 regression loss. Additionally, we adopt the same hard negative mining strategy

as [10]. However, we do not include their full sampler. They sample patches from the image with a minimum overlap of [0.3, 0.5, 0.7, and 0.9] with a ground truth box. In contrast, we sample patches with overlaps of [0.7 and 0.9] because it's too difficult to predict the pose otherwise.

4. Experiments

To evaluate our model, we ran several experiments for detection and pose estimation on two different datasets: Pascal 3D+ and a dataset that we collected for detection and pose estimation in real-world scenes. We found that our SSD model performs comparably to the state-of-the-art two-stage pipeline methods; this is surprising, considering the 46x speedup that our method has. We evaluate our results using the AVP metric proposed by Xiang *et al.* [21]. AVP is an extension of the standard AP metric used to evaluate object detection. For object detection, the AP metric labels a prediction as a true positive if its bounding box has $IoU > 0.5$ with the ground truth bounding box and the correct class label. AVP adds an additional requirement that the predicted pose label must also be correct. AVP is evaluated at different levels of discretization of the pose space into N_θ bins: 4, 8, 16, and 24 bins.

4.1. Pascal 3D+ Dataset

The Pascal 3D+ dataset [21] is made up of images from the Pascal [2] and ImageNet [14] datasets that have been labeled with both detection and continuous pose annotations for the 12 rigid object categories that appear in Pascal VOC12 [2] Train and Val set. In our experiments on this dataset, we follow the standard split and use the labeled Pascal Val images for evaluating our models.

The first set of experiments discussed below justify several design decisions selected for our base model. The sec-

ond set of experiments compares our best models with previously published methods.

4.1.1 Training Choices

Share vs. Separate

First, we compare training on shared and separate pose estimation (Table 1). The shared model, as anticipated, provides significant speed improvements over the separate model, as explained in Sec. 3. Surprisingly, however, we observe that our share model performs better than the separate method. We hypothesize that this is because training a separate pose estimator requires more data, and Pascal 3D+ only has an average of 3091 images per class. Based on these findings, the remaining experiments use only the share model.

Method	4 View	8 View	16 View	24 View
Share 300	48.1	42.3	31.9	27.7
Separate 300	47.6	40.6	29.8	25.5

Table 1. Share vs Seperate.

300x300 vs. 500x500

Our next experiment analyzes the effects of using higher resolution images for training. Table 4 shows the performance of our models using 300x300 and 500x500 images on each object class. In all cases we achieve greater than 2% improvement with the higher resolution 500x500 model for the average case. On the other hand, the 500x500 reduces the FPS from 46 to 17. Since our goal is to build a joint detector and pose estimator suitable for real-time systems, we use the 300x300 model in subsequent experiments.

Pascal + Imagenet vs. Pascal

Recall that the Pascal 3D+ dataset provides detection and pose ground truths for images from Pascal and Imagenet. The test set consists only of Pascal images, which presents a problem, as there are more labeled images in the Imagenet dataset than the Pascal dataset. Nevertheless, pose labels are an expensive resource to collect, so we cannot afford to ignore the additionally labeled Imagenet images. Furthermore, our deep learning approach benefits from the additional data. Consequently, it is essential that we use the labels from both the Pascal and Imagenet images.

If one naively trains on the union of the labeled Pascal and Imagenet images, then the results on the Pascal test set will be diminished by the presence of Imagenet images. To resolve this issue and counteract the effects of the Imagenet images, we replicate the Pascal images so that the number of Pascal and Imagenet images in our training set are equal, ensuring that the number of images from each are approximately the same in each sampled minibatch.

Method	4 View	8 View	16 View	24 View
Share 300	48.1	42.3	31.9	27.7
Share 300 Pascal	43.2	36.3	22.8	20.7

Table 2. Training with vs. without ImageNet annotations.

To verify the benefit of training with both the Pascal and Imagenet images, we trained a shared 300 model using both Pascal and Imagenet training images as well as a Shared model using only Pascal training images (Table 2). As hypothesized, the absence of Imagenet images negatively affects the performance of our model; therefore, the remaining experiments are trained with both Pascal and Imagenet images.

Fine-grained training vs. Coarse-grained training

We also present results training on 24 bin pose discretization and testing on 4 and 8 bins (Table 3). Theoretically, the 24 bin model tested on a coarser discretization of angles would provide results equal to models trained directly for 4 or 8 bins.

