Synthesising large, low cost and diverse datasets for robust semantic segmentation in self-driving tasks

Pau Dietz Romero[1, 2], Merlin David Mengel[1], and Jakob Czekansky[1]

[1]University of Applied Sciences Mittelhessen, Gießen, Germany
[2]University of Siegen, Germany

May 1, 2022

Abstract: Robust scene understanding algorithms are crucial for the success of autonomous navigation. The supervised learning of semantic segmentation unfortunately requires large and diverse datasets. For some self-driving tasks, such as navigating a robot within an industrial facility, freely available datasets are not available, and manual annotation of large datasets is impractical for smaller development teams. While there are approaches to automatically generate synthetic data, they can be computationally expensive, require significant preparation effort, or miss a wide variety of features. This paper presents a new framework for generating synthetic datasets with high variance for low computing demands that can be easily adapted to different self-driving tasks The details of the framework can be found at https://github.com/cITIcar/SAD-Generator. As a demonstration, this approach was applied to a semantic segmentation task on a miniature road with random obstacles, lane markings, and disturbing artifacts. A U-Net was trained using synthesized data and later fine-tuned with a small amount of manually annotated data. This resulted in an improvement of 2.5 percentage points in pixel accuracy (PA) and 11.19 percentage points in mean intersection over union (mIoU).

1 Introduction

We put forward the hypothesis that the overall acceptance of autonomous navigation is closely tied to the performance of scene understanding in extreme and unforeseen circumstances. Understanding the surroundings helps the system to avoid obstacles, locate the drive path and react to other entities. This task can be accomplished through semantic segmentation, which assigns each pixel of an image to a specific semantic category. Semantic segmentation has already demonstrated impressive results in the past [1, 2, 3, 4]. Pixel-wise predictions can be achieved using a Neural Network architecture called Fully Convolutional Network (FCN) firstly mentioned by Long et al. [5].

A robust algorithm keeps a high accuracy even in situations with bad lighting, disturbing artefacts or obstacles. The training of supervised Neural Networks requires huge datasets, representing the variety of situations in inference. Annotating such a large quantity of data is time consuming and expensive, representing a bottleneck for small development projects with tight time and financial resources. Too small datasets result in deteriorated performance due to overfitting of the Neural Network [7].

To match the need for data to train semantic segmentation algorithms many large automotive datasets have been published like the Cityscapes [8], the KITTI [11] or the Toronto dataset [12]. For navigation tasks that do not occur in urban streets, but for instance in industrial environments, these available datasets are not applicable.

Another approach is generating synthetic data [9]. The idea is to create artificial images that resemble the real world. The program generates the images for a given ground-truth, so annotation is simple, the computational complexity is domi-

nated by producing a realistic image. Using synthetic data, even rare and thus under-represented situations may occur arbitrary often.

In the past many automotive synthetic datasets based on game engines and graphic simulators have been created, such as the Synthia [9], the Virtual KITTI 2 [13] and the GTA [10] dataset. Unfortunately they are computational expensive to generate and mostly demand artistic modelling of a detailed environment [7]. The high expense necessary to generate one photo-realistic annotated sample contradicts the primary selling point of synthetic data, that labelled data is available for nearly free.

To counter the high expense of synthetic data Tobin et al. [14] introduced the method of domain randomization (DR). This approach creates a non photo-realistic environment with many randomly generated features to force the network to learn the relevant features of an image.

Tremblay et al. [7] extended DR to object detection and generated low cost synthetic samples with great variance. Figure 1 shows an example of a generated image by DR. Their images have been created in the following work flow:

- A random number of objects of interest are placed in a random orientation inside a 3D scene.
- A random number of distracting objects are placed in a random orientation inside a 3D scene.
- Random colors and textures are applied to the objects.
- A random background is added to the scene.
- Random light is projected on to the scene.
- A random perspective of the scene is chosen.

They compared the effectiveness of their dataset with the photo-realistic, synthetic dataset virtual KITTI [13]. The original KITTI dataset [11] was used as test dataset. Deep learning networks performed better when trained on their dataset than when trained on virtual KITTI. Furthermore, they demonstrated that training their dataset plus fine tuning the network with real world data outperformed networks only training with real world data.

