POVa – Computer Vision (in English)

Car Pose Estimation

Ladislav Ondris (xondri07@stud.fit.vutbr.cz)
and Peter Zdravecký (xzdrav00@stud.fit.vutbr.cz)

July 19, 2024

1 Introduction

Object orientation prediction gives useful information about the scene and objects contained. Such information is necessary for many industry tasks such as industrial automation, predicting the orientation of components, or visual surveillance, providing valuable information about the objects being watched.

Such tasks are impractical to do manually. Computer vision has made tremendous progress in the past few decades. Especially, current research in *Deep Learning (DL)* techniques brings promising results in various computer vision fields, including object orientation prediction. With the availability of pre-trained models, it only makes sense to focus efforts on applying them to the task at hand.

In our work, we pose the question ***Which of the rotation representations is ideal for orientation prediction of objects?*** Specifically, we train a deep learning model and evaluate the different representations on a dataset of images of passing cars on highways. We compare five representations—geodetic, Euler, quaternion, axis-angle, and SVD.

Our experiments clearly show that the axis-angle representation outperforms all the other ones by a large factor in both the mean and median scores of the Angular Distance metric.

2 Related work

Estimating the object pose is still relevant in current research. Deep learning shows very good performance for this task. Many works tackle the 6D object pose estimation, comprising of predicting the rotation and translation. DenseFusion [7] is one such example utilizing RGB-D images. HybridPose [5] exploits different geometric information in the image, including keypoints, and edge vectors. Some methods approach the problem by first detecting the target object, followed by a prediction of the orientation. Juránek [2] proposed a detector as a sequence of decision trees that is also capable of predicting the pose of the object.

The performance of a model largely depends on the chosen rotation representation, as was shown by Levinson et al. [3]. They explore the viability of using an SVD orthogonalization as opposed to the traditional ones like quaternions, Euler angles, or the angle-axis. They show that SVD achieves a state-of-the-art performance. An important property of all representations having four or fewer dimensions is they are discontinuous, which may result in a loss of performance for gradient-based optimization.

3 Task Definition and Solution Proposal of our Work

Our goal was to compare various rotation representations. Mainly, we wanted to perform our own experiments in order to acquire practical experience in predicting the orientation of an object.

Formally, the task is to find a function f by optimizing for parameters θ, which maps an input image $\mathbf{I} \in \mathbb{R}^{W \times H \times 3}$ to an orientation prediction $\mathbf{O} \in \mathbb{R}^{T}$ where T is the size of the vector needed to represent the orientation of the object:

$$\mathbf{O} = f(\mathbf{I}, \theta) \tag{1}$$

.

Figure 1 shows what we mean by orientation prediction. A unit sphere can be visualized around the target object. The orientation is then defined as a unit vector pointing in the direction of the camera in its local coordinate system.

The view vector can be converted to various orientation representations. In our work, we focus on four of them. Our major experiment is, therefore, to compare their scores. In our work, we

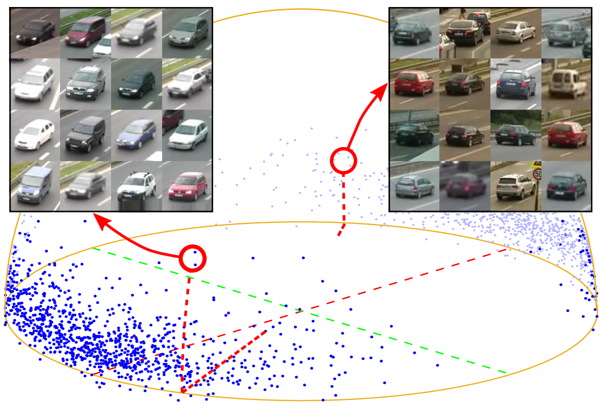

Figure 1: The figure demonstrates the orientation prediction of an object. The orientation is defined by the view vector from the object to the camera. The view vector can be converted to various representations. Adopted from [2].

compare geodetic, Euler angles, quaternions, and angle-axis.

Geodetic representation is what we call the tuple of pitch $\in [-\pi/2, \pi/2]$ and yaw $\in [-\pi, \pi]$. As we are disregarding the *roll* rotation, the pitch and yaw can be thought of as latitude and longitude locations on the unit sphere, respectively. Both rotations are independent of each other.