In our case we find that the 24 bin model performs comparably to the 4 and 8 bin models; however, in all cases the coarse-grained training performs better.

Method	4 View	8 View
Share 300	45.5	40.9
Share 300 24-V	43.4	34.7
Share 500	48.0	43.3
Share 500 24-V	47.0	38.0

Table 3. 24 view model tested on other binnings.

4.1.2 Comparison to state-of-the-art

Su *et al*. [18] represents the state-of-the-art on pose estimation while Tulisani *et al*. [19] represent the state-of-the-art for object detection and pose estimation on Pascal 3D+. Both methods follow a similar two-stage pipeline for object detection and pose estimation: the first stage uses RCNN for detection, and the second stage uses a CNN finetuned for pose estimation.

In table 5 we compare the speed of our model to [19]. We achieve a very significant 46x increase in speed, which opens up a wide range of settings in which the system can be used. Furthermore, in Table 4 we demonstrate comparable results in accuracy to [19] and even surpass [18].

It is important to note that [18] renders more than 2 million synthetic images, which is approximately 50 times larger than the original Pascal 3D+ dataset. These additional images are particularly useful for deep learning methods. Therefore, we suspect that using a large of amount of

Methods	aero	bicycle	boat	bus	car	chair	table	mbike	sofa	train	monitor	Avg.
Joint Object Detection and Pose Estimation (4 View AVP)												
VDPM [21]	34.6	41.7	1.5	26.1	20.2	6.8	3.1	30.4	5.1	10.7	34.7	19.5
DPM-VOC+VP [11]	37.4	43.9	0.3	48.6	36.9	6.1	2.1	31.8	11.8	11.1	32.2	23.8
RCNN+Alex [18]	54.0	50.5	15.1	57.1	41.8	15.7	18.6	50.8	28.4	46.1	58.2	39.7
VpKps [19]	63.1	59.4	23.0	69.8	**55.2**	**25.1**	24.3	61.1	**43.8**	59.4	55.4	49.1
Ours Share 300	63.6	54.7	25.0	67.7	47.3	10.8	38.5	59.4	41.8	65.0	55.8	48.1
Ours Share 500	**64.6**	**62.1**	**26.8**	**70.0**	51.4	11.3	**40.7**	**62.7**	40.6	**65.9**	**61.2**	**50.7**
Joint Object Detection and Pose Estimation (8 View AVP)												
VDPM [21]	23.4	36.5	1.0	35.5	23.5	5.8	3.6	25.1	12.5	10.9	27.4	18.7
DPM-VOC+VP [11]	28.6	40.3	0.2	38.0	36.6	9.4	2.6	32.0	11.0	9.8	28.6	21.5
RCNN+Alex [18]	44.5	41.1	10.1	48.0	36.6	13.7	15.1	39.9	26.8	39.1	46.5	32.9
VpKps [19]	57.5	54.8	18.9	59.4	**51.5**	**24.7**	20.5	**59.5**	**43.7**	53.3	45.6	44.5
Ours Share 300	57.6	50.8	**20.9**	58.4	43.1	9.1	34.2	52.3	37.2	55.6	46.7	42.4
Ours Share 500	**58.6**	**56.4**	19.9	**62.4**	45.2	10.6	**34.7**	58.6	38.8	**61.2**	**49.7**	**45.1**
Joint Object Detection and Pose Estimation (16 View AVP)												
VDPM [21]	15.4	18.4	0.5	46.9	18.1	6.0	2.2	16.1	10.0	22.1	16.3	15.6
DPM-VOC+VP [11]	15.9	22.9	0.3	49.0	29.6	6.1	2.3	16.7	7.1	20.2	19.9	17.3
RCNN+Alex [18]	27.5	25.8	6.5	45.8	29.7	8.5	12.0	31.4	17.7	29.7	31.4	24.2
VpKps [19]	**46.6**	**42.0**	12.7	**64.6**	**42.7**	**20.8**	18.5	38.8	**33.5**	**42.5**	32.9	**36.0**
Ours Share 300	45.4	33.4	13.7	52.9	32.9	5.3	**27.2**	38.8	27.3	37.4	**36.2**	31.9
Ours Share 500	45.9	39.6	**14.0**	54.0	35.4	7.4	26.4	**40.4**	29.2	41.5	35.8	33.6
Joint Object Detection and Pose Estimation (24 View AVP)												
VDPM [21]	8.0	14.3	0.3	39.2	13.7	4.4	3.6	10.1	8.2	20.0	11.2	12.1
DPM-VOC+VP [11]	9.7	16.7	2.2	42.1	24.6	4.2	2.1	10.5	4.1	20.7	12.9	13.6
RCNN+Alex [18]	21.5	22.0	4.1	38.6	25.5	7.4	11.0	24.4	15.0	28.0	19.8	19.8
VpKps [19]	**37.0**	**33.4**	10.0	54.1	**40.0**	**17.5**	19.9	**34.3**	**28.9**	43.9	22.7	**31.1**
Ours Share 300	35.7	23.6	**10.8**	51.7	33.8	6.2	**23.6**	26.9	20.4	**46.9**	**25.3**	27.7
Ours Share 500	33.4	29.4	9.2	**54.7**	35.7	5.5	22.9	30.3	27.5	44.1	24.3	28.8

