Towards Pose Estimation of 3D Objects in Monocular Images via Keypoint Detection

Debidatta Dwibedi
The Robotics Institute
Carnegie Mellon University
debidatd@andrew.cmu.edu

Abstract

The availability of a large number of crowdsourced CAD models of objects can be leveraged to solve the problem of pose estimation of 3D objects in monocular images. Convolutional Neural Networks(CNNs) perform to the best of their capability when they have been trained on a large amount of labeled data. We explore how 3D models can be used to generate lots of training images and annotations in the form of keypoint locations. We propose to use CNNs to first detect keypoints in rendered images. Once, we have a correspondence between 2D points in a test image and the 3D points on the CAD model, we can align 3D models in 2D images.

1. Introduction

Consider the scenario where we are required to build a computer vision system to estimate the pose of some objects of interest. If we are given the 3D models of all these objects, how do we then go about solving the problem of object detection and pose estimation? The advent of depth sensors has given tremendous impetus to tackle this problem. It becomes significantly easier to detect objects(especially in cluttered scenes) in point clouds as compared to monocular images. However, there can be many scenarios where even depth sensors fail to be effective(outdoors or in case of metallic objects). The above system might be used in virtual and augmented reality applications where we might not always have the luxury of having depth as input. On the other hand, a good estimate of the object's pose will allow the system to perform 2D-3D alignment between the object in an image and its CAD model. This can be used to reconstruct the scene with at least the objects of our interest. Hence, it is still a worthwhile exercise to look at the problem of pose estimation of 3D objects in monocular images.

Today many online platforms like 3D Warehouse, GrabCAD etc. host millions of crowdsourced CAD models. These repositories can prove to be a source of large amounts of training data. ShapeNet[2] provides a huge dataset of such models organized by object category. It is possible to look up CAD models of objects of daily use. Even in cases where the exact CAD model of the object is not available, it can be reconstructed by using lots of views of that object or by using a good quality laser scanner. Once, we have the model for the object we should be able to leverage it to solve the task of object detection and pose estimation jointly.

Convolutional Neural Networks(CNNs) have proven effective in many applications in computer vision. However, for CNNs to be able to extract the features most useful for the task of pose estimation of an object would require lots of images with reliable annotations at a very fine level. It won't be economical for humans to provide this level of supervision for many objects. The availability of 3D models and rendering software like Blender etc. allows us to theoretically generate infinite training data with automatically labeled annotations.

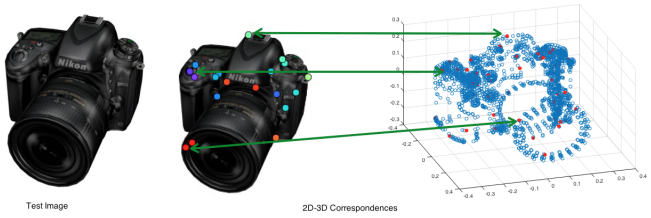

Figure 1. Keypoint Detection in objects means finding correspondences between 2D pixels and 3D points on the model.

In Fig. 1, we can see an overview of the task we want to solve. We propose to leverage the power of CNNs by using them to detect keypoints on the object. These keypoints are selected from the 3D points on the mesh of the CAD model of the object. Once they are detected, we can use the correspondences to determine the required transformation matrix to estimate the pose of the object. In this project, we first attempt to detect keypoints of objects using only

rendered data. But our initial approach does not transfer trivially from rendered images to real images. But feature for the task of coarse pose estimation transfers well. We hope to take advantage of this by using the coarse estimate of the pose to detect keypoints and then refine both the pose estimation and keypoint detection in a second refinement step.

2. Related Work

CNNs have revolutionized the field of computer vision. Krizhevsky et al's now landmark paper [5] describing their neural network based approach for the task of image classification on the ImageNet dataset, paved the way for CNNs to be applied successfully in a variety of vision tasks like semantic segmentation, object detection and image captioning. Researchers have also looked into how CNNs can be used for the task of pose estimation. RenderForCNN[9] solves the problem of pose estimation by converting it into a classification problem. An object rotated at different orientations can be thought of as being a different class altogether. They showcase how using only rendered images they were able to outperform existing state of the art methods in 3D object pose estimation. One reason for this might just be that the real data might not have enough variation in terms of views, lighting and models as the amount of images where 3D objects are labeled along with their pose is less.

