3D Pose Regression using Convolutional Neural Networks

Siddharth Mahendran
siddharthm@jhu.edu

Haider Ali
hali@jhu.edu

René Vidal
rvidal@cis.jhu.edu

Center for Imaging Science, Johns Hopkins University

Introduction 3D pose estimation is vital to scene understanding and a key component of many modern vision tasks like autonomous navigation. It is also a very old and well-studied problem and many different approaches have been proposed for this very challenging task. Due to space constraints, we focus our discussion on two recent state-of-the-art approaches based on Convolutional Neural Networks (CNNs): Viewpoints&Keypoints (V&K) [9] and Render-for-CNN (Render) [8]. In both these works, the 3D pose space is discretized into bins and a pose-classification problem is solved using standard architectures, VGG-net [7] for [9] and Alexnet [5] for [8], with an additional fully connected (FC) layer that predicts the pose label. V&K [9] augments training data using jittered bounding boxes with some overlap whereas Render [8] uses millions of images rendered from 3D models for training and real images for finetuning. Both train all classes jointly with weights shared across all layers except the output layer.

Contributions We argue that *3D pose is continuous and can be solved in a regression framework instead*. The challenge is that 3D pose space is non-Euclidean, hence CNN algorithms need to be modified to account for the nonlinear structure of the output space. Our contributions are three-fold: (i) a suitable representation for 3D pose that also allows us to use a natural non-linearity at the output layer and an appropriate geodesic loss, (ii) data augmentation that is relevant for the task of 3D pose regression and (iii) an algorithm to train and finetune a modified VGG network that shows competitive performance on the PASCAL3D+ dataset. In this work, like in [9, 8], we assume that we have a bounding box in the image around an object of interest, like the output of a 2D detector, and find the 3D pose of the object inside the bounding box. More specifically, we are interested in estimating the rotation transformation between object and camera. We note that pose regression is common practice in human pose estimation and that other non-CNN based approaches do object pose regression. For instance, [4] regresses camera pose with a quaternion representation but uses euclidean loss instead of a geodesic loss.

Representing 3D Rotations A rotation of angle θ about axis v, $\|v\|_2 = 1$, can be represented as $R = \exp(\theta\hat{v})$, where $\hat{v}$ is the skew-symmetric operator $[[0, -v_3, v_2], [v_3, 0, -v_1], [-v_2, v_1, 0]]$ of the vector $v = (v_1, v_2, v_3)^T$ and exp is the matrix exponential which can be simplified to $R = I_3 + \sin\theta\hat{v} + (1 - \cos\theta)\hat{v}^2$ using the Rodrigues' rotation formula. So, any rotation matrix R has a corresponding 3D representation $y = \theta v$ and vice-versa. We also restrict $\theta \in [0, \pi)$ which ensures a one-to-one correspondence between the rotation matrix R and its representation. We refer the reader to [6] for more details.

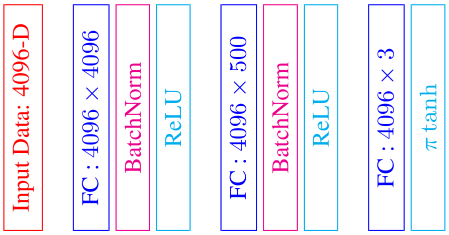
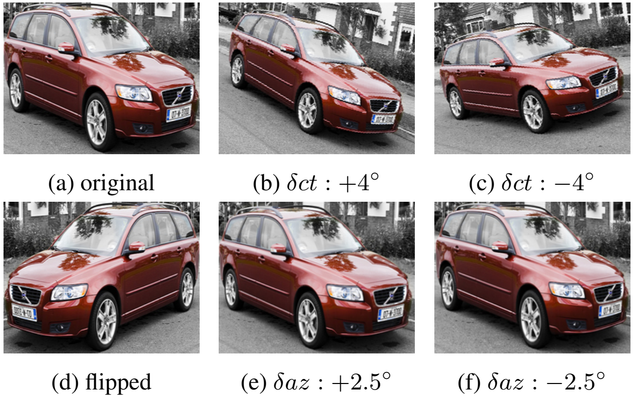


Figure 1: Pose Network

Network Architecture and Loss function Our network is a modification of the VGG-M network [3] and has two parts, a feature network and a pose network. The feature network is identical to the VGG-M upto layer FC6 and is initialized using pre-trained weights. The pose network takes as input the 4096-dim output of the feature network and has 3 additional fully connected layers as outlined in Fig. 1. The feature network is shared across all object categories but each category has its own pose network. Note that this is similar to [9, 8] except that we branch out at FC6 whereas they branch at FC7. The output of the pose network is θv and we model the constraints $\theta \in [0, \pi]$ and $v_i \in [-1, 1]$ using a $\pi \tanh$ non-linearity. An additional advantage of modeling pose in the continuous domain is that we can now use the more appropriate geodesic loss instead of the cross entropy loss for pose-classification or the mean squared error for standard regression. We optimize the geodesic error between the ground-truth rotation R and the estimated rotation $\hat{R}$, given by $\mathcal{L}(R, \hat{R}) = \frac{\|\log R^T\hat{R}\|_F}{\sqrt{2}}$. Here the matrix logarithm, can be simplified to get $\mathcal{L}(R, \hat{R}) = \left|\cos^{-1}\left[\frac{1}{2}(\text{trace}(R^T\hat{R}) - 1)\right]\right|$.

