

Where to drive? Free Space Detection with one Fisheye Camera

Tobias Scheck, Adarsh Mallandur, Christian Wiede, Gangolf Hirtz

Chemnitz University of Technology

ABSTRACT

The development in the field of autonomous driving goes hand in hand with ever new developments in the field of image processing and machine learning methods. In order to fully exploit the advantages of deep learning, it is necessary to have sufficient labeled training data available. This is especially not the case for omnidirectional fisheye cameras. As a solution, we propose in this paper to use synthetic training data based on Unity3D. A five-pass algorithm is used to create a virtual fisheye camera. This synthetic training data is evaluated for the application of free space detection for different deep learning network architectures. The results indicate that synthetic fisheye images can be used in deep learning context.

Keywords: Free Space Detection, Fisheye Camera, Deep Learning, CNN, Synthetic Data Creation.

1. INTRODUCTION

In recent years, the idea of autonomous driving has gained tremendous momentum. All automotive companies and suppliers have set up large research departments to turn the dream of driver-less driving into reality. The development is currently between levels 2 (Partial automation) and 3 (Conditional automation) of the level of automation. These includes systems such as adaptive cruise control, parking assistance or driving on highway. In order to reach levels 4 (High Automation) and 5 (Full automation), however, further challenges have to be mastered. A fundamental challenge is the detection of free spaces in which the car can generally move. These can be streets, side walks or meadows but not parking vehicles, lanterns, trees or people occupying the place. Therefore, it is essential for the navigation to detect these free spaces. This is not sufficiently fulfilled by existing systems. In addition, many external sensors are required.

In this paper, we propose a way to detect free spaces using a single omnidirectional camera with a fisheye lens. Fisheye cameras can capture a wide field of view. This is at the expense of strong tangential and radial disturbances in the image. For a proper training with machine learning algorithms a huge amount of training data is necessary. However, less or no labeled training data exists for fisheye pictures. Especially the semantic segmentation requires a pixel by pixel labeling. This cannot be done manually without an extreme amount of resources. Instead, we suggest to use synthetic data for the training. If these are as realistic as possible, these images can be used for training and create a model working on real world data. In order to create these images urban traffic scenes by Unity3D are used. This environment can realize different backgrounds and dynamic driving scenarios. Furthermore, we propose a five-pass algorithm in combination with synthetic data to generate virtual fisheye images and a transfer learned convolutional neural network (CNN) to generate a semantic segmentation.

This work is structured as follows: In Sect. 2, the related work of deep learning based segmentation and synthetic data acquisition is outlined and the research gap highlighted. The synthetic data generation by means of Unity3D is described in Sect. 3. This is followed by the presentation of the deep learning in Sect. 4. In Sect. 5, we present our experimental results, which is accompanied by a discussion. Finally, we summarise our outcomes and outline future work.

2. RELATED WORK

Identifying drivable areas and detection of surrounding obstacles is a crucial task for autonomous driving. A mixture of different sensors such as radar, lidar, camera, high precision GPS and prior map information is used to determine free space in current autonomous vehicles [1, 2]. Research in autonomous driving started in Europe in 1986 with the PROMETHEUS project [3] involving more than 13 vehicle manufacturers and several universities from Europe. VaMP driverless car was a notable result of this project which covered 95 % of about a distance of 1,600 km fully automatically [4] and is considered as one of the first autonomous cars [5]. Projects such as these were usually computer vision systems for lateral and longitudinal vehicle guidance [6, 7], lane departure warning [8] or collision avoidance [9]. The success of PROMETHEUS

and other similar projects [9, 10] influenced researchers to delve into more complex urban environments from the much simpler highway scenarios. The term urban environment means that the focus is towards the problem of obstacle detection or free space detection rather than vehicle guidance. Franke et al. [11] propose an "Intelligent Stop & Go system", using depth-based obstacle detection and tracking from stereo images for urban environment. In the 2000s, the problem of road detection was solved by using low-level features such as color, shape [12] and texture [13] of the road. Detection of road borders or road markings using laser [14], radar [15], Hough transform [16] etc. is also used as a technique to detect drivable space. However, such algorithms fail under severe lighting variations and depend strongly on the structure of the roads. Hence, more and more algorithms relied on geometric modeling using stereo images. Badino et al. introduce a method for free space detection in complex scenarios using stochastic occupancy grids [17]. These grids are built using data from stereo cameras and are integrated over time with Kalman filters. Occupancy grids are in a way segmentation between free and occupied regions. Apart from segmentation of pixels, other representations such as Stixels [18], which groups objects with vertical surfaces, have also been used for the task of urban traffic scene understanding.

