LEARNING THE 3D POSE OF VEHICLES FROM 2D VEHICLE PATCHES

C. Koetsier[1,*], T. Peters[1], M. Sester[1]

[1] Institute of Cartography and Geoinformatics, Leibniz University Hannover, Germany
(koetsier, peters, sester)@ikg.uni-hannover.de

KEY WORDS: Pose Estimation, Deep Learning, Trajectory Extraction, Surveillance Video Analysis, Trajectory Analysis

ABSTRACT:

Estimating vehicle poses is crucial for generating precise movement trajectories from (surveillance) camera data. Additionally for real time applications this task has to be solved in an efficient way. In this paper we introduce a deep convolutional neural network for pose estimation of vehicles from image patches. For a given 2D image patch our approach estimates the 2D coordinates of the image representing the exact center ground point (cx, cy) and the orientation of the vehicle - represented by the elevation angle (e) of the camera with respect to the vehicle's center ground point and the azimuth rotation (a) of the vehicle with respect to the camera. To train a accurate model a large and diverse training dataset is needed. Collecting and labeling such large amount of data is very time consuming and expensive. Due to the lack of a sufficient amount of training data we show furthermore, that also rendered 3D vehicle models with artificial generated textures are nearly adequate for training.

1. INTRODUCTION

Maps contain important information to navigate and route vehicles. For autonomous vehicles, this information about their environment must be very accurate and up-to-date in order to directly interpret and evaluate the environment measured by sensors. The richer the information is, the better a vehicle can judge the situation, predict next steps and react. The surrounding of the vehicle can significantly influence the driving situation. Which environmental conditions lead to unsafe driving behaviour is not always clear. Therefore, it is important to investigate how such situations can be reliably detected, and then search for their triggers. It is conceivable that such insecure situations (e.g. near-accidents, sudden u-turns, avoiding obstacles) are reflected, for example, as anomalies in the movement trajectories of road users (Huang et al., 2014). Collecting real world traffic data in driving studies (e.g. (Barnard et al., 2016)) is very time consuming and expensive. On the other hand, a lot of roads or public areas are already monitored with video cameras. In addition, nowadays more and more of such video data is made publicly available over the internet so that the amount of free video data is increasing.

Figure 1. Detection bounding boxes of pedestrians (left) and vehicles (right)

Previous research (e.g (Koetsier et al., 2019)) exploited the use of such kind of opportunistic VGI by creating a real time surveillance camera pipeline to extract road user trajectories from mono-camera videos. The framework is based on a single shot neural network YOLO (Redmon et al., 2016) where road users are located within bounding boxes in single image frames. To track the road users and extract their trajectories a specific point of this bounding box has to be chosen as real center ground point. While it works well for pedestrians to choose the center point at the bottom of the bounding box to estimate the real center ground point - because of the pedestrians small stand space, it is inaccurate to use the same point for vehicles as shown in Figure 1. Due to the the view angle of a surveillance camera onto a scene, depending on where a vehicle is located and how it is orientated in the scene, the real center ground point changes within the detected vehicles bounding box. This fact causes inaccurate trajectories choosing a fixed point of the vehicles bounding box in (Koetsier et al., 2019).

The exact position of the center ground point for a vehicle in an image can be determined, if the 3D geometry of the situation is known. However, this is often not the case. Thus, the aim of this research is to improve the accuracy of the trajectory extracted from surveillance camera data by learning the center position and heading of a vehicle just from its 2D projection in mono-camera images.

Other research in the field of vehicle pose estimation for mono-camera images aims at finding so called landmarks (e.g. (Zhang et al., 2020) or key points (e.g. (Coenen & Rottensteiner, 2019) in a given camera image and matching those to a given 3D vehicle model to reconstruct 3D coordinates respectively a 3D scene. While the results of these works show, that it is possible to precisely estimate vehicle poses in mono-camera images, they are computational expensive and thus do not fulfill the requirement of real time capability.Further research, like (Xiang et al., 2017) and (Tekin et al., 2018) present convolutional neural networks to efficiently estimate an objects location respectively pose in mono-camera images similar to our approach for different domains - but not for vehicles. To apply those works or similar neural networks to localize vehicles in surveillance camera data a sufficient amount of labeled training data is needed.

