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

Abstract

3D pose estimation is a key component of many important computer vision tasks such as autonomous navigation and 3D scene understanding. Most state-of-the-art approaches to 3D pose estimation solve this problem as a pose-classification problem in which the pose space is discretized into bins and a CNN classifier is used to predict a pose bin. We argue that the 3D pose space is continuous and propose to solve the pose estimation problem in a CNN regression framework with a suitable representation, data augmentation and loss function that captures the geometry of the pose space. Experiments on PASCAL3D+ show that the proposed 3D pose regression approach achieves competitive performance compared to the state-of-the-art.

1. Introduction

A 2D image is a snapshot of the 3D world and retrieving 3D information from images is an old and fundamental challenge in computer vision. One way to describe the underlying 3D scene is to report the 3D pose of all the objects present in the scene. This task is known as 3D pose estimation and it is a key component of vision problems such as scene understanding and 3D reconstruction. It also plays a vital role in modern vision challenges such as autonomous navigation, where the ability to quickly and reliably recover the 3D pose of other automobiles, pedestrians and objects relative to the camera is very important.

The term 3D pose refers to the transformation between the object and the camera and is often captured using 6 parameters: azimuth az, elevation el, camera-tilt ct, distance to the camera d and image translation (px, py). In this work however, we are not interested in the full 6-dof pose, but in the rotation transformation R between the object and the camera, which is captured by the first three parameters, i.e. $R(az, el, ct)$. Note that we also make a distinction between the tasks of 3D pose estimation and 2D detection. We assume that we have the output of a 2D detection system or an oracle that gives us a bounding box around the object in an image. We then process the image patch inside the bounding box to predict the rotation R. We do this by using a deep convolutional neural network (CNN) that regresses the 3D pose given this 2D image patch.

Related Work There is a rich literature on 3D pose estimation from a single image, from the earlier work of [16] to the more recent work of [14, 8]. Due to space constraints, we concentrate our review on CNN-based methods, which can be grouped into two categories. Methods in the first category, such as [21] and [13], predict 2D keypoints from an image and then use 3D object models to predict the 3D pose given these keypoints. Methods in the second category, such as Viewpoints and Keypoints (V&K)[20] and Render-for-CNN [17], which are closer to what we do, predict 3D pose directly given an image. Both of these methods discretize the pose space into bins and solve a pose classification problem. They have a similar network architecture, which is shared across object categories up to the second-last layer and a separate output layer for every category. While V&K [20] uses a standard cross-entropy loss for classification, Render-for-CNN [17] uses a weighted cross-entropy loss that respects the circular symmetry of angles. While V&K [20] uses jittered bounding boxes with sufficient overlap to augmented annotated training data, Render-for-CNN [17] uses rendered images with a well-sampled distribution over pose space, random crops, and backgrounds. Another method in the second category is [7], which studies multi-view CNN models for joint object categorization and pose estimation, and their models also solve for pose labels.

Contributions In this work, we argue that since *the 3D pose space is continuous, the pose estimation problem can be solved in a regression framework* rather than breaking up the pose space into discrete bins. The challenge is that the 3D pose space is non-Euclidean, hence CNN algorithms need to be modified to account for the nonlinear structure of the output space. Our key contribution is to develop a CNN regression framework for solving the 3D pose estimation problem in the continuous domain by designing a suitable representation, data augmentation and loss function that respect the non-linear structure of the 3D pose space.

More specifically, we use a modified VGG network architecture that consists of a feature network that is shared

	V&K [20]	Render-for-CNN [17]	Ours
Problem formulation	Classification	Fine-grained classification	Regression
Representation	Discretized angles (21 bins)	Discretized angles (360 bins)	Axis-angle / Quaternion
Loss function	Cross-entropy	Weighted cross-entropy	Geodesic loss
Data augmentation	2D jittering	Rendered images	3D pose jittering + rendered images
Network architecture	VGG-Net (FC7)	AlexNet (FC7)	VGG-M (FC6)

Table 1: A comparison of the state-of-the-art methods and our proposed framework

between all object categories and a pose network that is specific to each category. The pose network models 3D pose using an appropriate representation, non-linearity and loss function. We study two representations in particular, axis-angle and quaternions, and model their constraints using non-linearities in the output layer. Our loss function is a geodesic distance on the space of rotation matrices. We also propose a data augmentation technique that is more suitable for regression compared to jittering. We also present experiments on the Pascal3D+ dataset together with an ablation analysis of our various design choices, which show competitive performance with respect to state-of-the-art methods. We present a comparison of our proposed framework with current state-of-the-art methods in Table 1.

To the best of our knowledge, our work is the first one that does 3D object pose regression using CNNs with axis-angle/quaternion representations and geodesic loss functions, and shows good performance on a challenging dataset like Pascal3D+ [22]. We also note that 3D pose regression is commonly used in human pose estimation, to regress the joint locations of the human skeleton. Quaternions have also been used to represent 3D pose for camera localization in [11, 10, 9], but these works ignore the unit-norm constraint for computational ease and use a mean-squared or reprojection loss, whereas we incorporate the constraint into the network and use a geodesic loss.

2. 3D Pose Regression using CNNs

In this section, we describe our regression framework in detail. We first describe the two representations of 3D rotation matrices we use: axis-angle and quaternions, and the corresponding non-linear activations and loss functions. We then describe our network architecture. Finally, we present our proposed data augmentation strategy.