In the past decade, segmentation of road scenes using computer vision techniques has been considered as a standard approach to detect free spaces [19]. There are techniques ranging from Watershed Algorithms [20], Dense Stereo Maps [21], Structure from Motion (SfM) [22] to global methods such as Conditional Random Fields (CRF) [23] or Boosting Algorithms [24]. Inspired from the success of CNNs in tasks such as classification, researchers have found ways to apply CNNs in the task of road segmentation [19, 25, 26]. Alvarez et al. used CNN based algorithm to recover 3D scene layout of a road image [19]. Hereby, the problems of overfitting and manual label generation are solved by using noisy labels generated from a classifier, which is trained on a general image dataset. The authors in [25] introduce convolutional patch networks, which perform a pixel-wise labeling of input road scene images, by first applying patch segmentation or classification of image patches. The deconvolution networks used in [26] show a good performance on the KITTI benchmark. However, this approach is computationally expensive and is not suited for real-time road segmentation [27]. The most notable work in the field of semantic segmentation are the fully convolutional networks (FCN) [28], which are realized by replacing fully connected layers of a CNN by convolutional layers and then up-sampling the resulting feature map into a pixel-wise segmented image thus enabling end-to-end learning. The absence of fully connected layers meant that the input can be of any size and that the spatial information is preserved. Furthermore, the resulting architecture has a smaller number of trainable parameters, thus achieving better performance than patch networks.

Although state-of-the-art semantic segmentation networks achieve excellent accuracy, maximum information about the surrounding area is necessary for safe navigation. Fisheye cameras provide a wider field of view than narrow-angle pin-hole cameras in complex urban traffic scenes and are becoming more popular in vehicles since they are cheap and easy to handle [29, 30, 31]. However, fisheye images are distorted due to strong perspective projections and are consequently unwarped for practical usage. This unwarping process decreases the image quality, especially at the image boundaries. Thus, CNNs trained on pin-hole camera images do not perform well on fisheye images. Several research works have been carried out to develop algorithms that directly apply on distorted fisheye images [32, 33, 34, 35, 36]. The main problem of these algorithms is the lack of labeled fisheye urban scene datasets that have good quality and rich scenes. Therefore, Qian et al. [36] devise adversarial method of training CNNs using images from pin-hole camera images dataset to detect pedestrians in fisheye images. An adversarial network generates fisheye pedestrian features, which are difficult for the detector to classify. However adversarial networks are considered difficult to train because of various reasons such as unstable parameters, sensitivity to hyperparameters, mode collapses, vanishing gradients and overfitting [37]. In [33, 34, 35], the authors synthetically generate fisheye images with labels from Cityscapes [38] and SYNTHIA [39] datasets to train a CNN. The problem is that a transformation from a perspective to a fisheye image, loses image details in the centre during warping. Our work will solve this problem by devising a method to generate realistic synthetic fisheye images along with the labels based on 5-pass rendering algorithm implemented in Unity3D. It solves the problem of dataset generation as well as that of training a CNN directly on distorted fisheye images by providing a way to generate fisheye datasets of different urban traffic scenes.

3. SYNTHETIC DATA CREATION

In this section, we describe the generation of fisheye images along with their labels by a 5-pass rendering algorithm by using Unity3D. Currently, different types of cameras such as perspective or omnidirectional cameras are used for autonomous driving. Omnidirectional cameras provide a wider field of view (FOV) than perspective cameras thus eliminating the need for more cameras or mechanically rotatable cameras. An ideal omnidirectional camera can capture light from all directions