Although there are datasets like KITTI (Geiger et al., 2013) and Waymo (Sun et al., 2019) containing the necessary information needed to create labeled vehicle image patches with pose information for training a pose estimation network, they are recorded from the perspective of (self-) driving cars. Thus they

*Corresponding author

do not include all possible viewing angels to recorded vehicle, which are required in the domain of surveillance camera data. To our knowledge there is no such dataset of real image patches with labeled pose information of vehicles covering all or at least nearly all possible viewing angles.

Collecting and (manually) labeling a large amount of this data is very time consuming and expensive. The goal of this paper therefore is to (a) create a 2D image dataset from 3D vehicle models with arbitrary, but known center ground point and orientation to (b) adapt a deep convolutional neural network for pose estimation to the domain of cars by training custom models using such input images and (c) evaluate the performance of the trained models to show that those neural networks can be trained by non real data using rendered 3D vehicle models, solving the training data issue.

2. METHOD

As introduced, the focus of the paper is on the second block of Figure 2: for a given 2D image patch, our approach tries to retrieve 2D coordinates of the image representing the exact center ground point (cx, cy) and the orientation of the vehicle - represented by the elevation angle (e) of the camera with respect to the vehicle's center ground point and the azimuth rotation (a) of the vehicle with respect to the camera, by keeping real time ($\geq$ 30 frames per second) processing speed of a pipeline like in (Koetsier et al., 2019). The 2D image coordinates will be transferred with a given homography into world coordinates - assuming a planar road surface in the field of view.

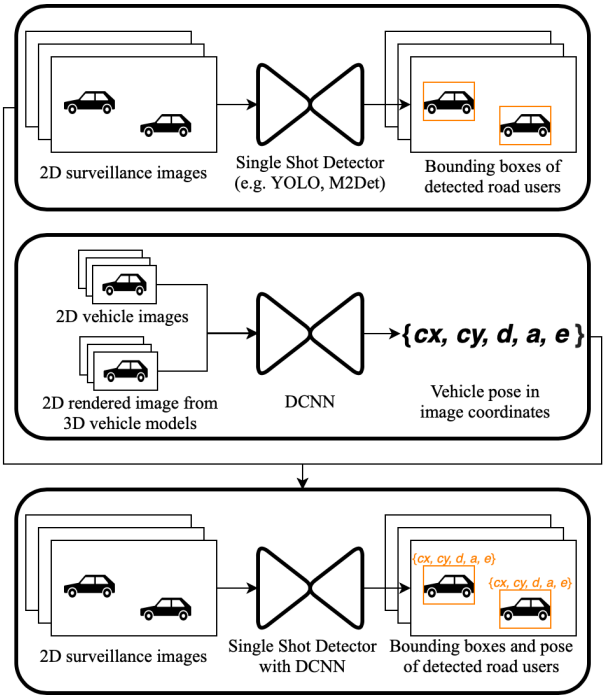

Figure 2. Framework overview

The architecture of our pose estimation network is based on the deep convolutional neural network (DCNN) ResNet-18 (He et al., 2016). As shown in Figure 3 we removed the last layer for classification and added an adaptive average pooling in order to reduce the different vehicle-image sizes to a fixed output of size $2 \times 2 \times 512$. This feature-layer is then reshaped to 2048 and fed to two consecutive fully connected layers with an final output size of 6. In order to estimate the azimuth rotation a and elevation e angles of the car we did not directly minimize the angle difference but approximated the sin and cos of the angles with an tangens hyperbolicus ($tanh$) instead. This should yield better results because we circumvent the problem of circular and signed angles. The angle can then be reconstructed using arctus tangens ($atan2$) of the approximated sin and cos angle.

