ProcSy: Procedural Synthetic Dataset Generation Towards Influence Factor Studies Of Semantic Segmentation Networks

Samin Khan Buu Phan Rick Salay Krzysztof Czarnecki
University Of Waterloo
Waterloo, ON, Canada
{sa24khan, buu.t.phan}@uwaterloo.ca {rsalay, kczarnec}@gsd.uwaterloo.ca

Abstract

Real-world, large-scale semantic segmentation datasets are expensive and time-consuming to create. Thus, the research community has explored the use of video game worlds and simulator environments to produce large-scale synthetic datasets, mainly to supplement the real-world ones for training deep neural networks. Another use of synthetic datasets is to enable highly controlled and repeatable experiments, thanks to the ability to manipulate the content and rendering of synthesized imagery. To this end, we outline a method to generate an arbitrarily large, semantic segmentation dataset reflecting real-world features, while minimizing required cost and man-hours. We demonstrate its use by generating ProcSy, a synthetic dataset for semantic segmentation, which is modeled on a real-world urban environment and features a range of variable influence factors, such as weather and lighting. Our experiments investigate impact of the factors on performance of a state-of-the-art deep network. Among others, we show that including as little as 3% of rainy images in the training set, improved the mIoU of the network on rainy images by about 10%, while training with more than 15% rainy images has diminishing returns. We provide ProcSy dataset, along with generated 3D assets and code, as supplementary material [1].

1. Introduction

In recent years, there has been significant research and development in the domain of semantic segmentation — the act of assigning class labels to pixels in an image. However, much of this pertains to ideal weather and lighting conditions. This poses a real problem in gauging the performance and robustness of deep learning networks that are trained for semantic segmentation of road scenes. Thus, comprehensive datasets that include *scene influence factors* [7] such as weather and lighting are essential.

[1]ProcSy: https://uwaterloo.ca/wise-lab/procsy

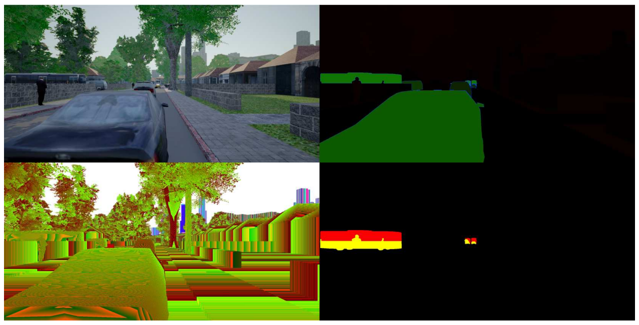
Figure 1. ProcSy dataset sample frame (from top-left going across then down): RGB image, GT_ID image, depth map, 1 occlusion map

In the real world, it is seldom the case that a road scene can be recaptured multiple times in various weather/seasons with varying lighting patterns, while keeping scene features, such as dynamic objects, the same. There have been advances in approaching this as a domain adaptation problem, but these tend to be expensive. For instance, Dark Cityscapes dataset [24] required the recording of various Zurich streets at different times of the day in order to apply guided style transfer.

Creating road scene weather and lighting variation is much more feasible and repeatable in a virtual environment. Research has progressed either via reverse-engineering existing commercial games [1, 21] or building road scene simulators from the ground-up [9]. Today, an open-source autonomous driving simulator such as CARLA can be used for influence factor variations and the data collection pipeline.

In this paper, we follow this approach and leverage the benefits of *procedural modeling* to create a virtual driving environment that replicates a real-life area of the world. The semi-automated nature of 3D procedural modeling tools enables us to rapidly model vast city areas for an infinite possibility of dataset creation and rendering, with a minimal effort.

Specifically, we make the following contributions:

(1) We present a novel workflow to replicate a real-world operational design domain in order to generate a semantic segmentation dataset with weather and lighting variations.

(2) We generate a sample synthetic dataset, ProcSy, consisting of 11,000 unique image frames that represent a 3 km^2 area of urban Canada (Fig. 1). This dataset also comprises semantic segmentation ground truth data, depth data, and vehicle occlusion maps. The dataset includes environmental influence factor variations in the form of rain, cloud, puddles, and Sun angle.

(3) We demonstrate the usefulness of ProcSy by using it to analyze the performance of an off-the-shelf deep-neural network, Deeplab v3+. We show the effects correlation of environmental factors, depth, and occlusion on the network's predictive capabilities.

The paper is structured as follows. In Sec. 2 we provide the required background. In Sec. 3 we show related work in semantic segmentation and inclement weather road-scene datasets. Sec. 4 describes the technical approach we have taken for generating the ProcSy dataset. In Sec. 5 we present three experiments using ProcSy to analyze the effects of influence factor variations on the performance of Deeplab v3+. Finally, we present conclusions and our vision for future research work in Sec. 6.

2. Background

2.1. Semantic Segmentation Network

Semantic segmentation of road scenes is an important domain of research in autonomous vehicle (AV) perception. Raw camera imagery is stored as a pixel matrix of RGB color values. The semantic segmentation task assigns each pixel into one of several classes of objects that are directly relevant to road scenes.

The research community has embraced Cityscapes Dataset, the seminal work of Cordts et al. [6], as the standard in benchmarking road-scene semantic segmentation networks. For the purposes of our experimentation, we focus on the 19 classes that are outlined by Cityscapes as relevant road scene classes.