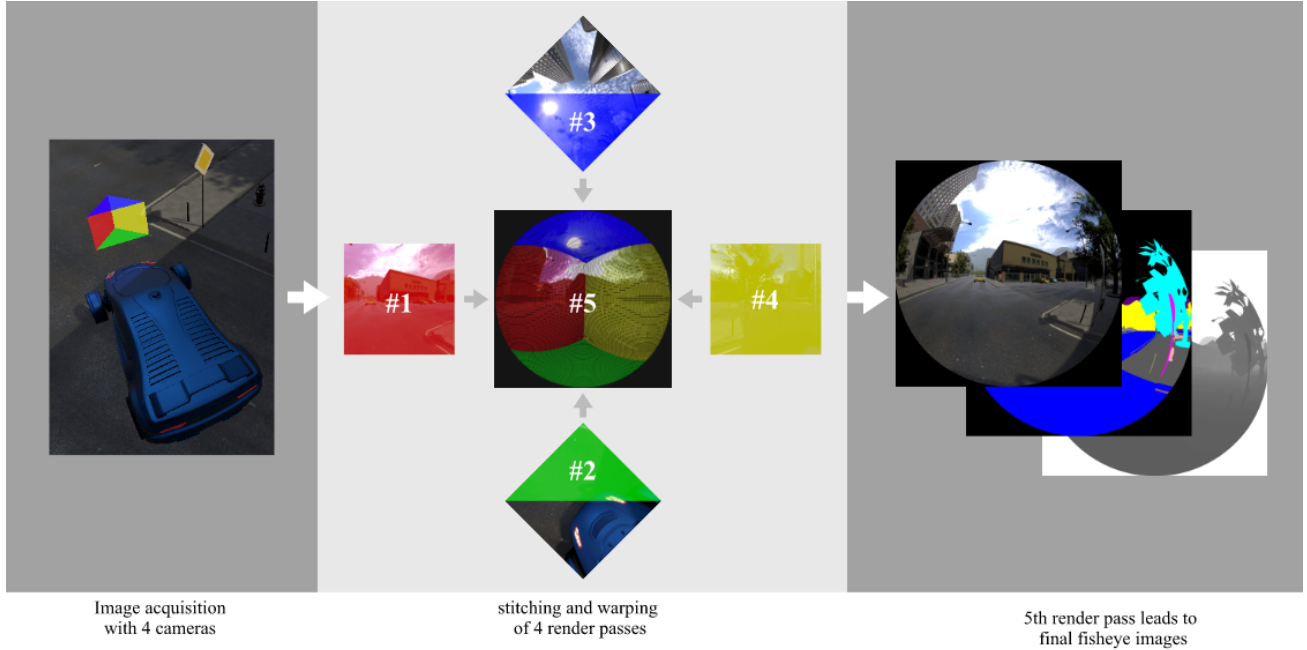


Figure 1: Pipeline of synthetic data acquisition beginning with 4 cameras, each camera projection represents a face of a cube. In the second step, each projection result is used as a texture for a precomputed mesh to form a fisheye image. The previous steps are performed for each type of synthetic data (RGB image, segmentation mask and depth map) and exported as an image.

falling onto the focal point, thus covering a full sphere (360° FOV). There are various omnidirectional cameras ranging from catadioptric cameras with curved mirrors to dioptric cameras with purely dioptric fisheye lenses. The name fisheye lens was coined by R. W. Wood to refer any lens capable of imaging the entire hemisphere in object space onto a finite circle in the focal plane [40]. Fisheye lenses are designed to cover the whole hemispherical field in front of the camera with a FOV of about 180° . Such fisheye lens cameras cannot be modeled by a linear projection due to very high distortions in the images caused by refractions in fisheye lenses. The distortions in fisheye images where straight lines appear curved are known as radial symmetric distortions and are usually caused by the shape of the lens [41]. Furthermore, these distortions are not uniformly distributed over all spatial areas [32]. There are several models to describe the behavior of fisheye lenses such as equidistant projection, stereographic projection, equisolid angle projection and orthogonal projection. However, real lenses do not exactly follow the designed projection model [42]. Using such models, the authors in [33, 34, 35, 43] generated synthetic fisheye images from existing datasets with standard perspective projection images captured from a single camera. Instead of transforming existing perspective projection images to fisheye projection images, our method directly synthesizes fisheye projection images using virtual cameras within Unity3D environment.

In order to generate a dataset for training, Microsoft Airsim simulator [44] designed for Unity3D is used. Airsim provides a plugin with Python interface and a vehicle asset to be used in any Unity3D environment. To help Artificial Intelligence (AI) research, Unity3D offers free Windridge City environment inside which Airsim asset vehicle can be imported. The vehicle can be controlled either by keyboard or through the Airsim Python interface. The generation of fisheye images is based on well-known technique of rendering to cubic maps, where a 360° view of the entire scene is rendered onto a cube. In our case, we render to four faces of a cube by using each a perspective camera with a FOV of 90° , as shown in Fig. 1. The image captured by each camera is rendered into a texture, which is then used for the corresponding precomputed mesh to form a fisheye image. The texture coordinates of these four meshes are designed in such a way that they capture the radial symmetric distortions and produce a fisheye projection. Further information about the mesh generation can be obtained such as described in [45]. This is a computationally inexpensive method compared to a single pass rendering algorithm with a vertex shader that pre-distorts the geometry of the world to produce a fisheye projection [45].

In the five-pass rendering algorithm, four orthogonal cameras each with 90° FOV are created inside Unity3D and are placed above the virtual car. The fifth camera is orthographic and is positioned to capture the fisheye projection formed by the combination of the four meshes. During one frame update procedure of Unity3D, for each orthogonal camera, three additional orthogonal hidden cameras at the same position are created: one for the source image rendering, one for the label image rendering and one for the depth image rendering. All these three hidden cameras render to the same texture as their parent cameras. This approach is based on [46] and we modified it to use it with the five-pass rendering algorithm. In order to render the label image, the layer id of each object is mapped to a color and is passed to the vertex shader for rendering. The depth label is computed inside the vertex shader by calculating the normalized distance value of each pixel from the near plane and assigning it to that pixel intensity. Therefore, for one image-saving cycle, three fisheye projections are obtained: source fisheye image, label fisheye image and depth fisheye image. To preserve the floating-point values in the depth image, the .exr file format is used.

Each object inside Unity3D is assigned with a layer. Thus, the total number of classes in the training set is equal to the number of layers in the Unity3D environment. Unity3D supports a maximum of 32 layers. These layer IDs along with their names and mapped colors are exported as JSON file in order to retain the mapping information. The files were saved at a rate of one frame per second in order to reduce redundant images and the CPU load. In order to train a classifier, 12028 images were generated by driving the virtual car around the Windridge city. We call the generated dataset *OmniCity* and the usage of a 80/20 training/validation split ratio results in 9623 training and 2405 validation images. Thus, the proposed method provides a flexible way to generate a dataset in a urban driving environment.

4. DEEP LEARNING

This section introduces the CNN free space detection architectures and the hyperparameters used for trainings. The selected learning strategies and data augmentation methods during the trainings are described as well.

