Procedural Modeling and Physically Based Rendering for Synthetic Data Generation in Automotive Applications

Apostolia Tsirikoglou[1,*] Joel Kronander[1] Magnus Wrenninge[2,†] Jonas Unger[1,‡]
[1]Linköping University, Sweden [2]7D Labs

Figure 1: Example images produced using our method for synthetic data generation.

Abstract

We present an overview and evaluation of a new, systematic approach for generation of highly realistic, annotated synthetic data for training of deep neural networks in computer vision tasks. The main contribution is a procedural world modeling approach enabling high variability coupled with physically accurate image synthesis, and is a departure from the hand-modeled virtual worlds and approximate image synthesis methods used in real-time applications. The benefits of our approach include flexible, physically accurate and scalable image synthesis, implicit wide coverage of classes and features, and complete data introspection for annotations, which all contribute to quality and cost efficiency. To evaluate our approach and the efficacy of the resulting data, we use semantic segmentation for autonomous vehicles and robotic navigation as the main application, and we train multiple deep learning architectures using synthetic data with and without fine tuning on organic (i.e. real-world) data. The evaluation shows that our approach improves the neural network's performance and that even modest implementation efforts produce state-of-the-art results.

*apostolia.tsirikoglou@liu.se
†magnus@7dlabs.com
‡jonas.unger@liu.se

1. Introduction

Semantic segmentation is one of the most important methods for visual scene understanding, and constitutes one of the key challenges in a range of important applications such as autonomous driving, active safety systems and robot navigation. Recently, it has been shown that solutions based on deep neural networks [18, 32, 19] can solve this kind of computer vision task with high accuracy and performance. Although deep neural networks in many cases have proven to outperform traditional algorithms, their performance is limited by the training data used in the learning process. In this context, data itself has proven to be both the constraining and the driving factor of effective semantic scene understanding and object detection [29, 32].

Access to large amounts of high quality data has the potential to accelerate the development of both new deep learning algorithms as well as tools for analyzing their convergence, error bounds, and performance. This has spurred the development of methods for producing synthetic, computer generated images with corresponding pixel-accurate annotations and labels. To date, the most widely used synthetic datasets for urban scene understanding are SYNTHIA [27] and the dataset presented by Richter et al. [26]. Both datasets use hand-modeled game worlds and rasterization-based image synthesis. It is worth noting that none of these previous studies have considered, in-depth, the way

in which the virtual world itself is generated. Instead, focus has been put on the capture of existing 3D worlds and analyzing the performance of the resulting data.

In recent years, game engines have steadily improved their ability to render realistic images. One recent example is the short film Adam[1], which was rendered in the Unity engine. However, it does not take long for a trained eye to spot the inconsistencies in these types of real-time renderings. In contrast, much of current visual effects in film feature imagery that even professionals cannot tell apart from reality. So far, current studies using synthetic data all involve mixing or fine tuning with organic datasets in order to achieve useful results. The *domain shift* of the data is both discussed in these studies and obvious when viewing the images. Given that deep neural networks are reaching and sometimes exceeding human-level perception in computer vision tasks, it follows that synthetic data eventually needs to be as realistic as real data, if it is to become a useful complement, or at some point, a substitute. To that end, we avoid the use of the term "photo-realistic" throughout this paper; although our method is grounded in physically based image synthesis methods that can enable extremely realistic results, and although we achieve state-of-the-art results, neither ours nor the compared datasets can currently claim to be photo-realistic. Instead, we aim to make the reader conscious of the need to thoroughly analyze and evaluate realism in synthetic datasets.

In this paper we use state-of-the-art computer graphics techniques, involving detailed geometry, physically based material representations, Monte Carlo-based light transport simulation as well as simulation of optics and sensors in order to produce realistic images with pixel-perfect ground truth annotations and labels, see Figures 1 and 2. Our method combines procedural, automatic world generation with accurate light transport simulation and scalable, cloud-based computation capable of producing hundreds of thousands or millions of images with known class distributions and a rich set of annotation types. Compared to game engine pipelines, our method uses principles from the visual effects and film industries, where large scale production of images is well established and realism is paramount.

Whereas image creation in film generally aims to produce a sequence of related images (i.e. an animation), we note that synthetic datasets instead benefit from images that are as diverse as possible. To that end, our system procedurally generates an entirely unique world for each output image from a set of classes representing vehicles, buildings, pedestrians, road surfaces, vegetation, trees and other relevant factors. All aspects of the individual classes, such as geometry, materials, color, and placement are parameterized, and a synthesized image and its corresponding annotations constitute a sampling of that parameter space. In the following sections, we demonstrate that this approach outperforms existing synthetic datasets in the semantic segmentation problem on multiple state-of-the art deep neural network architectures.

[1] https://unity3d.com/pages/adam

2. Background and Related work

The most common approach for producing training data with ground truth annotations has been to employ hand labeling, e.g. CamVid [3], Cityscapes [5], the KITTI dataset [9], or the Mapillary Vistas Dataset[2], which is both time consuming and complex to orchestrate. In Brostow et al. [3] and Cordts et al. [5] it is reported that a single image may take from 20 to 90 minutes to annotate. Another problem inherent to manual annotation is that some objects and features are difficult to classify and annotate correctly, especially when the image quality and lighting conditions vary. In the Cityscapes dataset, partially occluded objects, such as a pedestrian behind a car, are sometimes left unclassified, and the annotation of edges of dynamic objects, vegetation and foliage are often cursory.

Although several real-world, organic data sets are available, there has been a need to address the issue of data set bias [34, 16], by going beyond the thousands of hand labeled images they consist of, and in a controlled way ensuring a wider and generalizable coverage of features and classes within each training image. As analyzed in detail by Johnson et al. [13], game-based image synthesis has matured enough that the performance of deep learning architectures can be improved using computer generated footage with pixel accurate annotations. Synthetic data has been successfully used in a range of application domains including prediction of object pose [33, 21, 11], optical flow [6], semantic segmentation for indoor scenes [12, 36], and analysis of image features [2, 15].

Previous methods for data generation in automotive applications have largely used computer game engines. Richter et al. [26] and Johnson et al. [13] used the Grand Theft Auto (GTA) engine from Rockstar Games, and Ros et al. [27] and Gaidon et al. [8] used the Unity development platform[3] for the SYNTHIA and Virtual KITTI data set respectively. Other examples of using video game engines are presented by Shafaei et al. [30] who are using an undisclosed game engine, and Qui et al. [25] who recently introduced the UnrealCV plugin, which enables generation of image- and accompanying ground truth annotations using Unreal Engine[4] from Epic Games. Although it provides relatively easy access to virtual worlds, the game engine approach to synthetic data generation is limited in several ways. First, pre-computations with significant approximations of the light transport in the scene are required in order

[2] www.mapillary.com
[3] www.unity3d.com
[4] www.unrealengine.com

Figure 2: **Left:** Three example images generated using our approach. **Middle:** The corresponding per-pixel class segmentation. **Right:** The instance segmentation. The procedural world modeling approach does not generate images from a fixed world, but rather samples from combinations of the input class instances described by the experimental design and scope.

to fit the world model onto the GPU and enable interactive rendering speeds. Consequently, these methods do not scale well when the ratio of final images to number of scenarios (i.e. game levels) is low. Secondly, even though the 3D world in many cases may be large, it is hand modeled and of finite size. This not only means that it is time consuming and costly to build, but also that the coverage of classes and features is limited, and that dataset bias is inevitable. This is obvious if we consider the limit case: the more images we produce, the more similar each image becomes to the others in the dataset. Conversely, if a large number of scenarios were built, the cost of pre-computing light transport within each scene would not be amortized over a large enough set of images to be efficient. Finally, some important aspects such as accurate simulation of sensor characteristics, optics and surface and volumetric scattering (and even some annotations) may be impractical to include in rasterization-based image synthesis.