Euler angles are special in that the rotations are not independent. They consist of rotations along three axes. **Quaternion** is yet another representation consisting of four numbers. The last representation is the **angle-axis**, which consists of a vector defining the axis of rotation and an angle of rotation around this axis. Another approach is using **SVD orthogonalization** [3] for the rotation estimation by producing a rotation matrix.

3.1 COD20k Dataset

The COD20K dataset contains training and testing data for the pose estimation of cars on highways. It consists of approximately 19,061 training images with 40,590 annotated car instances and 1,128 testing images with 5,257 instances. The dataset was created by researchers at the Brno University of Technology [4].

3.1.1 Preparation

Our task does not involve the detection task. Thus, we preprocess the original COD20k dataset by cropping individual cars from all images using the provided bounding box annotations. For each crop, we determined the pitch and yaw angles from the provided view vectors. Figure 2 demonstrates this preprocessing step.

We obtained from this process the final number for the train and test part of the dataset. The dataset contains approximately 81,500 and 10,500 images for the train and test split, respectively. Examples are shown in Figure 3. We use 20% of the training data for validation during training.

3.2 Our Approach

We employ an architecture that consists of a backbone encoder and a prediction head. The encoder takes an image as input and produces a one-dimensional vector, representing the input image. The prediction head takes this representation and outputs raw orientation prediction.

We employ two pretrained backbones in our work—EfficientNetV2-S [6] , and ResNet50 [1]. We use PyTorch-provided `IMAGENET1K_V2` weights for both backbones. We flatten the output at the end. The output dimension is 1280 and 2048 for EfficientNet and ResNet, respectively.

Our prediction head consists simply of two linear layers. The first linear layer takes the flattened tensor from the encoder and outputs a lower-dimensional vector. This vector is then processed by a final layer, outputting the orientation prediction vector.

We use the Mean Squared Error loss for the orientation predictions.

4 Experiments and Results

Our main experiment is comparing five orientation representations. We trained several variations of the proposed model using different backbones and prediction heads. Then, we evaluated it using the angular distance metric.

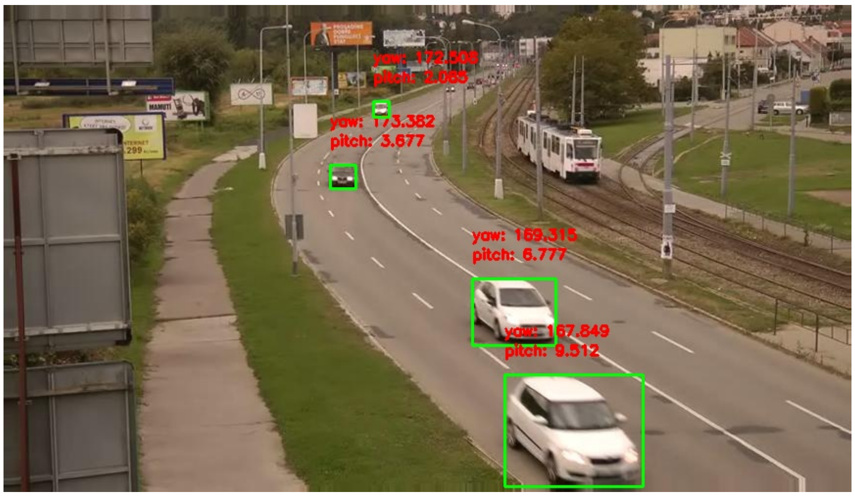

Figure 2: The figure shows an example from the COD20k dataset. We crop the cars using provided bounding box annotations and generate an annotation in terms of pitch and yaw.