Crivellaro et al.[3] present the idea of using CNNs to detect keypoints on an object. They manually mark the few keypoints that they are interested in and present a novel approach to estimates the pose using the keypoints. The training is done on a set of registered images of the target object under different poses and lighting. While the approach is novel, one might desire to remove the fine-grained keypoint annotation step from the pipeline.

CNNs have also been used in a variety of ways[13] [11] [7] [1] for the task of keypoint detection for human pose estimation. These approaches have been shown useful to detect human pose by estimating the location of keypoints of people. While the approaches are different, essentially the task of human pose estimation is that of structured output prediction. The output space for keypoint detection in 3D objects is also structured because of the geometric constraints of a rigid object. We feel that the approaches used in human pose estimation can be leveraged to estimate pose or viewpoints of objects.

The closest work which directly uses keypoints on objects to do pose estimation is that of Viewpoints and Keypoints[12]. One of the key insights of the paper is that the viewpoint of the paper directly decides the visibilty of keypoints and hence, the visibility of keypoints can be used to decide the viewpoint or equivalently the pose of the object. They use both appearance and viewpoint to estimate keypoint likelihood at each pixel in the image. Our work differs from this in two aspects: the keypoints need not be pre-defined or annotated manually in many images. Also we stick to one instance of a category as a "keypoint" can be defined without ambiguity only for a single instance.

3. Keypoint Detection using Synthetic Training Data

We have the model of the object we are interested in. We will stick with the running example of the model of a Nikon D600 camera from the ShapeNet repository for the following section. Now we are faced with the following questions that we must answer before moving ahead:

1. What are the keypoints for this object?
2. How do we generate training data from the 3D model?
3. How do we train a CNN for the task of keypoint detection?

3.1. Definition of Keypoint

Ideally, we want keypoints that are visually discriminative and geometrically informative spread evenly over the surface of the object of interest. Conventionally, keypoints refer to semantically consistent points across many instances of a category. For human pose estimation, head, elbows, knees, wrist and shoulders serve as keypoints. However, the nose of an aeroplane might look drastically different for different instances of the same category. Similarly, in the PASCAL3D dataset there are annotations on racing cars for right headlights but in reality there are no headlights on those cars. To avoid such ambiguity, we define keypoints as a subset of randomly sampled 3D points in the point cloud of the model. Some of these points will not be useful because they will not satisfy the two criteria mentioned above. We propose that the CNN can be used to decide which keypoints should be used for a particular object. We first train a model to detect the entire chosen subset of keypoints. We drop the keypoints whose detection rate is below a threshold on a held out training set of rendered images. We retain only the "good" keypoints for any task built on top of keypoint detection like pose estimation.

3.2. Training Data Generation

We use RenderForCNN to generate synthetic training images. By varying lighting conditions, viewpoints and distance of camera from the object, 13680 images of the camera are generated. Random natural backgrounds are applied from the SUN dataset. The rendering code is modified to provide additional labels for the keypoints. For each selected 3D keypoint, the camera's projection matrix is used to find the corresponding 2D pixel coordinate in the image. The label also encodes if the keypoint is visible or not.

Along with these labels, we have the azimuth, elevation and tilt of the object in that image.

3.3. Network Architecture

We had the option of training a Fully Convolutional Network[6] to output a heatmap corresponding to the image which represents the probability of a pixel being a particular keypoint. We ran some initial experiments using an FCN which showed encouraging results. One interesting aspect of using a FCN was that since the output was a probability value for each pixel it resulted in heatmaps with multiple modes. Similar multi-modal heatmaps also occur in pose estimation of humans. This is resolved by first doing a coarse estimation of the keypoints and then using another network to determine the fine estimation. Researchers also suggest using probabilistic graphical models[10] to resolve this problem. While this is an interesting approach and warrants further research, training a FCN is typically slower as compared to a network with fully connected layers that does regression to the coordinates of the object.

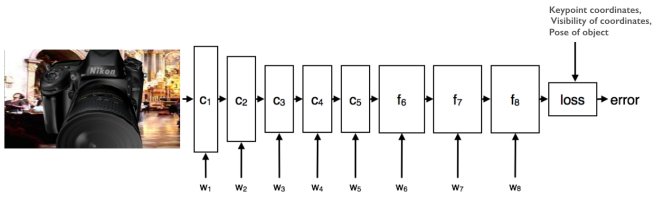

Figure 2. VGG CNN M network is used with four outputs: coordinates of the keypoints, visibility of the keypoints and the azimuth and elevation angles of the object in the image.