Another approach for increasing variability and coverage in organic, annotated footage is to augment the images with synthetic models. Rozantsev et al. [28] proposed a technique where 3D models were superimposed onto real backgrounds through simulation of the camera system and lighting conditions in the real scene for improving the detection of aerial drones and aircrafts. Other interesting examples of data augmentation using image synthesis include pedestrian detection presented in Marin et al. [20], the GAN-based gaze and hand pose detection described by Shrivastava et al. [31], and rendering of 3D car models into background footage for segmentation tasks described by Alhaija et al. [1]. Although a smaller set of hand-annotated images can be augmented to include a wider variation, this approach is best suited for coarse annotations and does not generalize well to large image volumes and semantic segmentation tasks.

In contrast to previous methods for generation of annotated synthetic training data, we employ procedural world modeling (for an overview, see Ebert et al. [7]). The benefit is that the creation of the 3D world can be parameterized to ensure detailed control over class and feature variations. In our system, the user input is no longer a large, concrete 3D world, but rather a composition of classes and a scenario scope, from which the system creates only what is visible from each image's camera view. For image synthesis, we use physically based rendering techniques based on path tracing and Monte Carlo integration [14]. This allows accurate simulation of sensors, optics, and the interaction between the light, materials and geometry in the scene. The next section gives an overview of our system and the underlying techniques used to generate synthetic training data.

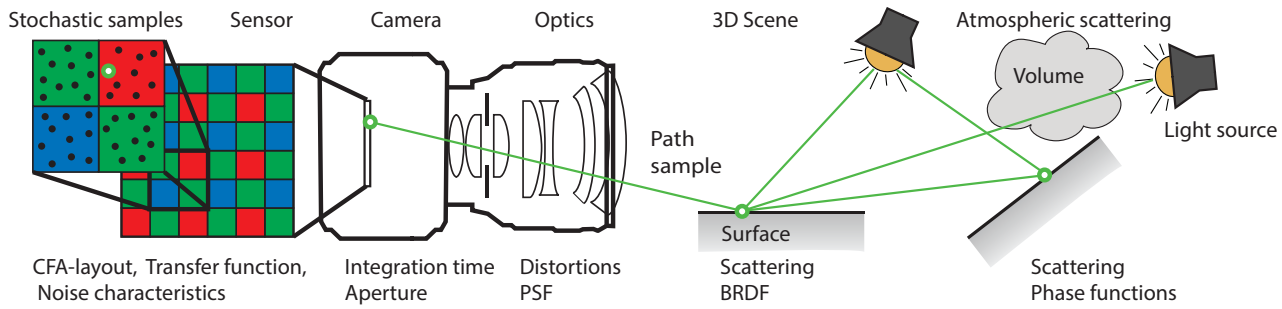

Figure 3: Illustration of the principles of path tracing. Sample paths are stochastically generated in the image plane and traced through the scene. At each interaction the light emitting objects are sampled and their contribution summed up. The technique enables accurate simulation of sensor characteristics and the color filter array (CFA), the effect of the optical system (PSF, distortion, etc.), complex geometries and scattering at surface and in participating media. Path tracing is computationally expensive, but parallelizes and scales well, and is made feasible through Monte Carlo importance sampling techniques.

3. Method

Image synthesis requires a well-defined model of the 3D virtual scene. The model contains the geometric description of objects in the scene, a set of materials describing the appearance of the objects, specifications of the light sources in the scene, and a virtual camera model. The geometry is often specified in terms of discretized rendering primitives such as triangles and surface patches. The materials define how light interacts with surfaces and participating media. Finally, the scene is illuminated using one or several light sources, and the composition of the rendered frame is defined by introducing a virtual camera.

One way to consider realism in synthetic imagery is as a multiplicative quantity. That is, in order to produce realistic synthetic images we must ensure that each of steps involved are capable of producing realistic results. For example, there is little hope of achieving realistic imagery even when employing physically accurate light transport if the geometry is of poor quality, and vice versa. To this end, we identify five orthogonal aspects of realism that are key to achieving the goal:

1. Overall scene composition
2. Geometric structure
3. Illumination by light sources
4. Material properties
5. Optical effects

Our method addresses each of these aspects. Realism in the overall scene composition along with the geometric structure and the material properties is addressed by the rule set used in the procedural modeling process. Realism in the illumination and material interaction is addressed by physically based light transport simulation and optical effects are modeled using point spread functions in the image domain.

3.1. Physically based light transport simulation

The transfer of light from light sources to the camera, via surfaces and participating media in the scene, is described by light transport theory, which is a form of radiative transfer [4]. For surfaces, the light transport is often described using the macroscopic geometric-optics model defined by the rendering equation [14], expressing the outgoing radiance $L(\vec{x} \rightarrow \omega_o)$ from a point $\vec{x}$ in direction ω_o as

$$L(\vec{x} \rightarrow \omega_o) = L_e(\vec{x} \rightarrow \omega_o) + \underbrace{\int_\Omega L(\vec{x} \leftarrow \omega_i)\rho(\vec{x}, \omega_i, \omega_o)(\vec{n} \cdot \omega_o)d\omega_i}_{L_r(\vec{x} \rightarrow \omega_o)}, \quad (1)$$

where $L(\vec{x} \leftarrow \omega_i)$ is the incident radiance arriving at the point $\vec{x}$ from direction ω_i, $L_e(\vec{x} \rightarrow \omega_o)$ is the radiance emitted from the surface, $L_r(\vec{x} \rightarrow \omega_o)$ is the reflected radiance, $\rho(\vec{x}, \omega_i, \omega_o)$ is the *bidirectional reflectance distribution function* (BRDF) describing the reflectance between incident and outgoing directions [22], Ω is the visible hemisphere, and $\vec{n}$ is the surface normal at point $\vec{x}$.

Rendering is carried out by simulating the light reaching the camera. This requires solving the rendering equation for a large number of sample points in the image plane which is a set of potentially millions of interdependent, high-dimensional analytically intractable integral equations. Solving the rendering equation is challenging since the radiance L appears both inside the integral expression and as the quantity we are solving for. The reason for this is that the outgoing radiance at any one point affects the incident radiance at all other points in the scene. This results in a very large system of nested integrals. Formally, the rendering equation is a Fredholm integral equation of the second kind, for which analytic solutions are impos-

sible to find in all but the most trivial cases. In practice, the rendering problem can be solved in a number of ways with different light transport modeling techniques, the most common of which can be divided into two main categories: *rasterization* which is the method generally used in GPU rendering, and *path tracing* in which equation 1 is solved using Monte Carlo integration techniques which stochastically construct paths that connect light sources to the virtual camera and compute the path energy throughput as illustrated in Figure 3.

The rendering system used in this paper relies on path tracing as it is the most general rendering algorithm. In the limit, path tracing can simulate any type of lighting effects including multiple light bounces and combinations between complex geometries and material scattering behaviors. Another benefit is that it is possible to sample the scene being rendered over both the spatial and temporal dimensions. For example, by generating several path samples per pixel in the virtual film plane it is possible to simulate the area sampling over the extent of each pixel on a real camera sensor, which in practice leads to efficient anti-aliasing in the image and enables simulation of the point spread function (PSF) introduced by the optical system. By distributing the path samples over time by transforming (e.g. animating) the virtual camera and/or objects in the scene, it is straightforward to accurately simulate motion blur, which is a highly important aspect in the simulation of the computer vision system on a vehicle. Path tracing is a standard tool in film production and is implemented in many rendering engines. For an in-depth introduction to path tracing and the plethora of techniques for reducing the computational complexity such as Monte Carlo importance sampling and efficient data structures for accelerating the geometric computations, we refer the reader to the textbook by Pharr et al. [23].