We run our experiments with DeepLab v3+ [4]. The network is currently amongst state-of-the-art for Cityscapes' semantic segmentation task with a peak 82.1% mIoU accuracy. They achieve this by implementing atrous convolutions into the network architecture [3] and applying depthwise separable convolution to the atrous spatial pyramid pooling and decoder modules [5]. We use a ResNet-50 backbone architecture [14] that has been pre-trained on ImageNet dataset [8]. This requires less time to train than ResNet-101, thus allowing for quicker training and testing iterations.

2.2. Procedural Modeling

Procedural modeling is a 3D modeling paradigm based on sets of rules that are iterated upon. These rulesets are derived from Lindenmayer Systems (L-systems) [20]. An L-system is a type of formal language that was first developed by Hungarian botanist Aristid Lindenmayer in 1968. L-system is an iterative, parallel rewrite system. There is an initial axiom string from which the pattern propagates, and there exist rules to translate the string into generated structures after every iteration.

Parish et al. published seminal work towards automatic procedural modeling of cities [18]. This consists of taking the aforementioned L-system and extending it — called CGA Shape grammar. With their CityEngine urban planning tool, based around CGA Shape grammar rules, they have enabled professionals in various industries (from urban planners to movie and video game artists) to rapidly prototype large vistas while cutting costs in the process.

Procedural modeling can also be leveraged for semantic segmentation. This technique (using CityEngine) allows a single user to rapidly create a virtual rendition of a real-world map region in a matter of hours (Sec. 4).

3. Related Work

3.1. Inclement Weather Datasets

Segmentation datasets for road scenes have been around as early as 2008 with Brostow et al. introducing CamVid Dataset [2]. The Cityscapes Dataset [6], introduced in 2015, represented a leap in the scale and quality of the publicly available real-world datasets for semantic segmentation of road scenes. However, a common criticism of these preliminary datasets is that they lack in quantity of finely-annotated images. Another criticism is the lack of variation in weather and lighting conditions. These earlier datasets presented imagery in ideal daytime conditions with little to no signs of inclement weather. In contrast, our dataset employs gradual variations in weather and lighting conditions in its scenes.

Raincouver dataset [25], published in 2017, is perhaps the first publicly available dataset to contain rainy driving scenes at different times of day. This dataset contains only 326 finely annotated images, and is meant to supplement pre-existing datasets such as Cityscapes. In comparison, our dataset contains 11,000 finely-annotated images.

The Mapillary Vistas dataset was published in 2018 [17]. This dataset contains 25,000 finely-annotated, non-temporal images from many different geographical locations and conditions. However, a caveat with this dataset is the lack of metadata pertaining to the content of each image. This poses a problem in measuring effects of varying conditions on the performance of a semantic segmentation network. Our dataset has a gradual variation in quantifying rain amount present in scenes (Sec. 4.5).

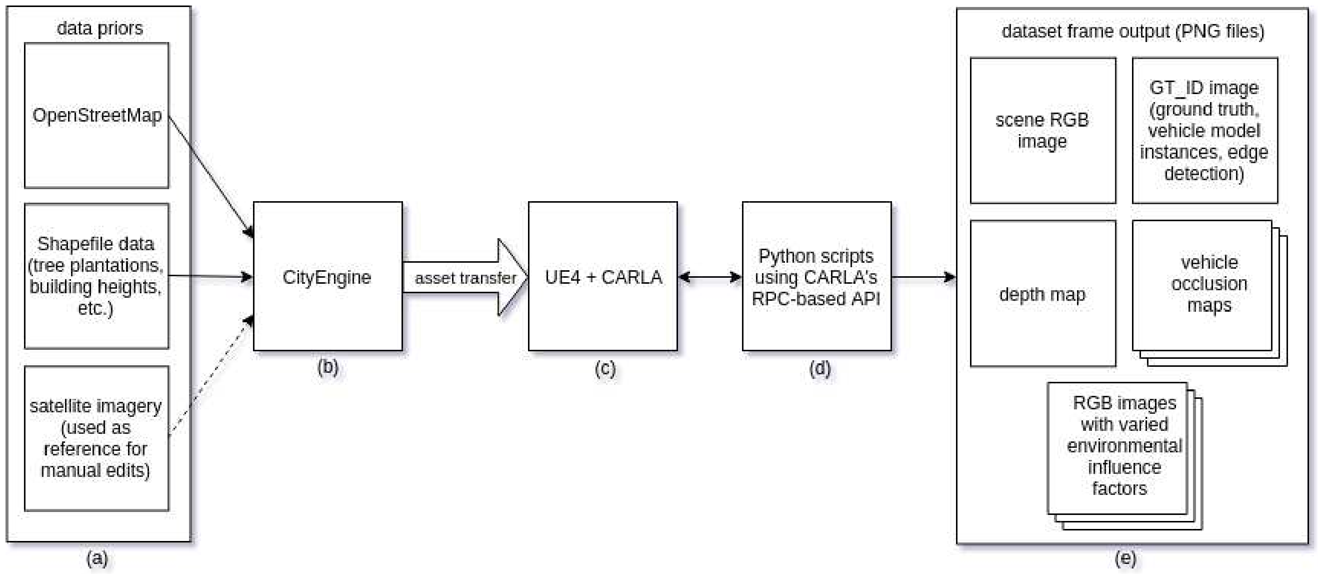
data priors
OpenStreetMap
Shapefile data (tree plantations, building heights, etc.)
satellite imagery (used as reference for manual edits)
(a)
CityEngine
(b)
asset transfer
UE4 + CARLA
(c)
Python scripts using CARLA's RPC-based API
(d)
dataset frame output (PNG files)
scene RGB image
GT_ID image (ground truth, vehicle model instances, edge detection)
depth map
vehicle occlusion maps
RGB images with varied environmental influence factors
(e)