We use a VGG-CNN-M-1024 network(Fig. 2 which was introduced by Simonyan et al.[8]. The network is modified to produce four outputs which are trained in the following manner:

1. Keypoint coordinates for each keypoint with euclidean loss

2. Visibilty of keypoint coordinates with log loss

3. Azimuth angle of object(divided into bins of 5 degrees) with log loss

4. Elevation angle of object(divided into bins of 5 degrees) with log loss

The last two losses are added because we have training data for both these tasks as we render the training set and the task of predicting keypoint visibility is closely related to the task of predicting the pose of the object. Experimentally it was found out that the pose estimation task boosts keypoint detection.

4. Experiments

4.1. Setup

The 13680 rendered images are separates into a training set of 12680 images and test set of 1000 images. The task for which evaluation was carried out was keypoint detection using the PCK metric for evaluation. PCK is calculated by finding the number of times a keypoint is correctly predicted. It is assumed to be correct if it lies within a normalized distance α from the ground truth. The distance is normalized by the size of the object in the image. However, for 3D objects many a time the keypoint is not visible in the image. Hence, it is easy to get a high PCK value if the network predicts the keypoint is not present in the image all the time. To get a better measure of the keypoint detection task, we use VisPCK which is PCK when it is given that the keypoint is visible in an image.

In the second experimental setup, we train a network(VGG CNN M 1024 architecture) on a number of models of cars to estimate the pose of the car without any real images.

4.2. Results

Table 1 shows the results of some good keypoints. Only six are shown due to lack of space. Note that the threshold we are going for is very low as even if we might be close in the normalized distance space we can match with a nearby keypoint instead. Hence, PCK and visPCK values are reported at an α of 0.05. From the table, it can be seen that the additional supervision of pose, boosts the keypoint detection rates by reducing the number of false positives.

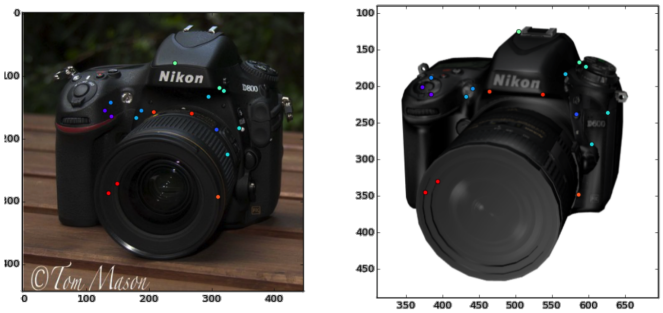

Figure 3. Keypoint detection does not transfer trivially to real images. Here on the left is a real image of the same camera and the colour indicates the index of the keypoint.

While the results for keypoint detection in the rendered image test set is good, it does not transfer trivially to real images. Figure 3 shows a typical example of how the keypoint detection fails. While some local structure is retained in the relative positions of predicted keypoints that are nearby. This motivates us to work on a two step process to detect keypoints. The idea is that getting a rough esti-

Keypoint ID	PCK@0.05 w/o Pose	PCK@0.05 w Pose	VisPCK@0.05 w/o Pose	VisPCK@0.05 w Pose
45	0.51	0.94	0.68	0.67
16	0.38	0.93	0.66	0.66
7	0.46	0.94	0.66	0.62
17	0.50	0.94	0.61	0.61
6	0.35	0.92	0.64	0.61
42	0.39	0.91	0.60	0.55

Table 1. PCK and VisPCK metrics for the good keypoints. w/o pose means the network was trained without supervision of pose.

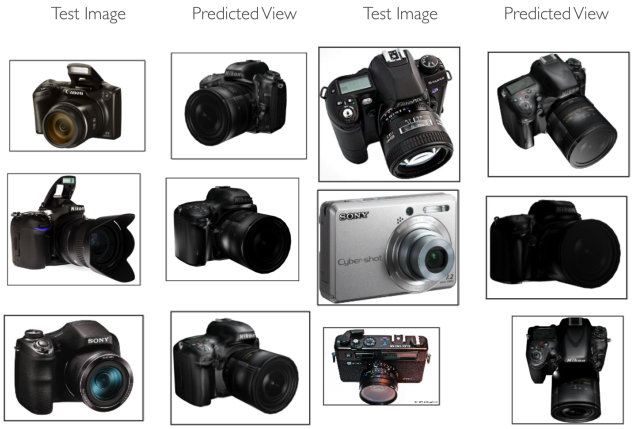

Figure 4. Coarse pose estimation works well qualitiatively and even detects pose in different types and colours of cameras.