Their approach is highly scalable and efficient when the objects of interest are in the foreground of the image rather than the background. In their work the ground plane is not annotated with a relevant object class. The images inserted on their ground plane were collected manually. But for the task of semantic segmentation in autonomous navigation the ground plane plays a major role. The relevant visual information are not only the objects in the foreground (such as obstacles) but also features on the ground plane (for instance lane markings).

Figure 1: A generated image with Domain Randomization [7]. In their work they trained object detection for a real world urban environment

Our work enhances the approach of DR to automate the creation of random ground planes that are accurately labelled. We present a framework that demands less design effort adapting it to different self-driving tasks on flat surfaces with visible lane markings. Our approach is highly scalable, runs cost effective on a standard CPU and creates a great variance of random features. It is an ideal solution for small teams developing autonomous robots. The ground plane is automatically generated by placing pre-annotated tiles in a useful manner. After the scene is built the plane gets post processed by adding disturbing artefacts like reflections, dust and textures.

Our framework is not meant to challenge synthetic datasets for urban streets like virtual KITTI [11] or SYNTHIA [9] that take place in complex and non flat environments. It is especially designed for simpler navigation tasks on flat surfaces, such as transport robots in a industrial facility. Our work lowers the hurdle for small development teams to tackle the need for large datasets.

2 Framework for Synthetic Data

The essential idea of this framework is, that many navigation tasks happen in closed and and controlled environments on a flat ground. In these scenarios, the crucial information for the self-driving entity lies in lane markings and the location of obstacles, such as other robots. The framework

generates annotated data with great variety for low computational cost. However, manual image annotation is still required for fine-tuning Neural Networks.

The framework simplifies the environment of the vehicle to following five basic elements (also called chunks): straight lines, 90° left turns, 90° right turns, intersections and an empty chunk. The user of the framework adds as many images to a chunk as appropriate. A chunk can have multiple images, but only has one annotation. Each image must be in bird's eye view and have a consistent scale. Figure 2 shows an example of a chunk image with its annotation. Adding different images to a chunk increases the variety of the data.

Figure 2: An image with its annotation representing a intersection chunk. The pixels of the image are assigned to a semantic class by their color (class 'left lane': yellow, class 'right lane': green, class 'obstacle': red, class 'intersection': orange).

The program picks random chunks and assembles them successively to create a continuous road with a defined length. Every chunk is put in an orientation and location that is compatible with its predecessor. All the space not covered by the road is filled with empty chunks to gain a resulting image in rectangular shape. This process is done simultaneously for the image and annotation. Figure 3 shows a possible placement of chunks. The result of this operation is an artificial image of the ground plane and its exact annotation. Such a ground plane is depicted in figure 4. The course of the road is highly flexible. For every new chunk the program can chose one of four different chunks. The amount of possible combinations that can be achieved in the best case by a road with a length of n chunks is $p_n = 4^n$. The program prohibits the chunks to overlap (the snake biting in its own tail), so the actual number is slightly lower than 4^n. Consider following examples: $p_5 = 1024, p_7 = 16384$

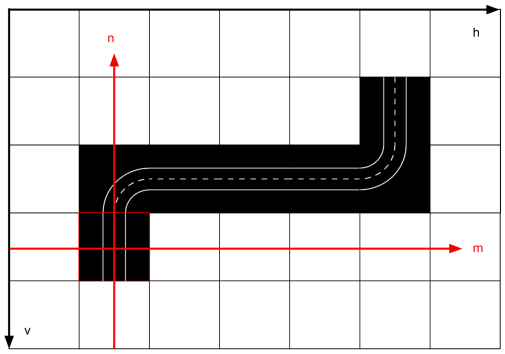

Figure 3: The framework arranges the chunks to a road like in a board game. The mn- coordinate system defines the position and orientation of a virtual car (red). The hv- coordinate system represents the image (black)

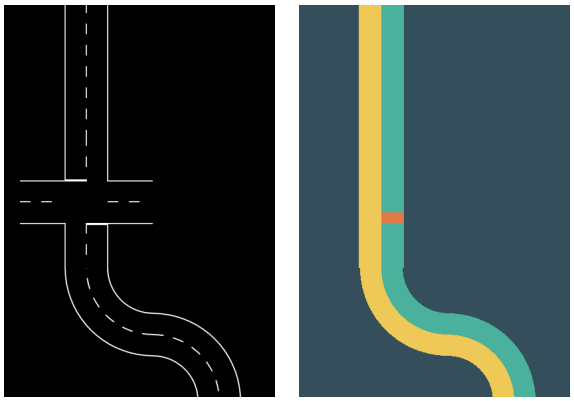

Figure 4: Example of an image and its annotation showing the synthetic generated ground plane in bird's-eye-view.