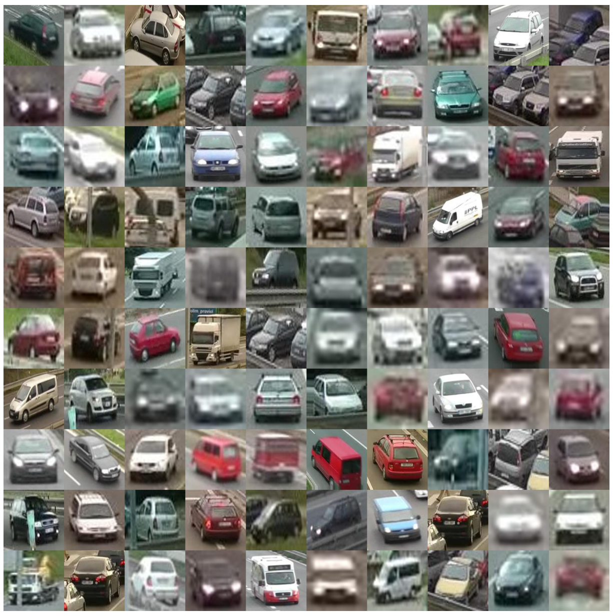

Figure 3: Examples from the processed COD20k dataset.

4.1 Training Setup and Data Augmentation

We used the Adam optimizer with the learning rate initially set to $1e{-}3$. The ReduceLROnPlateau scheduler reduced the learning rate by a factor of 0.1 when the validation loss was not decreasing for two consecutive epochs. We employed early stopping after 10 epochs of no improvement on a validation loss.

We resize input images to 256×256 and perform no further data augmentation. The labels are normalized either to [0, 1] or [-1, 1] range depending on the representation.

4.2 Measured Metrics

The different representations must be converted to a single unified representation for a fair comparison. We used a representation that allows the calculation of deviation in terms of degrees on a unit sphere.

Angular distance, or the central angle, is the angle between the orientation of two vectors in three-dimensional space. We convert each representation to pitch and yaw, from which we can determine the position on the unit sphere and calculate the angular distance between the predicted and expected values, as depicted in Figure 4.

The angular distance between two points, P and

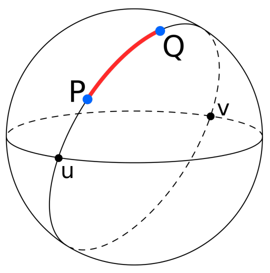

Figure 4: Angular distance (central angle) is the angle between two points on a sphere. On a unit sphere, the depicted red line connecting the two points along the surface of the sphere equals the central angle. Two antipodal points, u and v, are also shown.

Q, defined by their latitude (ϕ) and longitude (λ) is calculated using the Haversine formula as follows:

$$\mathtt{hav}(\theta) = \sin\left(\frac{\Delta_\phi}{2}\right)^2 + \cos(\phi)\cdot\cos(\lambda)\cdot\sin\left(\frac{\Delta_\lambda}{2}\right)^2 \tag{2}$$

where $\Delta_\phi = P_\phi - Q_\phi$ and $\Delta_\lambda = P_\lambda - Q_\lambda$.

Finally, isolating the angle θ:

$$\theta = \cos^{-1}\left(1 - 2\cdot\mathtt{hav}(\theta)\right) \tag{3}$$

gives the angular distance.

4.3 Naive, Theoretical Model

The dataset contains only a limited range of pitch and yaw rotations, as shown by a density function in Figure 5. The two blobs in the density function represent cars facing the camera and facing away from the camera. Arguably, a **naive model** could easily predict the most occurring rotations (the two peaks in the density function), while still providing relatively good results. We can calculate the mean error the model would make if it always predicted one of the closer peaks of the density function. The evaluation of such a model on a test set achieves median AD error, mean AD error, and standard deviation of **0.277**, **0.393**, and **0.310**, respectively. All metrics are reported in radians.

4.4 Overall Results

Table 1 contains the reported results of the angular distance metric computed over all cars within the COD20k test set. The results suggest that the choice of orientation representation has an impact on the performance of the model. Particularly, the axis-angle representation performs significantly better than the other representations for both convolutional backbones. Despite our anticipation that the SVD approach would surpass classical methods due to the absence of discontinuities, its performance turned out to be comparable to that of the classical approaches. We achieved the best result for the axis-angle representation for the EfficientNet backbone.

Each representation outperformed the naive theoretical model by a large margin, which shows that the trained models learned to predict the rotation of cars successfully. While the naive model yielded a median AD error of 15.9 degrees, our best model achieved a notably lower error of 1.6 degrees.