Figure 2. Generation pipeline for the ProcSy dataset: a) data priors that are used for procedural modeling; b) CityEngine is the program used to generate three-dimensional, procedurally-generated world map; c) UE4 and CARLA are used for realistic lighting and weather effects; d) dataset generation through CARLA is controlled via Python-based scripting; e) each frame of our dataset have the outlined images rendered

Berkeley Deep Drive [27] provides a finely-annotated dataset comprised of 5683 images. This dataset has more weather and lighting variations than Cityscapes, but is noted to contain severe labeling inconsistencies in dark regions [24]. Similar to Mapillary, the BDD100K dataset does not contain metadata with which the variations can be measured. Thus, this dataset is also not conducive to measuring the performance of a network against controlled weather and lighting variations. In contrast, our dataset contains metadata about both weather and lighting variations, using a consistent annotation scheme (Sec. 4.2).

3.2. Synthetic Data Generation

In attempting to study effects of various influence factors, including weather and lighting conditions, the realm of synthetic dataset generation appears promising. This is because variations in a synthetic environment are easier to control and quantify. One such example dataset is Virtual KITTI [11] released in 2016. It is a virtual recreation of the KITTI dataset [12] with the additional benefit that they replicate each frame in 8 different weather and lighting variations. In total, Virtual KITTI currently has 21,260 frames [16] of semantically labeled data. In their paper, they provide an impact analysis of weather and imaging conditions on object tracking algorithms. However, they do not provide any analysis on semantic segmentation, as we have done in our experiments. As another example from 2016, Ros et al. published the SYNTHIA dataset [23], which contains 13,400 random road scene annotated images that are synthetically generated with various lighting and weather conditions. This dataset was updated in 2017 with the release of SYNTHIA-SF containing 2224 new images [15]. However, the dataset does not provide metadata quantifying lighting and weather variations, which are necessary to study their influence on predictions.

Another approach has been to leverage an existing game environment. Examples include Playing for Data and subsequently Playing for Benchmark datasets, developed using the Grand Theft Auto V game environment by Richter et al. in 2016 and 2017 [21, 22]. In total, the latest state of their research contains 254,064 images. Angus et al. proposed a different and more salable method of data generation from GTAV in 2018, and published a dataset with more than 1,000,000 images in ideal weather and lighting conditions [1]. A drawback of using existing games is that their optimized and closed-source nature does not allow for easily controlling both the content and the rendering process.

4. Generating the ProcSy Dataset

In this section, we detail the steps summarized in Fig. 2 for generating the ProcSy dataset.

4.1. Procedural Modeling for World Generation

The generation of ProcSy begins with creating a virtual environment from which to capture road scene imagery. Our intent is to build a dataset with emphasis on scene variation, therefore we do not focus on temporal frames. Instead, we collect images by teleporting the camera around a statically built 3D environment. We choose to only focus on building an environment with statically placed assets. This eliminates the effort needed to develop vehicle and pedestrian animation and traffic system controls.

We used a procedural modeling tool called CityEngine (Fig. 2b). As explained in Sec. 2.2, procedural modeling takes a set of grammar rules and iterates to create rich patterns. In the case of CityEngine, these patterns represent a three-dimensional derivation of city-scale maps.

OpenStreetMap data [13] is fed into CityEngine as prior input (Fig. 2a). This data contains macroscopic information about building footprints and road networks within a city region. We chose a 3 km^2 region of urban Canada (Fig. 3) as the template to model our virtual environment. Additional data priors were fed into CityEngine using Shapefiles [10] from municipal open databases. These data priors include such things as building heights and tree plantations.

CityEngine stitches data priors together in order to create a three-dimensional representation of the real-world counterpart. For ProcSy, it took about one-person hour to identify and collect the relevant priors. The time-consuming aspect of procedural modeling is to restructure road networks by accounting for lane-level details. Such reconstruction is needed because OpenStreetMap does not contain lane-level data, such as number of lanes, street/lane-width, or heights of overpasses and highways.

Due to data unavailability, CityEngine cannot automatically create exact road network layouts. A solution to this may be to use high-definition (HD) maps with lane-level details. However, HD map data in an open-source format is still not readily available in the public domain. Currently, the most cost-effective approach to this problem is using openly available satellite imagery of road networks. Thus, we used satellite imagery as a reference to manually adjust the road graphs in CityEngine. This step took one person approximately 40 hours for the 3 km^2 geographical area.

4.2. Generating Ground Truth and Depth Data

After the 3D environment has been generated and populated with static vehicle and pedestrian models in CityEngine, these are exported to Unreal Engine 4 (UE4; Fig. 2c). Doing so enables us to have more interaction with the rendering pipeline. CityEngine uses a traditional rendering approach with a simplified lighting model to allow real-time rendering of massive amounts of geometry. This means that end result of the render has limited realism. Also, CityEngine's renderer currently does not support simulating rain and other adverse weather effects.