In the following step the artificial image gets post-processed to randomly alter its appearance. Following post-processing steps are made:

- Random particles are added to the image to represent dust or dirt on the road.
- Random images are placed on the ground plane to simulate larger interfering artifacts such as reflections. The program picks the artifacts from a source folder to which the user can add arbitrary images.
- Random contrast and brightness is applied to the image.
- Random areas of the lane markings are covered by black patches to hide visual information.
- Random objects of other classes are randomly inserted on the image and annotation.

In the next step the program transforms the images from bird's-eye-view into camera perspective. Figure 5 shows an image in camera perspective with its corresponding label. The data generator moves a fictitious car along the road and takes pictures from its position and orientation in camera perspective.

The chunks representing the road are structured in a FIFO list. As soon as the car leaves its initial chunk so that it is no longer visible to the camera, it is removed from the ground plane. Simultaneously a new chunk gets randomly added to the the end of the road. The program adds optical artifacts such as Gaussian blur to the camera images. The application runs with around 25 FPS on our Notebook CPU (Intel Core i7-8750H @ 2.2GHz).

Figure 5: A synthetic image with its annotation in camera perspective

3 Evaluation

The benefit of this framework for computer vision performance was tested on a racetrack similar to the one used in a miniature self-driving car competition (www.tu-braunschweig.de/carolocup). The car has to segment the drive lane, obstacles and intersections even when artefacts are disturbing their visibility. A training, validation and test dataset consisting of 350, 75 and 75 samples, respectively, were manually collected and annotated. A synthetic training and validation dataset consisting of 30000 and 1000 samples, respectively, were generated. The network chosen for evaluating the datasets is the U-Net introduced by Ronneberger et al. [6]. During pretraining a larger learning rate (lr = 0.001) was chosen. For the fine tuning the learning rate was reduced (lr = 0.0001). The Neural Network was trained on different subsets of the training and validation datasets and later evaluated on the test dataset with the metrics mean intersection over union (mIoU) and pixel accuracy (PA). The following tables show the resulting performance of those training configurations.

Pretraining the network with a synthetic training dataset and a mixed validation dataset yielded the best performance. A mixed dataset contains real and synthetic data. Figures 7 and 9 compare predictions for training only on real data and training on mixed data.

Data Configuration	Training Data	Validation Data	PA [%]	mIoU [%]
Pre-Training:	real	real	0.9660	0.7113
Fine Tuning:	none	none		

Table 1: Results of training configuration real-real-none-none

Data Configuration	Training Data	Validation Data	PA [%]	mIoU [%]
Pre-Training:	synthetic	synthetic	0.6425	0.3845
Fine Tuning:	real	real	0.9526	0.6051

Table 2: Results of training configuration synthetic-synthetic-real-real

Data Configuration	Training Data	Validation Data	PA [%]	mIoU [%]
Pre-Training:	mixed	mixed	0.9644	0.7015
Fine Tuning:	mixed	real	0.9670	0.7141

Table 3: Results of training configuration mixed-mixed-mixed-real

Data Configuration	Training Data	Validation Data	PA [%]	mIoU [%]
Pre-Training:	synthetic	mixed	0.9389	0.5369
Fine Tuning:	synthetic	real	0.9915	0.8232

Table 4: Results of training configuration synthetic-mixed-synthetic-real

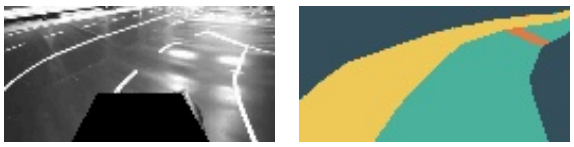

Figure 6: Example of a real world image (left) with its manually annotated label (right) from the test dataset.

Figure 7: Two different predictions from the image in figure 6: The image on the left is a segmented output of a network trained only on real world data. On the right is the output of a network trained with synthetic and real data. This example demonstrated how synthetic data improve the classification of the under-represented class 'intersection'.