VOC 2012 AVP	5 degrees per bin	RenderForCNN
4 Views	49.7%	41.8%
8 Views	43.9%	36.6%
16 Views	37.9%	29.7%
24 Views	33.9%	25.5%

Table 2. Quantitative Analysis of Viewpoint Estimation on Cars using RCNN[4] detection boxes

mate of the pose would help us get a good initial guess for the keypoints. The second step would be to refine the predictions from the first stage. To do this we did some qualitative tests to see if the pose is being predicted correctly on real images when training is only done on rendered images. Coarse pose estimation is a task that is not as fine grained as keypoint detection and transfers really well as can be seen in Fig. 4 which are images of different cameras from Google Image Search. In Fig. 5 it can be seen that the network is able to recover the rough pose of the object if the ground truth box containing the camera is given. What is interesting is that only one model of a camera was used for training and somehow the network is able to predict the pose in different cameras that look very different from the one it is trained on. One reason why this might be happening is that the network is able to realize that the coarse pose of the object is encoded in global features like the silhouette of the object. To test the hypothesis that coarse pose estimation is a task

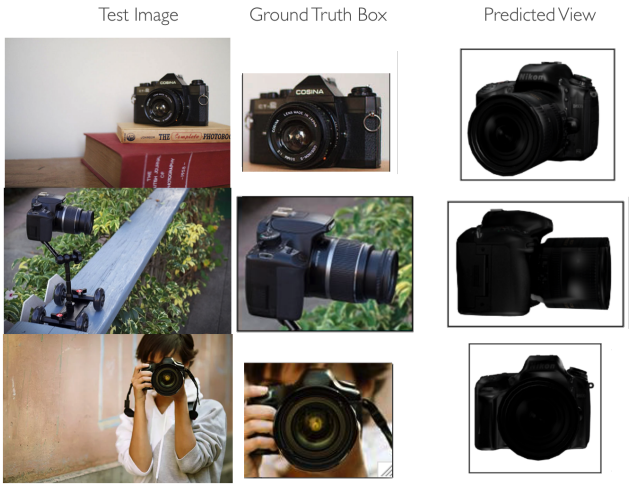

Figure 5. Qualitative pose estimation results on images of cameras in the wild when the ground truth box is provided.

that can be transferred to real images quantitatively, we rendered images of cars and tested on the PASCAL VOC 2012 validation dataset using annotations from PASCAL 3D[14]. Using no rendered images and the extra assumption of no tilt, we were able to get competing results with RenderForCNN.

5. Conclusion and Future Work

The project was an attempt to estimate the pose of an object using keypoints and rendered data only. We trained a network to predict keypoints on objects from only rendered data. The key idea is to use the keypoints to estimate the pose in real images. Doing this would lead to the elimination of the tedious and expensive manual annotation step from the pose estimation pipeline. While the system assumes that the object of interest is known beforehand but such a situation is not uncommon in robotics. We presented a method to select "good" keypoints on a CAD model. While the results on rendered images is promising, the task of pose estimation using keypoints in real images still remains to be done. We plan to move ahead with the task in two ways: use features pooled from the lower layers for the task for keypoint localization and adding real images in the training data in addition to the rendered images.

References

[1] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. *arXiv preprint arXiv:1507.06550*, 2015.

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

[3] A. Crivellaro, M. Rad, Y. Verdie, K. Moo Yi, P. Fua, and V. Lepetit. A novel representation of parts for accurate 3d object detection and tracking in monocular images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4391–4399, 2015.

[4] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *Computer Vision–ECCV 2014*, pages 345–360. Springer, 2014.

[5] 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.

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

[7] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. *arXiv preprint arXiv:1603.06937*, 2016.

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

[9] 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 *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015.

[10] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*, pages 1799–1807, 2014.

[11] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660, 2014.

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

[13] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. *arXiv preprint arXiv:1602.00134*, 2016.

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