3.2. World generation using procedural modeling

An important notion in machine learning in general, and in deep learning in particular, is that of *factors of variation*. In their textbook, Goodfellow et al. [10] state in their introduction that "Such factors are often not quantities that are directly observed. Instead, they may exist as either unobserved objects or unobserved forces in the physical world that affect observable quantities." (pp. 4-5). This notion is directly parallel to a longstanding methodology often employed in art and film production, namely procedural modeling. In designing our method for creating virtual worlds, we consider the procedural modeling aspect to be a direct path towards producing datasets with precisely controlled factors of variation.

Previous methods for producing synthetic datasets generally employ a single virtual world in which a virtual camera is moved around. However, this approach yields datasets where some environmental parameters are varied but others are constant or only partially varied throughout the world. Ideally, all parameters would be varied such that each image can be made as different from others as possible. Unfortunately, the architecture of game engines do not lend themselves well to this type of wide variability due to the divergence between the needs of game play and those of synthetic dataset production.

In order to produce a highly varied dataset, our method instantiates an entirely unique virtual world for each image. Although more time consuming than re-using the same geometric construct for multiple images, it is made practical by generating only the set of geometry that is visible either directly to the camera, or through reflections and shadows cast into the view of the camera.

When constructing the virtual world, we define a set of parameters to vary as well as a *rule set* that translates the parameter values into the concrete scene definition. It is worth noting that this approach yields an exponential explosion of potential worlds. However, in the context of dataset generation, this is entirely beneficial and means that each added factor of variation multiplies rather than adds to the size of the parameter space.

The following list highlights some of the key procedural parameters used in producing the dataset:

- **Road** width; number of lanes; material; repair marks; cracks
- **Sidewalk** width; curb height; material; dirt amount
- **Building** height and width; window height, width and depth; material
- **Car** type; count; placement; color
- **Pedestrian** model; count; placement (in road, on sidewalk)
- **Vegetation** type; count; placement
- **Sun** longitude; latitude
- **Cloud cover** amount
- **Misc.** Placement and count of poles, traffic lights, traffic signs, etc.

Our virtual world uses a mixture of both procedurally generated geometry, as well as model libraries. For example, the buildings, road surface, sidewalks, traffic lights and poles are all procedurally generated and individually unique. For pedestrians, bicyclists, cars and traffic signs, we use model libraries, where the geometry is shared between all instances, but properties such as placement, orientation and certain texture and material aspects vary between instances. Despite using a small set of prototypes for these classes, the resulting dataset is still rich, due to the large variability in how these classes are seen. In addition, the rule set used to populate the virtual world includes expected contextual arrangements such as pedestrians in cross walks, cars and bicyclists crossing the road, etc.

The illumination of the scene is specified by a sun position and includes an accurate depiction of the sky, including cloud cover. This ensures that the lighting conditions at street level includes a continuous range of times of day, all potential light directions relative to the ego vehicle, as well as indirect light due to clouds and other participating media. Illumination is calculated in a high dynamic range, scene-referred linear RGB color space, ensuring that the virtual camera sees realistic light and contrast conditions.

Overall, we have chosen to focus the development effort for the evaluation dataset on the most relevant classes for automotive purposes: road, sidewalk, traffic light and sign, person, rider, car and bicycle. These classes exhibit greater model variation, geometric detail, material fidelity and combinatoric complexity compared to the other classes, which are present but less refined. The evaluations presented in the next section show that the higher realism and feature variation in the selected classes increases the accuracy of the semantic segmentation.

4. Evaluation

To evaluate the proposed data synthesis pipeline and the resulting dataset, we perform a series of experiments wherein we train different state-of-the-art neural network architectures using synthetic data, and combinations of synthetic and organic data from Cityscapes. We compare the performance of our dataset to the two well-known synthetic datasets by Ros et al. (SYNTHIA) [27], and Richter et al. [26]. Although it is difficult to say exactly how much time was spent producing these two virtual worlds, it is worthwhile to remember that one was produced by a research lab while the other represents the direct and indirect work of over 1,000 people.

4.1. Methodology

We use semantic segmentation as the benchmark application, and compare the performance obtained when the network is trained using our synthetic data, consisting of 25,000 images, to that obtained using SYNTHIA, with 9,400 training images, and the set of 20,078 images[5] presented by Richter et al. Figure 4 shows the distribution of classes in our dataset. We evaluate the performance of both the synthetic data alone, and by subsequent fine-tuning on the Cityscapes training set. The network performance is computed from inference results on the Cityscapes validation set and quantified using the intersection over union (IoU) metric. For the comparison we choose two architectures, and use the publicly available reference implementations without any further modifications:

[5]The DFCN and FRRN architectures require consistent resolution for all inputs, so we use the subset of the original 25,000 images that have identical resolution.

1. **DFCN** – A dilated fully convolutional network, as presented by Yu and Koltun [35], which proposes an exponential schedule of dilated convolutional layers as a way to combine local and global knowledge. This architecture integrates information from different spatial scales and balances local, pixel-level accuracy, e.g. precise detection of edges, and knowledge of the wider, global level. The architecture consists of the frontend module along with a context aggregation module, where several layers of dilation can be applied.

   The DFCN-frontend network was trained using stochastic gradient descent with a learning rate of 10^{-5}, momentum of 0.99 and a batch size of 8 for synthetic data. For organic data, we used a learning rate of 10^{-4} in baseline training and 10^{-5} in fine-tuning, with the same momentum and batch size. For both frontend baseline and fine-tuning trainings for all datasets each crop is of size 628×628. The DFCN-context network was also trained using stochastic gradient descent with a learning rate of 10^{-5}, momentum of 0.99, a batch size of 100 and 8 layers of dilation for SYNTHIA and our dataset and 10 layers for Richter et al. For Cityscapes we used a learning rate of 10^{-4}, 10 layers of dilation, and same momentum and batch size as for synthetic data. A maximum of 40K iterations were used during frontend baseline training, 100K iterations for frontend fine tuning and 60K iterations for context baseline training.

   The project's GitHub page[6] provides implementation details. Results are given for all classes available in the Cityscapes dataset, except wall, fence and train, which are not present in our dataset.

2. **FRRN-A** – A full-resolution residual network, as described by Pohlen et al. [24]. The network is trained from scratch with no weight initialization using the same dataset combinations as for DFCN. In this architecture we provide results for all 19 classes available in Cityscapes dataset with IoU scores computed at the native image resolution of the FRRN-A architecture. The reference implementation can be found on GitHub[7].

   The FRRN-A network was trained using ADAM [17] with a learning rate of 10^{-3} for organic data and 10^{-4} for synthetic data. The bootstrap window size was set to 512 for organic data and 8192 for synthetic data. The batch size was 3. A maximum of 100K iterations were used both during baseline training and fine tuning.