Unreal Engine 4 relies on Physically-Based Rendering (PBR) [19] to approximate a realistic lighting model. Textures and materials used throughout UE4 assets are rendered in an energy-conservative manner to simulate physics of light in the real world. Further, using an open-source simulation platform such as CARLA streamlines the process of dataset generation. CARLA leverages UE4's depth and stencil buffers to output scene depth and automatic semantic annotations. This provides us with the platform for our dataset generation (Fig. 2d-e).

4.3. Vehicle Instances and Occlusion Maps

We generate instance-level segmentation for the vehicle class. UE4 has the capability of rendering metallic material buffer output. We leverage the idea of PBR materials and UE4's metallic buffer output in order to label vehicle models as instance groups (i.e., all instances of a specific vehicle model have the same label). By assigning each vehicle model's material to a different specularity value, we can have a different metallic buffer id per vehicle model. Additionally, we create a post-effect edge detection filter using a combination of Sobel and Laplacian edge detection algorithms. The resulting edges can be used to further break up the instance groups into individual instances. A space-saving measure, we encode ground truth data annotations, vehicle per-model instance ids, and edge detections in the same PNG file (called the GT_ID image in Fig. 2e), where each takes up one of the RGB channels.

For our dataset, there are 43 unique vehicle models in use. We create a post-effects filter to generate occlusion maps of vehicle models. We step through each of the unique vehicle models and use this filter to output occlusions for each frame that contains the corresponding vehicle model. As such, for each frame of the dataset, we have a set of occlusion maps for vehicle models that appear in the frame. We use these occlusion maps in one of the experiments in Sec. 5.2.

4.4. Data Selection

Data points are selected such that the camera is always facing towards a road-scene. The binary map in Fig. 3 is derived from a top-down render of the 3 km^2 map region. In its original resolution of 4545×4541 pixels, there are 1,349,841 unique camera positions that can be chosen (white pixels). At each road-going (white) pixel, the top 2 optimal orientations were determined by a simple line tracing algorithm. A longer, unobstructed line trace (a black pixel causes obstruction) is assumed to be a more optimal orientation. Afterwards, one optimal orientation was chosen at random for each road-going pixel.

From the set of 1,349,841 road scenes, using uniform random sampling, crude renders were first taken to identify 11,000 clean frames for our experimental dataset. This identification process required a human-in-the-loop to ensure frames did not have spawning issues with statically placed assets such as vehicles and pedestrians. An exam-

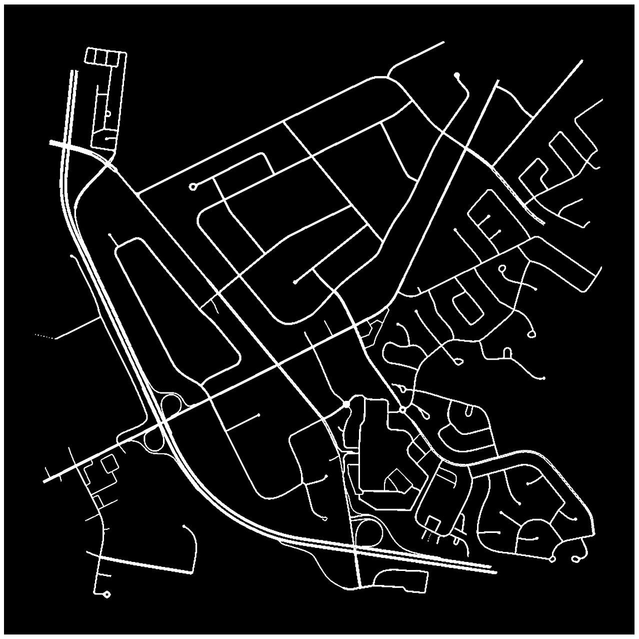

Figure 3. binary mask representing road-going pixels (white) of a 3 km^2 map region of urban Canada in top-down view; centered around (decimal degrees notation): 43.502507, -80.530754

ple non-clean frame would be the camera spawning inside a statically placed road-going vehicle, which means all or most of the frame would show only the vehicle class.

Once clean frames were identified, the dataset was split into a training set of 8000 frames, a validation set of 2000 frames, and a test set of 1000 frames. In Fig. 3, frames from the bottom-right quadrant were used for validation and test sets. Frames from the other 3 quadrants were used for the training set. The dataset was rendered in Cityscapes resolution (2048×1024 pixels) with the following outputs per frame: RGB image (2.5mb), GT_ID image (50kb), depth map (790kb), and an occlusion maps folder (approx. 60kb) containing n images where n refers to the number of unique vehicles that appear in the frame (Fig. 2e).

4.5. Influence Factors Variations

Our experimentation focuses on influence factor variations. We use CARLA's capability to generate depth maps, and we already discussed the generation of occlusion maps for vehicle class in Sec. 4.3. The remaining influence factors are environmental ones, which we achieve with help of existing UE4 functionality and CARLA. CARLA 0.9.1 introduced API calls to easily modify weather parameters and Sun position in real-time (Fig. 2d). For ProcSy, we selected three weather influence factors, namely rain, cloud, and puddle deposits (accumulation of water on road pavement). For each of these factors, we generate data for five different intensity levels of 0%, 25%, 50%, 75%, and 100%.