Table 4. Category specific results on Pascal 3D+.

similar synthesized data could improve the accuracy of the method presented here. We leave this as a potential future work.

We also provide qualitative results on test images from Pascal 3D+ in Fig. 4.

Method	Detector (FPS)	Pose (FPS)	Total (FPS)
VpsKps[19]	7 (F-RCNN[12])	0.713	0.647
Ours	-	-	46

Table 5. Speed comparison.

4.2. Household Dataset

In addition to using the Pascal 3D+ dataset, we collected over 3,700 images using a Kinect v2 sensor in five real-world household scenes. These five scenes include a bedroom and four open kitchen/living rooms. From these scenes we label four of the Pascal categories: chair, dining table, monitor, and sofa. See the supplementary material for additional information regarding the number of images and objects in each scene.

We are able to cheaply and easily collect labels for all images by first sparsely reconstructing each scene using COLMAP [16], and then densely reconstructing the scene using CMVS [4, 5]. Following reconstruction, we label each object in the dense 3D point cloud and project the labeled point cloud to a bounding box in each image. Using depth data from the Kinect, we are able to automatically adjust the bounding boxes projected from the point cloud to account for occlusion. Moreover, we are also able to get consistent object pose information for every image in the scene by labeling just a single image per object and using the camera positions from the reconstruction to discern the pose of the object in every image.

Method	Scene 1									
	4 View AVP					8 View AVP				
	chair	table	sofa	monitor	Avg	chair	table	sofa	monitor	Avg
Scratch	11.9	15.2	35.4	49.4	28.0	16.3	9.9	31.8	**45.1**	**25.8**
P3D	2.9	17.4	37.4	18.3	19.0	2.1	11.1	25.2	6.1	11.2
P3D Finetuned	**13.4**	**26.4**	**43.9**	**53.0**	**34.2**	**20.0**	**12.1**	**33.7**	24.6	22.6
Results on P3D+	10.8	38.5	41.8	55.8	36.7	9.1	34.2	37.2	46.7	31.8
	Scene 2									
	4 View AVP					8 View AVP				
	chair	table	sofa	monitor	Avg	chair	table	sofa	monitor	Avg
Scratch	18.1	36.7	44.5	45.3	36.1	13.1	14.8	42.1	**38.1**	27.0
P3D	9.9	35.1	59.7	45.7	37.6	5.0	25.1	44.9	34.8	27.4
P3D Finetuned	**28.7**	**43.9**	**63.6**	**66.4**	**50.6**	**16.1**	**25.4**	**56.3**	36.0	**33.4**
Results on P3D+	10.8	38.5	41.8	55.8	36.7	9.1	34.2	37.2	46.7	31.8
	Scene 3									
	4 View AVP					8 View AVP				
	chair	table	sofa	monitor	Avg	chair	table	sofa	monitor	Avg
Scratch	1.8	**28.8**	46.3	**66.1**	**35.7**	0.6	9.9	32.7	25.8	17.3
P3D	**7.6**	13.0	41.4	47.7	27.4	**8.0**	6.2	31.1	30.5	18.9
P3D Finetuned	5.3	27.7	**48.9**	56.3	34.6	4.4	**11.1**	**36.3**	**38.0**	**22.5**
Results on P3D+	10.8	38.5	41.8	55.8	36.7	9.1	34.2	37.2	46.7	31.8
	Scene 4									
	4 View AVP					8 View AVP				
	chair	table	sofa	monitor	Avg	chair	table	sofa	monitor	Avg
Scratch	22.6	22.9	-	10.5	18.6	11.4	28.7	-	12.2	17.4
P3D	**55.5**	**46.5**	-	1.6	**34.5**	**52.4**	**34.0**	-	3.1	29.8
P3D Finetuned	40.8	32.3	-	**17.6**	30.2	30.7	32.1	-	**27.4**	**30.1**
Results on P3D+	10.8	38.5	41.8	55.8	36.7	9.1	34.2	37.2	46.7	31.8
	Scene 5									
	4 View AVP					8 View AVP				
	chair	table	sofa	monitor	Avg	chair	table	sofa	monitor	Avg
Scratch	21.8	**21.2**	**47.0**	21.4	**27.9**	11.0	5.7	34.4	7.3	14.6
P3D	6.2	18.3	41.2	21.0	21.7	4.7	**9.0**	27.3	14.4	13.8
P3D Finetuned	**23.8**	5.4	41.8	**39.8**	27.7	**25.0**	4.5	**36.0**	**14.9**	**20.1**
Results on P3D+	10.8	38.5	41.8	55.8	36.7	9.1	34.2	37.2	46.7	31.8