Our primary goal is to explore how the performance of each

[6]https://github.com/fyu/dilation

[7]https://github.com/TobyPDE/FRRN

	Dataset	Road	Sidewalk	Building	Pole	Tr.Light	Tr.Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Motorcycle	Bicycle	Mean IoU
DFCN – frontend	S	0.44	18.54	35.25	15.00	0.00	0.00	64.91	0.00	72.09	49.34	2.81	60.51	0.00	**11.47**	0.06	0.83	20.7
	GTA	54.82	21.82	**66.37**	18.01	11.89	4.31	**79.02**	**30.43**	**72.99**	40.56	2.39	**73.49**	**11.33**	8.58	1.84	0.00	31.12
	O	**71.33**	**34.29**	63.33	**33.33**	**23.24**	**28.33**	72.58	5.99	67.22	**49.67**	**26.21**	50.97	7.10	5.19	**3.14**	**48.89**	**36.93**
	CS	96.41	75.97	90.04	50.61	50.16	65.17	90.67	54.31	90.85	72.34	44.69	89.99	40.62	59.08	43.81	68.57	67.71
	S + CS	96.46	75.77	90.10	50.13	49.65	64.81	90.51	55.72	91.22	73.41	**45.43**	90.28	45.78	**66.03**	47.06	68.87	68.83
	GTA + CS	96.62	76.97	**90.34**	**51.08**	**51.47**	64.86	**90.96**	**58.38**	91.02	73.44	44.24	90.59	45.75	63.16	46.59	69.00	69.03
	O + CS	**96.70**	**77.26**	90.30	49.58	51.34	**65.24**	90.85	57.34	**91.34**	**74.30**	45.25	**90.95**	**47.24**	65.69	**47.75**	**70.09**	**69.45**
DFCN – context	CS	96.48	76.96	90.24	51.16	51.18	66.99	90.98	57.83	91.74	71.76	46.13	90.04	47.00	65.52	45.02	64.76	68.97
	S	0.46	17.66	34.39	13.24	0.00	0.11	63.83	0.00	72.26	47.64	2.58	61.34	0.00	7.90	0.64	1.30	20.21
	GTA	54.07	22.10	60.95	20.25	20.52	4.54	**78.98**	**26.57**	**72.38**	43.95	1.10	69.19	**15.06**	**12.83**	**6.92**	0.00	31.84
	O	**74.42**	**37.85**	**70.26**	**32.91**	**27.32**	**28.85**	63.89	9.89	70.44	**52.76**	**26.23**	**73.73**	10.22	9.21	3.30	**45.50**	**39.80**

Table 1: DFCN [35] results on Cityscapes [5] (CS), SYNTHIA [27] (S), Richter et al. [26] (GTA) and Our data (O).

	Dataset	Road	Sidewalk	Building	Pole	Tr.Light	Tr.Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Motorcycle	Bicycle
DFCN – frontend	S	1.04	18.54	36.25	17.37	0.00	1.71	67.15	0.00	72.41	49.55	3.90	62.42	0.00	11.47	0.32	0.83
	GTA	63.45	24.32	**68.51**	18.54	16.22	5.98	**81.12**	**30.43**	**72.99**	41.59	3.68	**73.49**	**15.02**	**13.01**	2.47	0.00
	O	**78.42**	**38.21**	64.02	**33.44**	**24.61**	**32.57**	77.09	9.06	72.35	**51.98**	**26.93**	57.55	9.23	5.83	**3.14**	**49.29**
	CS	96.56	76.76	90.12	50.61	53.38	68.16	90.85	58.72	91.29	73.98	44.69	90.12	40.62	60.78	44.38	70.72
	S + CS	96.56	76.68	90.20	50.35	49.94	64.87	90.82	57.26	91.30	73.61	45.45	90.29	45.78	66.03	**48.56**	69.02
	GTA + CS	96.62	76.97	90.35	**51.08**	51.66	65.19	**91.11**	58.49	**91.64**	73.44	44.81	90.59	47.21	64.78	48.14	69.47
	O + CS	**96.72**	**77.26**	**90.45**	51.06	**52.22**	**66.09**	90.98	**58.87**	91.54	**74.59**	**47.27**	**90.96**	**47.64**	**66.70**	47.97	**70.23**
DFCN – context	CS	96.53	77.24	90.41	51.50	51.99	67.18	91.08	57.99	91.78	72.43	46.84	90.36	47.00	65.98	46.10	67.22
	S	0.48	17.73	34.52	13.36	0.00	0.24	64.01	0.00	72.30	47.66	2.63	61.36	0.00	8.44	0.65	1.35
	GTA	60.28	22.98	62.42	20.55	20.61	4.59	**78.98**	**26.57**	**72.47**	43.95	1.17	69.19	**15.06**	**12.83**	**7.05**	0.00
	O	**76.04**	**38.52**	**70.38**	**32.96**	**27.66**	**28.89**	67.15	9.93	70.46	**53.01**	**26.81**	**73.73**	10.23	9.28	3.39	**46.57**

Table 2: Best per-class IoU for all validation iterations on Cityscapes [5] (CS), SYNTHIA [27] (S), Richter et al. [26] (GTA) and Our data (O) for DFCN [35] architecture.

dataset varies across the different testing conditions and to analyze which properties persist across contexts. In particular, we chose the the DFCN network because of its high performance while still allowing weight initialization from an ImageNet-pretrained VGG architecture. On the other hand, the FRRN network must be trained from scratch, but has the capacity to attain quite high performance on the Cityscapes benchmark. We expect the FRRN architecture to be the most difficult for the synthetic datasets, whereas DFCN's use of the VGG weights may mask domain shift to some degree.

In both architectures, the results given represent the best validation iteration, with snapshots taken at each 2K iterations for DFCN and 1K iterations for FRRN.

4.2. Results and analysis

Table 1 presents results of the DFCN front-end module for pure synthetic training as well as versions with fine-tuning on the Cityscapes dataset. On pure synthetic training, our dataset scores 36.93% with Richter et al. at 31.12% and SYNTHIA at 20.7%. Although our method scores highest overall, the gains are largely in the classes on which development was focused, and it is clear that the other classes need further work, e.g. buildings and vegetation, where Richter et al. contains large variation and performs better.

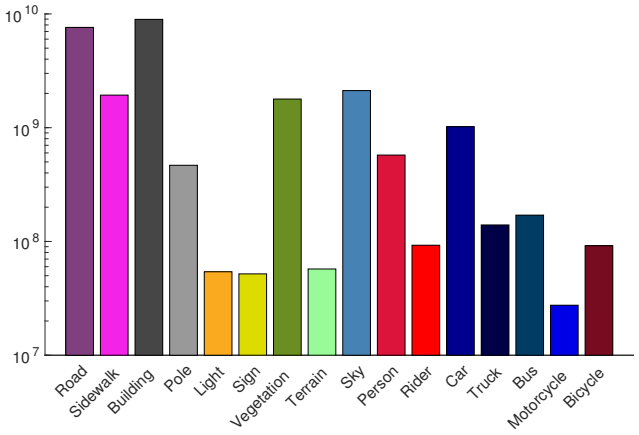

Figure 4: Number of annotated pixels per class in our dataset.

In the fine-tuning case, all three datasets improve upon the mean IoU score compared to the Cityscapes baseline. Overall, our dataset achieves an IoU of 69.45% when fine-tuning on Cityscapes. This is in comparison to 68.83% for SYNTHIA, and 69.03% for Richter et al.

Table 1 also gives the scores obtained by the DFCN con-

	Road	Sidewalk	Building	Wall	Fence	Pole	Tr.Light	Tr.Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle	Mean IoU
S	60.77	28.04	59.75	0.07	0.07	25.63	2.34	2.69	74.59	0.00	74.83	38.35	3.84	35.56	0.00	2.09	0.00	**1.92**	2.74	21.75
GTA	40.30	21.20	62.45	**7.17**	**6.85**	0.00	**11.03**	1.52	**75.40**	12.59	59.83	31.72	0.00	27.30	**14.91**	**7.47**	**7.98**	0.23	0.02	20.42
O	**85.84**	**44.45**	**67.05**	–	–	**29.34**	10.50	**24.45**	70.09	**13.51**	**80.10**	**50.67**	**20.25**	**60.51**	5.68	7.41	–	1.18	**20.91**	**31.15**
CS	97.49	80.43	90.41	41.47	43.77	61.39	61.95	71.58	91.34	**61.86**	**94.04**	75.11	51.36	92.90	56.56	64.52	46.59	42.62	69.53	68.15
S + CS	**97.58**	**81.04**	**90.81**	47.58	**50.49**	62.48	63.05	73.45	91.47	60.39	93.8	77.11	53.05	93.19	57.04	73.21	52.64	38.07	71.51	69.89
GTA + CS	96.90	77.17	90.71	**49.20**	48.62	62.42	61.58	72.34	91.25	60.93	93.84	75.53	53.77	**93.64**	**64.19**	73.13	**61.44**	**46.80**	70.96	70.76
O + CS	97.36	80.77	90.80	45.95	48.21	**63.57**	**64.73**	**76.16**	**91.60**	60.59	93.69	**77.41**	**55.36**	93.57	62.30	**74.43**	55.72	46.01	**71.93**	**71.06**

Table 3: FRRN-A [24] results on Cityscapes [5] (CS), SYNTHIA [27] (S), Richter et al. [26] (GTA) and Our data (O).