The detection of free spaces in a scene requires a general image understanding on pixel level. In our approach we are using CNNs designed for semantic segmentation tasks to train and classify pixel-wise. As shown in Fig. 2, the first model is based on the implementation of SegNet [47], which is an encoder-decoder architecture. This architecture gradually reduces the spatial dimension in the encoder part, while the decoder part behaves in the opposite direction. In concrete terms, this means as the CNN progresses, object details and spatial dimensions are restored.

DeepLabV3 was chosen as additional CNN architecture [49] with ResNet101 v2 [50] as feature extractor. Pooling layers, used in encoder-decoder architectures, are designed in such a way that they increase the field of view and aggregate necessary information but at the same time discards the local context. However, this information is fundamental to semantic segmentation and should not be dispensed with. To address this problem, the implementation of DeepLabV3 uses atrous convolutions. They are also known as dilated convolutions and allow an enlargement of the field of view without reducing the spatial dimension and this at the same time efficiently [51, 52].

Both architectures use RGB images as input with a resolution of 512×512 pixel. By using ResNet101 v2 as feature extractor, weights, pre-trained on ILSVRC2012-CLS image classification dataset [53], are used for DeepLabV3. The encoder-decoder weights are initialized using the technique described in Glorot et al. [54]. To train all architectures we use stochastic gradient descent (SGD) with a momentum of 0.9. An initial learning rate of 0.0001 for DeepLabV3 and 0.001 for the encoder-decoder architecture is selected. A training process can benefit from a reduction in the learning rate during training progress. To address this property, we use cosine decay [55] as our learning rate strategy. We train each architecture for 50 epochs with a mini-batch size of 4. In the training process, we shuffle the dataset after each epoch to ensure that each training example is used only once during an epoch. As objective function, cross-entropy loss is used for training which is summed over all pixel values in a mini-batch. The last layer for each architecture calculates a per-pixel propability using softmax. In this work, the CNNs distinguish only between free space and background.

In order to avoid over-fitting and ensure better generalization, data augmentation methods are applied during training [56, 57]. All methods depend on a random factor to decide a method is applied or not. We have selected horizontal flipping, brightness changing with a max_delta of 0.5 and random Gaussian noise with a mean of 0.0 and a standard deviation of 8.0. For a training with the generated synthetic data we also change the hue of a RGB image and the saturation.

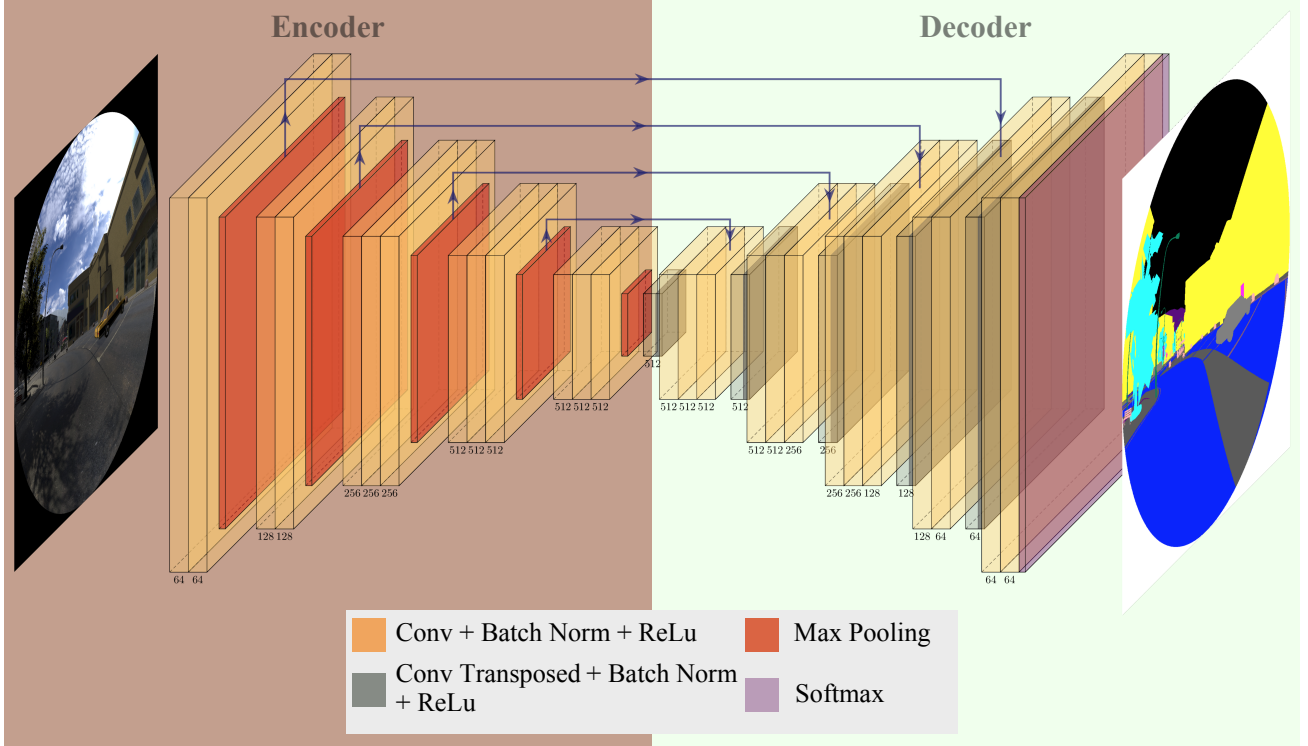


Figure 2: This overview visualizes the used encoder-decoder structure for free space detection. While the encoder produces sparse feature maps using max pooling, the decoder uses a transposed convolution for a learnable upsampling. Visualization based on [48]