2.1. Representing 3D Rotations

Any rotation matrix R lies in the set of special orthogonal matrices $SO(3) \doteq \{R : R \in \mathbb{R}^{3\times3}, R^TR = I_3, \det(R) = 1\}$. We can then define a geodesic distance between two rotation matrices, R_1 and R_2 as shown in Eqn. (1), where $\log$ is the matrix logarithm and $\|\cdot\|_F$ is the Frobenius norm. This is also the loss function we use in our networks, simplified depending on the representation.

$$d(R_1, R_2) = \frac{\|\log(R_1R_2^T)\|_F}{\sqrt{2}}. \quad (1)$$

Axis-angle A rotation matrix R captures the rotation of 3D points by an angle θ about an axis $v, \|v\|_2 = 1$. This can be expressed as $R = \exp(\theta[v]_\times)$, where $\exp$ is the matrix exponential and $[v]_\times$ is the skew-symmetric operator of vector v, i.e, $[v]_\times = \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix}$ for $v = [v_1, v_2, v_3]^T$. So, every rotation matrix R has a corresponding axis-angle vector $y = \theta v$ and vice-versa. We also restrict $\theta \in [0, \pi)$ and define $R = I_3 \Leftrightarrow y = 0_3$, which ensures a unique mapping between rotation matrix R and it's representation y. The matrix exponential can be simplified to $R = I_3 + \sin\theta[v]_\times + (1 - \cos\theta)[v]_\times^2$ using the Rodrigues' rotation formula. In the same way, Eqn. (1) can be simplified to get:

$$d_A(R_1, R_2) = \cos^{-1}\left[\frac{tr(R_1^TR_2) - 1}{2}\right]. \quad (2)$$

Note that $\|\log\big(\exp(\theta_1[v_1]_\times)\exp(\theta_2[v_2]_\times)^T\big)\|_F/\sqrt{2}$ looks very similar to $\|\theta_1v_1 - \theta_2v_2\|_2$, but it is not the same because $\exp(\theta_1[v_1]_\times)\exp(\theta_2[v_2]_\times)^T \neq \exp(\theta_1[v_1]_\times - \theta_2[v_2]_\times)$ in general. The equality holds only when the matrices $[v_1]_\times$ and $[v_2]_\times$ commute i.e. $v_1 = \pm v_2$.

Quaternion Another popular representation for 3D rotation matrices are quaternions. Given an axis-angle vector $y = \theta v$, the corresponding quaternion $q = (c, s)$ is given by $(\cos\frac{\theta}{2}, \sin\frac{\theta}{2}v)^T$. By construction, quaternions are unit-norm, $\|q\|_2 = 1$. Using quaternion algebra, we have $(c_1, s_1).(c_2, s_2) = (c_1c_2 - \langle s_1, s_2\rangle, c_1s_2 + c_2s_1 + s_1 \times s_2)$ and $(c, s)^{-1} = (c, -s)$ for unit norm $q = (c, s)$. Now, expressing Eqn. (1) in terms of quaternions q_1 and q_2, we have:

$$d(q_1, q_2) = 2\cos^{-1}(|c|) \quad \text{where} \quad (c, s) = q_1^{-1} \cdot q_2 \quad (3)$$

which we simplify to get:

$$d_Q(q_1, q_2) = 2\cos^{-1}(|\langle q_1, q_2\rangle|). \quad (4)$$

2.2. Network Architecture

The proposed network is a modification of the VGG-M network [4] and has two parts, a feature network and a pose network, as illustrated in Fig. 1. The feature network takes as input an Image I, and outputs the feature descriptor of the image, $\Phi_F(I; W_F)$, where W_F are the weights associated with the feature network. Our feature network is identical to the VGG-M upto layer FC6 and is initialized using pre-trained weights, learned by [4] for the ImageNet classification task [6]. The pose network takes the object category label c and the output of the feature network $\Phi_F(I; W_F)$ as input and predicts the object pose $y = \Phi_P(\Phi_F(I; W_F); W_P^c)$, where W_P^c denote the weights of the pose network associated with object category label c. It has 3 fully connected layers with associated activations and batch normalization as outlined in Fig. 2. The feature network is shared across all object categories but each category has its own pose network. Note that this is similar to [20, 17] except that we branch out at FC6 whereas they branch at FC7. Also, note that we use pose here interchangeably with rotation as there is a one-to-one correspondence (discussed in §2.1) between pose: axis-angle vector y or quaternion q, and rotation R.

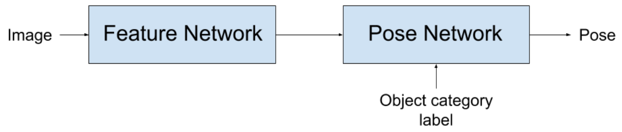


Figure 1: Overall network architecture which takes image I and object category label c as input and predicts object pose y. The feature network $\Phi_F(I; W_F)$ is shared across all object categories, while each category has its own pose network $\Phi_P(\Phi_F; W_P^c)$.