We trained our network $f(x)$ by minimizing the sum of the normalized mean squared errors as shown in the following equation:

$$\begin{aligned}\mathcal{L} = &\frac{1}{2}((sin(a) - f(x)_1)^2 + (cos(a) - f(x)_2)^2 + \\ &\frac{1}{2}((sin(e) - f(x)_3)^2 + (cos(e) - f(x)_4)^2 + \\ &(cx - f(x)_5)^2 + (cy - f(x)_6)^2\end{aligned} \tag{1}$$

Whereby $f(x)_i$ indicates the value of the output vector at the position $i \in \{1, \cdots, 6\}$. Furthermore the center ground point is normalized into a range of $[-1, 1]$ by using the maximum image dimension. In order to predict the center ground points we also used an $tanh$ output for every image axis and scaled the output to the final image size:

$$\begin{aligned}\hat{cx} &= f(x)_5 \cdot \frac{h}{2} + \frac{h}{2} \\ \hat{cy} &= f(x)_6 \cdot \frac{w}{2} + \frac{w}{2}\end{aligned} \tag{2}$$

Wherby $\hat{cx}$ and $\hat{cy}$ are the estimated ground points.

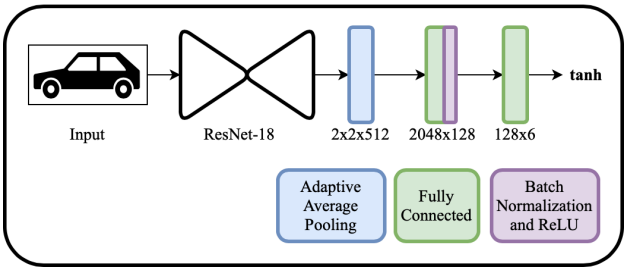

Figure 3. Deep convolutional neural network scheme. We removed the last layer of a ResNet-18 network and used it to encode the variable sized car images. The features are fixed in size with an adaptive average pooling and then fed to two consecutive fully connected layers. The car angles and the center ground points are estimated with the tanh of the output.

The network takes a vehicle image as input and estimates the pose parameters of the vehicle as explained above. Since the vehicle images are usually of different sizes, we have adaptively padded each batch to its maximum image size. Ultimately it is used on vehicles extracted from real surveillance camera data by an single shot detector like YOLO (Redmon et al., 2016) or M2Det (Zhao et al., 2019).

To train the deep convolutional neural network different data sources as described in chapter 3 were used: We extracted real car images with labeled pose information and rendered car images from 3D vehicle models. Since the car models are provided with no or unrealistic textures, we decided to create realistic renderings by using CycleGAN (Zhu et al., 2017) and pix2pixHD (Wang et al., 2018). We trained network models for

each dataset type separately as well as a model on combined real and rendered car images.

3. DATA

Waymo provides a large and diverse autonomous driving dataset (Sun et al., 2019), which is comprised of high resolution sensor data collected by Waymo self-driving cars in a variety of conditions. The dataset consists of 20s long segments with labeled 3D point clouds and corresponding but independently labeled 2D images, taken by five lidars as well as five cameras with a resolution of 1920 x 1280 pixel collected at 10Hz.

We sampled 800 segments of the dataset at 0.5Hz and extracted each *car patch*: a 2D car image with its minimal bounding box from the camera labels. Additionally the 2D bounding boxes from the camera labels were matched with the corresponding 3D bounding boxes of the lidar lables to create the following label information for each *car patch*:

- **distance (*d*):** the distance in meters from the camera to the car's center ground point.
- **elevation (*e*):** the elevation angel in degrees of the camera with respect to the car's center ground point in the range of -90°to 90°, where -90°equals a view from the bottom, 0°a view from the side and 90°a view from the top.
- **azimuth (*a*):** the azimuth rotation angel in degrees of the camera with respect to the car's center ground point in the range of 0°to 360°, where 0°equals a view at the car's front, 90°a view at the car's right side, 180°a view at the car's back and 270°a view at the car's left side.
- **center ground point (*cx, cy*):** the car's 3D center ground point projected to the 2D image coordinates of the *car patch*.
- **vehicle length:** the car's length in meters.
- **vehicle width:** the car's width in meters.
- **vehicle height:** the car's height in meters.

To filter wrongly labeled *car patches* and *car patches* where cars are highly occluded, we applied semantic image segmentation by using DeepLab (Chen et al., 2017) with the pretrained xception65_coco_voc_trainval[1] model. In the following this filtered and labeled dataset of approximately 25.000 *car patches* will be called *Waymo images*. Example patches are shown in Figure 4 on the left side.