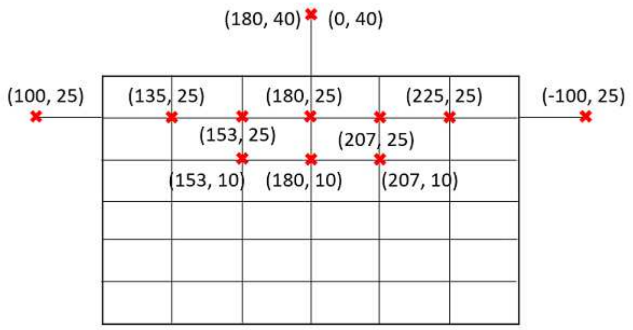

Figure 4. road scene frame approximately showing the Sun positions in and out of frame that are considered; red x's indicate Sun location and corresponding tuples show azimuth and altitude values used in CARLA

We also use the Sun's position in the sky as another influence factor (Fig. 4), consisting of the Sun's azimuth and altitude angles. In order to reduce the amount of variations to study, we first make note of positions in a road scene frame where the Sun's angle and coincident shadow effects are expected to have a meaningful impact. We note that below the horizon the Sun's angle is irrelevant. Also, a typical road scene is expected to have buildings or other architecture along the left and right sides. These generally approach a vanishing point near center of the frame. Therefore, we identify eight Sun positions within the frame that represent a V-shape in upper half of the frame. We also consider four Sun positions outside the frame.

With the identified environmental influence factor variations, we can create 5×5×5×12=1500 unique RGB images for each frame of our dataset. As explained in Sec. 5, we use a subset of this for our experimentation in order to enable faster experiment iterations.

5. Analysis with ProcSy Dataset

In this section, we present three experiments to demonstrate the usage of our synthetic data for understanding different effects of influence factors on semantic segmentation model's performance. For each of the three experiments we give the experimental objective, details, results, and actions suggested by the results.

5.1. Effects of influence factors on model's performance

In this experiment, we train and compare the performance of two Deeplab v3+ models: model A is trained with 8000 clean images, and model B is trained with 8000 images with equal proportion of three influence factors (rain, cloud, and puddles). More precisely, model B's training images are split into three equal parts, one per factor. Then each part is split in five equal sub-parts, one for a differ-

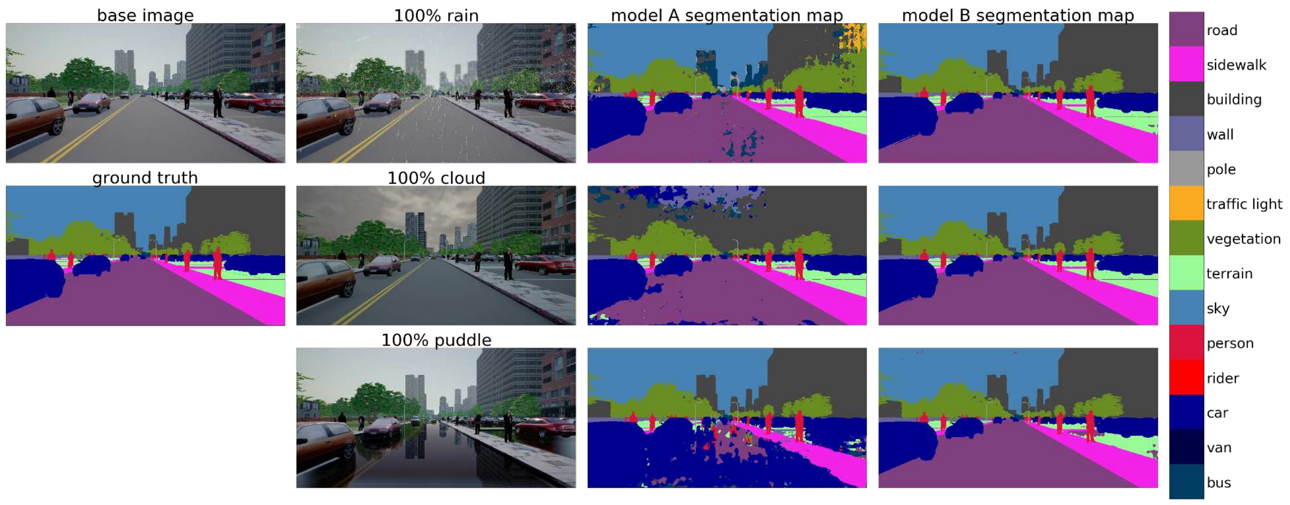

Figure 5. Performance of model A and B with different factors (each row); here, due to space constraints, we only show samples at 100% level for each influence factor

ent level of the given factor to be applied: 0%, 25%, 50%, 75%, and 100%. Each model is trained with 140,000 iterations with a batch-size of 16 and crop-size of 512×512. The performance of each model is evaluated by testing with different influence factors. The purpose of this experiment is two-fold. First, we would like to observe how the influence factors degrade the performance of model A. Second, it is important to know if model B is able to generalize across different influence factors and is more robust than model A. Otherwise, we would need to explore more advanced training methods to improve robustness, e.g., [28].