5. RESULTS AND DISCUSSION

In this section, we outline our experiments and discuss the corresponding result. In the first experiment, we use the training set of the Cityscapes dataset [38], containing 2975 images, to train both CNN architectures on perspective images of urban street scenes. Cityscapes contains segmentation masks for 30 classes. However, for free space detection we map the classes: road, sidewalk, parking, rail track and terrain to the class *free space* and everything else to the class *background*. The second experiment aims to simulate a fisheye distortion by mapping perspective images into fisheye images. Accordingly all images of Cityscapes are distorted with a focal length of 159 based on the approach described in Deng et al. [33], named Fisheye Cityscapes. In the third experiment, we use our approach for synthetic fisheye data generation to train all CNNs on OmniCity.

After training the architectures, we validate the performance using a sequence of real fisheye images. For this purpose, a subset of the fisheye data set DriveA sequence [58] was manually labeled, in a coarse manner, for our free space scenario*. In order to evaluate the 150 labeled images of this subset, the Intersection-over-Union (IoU) was used. The IoU is described as the quotient of the area of overlap of the ground truth and detected mask and area of union of both masks. During the evaluation all pixels beyond the fisheye FOV are labeled as void and ignored.

As shown in Tab. 1, the results for the CNN architectures trained on Cityscapes are those that achieve the highest IoU. To the contrary, the results for Fisheye Cityscapes decrease by 14.3 % for Deeplab v3 and by 2.58 % for the Encoder-Decoder structure. It seems that the distortions caused by a fisheye lense do not affect the free space detection results. This fact is supported by the results achieved by a training with OmniCity. The generated synthetic data, following the equiangular projection, achieves comparable results to Fisheye Cityscapes. The IoU decreases, compared to Cityscapes, by 7.45 % using Deeplab v3 and about 6.87 % with the Encoder-Decoder. However, it should be noted that, during training with OmniCity, none of the CNN architectures have ever seen real data.

*<https://gitlab.com/tschec/annotations>

Table 1: IoU (%) for free space detection after training using Cityscapes, Fisheye Cityscapes and our proposed synthetic dataset OmniCity.

Dataset	CNN	
	Encoder-Decoder	Deeplab v3
Cityscapes	93.1	84.6
Fisheye Cityscapes	90.7	72.5
OmniCity	86.7	78.3

An explanation for the poorer performance using fisheye distorted images can be the fact that the class free space is mainly not affected by those. Unlike objects, it is difficult to recognize a pattern in free space. Roads do not follow clear lines or terrain like meadows and fields rarely have flat surfaces, but these properties are most affected by distortion. It can be assumed that for more complex classes such as cars, people or buildings, the distortion has a greater impact than that studied in our scenario.