To our knowledge ShapeNet (Chang et al., 2015) is the largest collection of labelled 3D models. For our domain ShapeNetCore (v2) contains around 3500 car models from which we manually chose 100. With the help of PyTorch3D (Ravi et al., 2020) these models were used to render 2D *car patches* matching the *Waymo images*. For each *car patch* of the *Waymo images* a random 3D model is chosen and rendered with the exact same attributes as the given *car patches*. Example patches are shown in Figure 4 on the right side. In the following this labeled dataset will be called *ShapeNet images*.

Since the ShapeNetCore car models are provided with no or unrealistic textures we decided to create realistic renderings by using CycleGAN and pix2pixHD. For CycleGan we trained an own model for 50 epochs from scratch using the paired *Waymo* and *ShapeNet images*. For pix2pixHD we used the pretrained label2city_1024p[2] model.

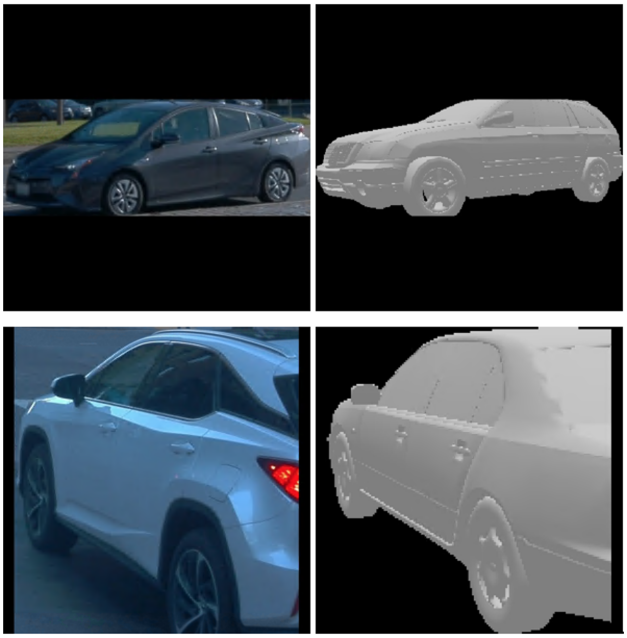

Figure 4. Waymo car patches (left) and corresponding rendered ShapeNet models (right)

With the help of the self-trained CycleGan model each *ShapeNet image* was textured. In the following this labeled datasets will be called *CycleGAN images*. Additionally CycleGAN and pix2pixHD were both applied to the manually chosen 100 ShapeNetCore vehicle models to create 5.000 textured *car patches*. For each *car patch* a random 3D vehicle model with known center ground point and orientation is rendered and a 2D image from the hemisphere with a precision of one degree around the vehicle is randomly selected from the range of 240°to 360°azimuth rotation and 0°to 25°elevation. In the following these labeled datasets will be called *CycleGAN images partial* and *pix2pixHD images partial* respectively. Example patches are shown in Figure 5.

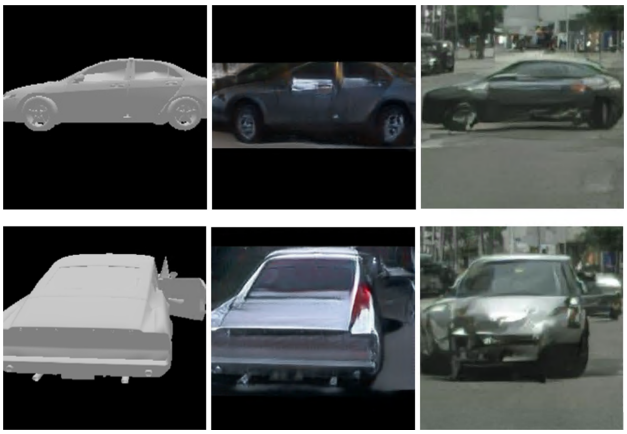

Figure 5. Rendered ShapeNet models (left) with CycleGAN texture (middle) pix2pixHD texture (right)

[1] https://github.com/tensorflow/models

[2] https://github.com/NVIDIA/pix2pixHD

4. EXPERIMENTS

Using our deep convolutional neural network and the datasets described in chapter 3 we trained and evaluated different models, namely:

