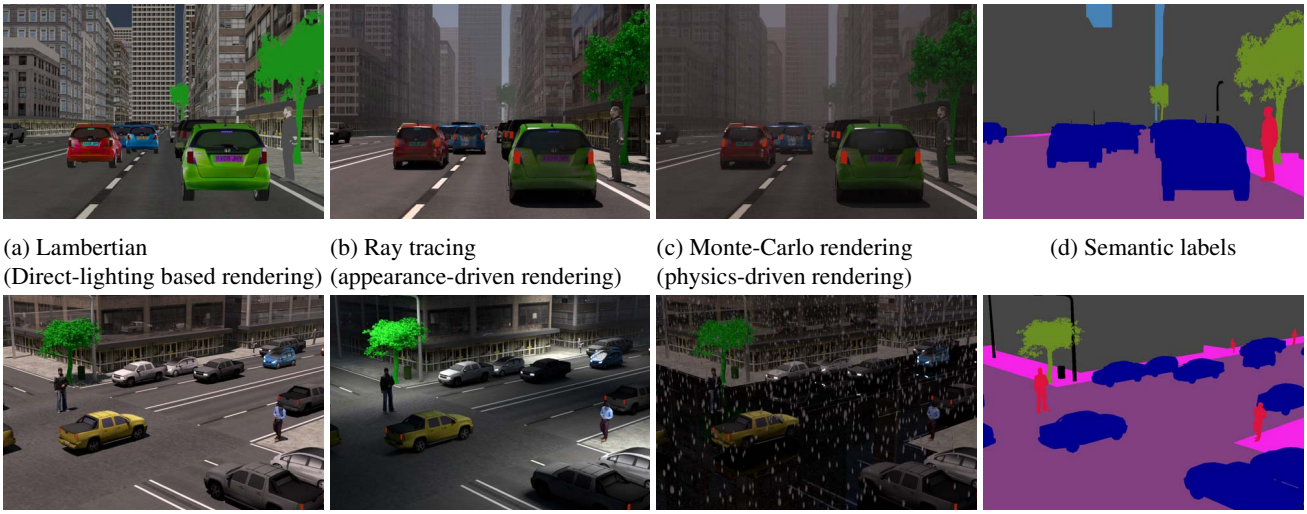
Model-driven Simulations for Computer Vision

VSR Veeravasarapu[1], Constantin Rothkopf[2], Ramesh Visvanathan[1]
[1]*Center for Cognition and Computation, Dept. of Computer Science, Goethe University, Frankfurt*
[2]*Center for Cognitive Science & Dept. of Psychology, Technical University Darmstadt.*



(a) Lambertian (Direct-lighting based rendering) (b) Ray tracing (appearance-driven rendering) (c) Monte-Carlo rendering (physics-driven rendering) (d) Semantic labels

(e) Day light (f) Night (g) Rain (h) Semantic labels

Figure 1: Rendering fidelity and Virtual scene diversity. This work aims to quantify the impact of photorealism and physics fidelity on transfer learning from virtual reality. (a)-(c): Images of same scene state rendered with different rendering engines. (e)-(g): Same scene under different lighting. (d) and (h) semantic labels. Color coding scheme for labels is same as [5].

Abstract

There is a growing interest to utilize Computer Graphics (CG) renderings to generate large scale annotated data in order to train machine learning systems, such as Deep convolutional neural networks, for Computer Vision (CV). However, there has been a long debate on the usefulness of CG generated data for tuning CV systems (even from the 1980's). Especially, the impact of modeling errors and computational rendering approximations, due to choices in the rendering pipeline, on trained CV systems generalization performance is still not clear. In this paper, we take a case study in traffic scenario to empirically analyze the performance degradation when CV systems trained with virtual data are transferred to real data. We: a) discuss a generative model coupled with 3D CAD shapes for scene instance synthesis and, b) explore system performance tradeoffs due to the choice of rendering engine (e.g. Lambertian shader (LS), ray-tracing (RT), and Monte-carlo path tracing (MCPT)) and their respective parameters. DeepLab, that performs semantic segmentation, is chosen as the CV system being evaluated. In our case study, involving traffic scenes, when the CV system is trained with CG data samples (that use MCPT or RT) and augmented with only 10% of real-world training data from CityScapes dataset, the performance levels achieved are comparable to that of training DeepLab with the complete CityScapes dataset. Use of samples from LS degraded the performance of DeepLab by 20%. Physics-based MCPT rendering improved the performance by 6% but at the cost of more than 3 times the rendering time.