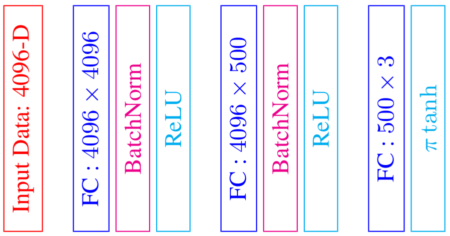
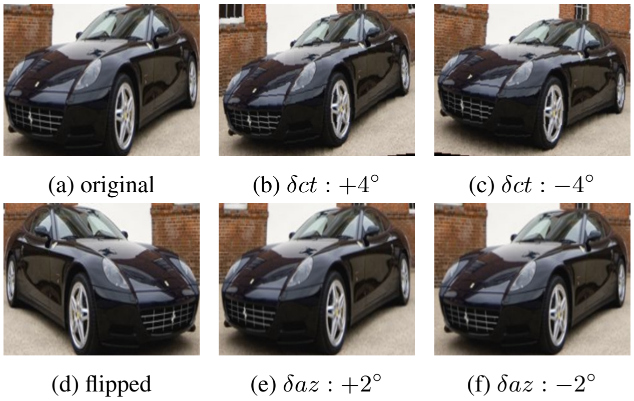


Figure 2: Pose Network for the axis-angle representation

For the axis-angle representation, 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 $\mathcal{L} = d_A(R, \hat{R})$ from Eqn. (2). For the quaternion representation, the output of the network is now 4-dimensional and the unit-norm constraint is enforced by choosing the non-linearity as L2 normalization. The corresponding loss function $\mathcal{L} = d_Q(R, \hat{R})$ is obtained from Eqn. (4).

2.3. Data Augmentation by 3D Pose Jittering

We assume that each image is annotated with a 3D rotation $R(az, el, ct) = R_Z(ct)R_X(el)R_Z(az)$, where R_Z and R_X denote rotations around the z- and x-axis respectively. Jittered bounding boxes (bounding boxes with translational shifts that have sufficient overlap with the original box), like in V&K [20], introduce small unknown changes in the corresponding R. Instead, we augment our data by generating new samples corresponding to known small shifts in camera-tilt and azimuth. We call this new augmentation strategy *3D pose jittering* (see Fig. 3). 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 project the 3D points at ground-truth pose and perturbed pose to get 2D correspondences, which are then used to estimate the homography. 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) $\delta az : +2°$ (f) $\delta az : -2°$

Figure 3: Augmented training samples from a car image

Along with these augmented images, we also use rendered images provided publicly by Render-for-CNN [17] [1] to supplement our training data. We present an analysis in §3.4 which shows that using these rendered images is important to reduce the problem of “un-seen” and “under-seen views”.

[1] https://shapenet.cs.stanford.edu/media/syn_images_cropped_bkg_overlaid.tar

Expt.	aero	bike	boat	bottle	bus	car	chair	dtable	mbike	sofa	train	tv	Mean
V&K [20]	**13.80**	17.70	**21.30**	12.90	5.80	9.10	14.80	15.20	**14.70**	13.70	8.70	15.40	13.59
Render [17]	15.40	**14.80**	25.60	9.30	**3.60**	**6.00**	**9.70**	**10.80**	16.70	**9.50**	**6.10**	**12.60**	**11.67**
Ours (axis-angle)	13.97	21.07	35.52	**8.99**	4.08	7.56	21.18	17.74	17.87	12.70	8.22	15.68	15.38
Ours (quaternion)	14.53	22.55	35.78	9.29	4.28	8.06	19.11	30.62	18.80	13.22	7.32	16.01	16.63

Table 2: A comparison of our framework with two state-of-the-art methods for the axis-angle and quaternion representations. We report the median geodesic angle error (lower is better). Best result in bold and second best in red (best seen in color).

3. Results and Discussion

In this section, we first discuss the dataset we use (§3.1) and how we train our network (§3.2). In §3.3, we present an experimental evaluation of our framework using image patches inside ground-truth bounding box annotations of un-occluded and un-truncated objects in an image (same protocol as V&K [20] - table 1 and Render-for-CNN [17] - table 2). In §3.4, we provide an analysis of various decision choices we make, like: (i) depth of feature network, (ii) choice of feature network, (iii) choice of optimization strategy, (iv) using rendered images for data augmentation, and (v) finetuning the network. Finally, in §3.5 we report performance using detected bounding boxes returned by Faster R-CNN [15] under various metrics.

3.1. Dataset

For our experiments, we use the Pascal 3D+ dataset (release 1.1) [22], which has 3D pose annotations for 12 common categories of interest: aeroplane (aero), bicycle (bike), boat, bottle, bus, car, chair, diningtable (dtable), motorbike (mbike), sofa, train, and tvmonitor (tv). The annotations are available for both VOC 2012 [1] and ImageNet [6] images. We use ImageNet data for training, Pascal-train images as validation data and evaluate our models on Pascal-val images. For every training image, we generate roughly 162 augmented samples with shifts in the camera-tilt (from $-4°$ to $+4°$ in steps of $1°$: x9), shifts in azimuth (from $-2°$ to $+2°$ in steps of $0.5°$: x9) and flips (x2).