- **Waymo-Full:** training and evaluation on all real car images (*Waymo images*). The aim of this is to provide a baseline for the subsequent experiments, where only a subset of possible orientations is used.
- **Waymo-Partial:** training and evaluation on the real car images (*Waymo images*) excluding all *car patches* with an azimuth rotation greater than 240°.
- **Waymo-CycleGAN:** improvement of *Waymo-Partial* by an additional training and evaluation on rendered car images with artificially generated textures from CycleGAN (*CycleGAN images partial*). We added the same amount of synthetic car images with the same angles as excluded in *Waymo-Partial* (i.e. azimuth rotation greater than 240°), so that the synthetic data should replace the excluded data.
- **Waymo-Pix2Pix:** same as *Waymo-CycleGAN* but instead of using artificial generated textures from CycleGAN, the textures are generated with pix2pixHD (*pix2pixHD images partial*).

All models were trained for 20 epochs with a batch size of 5. Therefore the above mentioned datasets, namely *Waymo images*, *ShapeNet images*, *CycleGAN images*, *CycleGAN images partial* and *pix2pixHD images partial* were each split randomly into training (85%) and validation (15%) sets. The trained network models were finally tested against a test set parallel to the *Waymo images* consisting of approximately 2100 *car patches* (∼10% of the number of training images). In the following this labeled datasets will be called *Full-Waymo test set*. Furthermore we created a second test set by excluding all angles between 240°and 360°from the *Full-Waymo test set*. In the following this labeled datasets will be called *Partial-Waymo test set*. The expectation is that our network will perform better with *Waymo-Cyclegan* than with *Waymo-Partial* due to the additional introduced synthetic data in former. This will be particularly visible on the second test set ($a : 240° - 360°$) because it contains new viewing angles.

5. RESULTS & DISCUSSION

Table 1 shows the results of the performed experiments. For each network model the average precision of the azimuth rotation and elevation angle (in degree) as well as the center ground in pixel (using the Euclidean distance) is given. Additionally we included the row 'Naive (mean)', representing an pose estimator always returning the validation sets mean values. Since the results of the respective evaluation and test sets are very similar, in Table 1 only the results for the full ($0° - 360°$) and partial ($240° - 360°$) test sets are presented.

The experiment baseline (*Waymo-Full*) of our deep convolutional neural network has an average accuracy of 11.10°for the azimuth rotation, 1.52°for the elevation angle, 22.06 pixel for the center ground x and 14.34 pixel for the center ground y coordinate when evaluating on the *Full-Waymo test set* and comparable results for the *Partial-Waymo test set*. With respect to the corresponding naive baselines *Waymo-Full* has higher accuracies for the azimuth rotation, elevation angle, center ground x and center ground y coordinate in comparison to the naive baseline, significantly for both the *Full-Waymo test set* and the *Partial-Waymo test set*. This demonstrates our network is able to estimate vehicle poses from *car patches*.

Full-Waymo test set ($a : 0° - 360°$)				
Method	$\overline{a}$	$\overline{e}$	$\overline{cx}$	$\overline{cy}$
Waymo-Full	11.10°	1.52°	22.06px	14.34px
Waymo-Partial	22.42°	1.54°	25.60px	14.59px
Waymo-CycleGAN	23.44°	1.90°	22.42px	20.87px
Waymo-Pix2Pix	23.27°	2.78°	21.69px	31.30px
Naive (mean)	78.79°	2.82°	84.55px	80.83px
Partial-Waymo test set ($a : 240° - 360°$)				
Method	$\overline{a}$	$\overline{e}$	$\overline{cx}$	$\overline{cy}$
Waymo-Full	10.75°	1.42°	21.56px	15.30px
Waymo-Partial	121.22°	2.10°	30.10px	13.09px
Waymo-CycleGAN	72.09°	2.51°	18.00px	26.56px
Waymo-Pix2Pix	73.95°	2.50°	17.78px	29.57px
Naive (mean)	78.79°	2.82°	84.55px	80.83px

Table 1. Mean pose estimation accuracies of our deep convolutional neural network for the *Full-* and *Partial-Waymo test set*

As expected the accuracies for the azimuth rotation and elevation decrease when using a training set excluding all *car patches* with an azimuth rotation greater than 240°(*Waymo-Partial*) in comparison to the experiment baseline (*Waymo-Full*). This is caused by the fact, that the network cannot generalize to unseen viewing angles. Consequently it fails to predict them during testing.