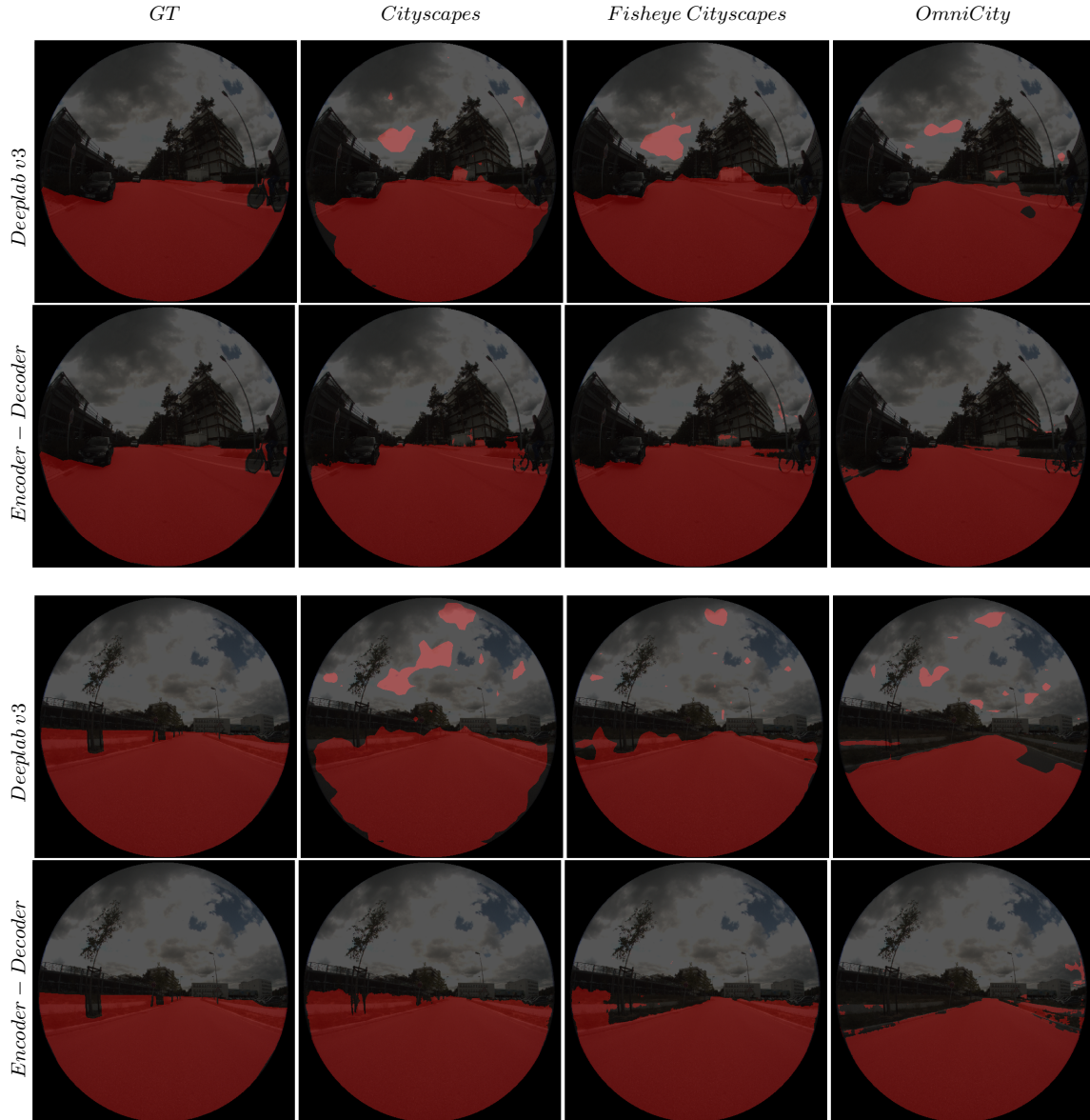


Figure 3: Qualitative results of the evaluated CNN architectures, trained on different datasets.

In Fig. 3, qualitative free space segmentation results for the experiments are shown. In general, Deeplab v3 is found to be slightly worse than the Encoder-Decoder architecture. However, this also reflects the results from Tab. 1. In the first selected scenario (row 1 and 2), a recognizable gap of quality between the results of all Encoder-Decoder architectures seems minimal. The difference in segmentation quality only becomes apparent in the last scenario (row 3 and 4). While the CNNs trained with Cityscapes and Fisheye Cityscapes are able to segment roads, footpaths and even terrain between them, the model trained with OmniCity is not detecting these accurately. This is particularly evident in the grassy areas and footpaths. One possible cause may be a lack of training examples with such constellations.

However, we could demonstrate that it is possible to learn deep learning architectures based on synthetic fisheye data and are able to apply that model to real world data. Nevertheless, further work as to be carried out in this field.

6. CONCLUSION

In this work, we proposed an approach for free space detection using a single fisheye camera to benefit from its wider FOV. Due the lack of omnidirectional training data, we use a data augmentation method, introduced in [33], to simulate a fisheye distortion on perspective images. Additionally, we described a method to generate synthetic omnidirectional images, segmentation masks and depth maps, using Unity3D. A new omnidirectional dataset, called OmniCity, with urban synthetic city scenes is introduced. Furthermore, an Encoder-Decoder and the Deeplab v3 CNN architecture in the context of free space detection was evaluated, which is commonly used for semantic segmentations tasks.

It can be shown that for free space detection on omnidirectional images, the use of fisheye distorted images for training is not mandatory, but can be advantageous in case no real world data is available. However, during our evaluation we have shown that CNNs trained only on synthetic data can achieve comparable results. In future research, the evaluation of more complex semantic segmentation scenarios should be investigated to address the usage of omnidirectional images. For this purpose, we will improve our OmniCity dataset to include a higher variation in urban scenes, different light and weather conditions, different vegetation and more classes like people and additional vehicles.

REFERENCES

- [1] Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., et al., “Junior: The stanford entry in the urban challenge,” *Journal of field Robotics* **25**(9), 569–597 (2008).
- [2] Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M., Dolan, J., Duggins, D., Galatali, T., Geyer, C., et al., “Autonomous driving in urban environments: Boss and the urban challenge,” *Journal of Field Robotics* **25**(8), 425–466 (2008).
- [3] Williams, M., “Prometheus-the european research programme for optimising the road transport system in europe,” in *[IEE Colloquium on Driver Information]*, 1–1, IET (1988).
- [4] Dickmanns, E. D. et al., “Vehicles capable of dynamic vision,” in *[IJCAI]*, **97**, 1577–1592 (1997).
- [5] Dickmanns, E. D., *[Dynamic vision for perception and control of motion]*, Springer Science & Business Media (2007).
- [6] Weber, J., Koller, D., Luong, Q.-T., and Malik, J., “New results in stereo-based automatic vehicle guidance,” in *[Proceedings of the Intelligent Vehicles’ 95. Symposium]*, 530–535, IEEE (1995).
- [7] Pomerleau, D. and Jochem, T., “Rapidly adapting machine vision for automated vehicle steering,” *IEEE expert* **11**(2), 19–27 (1996).
- [8] Franke, U., Mehring, S., Suissa, A., and Hahn, S., “The daimler-benz steering assistant: a spin-off from autonomous driving,” in *[Proceedings of the Intelligent Vehicles’ 94 Symposium]*, 120–124, IEEE (1994).
- [9] Thorpe, C., Hebert, M. H., Kanade, T., and Shafer, S. A., “Vision and navigation for the carnegie-mellon navlab,” *IEEE Transactions on Pattern Analysis and Machine Intelligence* **10**(3), 362–373 (1988).