	Road	Sidewalk	Building	Wall	Fence	Pole	Tr.Light	Tr.Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle
S	68.13	29.01	68.29	4.03	0.28	26.28	4.55	5.60	76.71	0.00	82.00	40.80	7.17	36.86	0.00	5.82	0.00	3.44	9.37
GTA	60.07	21.20	**70.11**	**8.80**	**15.79**	0.0	12.15	3.89	**77.40**	**19.65**	77.92	37.74	3.10	45.80	**20.29**	**16.79**	**17.50**	1.92	1.07
O	**90.87**	**47.36**	69.31	–	–	**33.26**	**13.58**	**26.84**	75.58	15.67	**83.19**	**54.38**	**26.10**	**70.18**	19.27	13.07	–	**4.81**	**28.31**
CS	97.65	81.49	90.61	48.56	46.69	61.92	62.42	72.33	91.43	61.86	94.30	76.08	53.55	93.46	57.97	69.56	50.71	47.16	70.47
S + CS	97.73	82.10	90.98	50.98	**52.23**	63.95	65.01	74.44	91.82	62.72	94.46	78.00	56.54	93.35	60.82	75.09	60.95	44.80	72.18
GTA + CS	97.73	81.57	91.03	**54.77**	50.54	63.00	64.03	73.72	91.66	61.18	**94.49**	76.75	56.58	**93.64**	65.59	76.50	**63.59**	47.90	71.65
O + CS	**97.82**	**82.25**	**91.08**	54.68	51.01	**64.50**	**65.83**	**76.66**	**91.92**	**63.14**	94.46	**78.20**	**57.28**	93.58	**66.10**	**77.38**	62.31	**48.51**	**72.74**

Table 4: Best per-class IoU for all validation iterations on Cityscapes [5] (CS), SYNTHIA [27] (S), Richter et al. [26] (GTA) and Our data (O) for FRRN-A [24] architecture.

text module, which acts on the front-end dense predictions in order to increase their accuracy both quantitatively and qualitatively. Our dataset scores 39.8%, achieving a further 7.2% improvement on mean IoU performance compared to the front-end result. In comparison, Richter et al.'s mean IoU score barely improves, and SYNTHIA regresses. We believe this is due to the fact that each of our training images are unique, giving the network more variation at the contextual level. In particular, one of the most important classes – car – improves to 73.73%, achieving the highest score in synthetic only training.

We note that although all Cityscapes classes are present in SYNTHIA, four achieve zero percent IoU (traffic light, traffic sign, terrain and truck) in frontend training and three of them (traffic light, terrain and truck) fail to improve in context training. This is due to the small number of total pixels occupied by these classes, and it highlights the importance of providing enough exemplars in the training dataset for the network to learn from. Likewise, the bicycle class is present in Richter et al., but is not well learned.

Table 3 shows results for the FRRN architecture on synthetic-only training as well as fine-tuning. Despite lacking three classes (wall, fence and train), our dataset achieves an IoU of 31.15%, with SYNTHIA at 21.75% and Richter et al. at 20.42%. Notably, the Richter et al. dataset performs worse than SYNTHIA in this architecture, and some classes that perform well with DFCN drop significantly on FRRN, e.g. Richter et al.'s car (73.49% to 27.30%) and person (40.56% to 31.72%). We attribute this to training the network from scratch, which highlights the domain shift between Richter et al. and Cityscapes. In contrast, our dataset achieves similar scores on both networks (50.97% to 60.41% and 49.67% to 50.67% for car and person, respectively), indicating less domain shift and less reliance on the initial VGG weights.

Fine tuning on the full set of Cityscapes images yields a score of 71.06% (a 4.3% relative increase) for our dataset, with 69.89% for SYNTHIA and 70.76% for Richter et al. In the FRRN case it is evident that the network shares its feature weights across classes; although our dataset contains no instances of the wall, fence or train classes, with fine-tuning on the Cityscapes dataset the network sees a relative increase in IoU of 10.8% for wall, 10.1% for fence and 19.6% for train. As expected, when a wider range of training data is given for the network to learn class **X**, class **Y** can benefit from improvements to the shared set of features. In fact, because those three classes are rarely occurring in the Cityscapes data, the percentage increase in performance is two to four times greater than for more common classes.

When considering the classes on which our development was focused (see Section 3.2), we see that we reach the highest score in 7 out of those 8 classes for both the DFCN and FRRN architectures on pure synthetic training. This type of targeting is especially desirable when considering an evolving dataset, and the procedural modeling approach ensures that further variation can be added as needed to specific classes without undoing previous work.

During training there are oscillations in individual class scores, which stem from the inherent competition between the classes to optimize the shared feature weights to suit each particular class' needs. Table 2 shows the difference between the best overall validation iteration (based on mean

IoU) and the best per-class IoU across all validation iterations for the DFCN architecture. Table 4 shows the same data for FRRN. Here we can see that most classes across all three datasets achieve a result that is higher than the Cityscapes baseline at some point during training, but that the iteration that provides the overall best mean IoU may have several classes performing below their respective optima. For the 16 classes included in our dataset, we achieve the best per-class IoU on 12 classes in DFCN-frontend and 14 classes in FRRN.

There are further conclusions to be drawn from each dataset's performance in the two respective architectures. While the dataset from Richter et al. achieves 50% higher mean IoU performance than SYNTHIA in the DFCN architecture, it is 6.1% behind SYNTHIA in the FRRN architecture. We attribute this to the VGG weight initialization used in DFCN, which carries features from pre-training on ImageNet, and which seem to complement features that are lacking in the Richter dataset itself. It is likely the case that low-level features exhibit a large domain shift in Richter et al., but that the greater variation in high-level features due to the extensive game world yields better training. This is further exemplified once fine-tuning is performed on Cityscapes: with the organic dataset added the network pre-trained on Richter et al. again outperforms SYNTHIA.

The three example images in Figure 5 show the behavior of the DFCN architecture trained on each of the synthetic datasets. The domain shift in the road surface and sidewalk classes is evident in both SYNTHIA and Richter et al., which both suffer from false predictions in large parts of the roadway. Although our dataset performs significantly better on classes such as road, sidewalk, traffic signs, bicycles, poles and motorcycles, it still has problems categorizing buildings correctly, likely due to insufficient variation for that particular class in our training set.

The FRRN examples in Figure 6 show similar outcomes as DFCN, although the structure of false predictions tends to be of higher frequency due to the residual network architecture. In particular, we see that the network, without weight initialization, is even more susceptible to domain shift than DFCN.

The images in Figure 7 show the results of fine-tuning the DFCN network on the full set of Cityscapes for our dataset in comparison with the predictions obtained by training DFCN on Cityscapes itself. Here, the improvements provided by pre-training on our synthetic data act as corrections, such as the vertical traffic sign in the bottom left image being corrected from a misprediction of 'person', and the detection of traffic signs in the middle of the same image, that the Cityscapes-only network fails to catch.