Table 6. Results on the Household Dataset. Evaluation on the household data, results are shown for our model trained on (P3D), then fine-tuned on the household data (P3D+finetune), and for our model trained only on the household data (scratch). We include the results of our P3D model on the P3D+ dataset in order to roughly compare the difficulty of the datasets for these categories.

4.2.1 Experiments on Household Dataset

Using our household dataset for real world scenes, we set out to evaluate how our detection and pose estimator trained on Pascal 3D+ performs on real-world scenes. Additionally, we compare training a detection and pose estimation model from scratch on the collected dataset (Scratch), using our detector trained on Pascal 3D+ (P3D), and finetuning the Pascal 3D+ detector on the training scenes (P3D Fine-tuned). When training our model on our Household Dataset, we train on four scenes and test on the held out scene.

Table 6 presents the results of these experiments. Un-surprisingly, our model generally performs best when first trained on the Pascal 3D+ dataset then finetuned on labels from our real-world scenes. We provide qualitative results in the supplementary material.

5. Conclusion

We have extended the fast SSD detector to estimate object pose. Our model achieves comparable accuracy to state-of-the-art methods on the standard Pascal 3D+ dataset[21], while offering at least a 46x speedup. The abil-ity to simultaneously detect and estimate the pose of an ob-
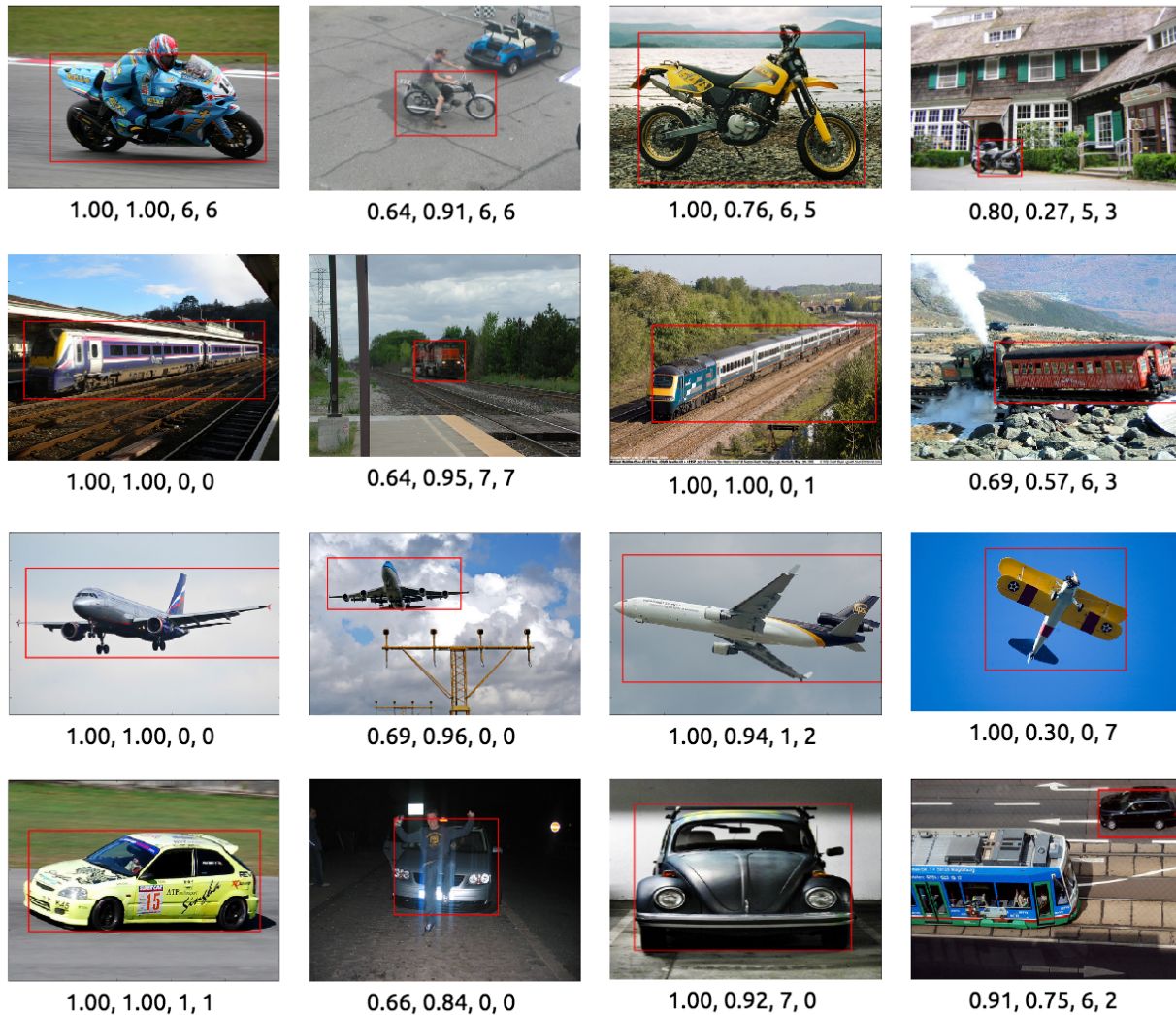
Figure 4. **Pascal 3D+ Qualitative Results.** Results on 8 bin detection and pose estimation on the Pascal 3D+ dataset. Each image has a corresponding detection class confidence, pose confidence, predicted pose label, and ground truth pose label, respectively. Columns one and two show correct pose predictions with high and low detection scores; column three shows pose predictions that are off by one bin; and column four shows some difficult examples where our system fails.

ject opens the door for a variety of use cases. Additionally, we have collected a dataset consisting of five everyday scenes with object bounding boxes and poses labeled. Our results on this dataset demonstrate that the model proposed in this paper has the capacity to quickly and accurately perform simultaneous object detection and pose estimation in real-world scenes.