In Fig. 6, we show mIoU of two models under different testing conditions, where all test images have a certain level of a influence factor. For model A, as we increase the level of each influence factor, its performance worsens. Initially, we found it surprising that cloud and puddle factors decrease the performance more than the rain factor does. The reason for this is largely due to the model inaccurately predicting sky as another class, such as car. This kind of error can be seen in Fig. 7, where we plot the IoU of several classes for each model. For example, in Fig. 7, the IoU values of car and sky classes reduce significantly with increased clouds compared to the other classes. Similarly, we can also see that puddles have a strong negative effect on person, car, and road classes, because the puddle factor creates reflection of objects on the ground. In contrast to model A, model B has generally stable performance across all and different levels of factors, which suggests that the model can generalize from the exposure to these factors in training. Fig. 5 shows samples of each model's prediction under different situations. We note that there is a small gap in overall and class-wise performance when testing on clean images; however, this can be explained by the fact

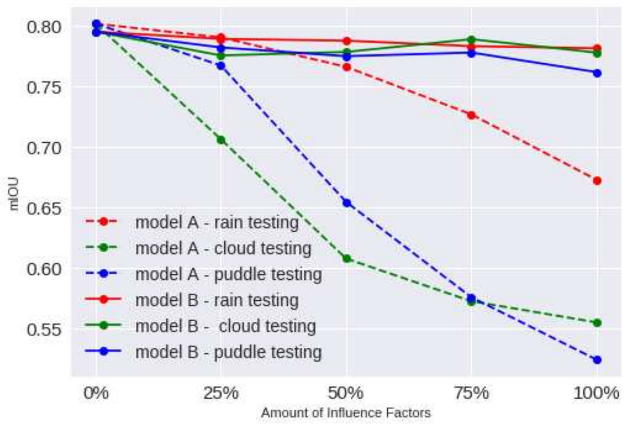

Figure 6. mIoU for each testing scenarios for two models, A and B; x-axis denotes the intensity level of a given influence factor in each scenario

that model B was not trained with as many clean images as model A. Overall, our results show that model B is more robust than model A.

5.2. Understanding the effect of occlusion and distance on network accuracy

The ProcSy dataset allows for the assessment of the quality of a network's prediction with respect to depth and occlusion. Performing this analysis on an existing real-world, autonomous driving dataset such as Cityscapes [6] is difficult since occlusion information is not available. However, this information can allow us to understand more about the reliability and weaknesses of a model.

In this experiment, we randomly choose 270 im-

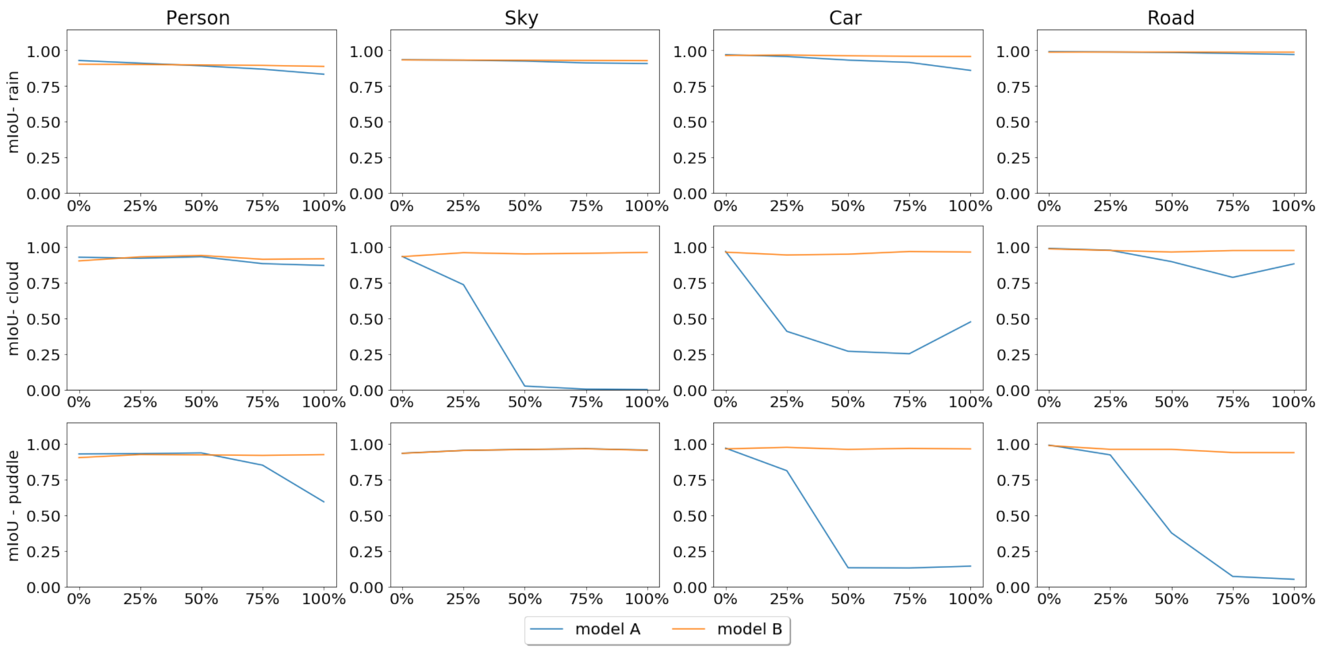

Figure 7. IoU values for 4 classes: person, sky, car and road; each row corresponds to each testing scenario (rain, cloud and puddle) and each column corresponds to each class