In Figure 8 we see fine-tuning results for the FRRN network. Here, the residual architecture provides more room for silhouette improvements, and we can see tightening of the predictions of 'person' in the bottom right image, as well as examples of the correction to both the wall and fence classes, as discussed previously. We can also see the limitations of judging network performance on hand-annotated data: both the organic-only and the fine-tuned networks correctly predict an additional traffic light on the left side of the image, which is not annotated in the ground truth image.

When considering the training of neural networks, we would expect a practitioner to choose a training regimen that will yield the highest possible performance, which may include a combination of weight initialization and a mixture of both organic and synthetic data. However, there are many uses for synthetic data besides training: labeled data can also be used for validation of models, for exploration of novel architectures, and for analysis of trained models. In these cases, neither weight initialization nor fine tuning can help bridge the domain shift of datasets with poor realism. GANs have shown some promise in this context, but their use in improving the realism of visual data is so far limited to cases of coarse annotations [31]. At the moment, the best way to produce high quality synthetic data is to engineer the realism into the data itself, rather than attempt to make an unrealistic dataset more realistic through machine learning means.

5. Conclusion and future work

This paper presented a new approach for generation of synthetic image data with per-pixel accurate annotations for semantic segmentation for training deep learning architectures in computer vision tasks. The image synthesis pipeline is based on procedural world modeling and state-of-the-art light transport simulation using path tracing techniques.

In conclusion, when analyzing the quality of a synthetic dataset, it is in general most telling to perform training on synthetic data alone, without any augmentation in the form of fine-tuning or weight initialization. Our results indicate that differences between datasets at the pure synthetic stage provide the best picture of the relative merits, down to the per-class performance. We also conclude that a focus on maximizing variation and realism is well worth the effort.

We estimate that our time investment in creating the dataset is at least three to four orders of magnitude smaller than the much larger virtual world from Richter et al., while still yielding state-of-the-art performance. We accomplish this by ensuring that each image is highly varied as well as realistic, both in terms of low-level features such as anti-aliasing and motion blur, as well as higher-level features where realistic geometric models and light transport comes into play.

In the future, we will analyze in more detail what impact the realism in the light transport simulation has on the neural network performance, to understand the trade-off between computational cost and inference results. Another in-

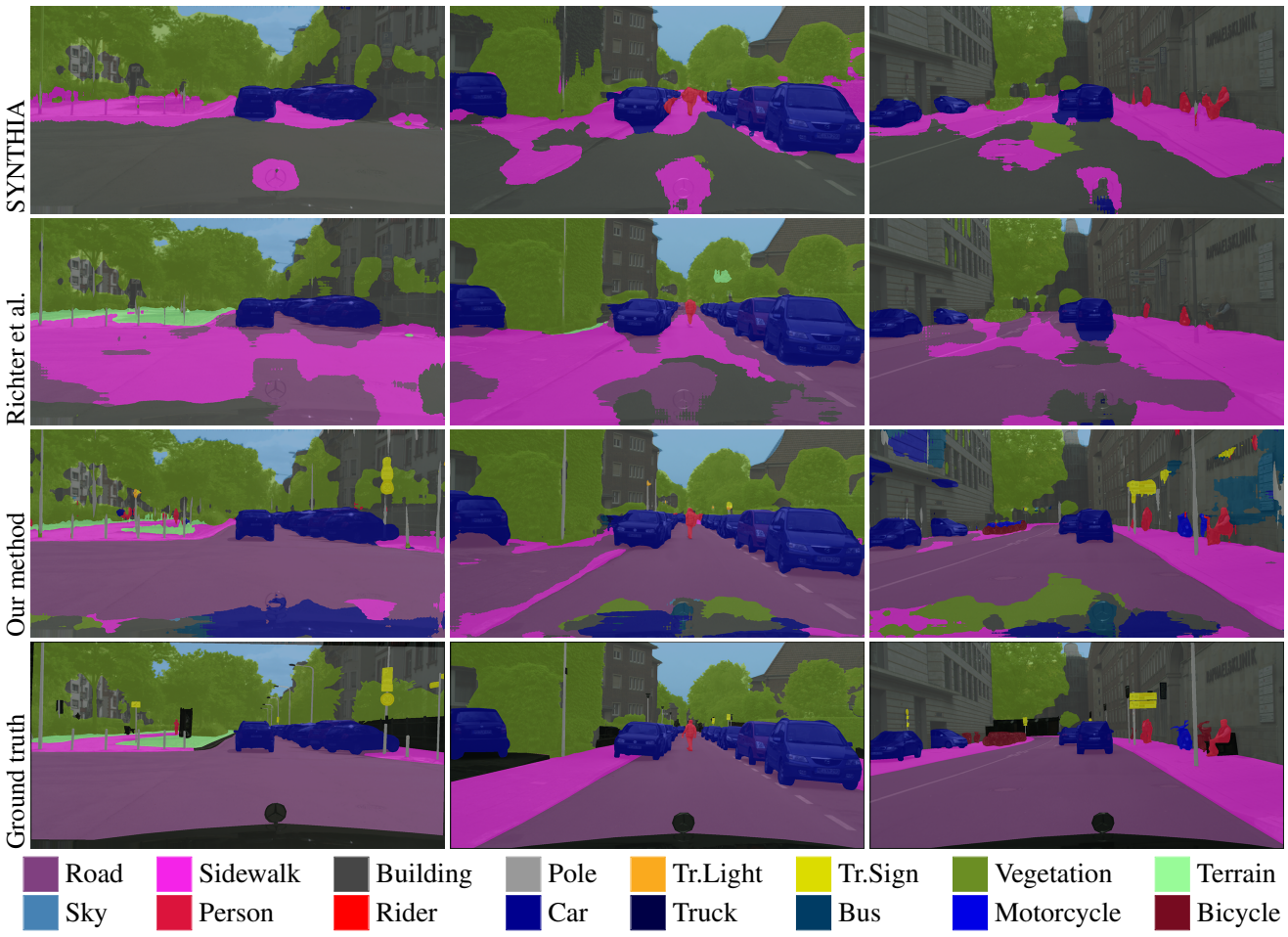

Figure 5: Results for the DFCN front-end architecture on pure synthetic data, corresponding to Table 1. Note the improved road surface and pedestrian segmentation, and the ability to identify traffic signs as well as poles, bicycles and motorcycles with our method.

teresting venue for future work will be to analyze realism's impact in other important computer vision tasks such as object recognition and feature tracking applications.

References

[1] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother. Augmented Reality Meets Deep Learning for Car Instance Segmentation in Urban Scenes. In *Proceedings of the British Machine Vision Conference*, Sept. 2017. 3

[2] M. Aubry and B. Russell. Understanding Deep Features with Computer-Generated Imagery. In *ICCV*, 2015. 2

[3] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recogn. Lett.*, 30:88–97, January 2009. 2

[4] S. Chandrasekhar. *Radiative Transfer*. Dover Books on Intermediate and Advanced Mathematics. Dover Publications, 1960. 4

[5] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 7, 8

[6] A. Dosovitskiy, P. Fischery, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. v. d. Smagt, D. Cremers, and T. Brox. FlowNet: Learning Optical Flow with Convolutional Networks. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 2758–2766, Washington, DC, USA, 2015. IEEE Computer Society. 2

[7] D. S. Ebert, S. Worley, F. K. Musgrave, D. Peachey, K. Perlin, and K. F. Musgrave. *Texturing and Modeling*. Academic Press, Inc., Orlando, FL, USA, 2nd edition, 1998. 3

[8] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual Worlds as Proxy for Multi-Object Tracking Analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2

[9] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets Robotics: The KITTI dataset. *International Journal of Robotics Research (IJRR)*, 32(11):1231–1237, Sept. 2013. 2