1. Introduction

Computational platforms have advanced both the fields of Computer Graphics (CG) and Computer Vision (CV) to a large degree in the last decade. The renaissance of data-hungry models such as Deep Convolutional Neural Networks (DCNN) has renewed the interest in utilizing CG based virtual worlds and rendering methods to generate large scale datasets to bypass the cumbersome task of manually annotating real world data, especially for pixel-

978-1-5090-4822-9/17 $31.00 © 2017 IEEE
DOI 10.1109/WACV.2017.123

wise labeling. CG simulations, along with automatically annotated groundtruth, can be exploited for vision system pipeline modeling, tuning, analysis and understand their trade-offs as a function of scene contexts and system parameters. However, it is not trivial to generate large amounts of high quality data as it often requires considerable human expertise in generating, selecting, and labeling scene states. Here, we develop a simulation platform that stochastically generates annotated data by leveraging recent advances in CG rendering engines, 3D CAD shapes, and probabilistic generative models. A stochastic generative model for traffic settings is developed for sampling. This is our first contribution.

Once CG generated data is available, an open fundamental question is about the utility of such data for tuning or training of CV systems; this is a long debated issue that has focused on the issues of modeling errors and computational approximations (see for instance, early discussions on the use of simulations for vision in late 80s [12, 11]). In the days of hand-engineered designs, explicit assumptions (priors) were at the core of the algorithms. The inputs that obey the assumptions are referred to as ideal inputs. Deviations from these assumptions are considered to be perturbations due to noise sources and modeled and tuned to the given data. Modeling assumptions underlying vision models significantly overlap with the ones that CG based simulations use (see for example, *Lambertian reflection* [20] and *Dichromatic scattering* models [19]). Thus, the CV community has been skeptical to use CG for learning as the data generated was assumed to be ideal or near-ideal inputs to CV and noise models are unrealistic [18]. Several questions that researchers often expressed are related to how photorealism and its physically based rendering approximations impact the generalization of the model trained on simulated data. However, several recent works [29, 33, 7, 24, 23, 25] have taken a fundamental assumption that the modern CG rendering methods are able to simulate visually highly realistic images/videos. Hence, they generate large scale annotated datasets to train modern CV systems, such as DCNNs. Modern training schemes use these annotated datasets along with real data sets to construct classifiers. To the best of our knowledge, it is still difficult to quantitatively understand the balance between virtual training and real data augmentation needed to achieve a given degree of system performance. In other words, the virtual-real data gap has not been quantified yet, even for a single application domain and task pair. In this work, we address this space in the context of semantic segmentation in traffic scenes. This is our second contribution.

Towards this end, we have developed a rendering platform that is integrated with several off-the-shelf rendering engines, ranging from classical to modern, and we generate diverse data with required annotations. A 3D scene instance sample is assumed to be a static scene with object configurations whose parameters are sampled from a probability distribution. The probability model is a spatial Marked Point processes (for 3D scene layouts) with Gibbs potentials (for inter-object constraints, for instance, spatial exclusion). A set of samples that are generated with the developed platform are shown in Figure 1. We use this platform to analyze the impact of modeling errors and computational approximations on the generality of trained DCNN, and to quantify its usefulness to derive conclusions that are likely to transfer to real-world settings. Specifically, we observe the performance (on a real world validation set) of a state-of-the-art system for semantic segmentation, i.e., DeepLab [4] after training it separately on several training sets rendered with different engines. We experiment with a direct lighting based renderer (Lambertian shader), Ray tracer (Mathematically simplified photorealistic method) and monte-carlo path tracer (Physics inspired photorealistic method). We also report results by augmenting the simulated training data with a few real world samples to correct the performance bias.

Related work: Here, we review relevant literature that utilized CG generated data to train CV systems. More exhaustive literature review can be found in our arXiv preprint [31]. This issue of photorealism for virtual-world based training has already been addressed in past work. For instance, [27, 29, 33] used basic rendering algorithms and scene parameters and concluded that simulations alone are not useful for tuning the respective vision systems, where as the work [18] used carefully designed indoor scene models (parameters) and advanced rendering algorithms to synthesize very realistic sensory data and concluded that graphics can be used. The work [3] showed that motion models and local spatial statistics, crucial for optic flow estimation, match with reality. Hence, they argued that the artificial animated (Sintel) data could be used to design and tune the flow estimators even though the data is not photo-realistic. Recently, [6] used a combination of real-data and synthetic objects to train a DCNN and provide empirical evidence of the usefulness of such training for real world settings. The main focus of these works is in the demonstration of the utility or lack of utility of simulation for vision systems design and the emphasis is on evaluation of the system as a black-box in an application context. Still, the degree of effectiveness of graphics rendered data for vision system design is an open question. A discussion [1] on a social network platform clearly conveys the diversity of opinion in the vision community about using graphics for learning. These apparent divergent conclusions can be explained away by the perspective that utility of CG for CV depends on closeness of simulation models to reality and invariant nature of