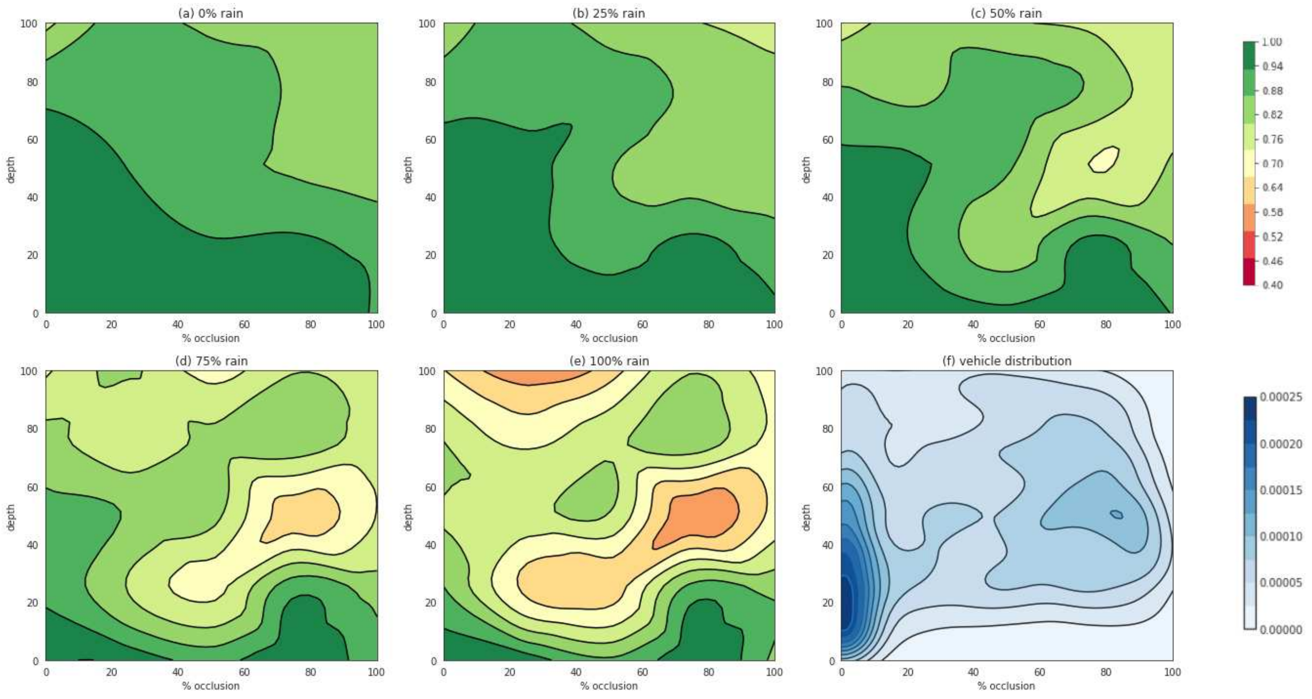

Figure 8. 'a-e' show model A's accuracy on vehicles according to occlusion level and depth maps of the test set; darker green color corresponds to higher accuracy; 'f' shows frequency of vehicles in training set; scales for these plots are shown in color bars at the right. The color bar for the distribution plot represents the distribution density.

ages with a total of 1200 vehicles in the test set. Also, we plot the distribution of vehicles in the training set according to occlusion and depth (Fig. 8f). Similar to Synscapes [26], we divide the predicted pixels into subsets of

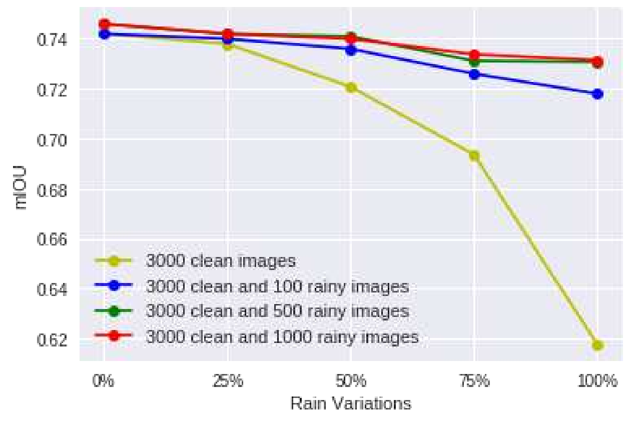

Figure 9. mIoU values for each model of Sec.5.3

$[0\%, 20\%]$, $[20\%, 40\%]$, $[40\%, 60\%]$, $[60\%, 80\%]$, $[80\%, 100\%]$ according to the depth and amount of occlusion of each vehicle. Then, we calculate accuracy for each subset, after which we use cubic spline interpolation to get a contour plot. We repeat the same process for different levels of rain in the images, as shown in Fig. 8.

For the no rain case (Fig. 8a), generally we see that a higher amount of occlusion or depth will lower the model's accuracy. Furthermore, the vehicle distribution shown in Fig. 8f indicates that depth and occlusion effects induce an irreducible source of error. Specifically in Fig. 8f, although the right-most concentrated cluster has higher amount of data than the bottom-most region, the model's accuracy at the right-most cluster in Fig. 8a is still lower than the accuracy in the bottom region where there is little vehicle data.

Also, we observe that in the region bounded by $[0\%, 20\%]$ occlusion and $[0\%, 60\%]$ depth, the model's accuracy is quite stable up until 50% rain amount (Fig. 8c). This region corresponds to the left-most cluster in the distribution map (Fig. 8f). In general, the effects of rain reduces the model's performance significantly, especially for regions (in Fig. 8f) with little data. However, the degradation effect of rain is counter-intuitive in some regions where the accuracies with high amounts of occlusion and depth are higher than regions with lower amounts of occlusion and depth. For example, in Fig. 8e, the model's accuracy in the top-right region is higher than the middle-right region. This is surprising and requires further investigation.