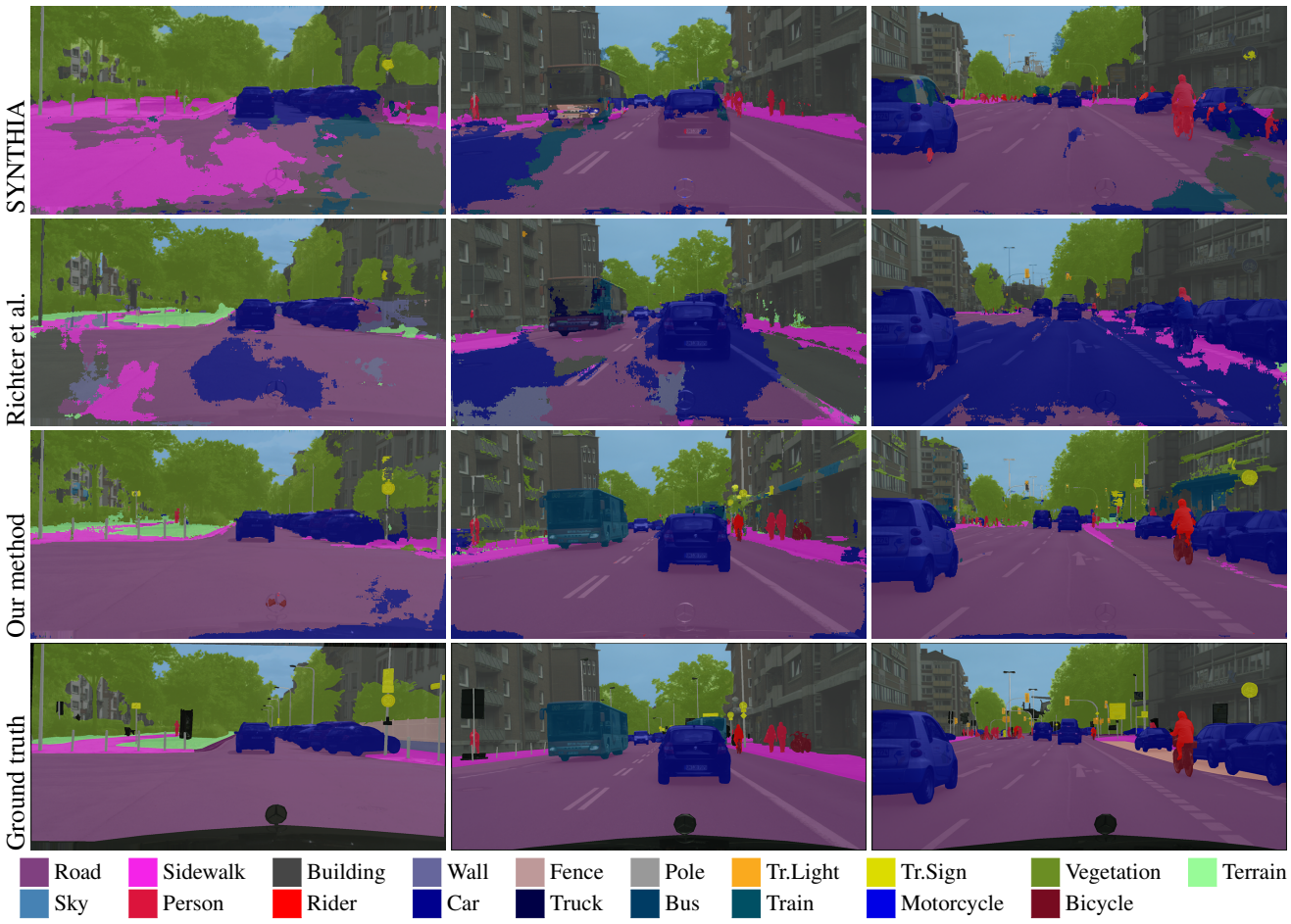

Figure 6: Results for the FRRN-A architecture, corresponding to Table 3. The more severe domain shift in SYNTHIA and Richter et al. is apparent, and only road, buildings, vegetation, and sky have IoU over 40%. Our dataset improves performance across nearly all classes, achieving at least 40% IoU on road, sidewalk, building, sky, person and car.

[10] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. 5

[11] S. Gupta, P. A. Arbeláez, R. B. Girshick, and J. Malik. Aligning 3D models to RGB-D images of cluttered scenes. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[12] A. Handa, V. Patraucean, V. Badrinarayanan, S. Stent, and R. Cipolla. Understanding Real World Indoor Scenes With Synthetic Data. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4077–4085, Las Vegas, USA, June 2016. 2

[13] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan. Driving in the Matrix: Can Virtual Worlds Replace Human-Generated Annotations for Real World Tasks? In *IEEE International Conference on Robotics and Automation*, pages 1–8, 2017. 2

[14] J. T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, Aug. 1986. 3, 4

[15] B. Kaneva, A. Torralba, and W. T. Freeman. Evaluating Image Features Using a Photorealistic Virtual World. In *IEEE International Conference on Computer Vision*, 2011. 2

[16] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba. Undoing the Damage of Dataset Bias. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part I*, ECCV'12, pages 158–171, Berlin, Heidelberg, 2012. Springer-Verlag. 2

[17] D. Kingma and J. Ba. ADAM: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 1

[19] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 1

[20] J. Marin, D. Vazquez, D. Geronimo, and A. Lopez. Learning Appearance in Virtual Scenarios for Pedestrian Detection. In *23rd IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 3