3.2. Training the Network

We train our network in two steps: (i) we train the pose network for every object category (keeping the feature network fixed) using augmented ImageNet trainval images as training data and Pascal-train images as validation data, and (ii) use this as the initialization to fine-tune the overall network with all object categories in an end-to-end manner using Pascal-train and ImageNet-trainval images with only flipped augmentation as our training data. While training the pose networks, we first minimize the mean squared error (MSE) for 10 epochs and then minimize the geodesic viewpoint error (GVE) for 10 epochs. Our loss is non-linear with many local minima and minimizing the MSE allows us to initialize the weights for the GVE minimization problem. We use the Adam optimizer with a learning rate schedule of $10^{-3}/(1 + epoch)$. Our code was written in Keras [5] with TensorFlow [2] backend.

3.3. Experimental Evaluation

As mentioned earlier, we use ground-truth bounding boxes of un-occluded and un-truncated objects in an image to evaluate our framework. As in V&K, we compute the geodesic angle between the ground-truth rotation and estimated rotation $d(R_1, R_2) = \frac{\|\log(R_1 R_2^T)\|_F}{\sqrt{2}}$ and present the median angle error (in degrees). We report the mean across three trials of the experiment corresponding to training the network from three different random initializations.

As can be seen in Table 2, we show competitive performance compared to V&K and Render-for-CNN, getting the lowest error for 1 category and second lowest error for 6 categories. We are able to do this in spite of solving a harder problem, of estimating 3D pose in the continuous domain.

3.4. Ablative Analysis

In this section, we present five experiments that provide insight into various design choices for our framework. Experiments (i)-(iv) report results after training only the pose networks (with the feature network fixed). Experiment (v) discusses the effects of finetuning the overall network.

(i) Depth of Feature Network: FC6 vs FC7 vs POOL5 Our feature network is identical to the VGG-M network and uses the output at the FC6 layer as input to the pose networks. This is different from V&K and Render-for-CNN which use the output at the FC7 layer of their feature networks. The rationale for using fewer layers is that a deeper network captures more invariances. This is because the VGG-M network is trained for classification of ImageNet Images, hence it is designed to be invariant to the object pose, which is a nuisance factor for the classification task. The question, however, is which layers of the network learn this invariance? The first few layers learn low-level features like edge detectors and simple shapes and deeper layers learn more complicated shapes. Similarly, we speculate that invariances like translation, color and scale are captured in the the convolutional layers, while pose invariances are learnt in the FC layers. Hence, features at FC7 are more invariant to pose compared to features at FC6 and

POOL5. This is also borne out by the results in rows 5-7 of Table 5 and Fig. 4 where we see that the pose estimation error is less for networks trained with FC6 features compared to FC7 features for all categories except diningtable. This is consistent with the behaviour observed in [7] and [3]. Even though POOL5 results are slightly better than FC6, we branch at FC6 due to significant increase in computation for marginal increase in performance (POOL5 features are 18432 dimensional compared to 4096 dimensional FC6 features).

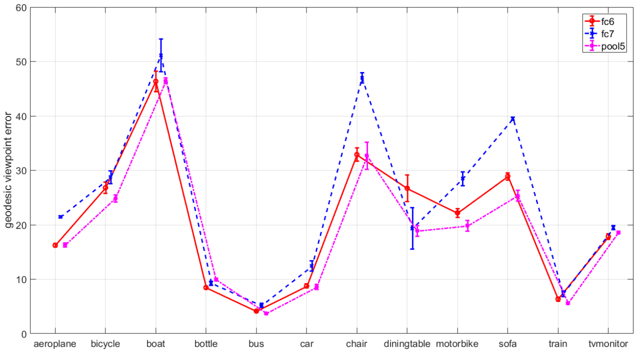


Figure 4: Median angle error for pose-networks trained with features extracted from the FC6, FC7 and POOL5 layers of the VGG-M network (axis-angle representation)

(ii) Type of Feature Network: VGG-M vs VGG16 Another decision choice is the use of the VGG-M network as our base-network. One could exhaustively search over all possible choices of pre-trained networks to decide which network is best suited for pose estimation. We chose not to do so, but compare the VGG-M and VGG16 networks which are two versions of the VGG architecture. We observe, in rows 8-9 of Table 5 and Fig. 5, that the VGG-M network performs better than the VGG16 network. At the same time, we observe that pose estimation performance is not significantly affected by the choice of the feature network. Interestingly, augmenting training data with rendered images (explained later) worsens the performance of the VGG16 network (see rows 12 and 16 of Table 5) whereas it improves the performance of the VGG-M network.

(iii) Optimization Strategy: MSE vs GVE vs Ours As mentioned earlier, we minimize the MSE for 10 epochs and then minimize the GVE for 10 epochs. We do this to avoid the problem of local optima for the non-linear loss function and representation we use. We now show a comparison of what would happen if we just minimize the MSE for 20 epochs or the GVE for 20 epochs. As can be seen from Fig. 6 and Rows 8-11 of Table 5, minimizing only the GVE leads us to bad local minima. However, initializing the GVE minimization with the result of the MSE minimization leads to significantly better performance. This phenomenon has also been observed in prior work on minimizing the geodesic distance in SO(3) [19].