The *Waymo-CycleGAN* experiment has higher accuracies for the azimuth rotation and elevation angle in comparison to *Waymo-Partial*, significantly for the *Partial-Waymo test set* and slightly worse for the *Full-Waymo test set*, but still does not reach the accuracies of *Waymo-Full*. This proves our assumption that rendered 3D vehicle models with artificially generated textures are helpful for training the pose estimation network for *car patches* not in the initial training dataset. In case of *Waymo-CycleGAN* the deep convolutional neural network could only learn to estimate car poses greater than 240°from the rendered 3D vehicle models. For the type of artificially generated texture we could not find any difference, *Waymo-CycleGAN* and *Waymo-Pix2Pix* share similar results.

Furthermore we investigated not only the networks average precision but also the azimuth rotation error with respect to the elevation angle as well as to the azimuth rotation itself. As exemplified at the top in Figure 6 and Figure 7, with increasing elevation angle the azimuth rotation error decreases. This means the deep convolutional neural network estimates vehicle poses more accurate on higher elevation angles (see also Table 2), which also confirms the intuition that higher elevation angles allow for a more precise pose and orientation determination.

This fact also applies for the azimuth rotation error with respect to the azimuth rotation, as presented at the bottom in Figure 6 and Figure 7. The deep convolutional neural network estimates vehicle poses more accurate for azimuth rotations around 0°and 180°. This correlates with the distribution of the azimuth rotation in the training, validation and test sets of the trained network models, which share the same distribution. Due to the

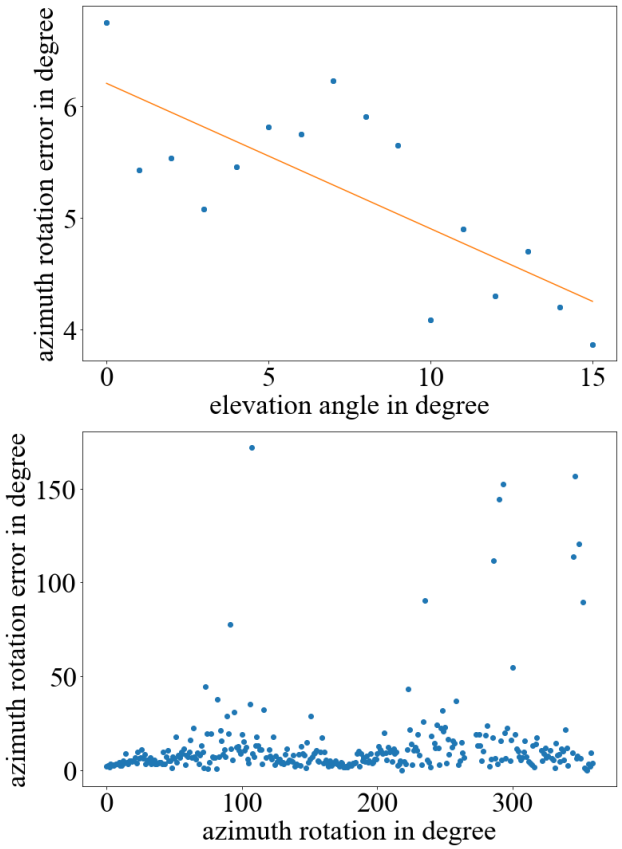

Figure 6. Median azimuth rotation error with respect to the elevation angle (top) and with respect to the azimuth rotation (bottom) of *Waymo-Full* for the *Full-Waymo test set*