[1]www.quora.com/How-useful-are-massive-virtual-game-environments-for-training-AI

feature transforms in the system to trained/validated. Recently, racing video game engines, (V-drift, Torcs and H-life etc.), have been used to produce synthetic annotated data for vision for automotive applications. However, these game engines were developed with real time rendering for gaming purposes. In general, most of the effects such as fog scenes, were mathematically simplified [22]. Also, the diversity of environments in the racing engines is also quite limited. Very recently, several independent research groups [7, 24, 23, 25] demonstrated that virtual-world based datasets can be a proxy to real world data to train and validate the vision systems. Our study (on impact of different rendering engines and their parameters on the system) complements the recent work by providing analysis on the impact of photorealism and computational approximations of modern CG renderings on modern CV systems.

Paper organization: Section 2 introduces the rendering processes and the probabilistic models of our platform, which are used to generate the data for our experiments in the traffic scene context. Vision system training procedure and datasets are described in Section 3. Section 4 first analyzes the deviations in the statistics of natural images and synthetic images that are rendered with direct-lighting (Lambertian) and photorealistic methods (Ray tracer and Monte-Carlo renderer). It also addresses the issues of our interest, i.e. impact of parameter choice in rendering process on the model's generalization in reality. Finally, we conclude with some insights from the experiments and relate our work with Systems philosophy in Section 5.

2. Model-driven Simulation Platform

For a systematic exploration, we require a controllable and parametric rendering platform to be able to render the data in the context of interest. To that end, we developed a rendering platform (cf Section 2.1) that (a) integrates the classical and modern rendering methods and, (b) exposes the scene and rendering parameters to the front-end scripting interface. These scene parameters and rendering parameters are then controlled or guided by probabilistic world models (cf Section 2.2) to generate diverse set of images to train or validate vision algorithms.

2.1. Parametric Rendering platform

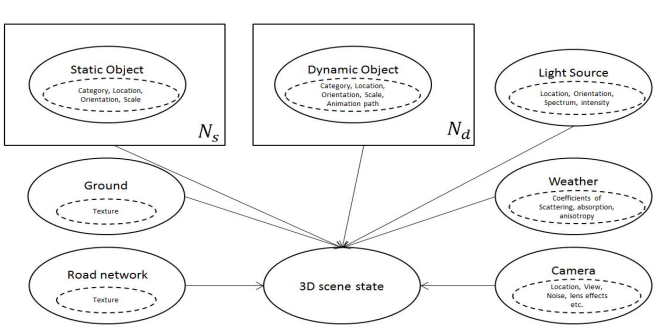
Here, we briefly discuss the rendering methods used in our experimentation, which range from local to global illumination methods and appearance-driven to physics-driven methods. The main goal of the rendering methods is to solve rendering equation [15]. There are many classes of rendering methods, depending on the details that rendering equation models (direct vs indirect lighting based methods) and the ways (assumptions and simplifications) in which they solve rendering equation (appearance-based vs physics-based methods). Direct-lighting based rendering schemes (e.g. Lambertian shaders) as well as Global illumination methods (e.g. Ray-tracing and Monte-Carlo Path Tracing) are utilized in our experiments. *MCPT* is a more physics grounded, but computationally intensive approach to rendering. *Ray tracing* is faster and more efficient, but it uses mathematical heuristics for the effects of global illumination/caustics. Additional information on these methods can be found in [31]. An open source rendering engine, *Blender* [1] meets several of our requirements. We integrated the other rendering and annotations methods using it's *Node* and *Scripting* interfaces.