Acknowledgements

References

[1] A. Collet, M. Martinez, and S. S. Srinivasa. The moped framework: Object recognition and pose estimation for manipulation. *Int. J. Rob. Res.*, 30(10):1284–1306, Sept. 2011.

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

[3] V. Ferrari, T. Tuytelaars, and L. Gool. Simultaneous object recognition and segmentation from single or multiple model views. *Int. J. Comput. Vision*, 67(2):159–188, Apr. 2006.

[4] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards internet-scale multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[5] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[6] A. Ghodrati, M. Pedersoli, and T. Tuytelaars. Is 2d information enough for viewpoint estimation?. In *BMVC*, 2014.

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

[8] D. Glasner, M. Galun, S. Alpert, R. Basri, and G. Shakhnarovich. Viewpoint-aware object detection and pose estimation. In *2011 International Conference on Computer Vision*, pages 1275–1282. IEEE, 2011.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. F. Fu, and A. C. Berg. Ssd: Single shot multibox detector. *arXiv preprint arXiv:1512.02325*, 2016.

[11] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Teaching 3d geometry to deformable part models. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3362–3369. IEEE, 2012.

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

[13] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *Int. J. Comput. Vision*, 66(3):231–259, Mar. 2006.

[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[15] S. Savarese and L. Fei-Fei. 3d generic object categorization, localization and pose estimation. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[16] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

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

[18] H. Su, C. R. Qi, Y. Li, and L. 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.

[19] S. Tulsiani and J. Malik. Viewpoints and keypoints. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1510–1519. IEEE, 2015.

[20] F. Viksten, R. Soderberg, K. Nordberg, and C. Perwass. Increasing pose estimation performance using multi-cue integration. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 3760–3767, May 2006.

[21] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision*, pages 75–82. IEEE, 2014.