The work of Juranek [2] reported the median error in azimuth (yaw) to be 8.6 degrees. Our model achieves an error of 1.01 degrees. Even though the work of Juranek worked towards a real-time solution performing both car detection and rotation estimation, our work shows the capability of using CNNs for rotation estimation.

As for the inference time, the model based on ResNet50 can process five images per second on an Intel Core i7 CPU, proving that the model can be used for real-time prediction if the number of cars in a single frame is not too great.

4.5 Further Analysis

We take the best-performing model and perform a further analysis of the distribution of the error across various properties of data.

First off, we provide a histogram in Figure 6 of the Angular Distance metric. We can see that even though the median is approaching zero, many predictions are far from the ground truth, increasing the mean error significantly. This divergence of mean and median is even more pronounced for representations other than axis-angle.

We also show the relationship between yaw (azimuth) and the reported mean Angular Distance error. From Figure 7, it is clear that the error depends highly on the rotation of the car. This can be the result of the imbalance of data in the training dataset.

Lastly, we wanted to explore the correlation between the error and the cropped image size Some images in the dataset are particularly small, which could affect the performance. As we can see in Figure 8, that is not what we see. In reality, the error is greater for larger images.

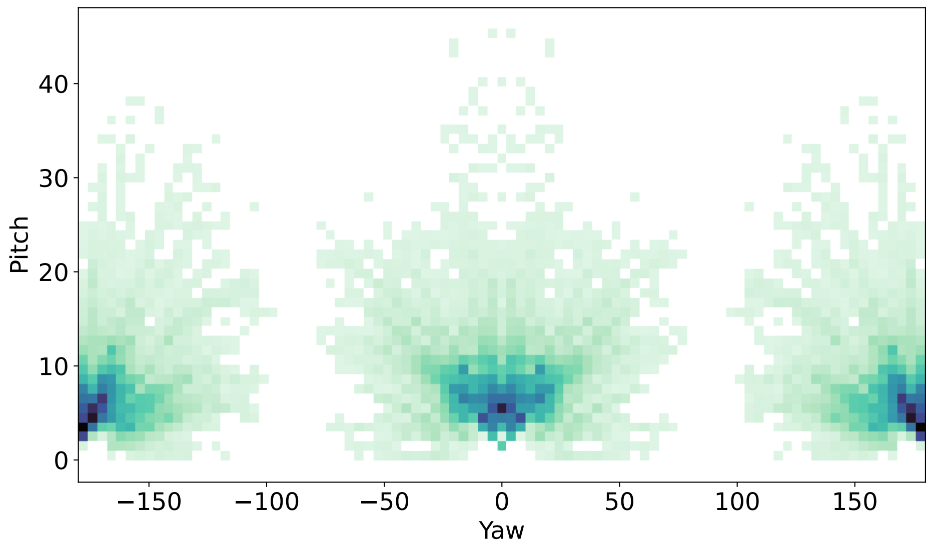

Figure 5: The COD20k dataset contains only a limited range of values of the yaw and pitch rotations. The model may be biased towards working in these ranges only. A naive model selecting one of two peaks correctly for the approaching and receding cars would result in a median error of 0.277 radians on the test partition.

Backbone	**Geodetic**	**Euler**	**Quaternion**	**Axis-Angle**	SVD
ResNet50	0.420 ± 0.702	0.443 ± 0.655	0.484 ± 0.808	$\mathbf{0.096} \pm \mathbf{0.133}$	0.432 ± 0.760
	0.079	0.111	0.071	**0.041**	0.077
EfficientNetV2-S	0.425 ± 0.730	0.448 ± 0.695	0.490 ± 0.832	$\mathbf{0.087} \pm \mathbf{0.138}$	0.446 ± 0.781
	0.083	0.099	0.070	**0.028**	0.068

Table 1: Angular distance metrics (mean $\pm$ standard deviation, and medians on a separate row) for ResNet50 and EfficientNetV2-S backbones on the test split of the COD20k dataset. All metrics are reported in radians.