2.2. Generative Models

Automatic generation of virtual scene states is an essential task, as manual creation of large scale diverse environments requires a lot of man hours from many designers. (Semi) automatic large content creation with procedural grammars has been attempted in [21] and a recent survey of these methods can be found in [26]. However, they might produce highly correlated scenes for two runs of the systems as scene generation is based on given axioms and deterministic logic. Probabilistic models are powerful way to describe the subjective priors and constraints on scene layout of man-made and natural objects. For example, the spatial arrangement of objects (for examples, buildings and trees) in a city can be described by a set of probabilistic constraints. Though several probabilistic scene models exist, we select a model of urban street scenes that leverages the existing 3D CAD object shapes to create large scale 3D layouts.

In this section, we restrict our goal to model urban traffic scene environments by formulating their space of layouts as probabilistic distributions and processes. Modeling scene geometry and object shape is getting easier due to the availability of large-scale 3D CAD shape repositories such as Google's 3D warehouses etc. We collected a variety of CAD shapes and textures from the web, and place them in a world's coordinate system according to the realizations of *Marked Poisson Processes* (*MPP*). Here, we first introduce the concepts of *MPP* which are underlying elements of our scene geometry model.

MPP for Scene Geometry A prior knowledge about spatial/temporal distributions of a set of objects can be modelled with stochastic processes such as point processes [14]. A realization of the point process consists of a random countable set of points $\{o_1, ..., o_n\}$ in a bounded region $\mathbf{O} \in R^d$. A marked point process couples a spatial point process Z with a second process defined over a *mark* space M such that some random marks or attributes $m_o \in M$ is associated with each point $o \in Z$. For example, a 3D marked point process of cuboid marks has elements of the form $o_i = (p_i, [c_i, l_i, b_i, h_i, \theta_i])$ specifying the location (p), object class (c), dimensions (l,b,h) and orien-



(a) Semantic diagram for our virtual world model: rectangle boxes represent MPP, N_s and N_d denote the number of static and dynamic objects respectively in the virtual world

(b) A few samples from our CAD repository

Figure 2: Virtual world models

tation (θ) of a specific bounding box (cuboid) in the scene. In this work, we use MPP to incorporate our prior knowledge about the spatial patterns in city geometries (extrinsic size and transformation of objects such as buildings and vehicles etc). Thus, the realization of the MPP in this work consists of an object location p defined on a bounded subset of R^2 (xz ground-plane), together with a mark m defining type and a 3D CAD shape to import and place at point p. In other words, we model the objects in the scene as a set of configurations from an MPP that incorporates prior knowledge such as expected sizes of buildings, trees, people and vehicles on the scene of knowledge about the scene regions where these objects will not appear. We denote the prior term for an object as $\pi(o_i)$, and assume independence among the objects. The mark process is assumed as independent from the spatial point process, so that priors in MPP would be factored as: $\pi(o_i) = \pi(p_i)\pi(m_i)$. However, this common approach ignores obvious and strong correlations between the size and orientation of objects. Hence, a conditional mark process is introduced for cuboids representing shape and orientations of a 3D bounding box, conditioned on the location and shape, leading to a factored prior of the form:

$$\pi(o_i) = \pi(l_i, b_i, h_i, \theta_i | c_i, p_i)\pi(c_i)\pi(p_i) \quad (1)$$

The prior for object class (type such as building, tree, pedestrian and vehicle classes etc) distributions are modelled with uniform distributions. This means object type follows a uniform distribution, and given object type, the shape dimensions (3D bounding box) are *i.i.d.* One can also bootstrap the domain models by learning the parameters through training using real world data.

Conditional mark process: We represent priors for $\pi(l_i, b_i, h_i, \theta_i | p_i, c_i)$ as simple parametric distributions (such as uniform and Gaussian) on the bounded regions. For example, the heights of pedestrian and vehicle are modelled with normal distributions with means 1.6 and 1.7meters respectively. θ_i (orientation with which object should be place on the ground or road) is allowed to vary from -180 to 180 degrees with uniform distribution.

3D CAD shapes and Textures as Marks. The object shapes (for given object identity) are randomly selected from the 3D CAD repositories and resized according to the sampled (l, b, h). We have collected a rich set of 3D models for each object category (buildings, grounds, pedestrians, and vehicles etc) from the web[2]. Some of the 3D CAD shapes are shown in Figure 2b. These 3D models are indexed according to their object category. However, parametric 3D object shape can be modelled with the distributions [9] such as Boltzmann machines [32] or Bernoulli mixture distributions [8]. Inter-object constrains such as spatial non-overlap and mutual alignment are incorporated with the help of Gibbs potentials between marks. In such case, point process is called as Poisson process [17] and more details are provided in [31].