Figure 8: Example of a real world image (left) with its manually annotated label (right) from the test dataset.

Figure 9: Two different predictions from the image in figure 8: The image on the left is a segmented output of a network trained only on real world data. On the right is the output of a network trained with synthetic and real data. This example demonstrated how synthetic data improve the classification of the under-represented class 'obstacle'.

4 Conclusion

Our framework proves to be an effective tool for small teams that develop self-driving robots in flat environments with visible lane markings. It is computational efficient and executes well on a regular notebook CPU. With minor effort, the framework can be adapted to a new self-driving task. When training a network on real world and synthetic data from our framework it outperforms networks being trained on real world data only. In particular the network improved the classification of under-represented classes when learning with synthetic data. Thus, our framework is a powerful tool especially in the early stages of small development projects. A further direction that should be examined is the training of recurrent FCNs. The program already generates temporally coherent road images, which can be used to learn temporal features in self-driving tasks.

References

[1] Li Wang and Xingxing Chen and Liangyuan Hu, Hui Li: *Overview of Image Semantic Segmentation Technology* (2019) 16th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)

[2] Rachmadio Noval Lazuardi and Daffa Sudrajat and Nafitri Aulia and Trio Adiono: *A System of Semantic Segmentation on An Autonomous Vehicle* (2019), 16th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)

[3] Alberto Garcia-Garcia and Sergio Orts-Escolano and Sergiu Oprea and Victor Villena-Martinez and Jose Garcia-Rodriguez: *A Review on Deep Learning Techniques Applied to Semantic Segmentation* (2017), arXiv:1704.06857 [cs.CV]

[4] Mennatullah Siam and Sara Elkerdawy and Martin Jagersand and Senthil Yogamani: *Deep Semantic Segmentation for Automated Driving: Taxonomy, Roadmap and Challenges* (2017), arXiv:1707.02432v2 [cs.CV]

[5] Jonathan Long and Evan Shelhamer and Trevor Darrell: *Fully Convolutional Networks for Semantic Segmentation* (2015) Proceedings of the IEEE conference on computer vision and pattern recognition 2015: 3431-3400

[6] Olaf Ronneberger and Philipp Fischer and Thomas Brox: *U-Net: Convolutional Networks for Biomedical Image Segmentation* (2015), arXiv:1505.04597v1 [cs.CV]

[7] Jonathan Tremblay and Aayush Prakash and David Acuna and Mark Brophy and Varun Jampani and Cem Anil and Thang To and Eric Cameracci and Shaad Boochoon and Stan Birchfield: *Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization* (2018), arXiv:1804.06516v3 [cs.CV]

[8] Marius Cordts and Mohamed Omran and Sebastian Ramos and Timo Rehfeld and Markus Enzweiler and Rodrigo Benenson and Uwe Franke and Stefan Roth and Bernt Schiele: *The Cityscapes Dataset for Semantic Urban Scene Understanding* (2016), Konferenz IEEE Computer Vision and Pattern Recognition (CVPR)

[9] German Ros and Laura Sellart and Joanna Materzynska and David Vazquez and Antonio Lopez: *The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes* (2016), Konferenz IEEE Computer Vision and Pattern Recognition (CVPR)

[10] Stephan R. Richter and Vibhav Vineet and Stefan Roth and Vladlen Koltun: *Playing for Data: Ground Truth from Computer Games* (2016), arXiv:1608.02192v1

[11] Andreas Geiger and Philip Lenz and Christoph Stiller and Raquel Urtasun: *Vision meets Robotics: The KITTI Dataset* International Journal of Robotics Research (IJRR) 32 (2013), pp. 1229-1235

[12] S. Wang, M. Bai, G. Mattyus, H. Chu, W. Luo, B. Yang, J. Liang, J. Cheverie, S. Fidler, and R. Urtasun: *orontocity: Seeing the world with a million eyes* arXiv preprint arXiv:1612.00423, 2016

[13] Yohann Cabon, Naila Murray, Martin Humenberger: *Virtual KITTI 2* arXiv:2001.10773, 2020

[14] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel.: *Domain randomization for transferring deep neural networks from simulation to the real world.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017. 1, 2