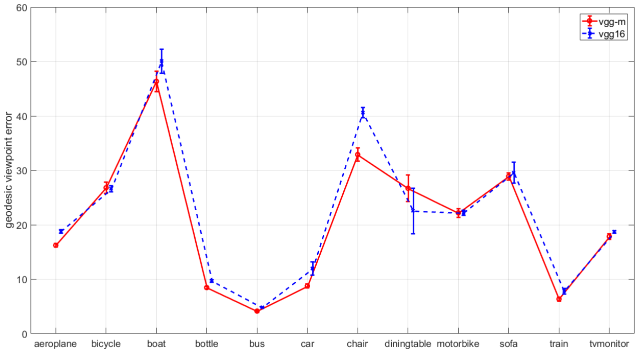


Figure 5: Median angle error under the VGG-M and VGG16 feature networks (axis-angle representation)

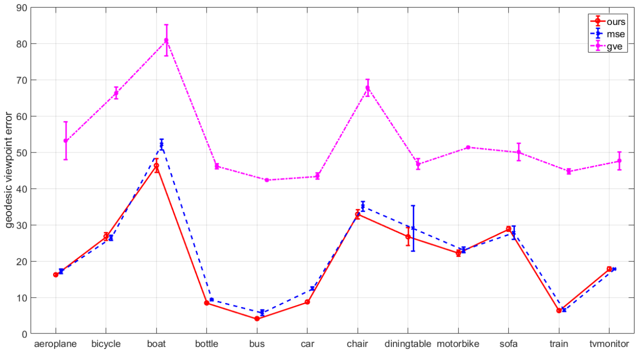


Figure 6: Median angle error under the different optimization strategies (axis-angle representation)

(iv) Data Augmentation using Rendered Images The number of images used for training, validation and testing are shown in Table 3. For training, the number of augmented samples used is roughly 162 times the number of images in the training set. We dig a little deeper into the viewpoint distribution to check if there are images in the training data that are 'close' to the images present in the testing data. This is done using two metrics:

$$\text{Cost1:} \quad \frac{1}{|\mathcal{D}_{Test}|} \sum_{i \in \mathcal{D}_{Test}} \min_{j \in \mathcal{D}_{Train}} d(\theta_i, \theta_j), \quad \text{and} \tag{5}$$

$$\text{Cost2:} \quad \frac{1}{|\mathcal{D}_{Test}|} \sum_{i \in \mathcal{D}_{Test}} \sum_{j \in \mathcal{D}_{Train}} [d(\theta_i, \theta_j) < \epsilon]. \tag{6}$$

Cost1 measures how close the nearest training sample is, in pose space, to each test sample. Cost2 on the other hand, measures how many training samples lie in an ϵ neighbourhood of each training sample. $\mathcal{D}_{Train}$ is the set of all original and flipped training images and $\mathcal{D}_{Test}$ is the set of all testing images. As can be seen by comparing Table 4 and row 3 of Table 5, we do well for categories that have many training examples in the ϵ neighbourhood of a test image, like bottle, bus, car and train, and don't do well for categories like bicycle, chair and motorbike that have few training examples in the ϵ neighbourhood. This is another way of saying that we do well for categories whose pose space is well sampled and worse for categories whose pose space is undersampled. Note that because we augment training images with small perturbations, the number of actual training samples close to a test sample will roughly be a multiple ($\sim$ 162) of the entries in column 3 of Table 4.

	Pascal3D+		
Category	Train	Val	Test
aeroplane	1765	242	244
bicycle	794	108	112
boat	1979	177	163
bottle	1303	201	177
bus	1024	149	144
car	5287	294	262
chair	967	161	180
diningtable	737	26	17
motorbike	634	119	127
sofa	601	38	37
train	1016	100	105
tvmonitor	1195	167	191

Table 3: Number of images in Pascal3D+

Category	Cost1	Cost2($\epsilon = 0.1$)	Cost2(+Rendered)
aeroplane	0.047	23.12	1008.74
bicycle	0.051	11.18	950.622
boat	0.023	58.74	1801.09
bottle	0.024	272.14	7733.42
bus	0.011	168.19	6468.37
car	0.012	217.21	3363.99
chair	0.061	16.07	1124.23
diningtable	0.026	39.71	2319.48
motorbike	0.059	9.55	2319.48
sofa	0.083	40.31	1733.97
train	0.068	213.84	5639.89
tvmonitor	0.029	74.15	3135.37

Table 4: Viewpoint distribution under our two metrics

One way to increase the number of training examples and reduce this discrepancy of unseen poses is to use rendered images with known poses that sample the pose space in a more uniform manner. We use the rendered data made available by Render-for-CNN [17]. As can be seen in Figs. 7 and 8, and rows 10-12 of Table 5, adding this rendered data helps reduce the errors significantly for categories like chair and sofa. This is observed for both the axis-angle and quaternion representations. Column 4 in Table 4 shows the updated Cost2 after including rendered images in $\mathcal{D}_{Train}$. Note that these numbers indicate neighbours in pose space and include images of varying sub-categories and appearances. Also, note that training purely on rendered images (row 14 of Table 5) is worse than training on augmented data and training with both augmented and rendered data jointly gives best results.