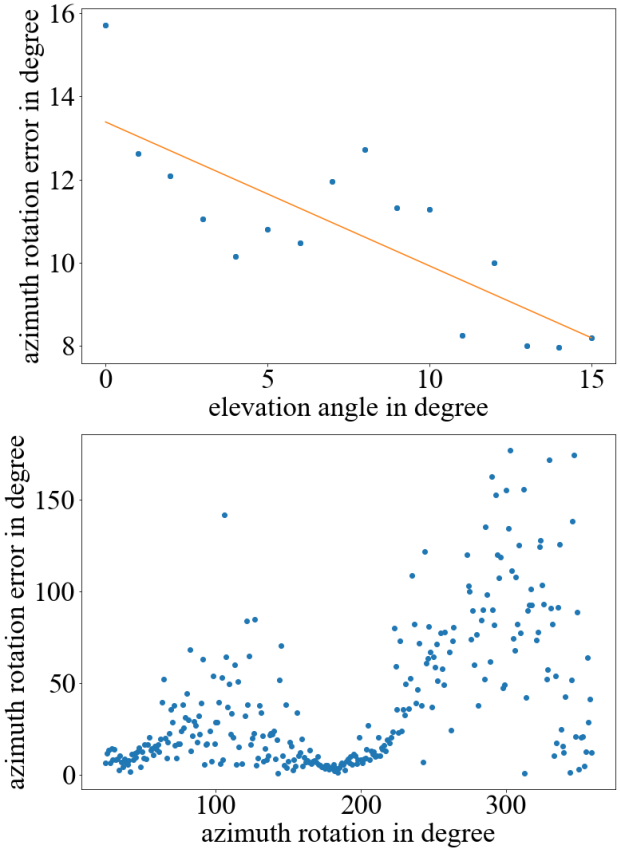

Figure 7. Median azimuth rotation error with respect to the elevation angle (top) and with respect to the azimuth rotation (bottom) of *Waymo-CycleGAN* for the *Full-Waymo test set*

Full-Waymo test set ($a : 0° - 360°$)			
Method	$\overline{a}$ (e<5)	$\overline{a}$ (5≥e<10)	$\overline{a}$ (e≥10)
Waymo-Full	13.95°	12.37°	09.22°
Waymo-CycleGAN	25.95°	24.17°	19.51°
Partial-Waymo test set ($a : 240° - 360°$)			
Method	$\overline{a}$ (e<5)	$\overline{a}$ (5≥e<10)	$\overline{a}$ (e≥10)
Waymo-Full	30.89°	15.23°	13.22°
Waymo-CycleGAN	81.74°	82.79°	69.75°

Table 2. Mean azimuth rotation accuracies of our deep convolutional neural network for the *Full-* and *Partial-Waymo test set* for different elevation angle classes

fact, that the *Waymo-images* are recorded from the perspective of (self-) driving cars, the *Waymo-images* have an unbalanced distribution of azimuth rotation and elevation angle (see Figure 8). The datasets contain more front- and backfacing cars than sidefacing ones for small elevation angles and thus a higher pose estimation precision due to the higher availability of training data for this viewing angles can be reached.

6. CONCLUSION & OUTLOOK

We showed that even with a simple custom deep convolutional neural network it is possible to estimate vehicle poses for car-patch-images by reaching accuracies of ∼10°for the azimuth rotation, ∼1.5°for the elevation angle and ∼20px for the center ground point. Additionally we demonstrated that those neural networks can be trained by non real data using rendered 3D vehicle models with artificial generated textures by CycleGAN and pix2pixHD.

Even though first experiments show promising results there is room for improvement. First, using real datasets with higher elevation angles (e.g. from road junction surveillance cameras), should be used in order to complement more perspectives with real data. Since in this work we only used a slightly modified version of ResNet-18 as deep convolutional neural network and focused on usage and generation of training data, next steps will deal with adapting more advanced networks, which are proven to estimate object poses more precisely, like (Xiang et al., 2017), to our domain of vehicles.

Furthermore, vehicle poses, which are estimated by the introduced deep convolutional neural network could be used in order to adapt and retrain object detection networks, like YOLO or M2Det, to directly predict the vehicle pose without estimating an object bounding box beforehand, like (Tekin et al., 2018).

Additionally, we want to use domain adaptation to close the domain gap between the synthetic and the real vehicle images. A possible solution would be to extend the training of CycleGAN on an large unpaired dataset of the synthetic vehicles and the images extracted by an object detection network of surveillance camera data. This approach would probably lead to more real-