Data Augmentation For every image, we have 3D pose annotated in the form of azimuth, elevation and camera-tilt angles. The corresponding 3D rotation is given by $R(az, el, ct) = R_Z(ct)R_X(el)R_Z(az)$ where R_Z and R_X denote rotations around the z- and x-axis respectively. Jittered bounding boxes, like in [9], introduce small unknown shifts in the corresponding R. Instead, we augment our data by generating new samples corresponding to known small shifts in camera-tilt and azimuth (see Fig 2). Small shifts in camera-tilt lead to in-plane rotations which are easily captured by rotating the image. Small shifts in azimuth lead to out-of-plane rotations which are captured by homographies estimated from 2D projections of 3D point clouds corresponding to the object. We generate a dense grid of samples corresponding to $R(az \pm \delta az, el, ct \pm \delta ct)$. We also flip all samples, which corresponds to $R(-az, el, -ct)$.



(a) original

(b) $\delta ct : +4°$

(c) $\delta ct : -4°$

(d) flipped

(e) $\delta az : +2.5°$

(f) $\delta az : -2.5°$

Figure 2: Augmented training samples from a car image

Experiments We train and test our algorithm on the PASCAL3D+ dataset [10] which has annotations for PASCAL 2012 trainval and Imagenet trainval images for 12 common categories. Like V&K [9], we train on the augmented data from Imagenet trainval and PASCAL train images, and test on PASCAL validation images. For every training image, we generate roughly 162 new augmented samples with shifts in the camera-tilt $-4 : 1 : 4$ (x9), shifts in azimuth $-2.5 : 0.5 : 2.5$ (x9) and flips (x2). In this abstract, we present the results of some preliminary experiments where we estimate the pose given ground-truth bounding boxes of objects that are not truncated or occluded (same protocol as [9]). We train our network in two steps: (i) we train the pose network (keeping the feature network fixed) using augmented Imagenet trainval images with 5-fold cross validation, and (ii) use this as the initialization to fine-tune the network with all classes jointly in an end-to-end manner using PASCAL-train and Imagenet-trainval images with only flipped augmentation. We present our results in Table 1 where we report the median geodesic viewpoint error between the estimated rotation and ground-truth rotation. As can be seen in the table, we get competitive results compared to the current state-of-the-art. Code was written using Keras [1] with TensorFlow [2] backend.

Class	[9]	[8]	ours-no tune	ours-with tune
aero	13.8	15.4	18.30	15.11
bike	17.7	14.8	24.27	20.87
boat	21.3	25.6	47.48	37.32
bottle	12.9	9.3	8.93	8.79
bus	5.8	3.6	4.31	4.14
car	9.1	6.0	9.00	7.86
chair	14.8	9.7	35.24	20.10
table	15.2	10.8	32.66	21.81
mbike	14.7	16.7	21.69	19.99
sofa	13.7	9.5	21.07	18.88
train	8.7	6.1	7.28	7.85
tv	15.4	12.6	17.44	16.68

Table 1: Median angle error for the different categories on PASCAL-val images. ours-no tune refers to the experiment where we use the pose network learnt from Imagenet data to test. ours-with tune is after the end-to-end fine-tuning across all categories

Conclusions and Future Work We have shown that with a suitable representation, loss function and data augmentation, we get competitive performance with respect to state-of-the-art pose-classification methods while solving a pose-regression task. We will continue this investigation into what is the most appropriate representation of 3D object pose while estimating it from a 2D image. In the future, we will extend our work to include performance using bounding boxes returned by 2D detection systems. We also find that our method doesn't work as well under the accuracy metric (percentage of images that have error $< 30°$) and we are investigating this behaviour. We have also made a lot of decision choices that we hope to extensively test, for example: network choice (VGG-M v/s VGG16), branching point (Pool5 v/s FC6 v/s FC7), and augmentations.

Acknowledgements This work was supported by NSF grant 1527340

[1] Keras. https://github.com/fchollet/keras, 2015.

[2] TensorFlow: Large-scale machine learning on heterogeneous systems. http://tensorflow.org/, 2015.

[3] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *BMVC*, 2014.

[4] A. Kendall, M. Grimes, and R. Cippolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. *ICCV*, 2015.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.

[6] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003.

[7] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[8] H. Su, C. Qi, Y. Li, and L. Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. *ICCV*, 2015.

[9] S. Tulsiani and J. Malik. Viewpoints and keypoints. *CVPR*, 2015.

[10] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. *WACV*, 2014.