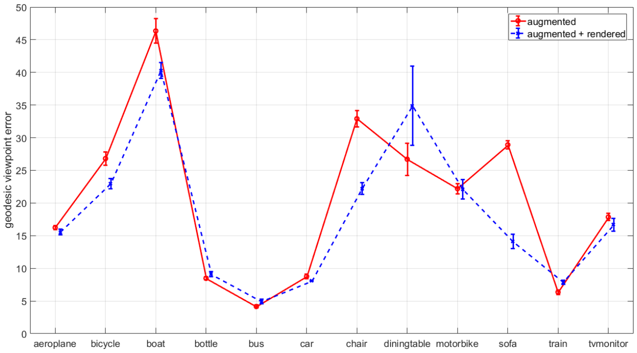


Figure 7: Median angle error under the axis-angle representation using rendered data

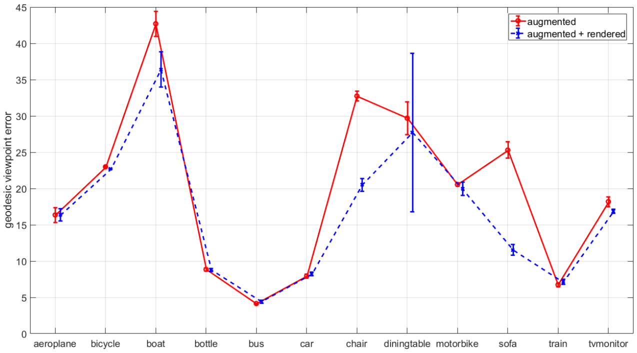


Figure 8: Median angle error under the quaternion representation using rendered data

(v) Finetuning the Joint Network As mentioned earlier, we train our network in a two-step procedure. We first train all pose networks with a fixed feature network and we then finetune the entire network. The finetuning step updates the pre-trained feature network for the task of pose regression. We minimize the geodesic viewpoint error for 30 epochs using the Adam optimizer with original and flipped images of ImageNet trainval and Pascal train images. We use a weighted loss inversely proportional to the number of images per object category. For the axis-angle representation, we used a learning rate of 10^{-5} and find an improvement of $\sim 3°$ in the median angle error averaged across all object categories, comparing rows 16 and 20 of Table 5 and Fig 9. For the quaternion representation, the optimization converges at a lower learning rate of 10^{-6}, but doesn't show a significant improvement after finetuning, comparing rows 17 and 21 of Table 5 and Fig. 10.

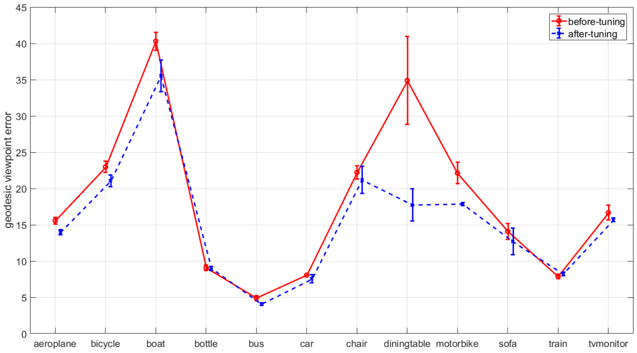


Figure 9: Median angle error under the axis-angle representation after fine-tuning the network

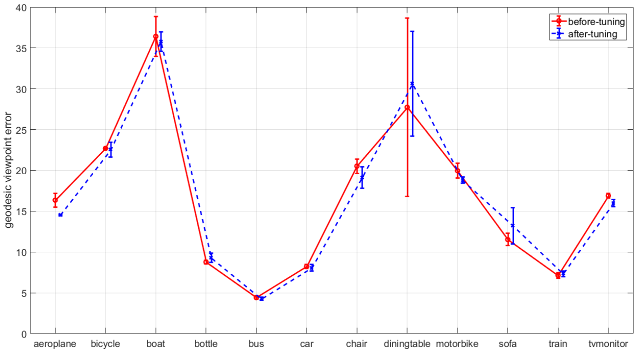


Figure 10: Median angle error under the quaternion representation after fine-tuning the network

3.5. Using Detected bounding boxes

The results presented so far have been obtained with ground-truth bounding boxes for un-occluded and un-truncated objects. We now present results on detected bounding boxes. We run the Faster R-CNN [15] detection code to get bounding boxes for test images and then run our trained models on the patches extracted from these bounding boxes to get a corresponding pose. For every ground-truth bounding box (with annotated 3D pose), we find the detected box with the largest intersection-over-union overlap and compute the median angle error between the ground-truth pose and the estimated pose. As can be seen from Fig. 11 and Table 6, we lose performance slightly $\sim 2^\circ$ in going from ground-truth bounding boxes to detected bounding boxes. We also compare the performance of our method with V&K [20] under the ARP_θ metric which requires sufficient overlap (intersection over union > 0.5) between detected and ground-truth bounding box as well closeness between ground-truth and predicted 3D pose, $\Delta(R_{gt}, R_{pred}) < \theta$. For this experiment, we use the detections provided publicly by V&K [2] for a direct comparison. We compare our performance with V&K under the $ARP_{\pi/6}$ metric in Table 7 and as can be seen, we perform slightly worse in all object categories. We also compare under the AVP metric, which requires predicted azimuth az to be close to the ground-truth azimuth, in Table 8. We perform slightly worse than Render-for-CNN and clearly worse than V&K under this metric. However, we are at a disadvantage here, because the other methods train networks that return azimuth labels directly for this experiment, whereas we still predict a continuous 3D pose, recover the azimuth angle from predicted rotation matrix and then bin it to get predicted azimuth label. Effectively, we're solving a much harder problem but still get comparable results.