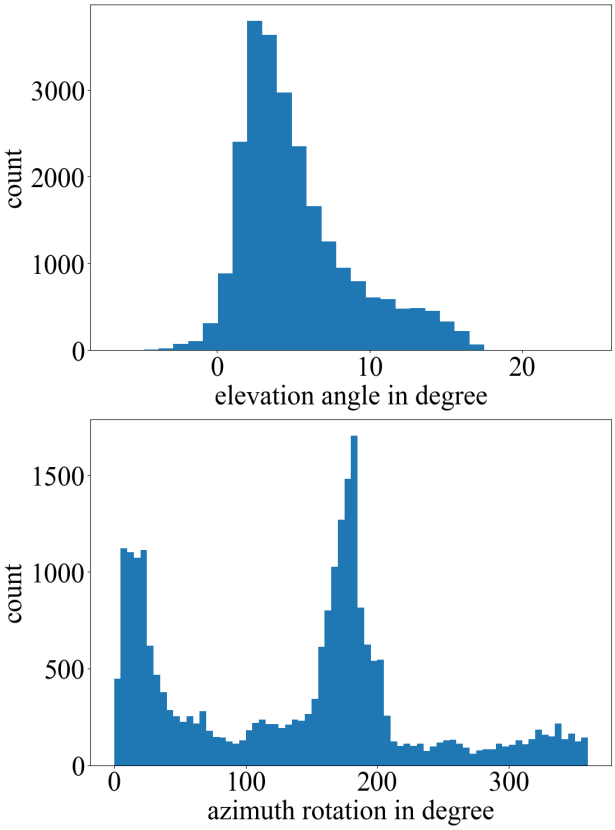

Figure 8. Distribution of the elevation angle (top, bin size of 1°) and azimuth rotation (bottom, bin size of 5°) in the *Waymo*, *ShapeNet* and *CycleGAN images*

istic looking vehicle images of ShapeNet. It is imaginable that instead of regressing the car-pose we could reduce the pose estimation to a classification problem that predicts discrete angles. We could then use adversarial discriminative domain adaptation (Tzeng et al., 2017) to directly decrease the domain gap in the feature space.

REFERENCES

Barnard, Y., Utesch, F., van Nes, N., Eenink, R., Baumann, M., 2016. The study design of UDRIVE: the naturalistic driving study across Europe for cars, trucks and scooters. *European Transport Research Review*, 8, 14.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F., 2015. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A. L., 2017. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40, 834–848.

Coenen, M., Rottensteiner, F., 2019. Probabilistic vehicle reconstruction using a multi-task cnn. *The IEEE International Conference on Computer Vision (ICCV) Workshops*.

Geiger, A., Lenz, P., Stiller, C., Urtasun, R., 2013. Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Huang, H., Zhang, L., Sester, M., 2014. A recursive bayesian filter for anomalous behavior detection in trajectory data. *Connecting a Digital Europe Through Location and Place*, Springer, 91–104.

Koetsier, C., Busch, S., Sester, M., 2019. Trajectory extraction for analysis of unsave driving behaviour. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W13, 1573–1578. https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-2-W13/1573/2019/.

Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W.-Y., Johnson, J., Gkioxari, G., 2020. Pytorch3d. `https://github.com/facebookresearch/pytorch3d`.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779–788.

Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., Anguelov, D., 2019. Scalability in perception for autonomous driving: Waymo open dataset.

Tekin, B., Sinha, S. N., Fua, P., 2018. Real-time seamless single shot 6d object pose prediction. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 292–301.

Tzeng, E., Hoffman, J., Saenko, K., Darrell, T., 2017. Adversarial discriminative domain adaptation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7167–7176.

Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Tao, A., Kautz, J., Catanzaro, B., 2018. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. http://dx.doi.org/10.1109/CVPR.2018.00917.

Xiang, Y., Schmidt, T., Narayanan, V., Fox, D., 2017. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*.

Zhang, S., Wang, C., He, Z., Li, Q., Lin, X., Li, X., Zhang, J., Yang, C., Li, J., 2020. Vehicle global 6-DoF pose estimation under traffic surveillance camera. *ISPRS Journal of Photogrammetry and Remote Sensing*, 159, 114 - 128.

Zhao, Q., Sheng, T., Wang, Y., Tang, Z., Chen, Y., Cai, L., Ling, H., 2019. M2det: A single-shot object detector based on multi-level feature pyramid network. *The Thirty-Third AAAI Conference on Artificial Intelligence,AAAI*.

Zhu, J.-Y., Park, T., Isola, P., Efros, A. A., 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. *The IEEE International Conference on Computer Vision (ICCV)*.