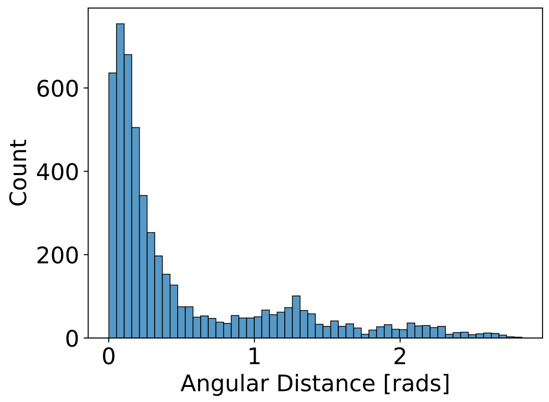

Figure 6: Histogram of the Angular Distance metric on the test set of the COD20k dataset.

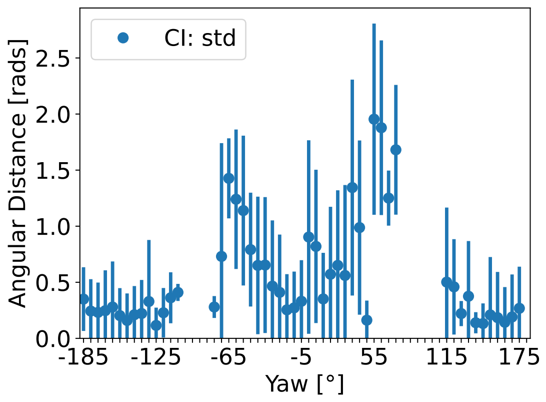

Figure 7: The relationship between yaw and the mean Angular Distance metric on the test set of the COD20k dataset.

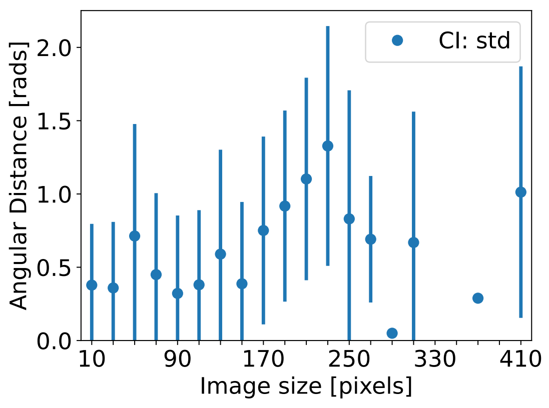

Figure 8: The relationship between image size and the mean Angular Distance metric on the test set of the COD20k dataset.

5 Conclusion

In this work, we aimed to predict the car's rotation in an image using various rotation representations. Our main experiment was to show which representation performs best. We employed two pre-trained convolutional backbones—ResNet50 and EfficientNetV2-S—and a prediction head consisting of linear layers. We trained this setup on images cropped from the COD20k dataset in all configurations, resulting in a total of ten models.

The results show that the best-performing representation is the axis-angle representation with a median Angular Distance error of 1.6 degrees. We compared our results to the work of Juranek and showed that we achieved significantly better results.

This work could be further improved by giving more importance to under-represented yaw angles in the dataset, improving the score for this data. We also believe that a significant inference time boost would be achieved if models with a smaller input image resolution were used given that most images in the test set have a median size of 50.

6 Acknowledgements

Computational resources were provided by the e-INFRA CZ project (ID:90254), supported by the Ministry of Education, Youth and Sports of the Czech Republic.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[2] Roman Juranek, Adam Herout, Marketa Dubska, and Pavel Zemcik. Real-time pose estimation piggybacked on object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[3] Jake Levinson, Carlos Esteves, Kefan Chen, Noah Snavely, Angjoo Kanazawa, Afshin Rostamizadeh, and Ameesh Makadia. An analysis of svd for deep rotation estimation. *ArXiv*, abs/2006.14616, 2020.

[4] Juránek Roman, Herout Adam, Dubská Markéta, and Zemčík Pavel. Real-time pose estimation piggybacked on object detection. In *ICCV*, 2015.

[5] Chen Song, Jiaru Song, and Qixing Huang. Hybridpose: 6d object pose estimation under hybrid representations. *CoRR*, abs/2001.01869, 2020.

[6] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. *CoRR*, abs/2104.00298, 2021.

[7] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. *CoRR*, abs/1901.04780, 2019.