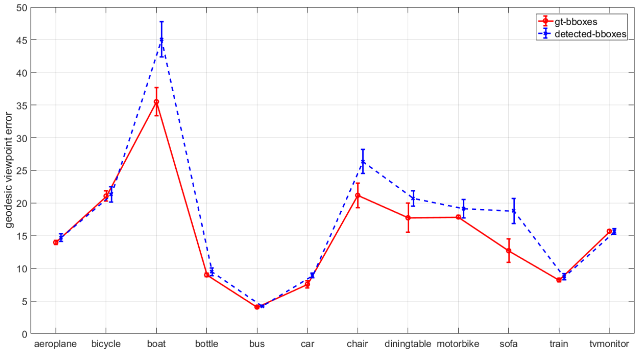


Figure 11: Median angle error under the axis-angle representation with ground-truth and detected bounding boxes

4. Conclusion

We have proposed a regression framework to estimate 3D object pose given a 2D image. We use axis-angle and quaternions to represent the 3D pose output by the CNN and minimize a geodesic loss function during training. We show competitive performance with current state-of-the-art methods and provide an analysis of different parts of our framework.

Acknowledgments This research was supported by NSF grant 1527340. The research project was conducted using computational resources at the Maryland Advanced Research Computing Center (MARCC). This work also used the Extreme Science and Engineering Discovery Environment (XSEDE) [18], which is supported by National Science Foundation grant number OCI-1053575. Specifically, it used the Bridges system [12], which is supported by NSF award number ACI-1445606, at the Pittsburgh Supercomputing Center (PSC).

[2] http://www.cs.berkeley.edu/ shubhtuls/cachedir/vpsKps/VOC2012_val_det.mat

#	Expt.	aero	bike	boat	bottle	bus	car	chair	dtable	mbike	sofa	train	tv	Mean
1	[20]	13.80	17.70	21.30	12.90	5.80	9.10	14.80	15.20	14.70	13.70	8.70	15.40	13.59
2	[17]	15.40	14.80	25.60	9.30	3.60	6.00	9.70	10.80	16.70	9.50	6.10	12.60	11.67
3	axis-angle	16.24	26.81	46.35	8.47	4.15	8.76	32.90	26.71	22.20	28.91	6.36	17.85	20.48
4	quaternion	16.35	22.99	42.71	8.85	4.15	7.93	32.74	29.70	20.55	25.29	6.73	18.20	19.68
5	fc6	16.24	26.81	46.35	8.47	4.15	8.76	32.90	26.71	22.20	28.91	6.36	17.85	20.48
6	fc7	21.45	28.75	51.13	9.26	5.19	12.42	47.00	19.34	28.50	39.49	7.34	19.47	24.11
7	pool5	16.30	24.85	46.46	9.93	3.72	8.56	32.68	18.91	19.81	25.33	5.59	18.57	19.23
8	mse(10)+gve(10)	16.24	26.81	46.35	8.47	4.15	8.76	32.90	26.71	22.20	28.91	6.36	17.85	20.48
9	mse(20)	17.24	26.42	52.12	9.33	5.79	12.44	35.12	29.02	23.08	27.85	6.48	17.84	21.89
10	gve(20)	53.16	66.32	80.85	46.14	42.33	43.40	67.75	46.73	51.37	50.02	44.72	47.63	53.37
11	vggm	16.24	26.81	46.35	8.47	4.15	8.76	32.90	26.71	22.20	28.91	6.36	17.85	20.48
12	vgg16	18.76	26.62	50.07	9.69	4.81	11.97	40.62	22.55	22.20	29.56	7.81	18.71	21.95
13	augmented	16.24	26.81	46.35	8.47	4.15	8.76	32.90	26.71	22.20	28.91	6.36	17.85	20.48
14	rendered	27.31	24.83	53.25	12.97	10.15	13.84	26.76	33.47	27.19	14.21	13.38	19.58	23.08
15	both	15.56	22.98	40.29	9.09	4.92	8.06	22.21	34.88	22.13	14.09	7.88	16.67	18.23
16	row3 + render	15.56	22.98	40.29	9.09	4.92	8.06	22.21	34.88	22.13	14.09	7.88	16.67	18.23
17	row4 + render	16.35	22.70	36.41	8.77	4.42	8.24	20.53	27.73	19.96	11.53	7.14	16.89	16.72
18	row6 + render	19.43	29.76	49.25	9.37	5.85	10.89	35.14	30.06	26.69	20.06	8.82	17.44	21.90
19	row12 + render	19.65	27.61	49.26	9.85	4.89	12.13	46.66	30.76	23.12	36.80	8.71	19.72	24.10
20	row16 + finetune	13.97	21.07	35.52	8.99	4.08	7.56	21.18	17.74	17.87	12.70	8.22	15.68	15.38
21	row17 + finetune	14.53	22.55	35.78	9.29	4.28	8.06	19.11	30.62	18.80	13.22	7.32	16.01	16.63
22	row14 + finetune	16.00	21.29	39.26	9.85	3.98	7.82	22.19	22.90	18.87	12.18	7.27	16.76	16.53

Table 5: Median angle error under various experiments with ground-truth bounding boxes. Lower is better.