- [10] Morimoto, H., Koizumi, M., Inoue, H., and Nitadori, K., "Ahs road-to-vehicle communication system," in [*Proceedings 199 IEEE/IEEE/JSAI International Conference on Intelligent Transportation Systems (Cat. No. 99TH8383)*], 327–334, IEEE (1999).
- [11] Franke, U., Gavrila, D., Görzig, S., Lindner, F., Paetzold, F., and Wöhler, C., "Autonomous driving goes downtown," *IEEE Intelligent systems* (6), 40–48 (1998).
- [12] Sotelo, M. A., Rodriguez, F. J., Magdalena, L., Bergasa, L. M., and Boquete, L., "A color vision-based lane tracking system for autonomous driving on unmarked roads," *Autonomous Robots* **16**(1), 95–116 (2004).
- [13] Lombardi, P., Zanin, M., and Messelodi, S., "Switching models for vision-based on-board road detection," in [*IEEE Conference on Intelligent Transportation Systems*], 67–72 (2005).
- [14] Sparbert, J., Dietmayer, K., and Streller, D., "Lane detection and street type classification using laser range images," in [*ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 01TH8585)*], 454–459, IEEE (2001).
- [15] Ma, B., Lakshmanan, S., and Hero, A. O., "Simultaneous detection of lane and pavement boundaries using model-based multisensor fusion," *IEEE Transactions on Intelligent Transportation Systems* **1**(3), 135–147 (2000).
- [16] Yu, B. and Jain, A. K., "Lane boundary detection using a multiresolution hough transform," in [*Proceedings of International Conference on Image Processing*], **2**, 748–751, IEEE (1997).
- [17] Badino, H., Franke, U., and Mester, R., "Free space computation using stochastic occupancy grids and dynamic programming," in [*Workshop on Dynamical Vision, ICCV, Rio de Janeiro, Brazil*], **20**, Citeseer (2007).
- [18] Badino, H., Franke, U., and Pfeiffer, D., "The stixel world-a compact medium level representation of the 3d-world," in [*Joint Pattern Recognition Symposium*], 51–60, Springer (2009).
- [19] Alvarez, J. M., Gevers, T., LeCun, Y., and Lopez, A. M., "Road scene segmentation from a single image," in [*European Conference on Computer Vision*], 376–389, Springer (2012).
- [20] Beucher, S., Bilodeau, M., and Yu, X., "Road segmentation by watershed algorithms," in [*PROMETHEUS Workshop, Sophia Antipolis, France*], (1990).
- [21] Ladický, L., Sturgess, P., Russell, C., Sengupta, S., Bastanlar, Y., Clocksin, W., and Torr, P. H., "Joint optimization for object class segmentation and dense stereo reconstruction," *International Journal of Computer Vision* **100**(2), 122–133 (2012).
- [22] Sturgess, P., Alahari, K., Ladický, L., and Torr, P. H., "Combining appearance and structure from motion features for road scene understanding," in [*BMVC-British Machine Vision Conference*], BMVA (2009).
- [23] Passani, M., Yebes, J. J., and Bergasa, L. M., "Crf-based semantic labeling in miniaturized road scenes," in [*17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*], 1902–1903, IEEE (2014).
- [24] Vitor, G. B., Victorino, A. C., and Ferreira, J. V., "A probabilistic distribution approach for the classification of urban roads in complex environments," in [*IEEE Workshop on International Conference on Robotics and Automation*], (2014).
- [25] Brust, C.-A., Sickert, S., Simon, M., Rodner, E., and Denzler, J., "Convolutional patch networks with spatial prior for road detection and urban scene understanding," *arXiv preprint arXiv:1502.06344* (2015).
- [26] Mohan, R., "Deep deconvolutional networks for scene parsing," *arXiv preprint arXiv:1411.4101* (2014).
- [27] Oliveira, G. L., Burgard, W., and Brox, T., "Efficient deep models for monocular road segmentation," in [*2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*], 4885–4891, IEEE (2016).
- [28] Shelhamer, E., Long, J., and Darrell, T., "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.* **39**, 640–651 (Apr. 2017).