5.3. Estimating the amount of real-world data to be collected

Collecting and labeling more data in different weather conditions is the simplest way to improve model's robustness. However, this is a costly and laborious task, and one may want to know the optimal amount of data to collect. We show in this experiment how the ProcSy dataset can help us estimate this number.

We train and compare the performance of four different models, one with 3000 clean images, and the other three with the same 3000 clean images and an addition of 100, 500, or 1000 rainy images (each additional set containing equal distribution of images at a given level: 25%, 50%, 75%, 100%) respectively. We further note that rainy images are created by randomly taking and modifying weather conditions from the existing 3000 images, so that rain is the only indicative feature that suggests any difference in the performance between these models.

The results are shown in Fig. 9, where we can see that most of the improvement could be obtained by adding just 100 rainy images. For instance, when the amount of rain is 100%, it improves the performance by 10% (from 61.8% to 71.8% mIoU), whereas adding another 900 rainy images only improves performance by an additional 2%. Also, adding 1000 rainy images only gives us a slight increase in mIoU compared to adding only 500 images. Since raindrops cause occlusion effects in the images (a kind of irreducible source of error), this result suggests that adding more rainy images will likely not further improve the model's performance. By performing similar experiments for other influence factors, one can estimate the reasonable amount of real-world data that should be collected to improve the robustness of a given model.

6. Conclusions and Future Work

We presented an approach for rapidly producing a synthetic replica of a real-world locale using procedural modeling in order to generate road scenes with different influence factors and high-quality semantic annotations. We used the approach to create ProcSy, a dataset for semantic segmentation with different influence factors, including varying depth, occlusion, rain, cloud, and puddle levels.

Our experiments show the utility of ProcSy to study the effect of influence factors on the performance of Deeplab v3+, a state-of-the-art deep network for semantic segmentation. In our experiments, variations in puddle and cloud levels affected the networks performance surprisingly more significantly than rain levels. Further, including as little as 3% of rainy images in the training set led to large improvements of the networks robustness on such imagery (by about 10%), whereas adding more than 15% of rainy images showed signs of diminishing returns. This sort of knowledge can be useful to determine how much real-world data for different influence factors to collect. This remains an exploration point for future work.

While this paper studied effects of influence factors individually, future work should explore their combinations. Further exploration remains for understanding correlation of distribution densities of the factors on performance, as well as studying the effect of the factors in real-world data.

References

[1] M. Angus, M. ElBalkini, S. Khan, A. Harakeh, O. Andrienko, C. Reading, S. L. Waslander, and K. Czarnecki. Unlimited road-scene synthetic annotation (URSA) dataset. *CoRR*, abs/1807.06056, 2018. 1, 3

[2] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV (1)*, pages 44–57, 2008. 2

[3] L. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017. 2

[4] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 801–818, 2018. 2

[5] F. Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016. 2

[6] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 2, 6

[7] K. Czarnecki and R. Salay. Towards a framework to manage perceptual uncertainty for safe automated driving. In *SAFECOMP 2018 Workshops, WAISE, Proceedings*, pages 439–445, 2018. 1

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. 2009. 2

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 1

[10] E. ESRI. Shapefile technical description. *An ESRI white paper*, 1998. 4

[11] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. *CoRR*, abs/1605.06457, 2016. 3

[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 3

[13] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008. 4

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[15] D. Hernandez-Juarez, L. Schneider, A. Espinosa, D. Vazquez, A. Lopez, U. Franke, M. Pollefeys, and J. C. Moure. Slanted stixels: Representing san franciscos steepest streets. 2017. 3

[16] Naver. *Proxy Virtual Worlds*. `http://www.europe.naverlabs.com/Research/Computer-Vision/Proxy-Virtual-Worlds`. 3

[17] G. Neuhold, T. Ollmann, S. Rota Bulò, and P. Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *International Conference on Computer Vision (ICCV)*, 2017. 2

[18] Y. I. H. Parish and P. Müller. Procedural modeling of cities. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 301–308, New York, NY, USA, 2001. ACM. 2

[19] M. Pharr, W. Jakob, and G. Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016. 4

[20] P. Prusinkiewicz and J. Hanan. *Lindenmayer systems, fractals, and plants*, volume 79. Springer Science & Business Media, 2013. 2

[21] S. R. Richter, Z. Hayder, and V. Koltun. Playing for benchmarks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2213–2222, 2017. 1, 3

[22] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *European Conference on Computer Vision (ECCV)*, volume 9906 of *LNCS*, pages 102–118. Springer International Publishing, 2016. 3

[23] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. Lopez. The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes. 2016. 3

[24] C. Sakaridis, D. Dai, and L. V. Gool. Semantic nighttime image segmentation with synthetic stylized data, gradual adaptation and uncertainty-aware evaluation. *CoRR*, abs/1901.05946, 2019. 1, 3

[25] F. Tung, J. Chen, L. Meng, and J. Little. The raincouver scene parsing benchmark for self-driving in adverse weather and at night. *IEEE Robotics and Automation Letters*, PP:1–1, 07 2017. 2

[26] M. Wrenninge and J. Unger. Synscapes: A photorealistic synthetic dataset for street scene parsing. *arXiv preprint arXiv:1810.08705*, 2018. 7

[27] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell. BDD100K: A diverse driving video database with scalable annotation tooling. *CoRR*, abs/1805.04687, 2018. 3

[28] S. Zheng, Y. Song, T. Leung, and I. Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4480–4488, 2016. 6