Expt.	aero	bike	boat	bottle	bus	car	chair	dtable	mbike	sofa	train	tv	Mean
augmented	18.59	26.43	56.47	9.13	4.31	10.05	41.83	27.00	22.19	27.60	7.06	19.23	22.49
+rendered	17.38	23.32	54.11	10.10	5.22	9.39	25.45	21.98	20.88	18.13	8.27	17.78	19.33
+finetuned	14.71	21.31	45.07	9.47	4.20	8.93	26.36	20.70	19.16	18.80	8.72	15.65	17.76

Table 6: Median angle error under various experiments with detected bounding boxes and axis-angle representation.

Expt	aero	bike	boat	bottle	bus	car	chair	dtable	mbike	sofa	train	tv	Mean
[20]	64.0	53.2	21.0	-	69.3	55.1	24.6	16.9	54.0	42.5	59.4	51.2	46.5
Ours	61.95	49.07	20.02	35.18	66.24	49.89	19.78	15.36	49.38	40.92	56.68	49.87	42.86

Table 7: Comparision under the ARP metric for the results of the axis-angle + rendered + finetuned model. Higher is better.

Expt	aero	bike	boat	bottle	bus	car	chair	dtable	mbike	sofa	train	tv	Mean
[20]-4V	63.1	59.4	23	-	69.8	55.2	25.1	24.3	61.1	43.8	59.4	55.4	49.1
[20]-8V	57.5	54.8	18.9	-	59.4	51.5	24.7	20.4	59.5	43.7	53.3	45.6	44.5
[20]-16V	46.6	42	12.7	-	64.6	42.8	20.8	18.5	38.8	33.5	42.4	32.9	36.0
[20]-24V	37.0	33.4	10.0	-	54.1	40.0	17.5	19.9	34.3	28.9	43.9	22.7	31.1
[17]-4V	54.0	50.5	15.1	-	57.1	41.8	15.7	18.6	50.8	28.4	46.1	58.2	39.7
[17]-8V	44.5	41.1	10.1	-	48.0	36.6	13.7	15.1	39.9	26.8	39.1	46.5	32.9
[17]-16V	27.5	25.8	6.5	-	45.8	29.7	8.5	12.0	31.4	17.7	29.7	31.4	24.2
[17]-24V	21.5	22.0	4.1	-	38.6	25.5	7.4	11.0	24.4	15.0	28.0	19.8	19.8
Ours-4V	52.43	50.80	19.74	35.66	61.24	46.82	20.85	20.31	50.60	42.01	53.42	53.11	42.25
Ours-8V	42.98	37.96	13.18	34.61	41.59	38.66	16.13	12.55	37.94	33.19	43.00	40.43	32.68
Ours-16V	29.90	24.37	7.73	32.06	38.75	29.23	12.18	10.32	25.62	24.82	29.50	25.16	24.14
Ours-24V	21.71	14.21	5.62	29.44	29.16	25.15	9.16	6.98	18.94	15.47	26.38	17.97	18.35

Table 8: Comparision under the AVP metric for the results of the axis-angle + rendered + finetuned model. Higher is better. 4/8/16/24V refers to number of azimuth bins.

References

[1] The PASCAL Object Recognition Database Collection. http://www.pascal-network.org/challenges/VOC/databases.html.

[2] TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] A. Bakry, M. Elhoseiny, T. El-Gaaly, and A. Elgammal. Digging deep into the layers of cnns: In search of how cnns achieve view invariance. In *International Conference on Learning Representations*, 2016.

[4] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

[5] F. Chollet. Keras. https://github.com/fchollet/keras, 2015.

[6] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[7] M. Elhoseiny, T. El-Gaaly, A. Bakry, and A. Elgammal. A comparative analysis and study of multiview CNN models for joint object categorization and pose estimation. In *International Conference on Machine learning*, 2016.

[8] M. Hejrati and D. Ramanan. Analysis by synthesis: 3D object recognition by object reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[9] A. Kendall and R. Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *IEEE International Conference on Robotics and Automation*, 2016.

[10] A. Kendall and R. Cipolla. Geometric loss functions for camera pose regression with deep learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[11] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *IEEE International Conference on Computer Vision*, 2015.

[12] N. A. Nystrom, M. J. Levine, R. Z. Roskies, and J. R. Scott. Bridges: A uniquely flexible hpc resource for new communities and data analytics. In *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*, XSEDE '15, pages 30:1–30:8, New York, NY, USA, 2015. ACM.

[13] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis. 6-dof object pose from semantic keypoints. In *IEEE International Conference on Robotics and Automation*, 2017.

[14] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Teaching 3D geometry to deformable part models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[15] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[16] H. Schneiderman and T. Kanade. A statistical approach to 3D object detection applied to faces and cars. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[17] 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 *IEEE International Conference on Computer Vision*, 2015.

[18] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, and N. Wilkins-Diehr. XSEDE: Accelerating scientific discovery. *Computing in Science & Engineering*, 16(5):62–74, 2014.

[19] R. Tron and R. Vidal. Distributed image-based 3-D localization in camera sensor networks. In *IEEE Conference on Decision and Control*, 2009.

[20] S. Tulsiani and J. Malik. Viewpoints and keypoints. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[21] J. Wu, T. Xue, J. J. Lim, Y. Tian, J. B. Tenenbaum, A. Torralba, and W. T. Freeman. Single Image 3D Interpreter Network. In *European Conference on Computer Vision*, pages 365–382, 2016.

[22] 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*, 2014.