- [29] Wang, C., Zhang, H., Yang, M., Wang, X., Ye, L., and Guo, C., “Automatic parking based on a bird’s eye view vision system,” *Advances in Mechanical Engineering* **6**, 847406 (2014).
- [30] Liu, Y.-C., Lin, K.-Y., and Chen, Y.-S., “Bird’s-eye view vision system for vehicle surrounding monitoring,” in [*International Workshop on Robot Vision*], 207–218, Springer (2008).
- [31] Haltakov, V., Belzner, H., and Ilic, S., “Scene understanding from a moving camera for object detection and free space estimation,” in [*2012 IEEE Intelligent Vehicles Symposium*], 105–110, IEEE (2012).
- [32] Fremont, V., Bui, M., Boukerroui, D., and Letort, P., “Vision-based people detection system for heavy machine applications,” *Sensors* **16**(1), 128 (2016).
- [33] Deng, L., Yang, M., Qian, Y., Wang, C., and Wang, B., “Cnn based semantic segmentation for urban traffic scenes using fisheye camera,” in [*2017 IEEE Intelligent Vehicles Symposium (IV)*], 231–236, IEEE (2017).
- [34] Deng, L., Yang, M., Li, H., Li, T., Hu, B., and Wang, C., “Restricted deformable convolution based road scene semantic segmentation using surround view cameras,” *arXiv preprint arXiv:1801.00708* (2018).
- [35] Sáez, Á., Bergasa, L. M., López-Guillén, E., Romera, E., Tradacete, M., Gómez-Huélamo, C., and del Egidio, J., “Real-time semantic segmentation for fisheye urban driving images based on erfnet,” *Sensors* **19**(3), 503 (2019).
- [36] Qian, Y., Yang, M., Wang, C., and Wang, B., “Pedestrian feature generation in fish-eye images via adversary,” in [*2018 IEEE International Conference on Robotics and Automation (ICRA)*], 2007–2012, IEEE (2018).
- [37] Arjovsky, M., Chintala, S., and Bottou, L., “Wasserstein generative adversarial networks,” in [*International Conference on Machine Learning*], 214–223 (2017).
- [38] Cordts, M., Omran, M., Ramos, S., Scharwächter, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B., “The cityscapes dataset,” in [*CVPR Workshop on the Future of Datasets in Vision*], **2** (2015).
- [39] Ros, G., Sellart, L., Materzynska, J., Vazquez, D., and Lopez, A. M., “The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes,” in [*Proceedings of the IEEE conference on computer vision and pattern recognition*], 3234–3243 (2016).
- [40] Kingslake, R., [*A history of the photographic lens*], Elsevier (1989).
- [41] Tardif, J.-P., Sturm, P., and Roy, S., “Self-calibration of a general radially symmetric distortion model,” in [*European conference on computer vision*], 186–199, Springer (2006).
- [42] Kannala, J. and Brandt, S. S., “A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses,” *IEEE transactions on pattern analysis and machine intelligence* **28**(8), 1335–1340 (2006).
- [43] Sáez, A., Bergasa, L. M., Romeral, E., López, E., Barea, R., and Sanz, R., “Cnn-based fisheye image real-time semantic segmentation,” in [*2018 IEEE Intelligent Vehicles Symposium (IV)*], 1039–1044, IEEE (2018).
- [44] Shah, S., Dey, D., Lovett, C., and Kapoor, A., “Airsim: High-fidelity visual and physical simulation for autonomous vehicles,” in [*Field and service robotics*], 621–635, Springer (2018).
- [45] Bourke, P., “idome: Immersive gaming with the unity3d game engine,” *CGAT09 Computer Games, Multimedia and Allied Technology* **9**, 265–272 (2009).
- [46] “Unity-technologies / ml-imagesynthesis.” (Accessed on 06/22/2019).
- [47] Badrinarayanan, V., Kendall, A., and Cipolla, R., “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE transactions on pattern analysis and machine intelligence* **39**(12), 2481–2495 (2017).
- [48] Iqbal, H., “Harisqbal88/plotneuralnet v1.0.0,” (Dec. 2018).
- [49] Chen, L., Papandreou, G., Schroff, F., and Adam, H., “Rethinking atrous convolution for semantic image segmentation,” *CoRR* **abs/1706.05587** (2017).

- [50] He, K., Zhang, X., Ren, S., and Sun, J., “Identity mappings in deep residual networks,” in [*European conference on computer vision*], 630–645, Springer (2016).
- [51] Yu, F. and Koltun, V., “Multi-scale context aggregation by dilated convolutions,” *arXiv preprint arXiv:1511.07122* (2015).
- [52] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L., “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” *IEEE transactions on pattern analysis and machine intelligence* **40**(4), 834–848 (2017).
- [53] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L., “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision (IJCV)* **115**(3), 211–252 (2015).
- [54] Glorot, X. and Bengio, Y., “Understanding the difficulty of training deep feedforward neural networks,” in [*Proceedings of the thirteenth international conference on artificial intelligence and statistics*], 249–256 (2010).
- [55] Loshchilov, I. and Hutter, F., “Sgdr: Stochastic gradient descent with warm restarts,” *arXiv preprint arXiv:1608.03983* (2016).
- [56] Takahashi, R., Matsubara, T., and Uehara, K., “Data augmentation using random image cropping and patching for deep cnns,” *arXiv preprint arXiv:1811.09030* (2018).
- [57] Krizhevsky, A., Sutskever, I., and Hinton, G. E., “Imagenet classification with deep convolutional neural networks,” in [*Advances in neural information processing systems*], 1097–1105 (2012).
- [58] Eichenseer, A. and Kaup, A., “A data set providing synthetic and real-world fisheye video sequences,” in [*2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*], 1541–1545, IEEE (2016).