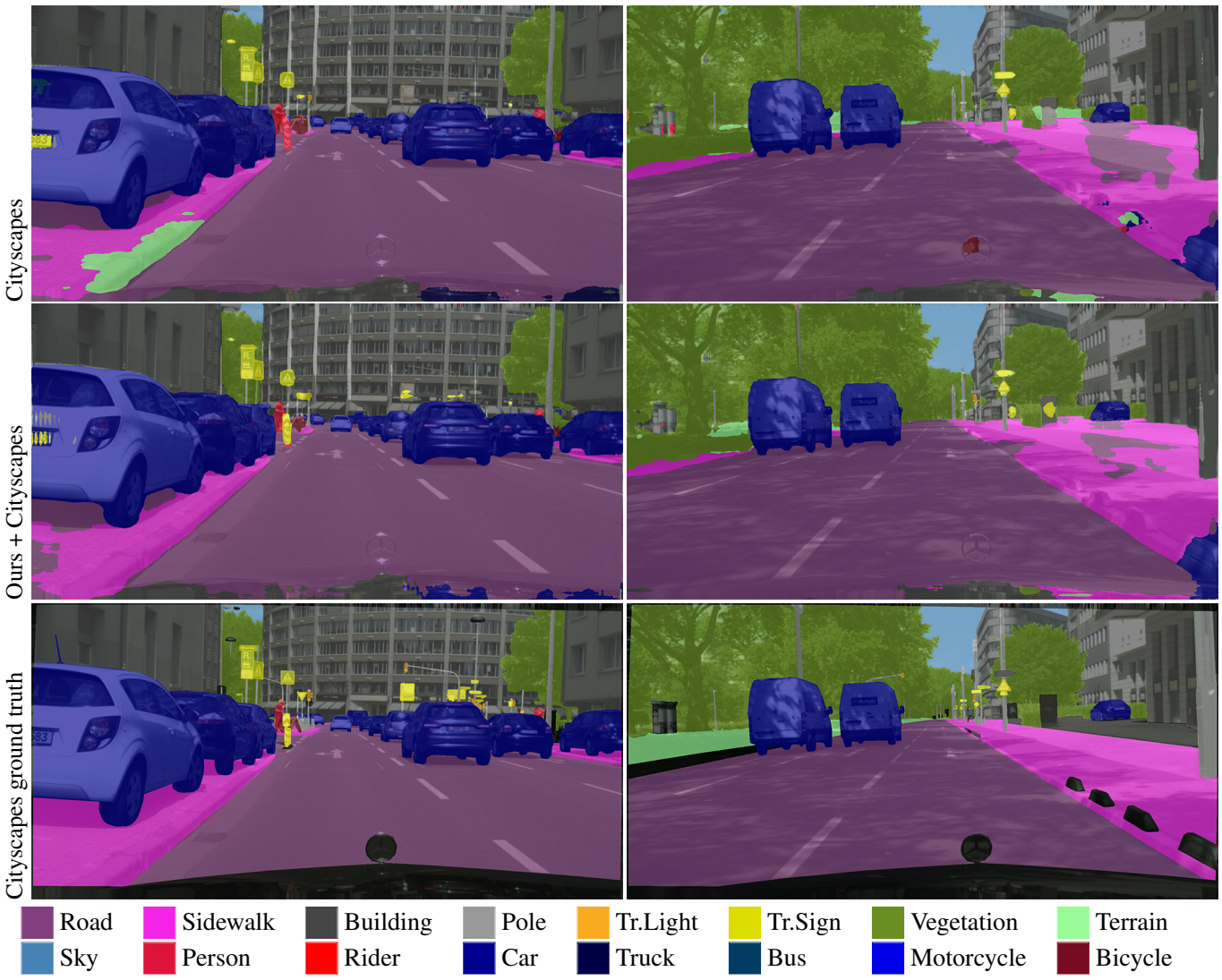

Figure 7: DFCN results with fine-tuning. Our dataset helps the network disambiguate between a street sign and a person in the left image, and allows the network to recognize a distant traffic light in the right image.

[21] Y. Movshovitz-Attias, T. Kanade, and Y. Sheikh. *How Useful Is Photo-Realistic Rendering for Visual Learning?*, pages 202–217. Springer International Publishing, Cham, 2016. 2

[22] F. E. Nicodemus. Directional Reflectance and Emissivity of an Opaque Surface. *Appl. Opt.*, 4(7):767–775, Jul 1965. 4

[23] M. Pharr and G. Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2010. 5

[24] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 2017. 6, 8

[25] W. Qiu and A. L. Yuille. UnrealCV: Connecting Computer Vision to Unreal Engine. In *Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III*, pages 909–916, 2016. 2

[26] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for Data: Ground Truth from Computer Games. In *European Conference on Computer Vision (ECCV)*, volume 9906, pages 102–118. Springer International Publishing, 2016. 1, 2, 6, 7, 8

[27] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1, 2, 6, 7, 8

[28] A. Rozantsev, V. Lepetit, and P. Fua. On rendering synthetic images for training an object detector. *Comput. Vis. Image Underst.*, 137(C):24–37, Aug. 2015. 3

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer*

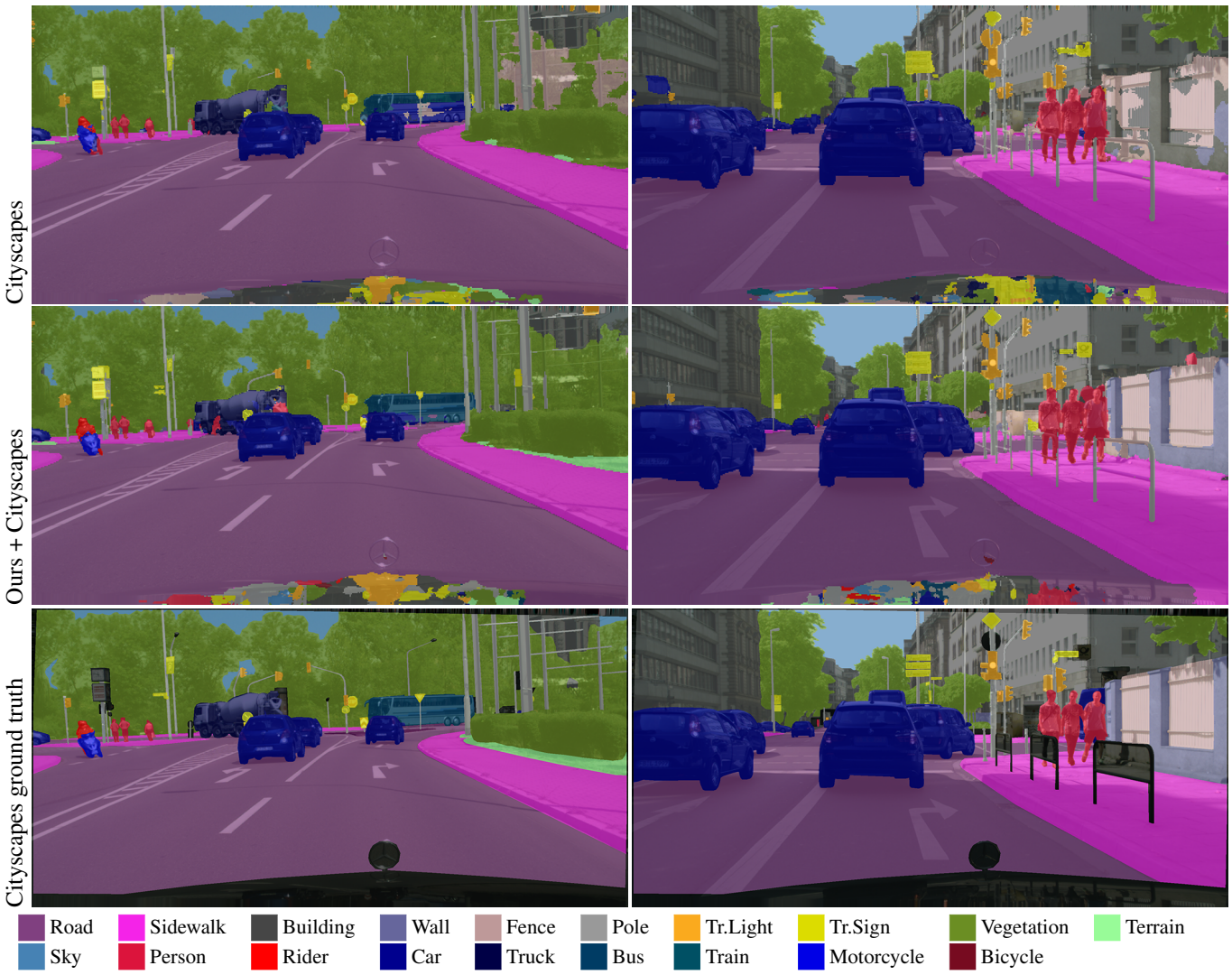

Figure 8: FRRN-A results with fine-tuning. In the left images we see that our dataset improves the vegetation and terrain recognition as well as that of the distant bus. In the right images we see significant improvement in the wall and fence classes, despite our dataset having no occurrences of either one.

Vision (IJCV), 115(3):211–252, 2015. 1

[30] A. Shafaei, J. J. Little, and M. Schmidt. Play and Learn: Using Video Games to Train Computer Vision Models. *CoRR*, abs/1608.01745, 2016. 2

[31] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from Simulated and Unsupervised Images through Adversarial Training. In *CVPR*, 2017. 3, 9

[32] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*, 2015. 1

[33] H. Su, C. R. Qi, Y. Li, and L. J. Guibas. Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015. 2

[34] A. Torralba and A. A. Efros. Unbiased Look at Dataset Bias. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1521–1528, Washington, DC, USA, 2011. IEEE Computer Society. 2

[35] F. Yu and V. Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. 6, 7

[36] Y. Zhang, M. Bai, P. Kohli, S. Izadi, and J. Xiao. DeepContext: Context-Encoding Neural Pathways for 3D Holistic Scene Understanding. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017. 2