3. Datasets and Training

Semantic Segmentation: In this work, we select a state-of-the-art system, i.e. DeepLab [4], and evaluate virtual-reality-based-training for semantic segmentation task in traffic scene context. Approximations made in virtual world models and rendering processes influence the statistics of rendered outputs, and, thus, biases the learned classifier. Here, we conduct experiments to study the following: (a) Effects of Photorealism: Here we compare results obtained from training data using Lambertian Shaders against photorealistic methods such as Ray-tracing and Monte-Carlo Path tracing. (b) Physics Fidelity: Here we compare results obtained by training with Ray-tracing vs Monte-Carlo Path tracing (MCPT). For instance, the *mcpt* datasets are physically more realistic (see [31] for contrast between mcpt

[2]3dwarehouse.sketchup.com, 3dmodelfree.com, quality3dmodels.net, tf3dm.com

and ray-tracing methods) than ray_trace, though they both look realistic to human-eye. (c) Impact of Computational Approximations: Here we evaluate the variation in performance as a function of number of monte-carlo samples in MCPT.

The experiments analyze the following issues: (a) how photorealism and computational-approximation (physically accuracy of photorealistic effects) influences the performance of the model on real world data? (b) how biased virtual-world-based-training is, compared to real-world-based-training? (c) when and where virtual-world data is more reliable? (d) how much real data is needed to correct the performance bias? Though, we work with a single DCNN-based architecture, our approach and insights are transferable to other data-driven systems and tasks as we treat the vision system as a blackbox in this work.

Data preparation: We simulate 5000 images sampled from the world model (described in Section 2.2) and rendered with different rendering settings, along with pixel-wise object labels (the classes include: vehicle, pedestrian, building, vegetation, road, ground, and sky) . To measure the impact of computational-approximation, we render these images lambertian shader, ray tracer and mcpt with increasing computational-approximation (spp) ranging 10 to 130 with a step size of 30 ($spp = 10 : 131 : 30$ $python\ notation$). It results in seven image datasets of same scene states. We use the terms *"lambertian"*, *"ray_trace"*, and *"mcpt_X"* to denote the sets rendered with lambertian shader, ray tracer and path tracer respectively, where X being number of *spp* used in rendering.

Real-world datasets: For comparison purposes, we use a real world dataset, CityScapes[5] which is recorded on the streets of several European cities. It provides a diverse set of videos with a public access to 3475 images (train-val) that has finer pixel-level annotations for semantic labels. We divide the database into two disjoint subsets for training, validation (3000 images, named as *CS_train*) and testing (475 images, named as *CS_val*) purposes. This dataset was adapted to the 7 class labels mentioned above.

DeepLab architecture: DeepLab [4] is an modified version of VGG-net to operate at original image resolutions, by making following changes: (a) replace the fully connected layers with convolutional ones, (b) skip the last subsampling steps and upsample the feature-maps by using *Atros* convolutions. It still results coarser map with a stride of 8 pixels. Hence, targets (semantic labels) during training are the ground truth labels subsampled by 8. During testing, bilinear interpolation followed by fully connected conditional random field (CRF) was used to get final label maps. We modify the last layer of DeepLab from 21-class to 7-class (including: vehicle, pedestrian, building, vegetation, road, ground, and sky).

Training: Our models are initialized with ImageNet pre-trained weights to skip longer training times. Stochastic gradient descent method and cross-entropy loss function are used with initial learning rate of 0.001, momentum of 0.9 and a weight decay of 0.0005. We use mini-batch of 4 images and learning rate is multiplied by 0.1 after every 2000 iterations. High-resolution input images are down-sampled by a factor 4. Training data is augmented by vertical mirror reflections and random croppings from the original resolution images, which yields four times the data. As a stopping criteria, we use a fixed number of SGD iterations (50,000) in all our experiments. In the CRF postprocessing, we use fixed parameters in the CRF inference process (10 mean field iterations with Gaussian edge potentials as described in the work [4]) in all reported experiments. The CRF parameters are optimized on a subset of 300 images, randomly selected from the dataset.

4. Experiments and Results

Input Image statistics: We start with comparing the image statistics of datasets to know how the computational-approximation propagates to input space for vision system. Here, we discuss about KL divergence between probability distributions of Gabor filter responses. The corresponding plot and other comparisons of natural image statistics are provided in [31]. We compute the Gabor responses of each set with Gabor filters of 31x31 size, but with varying scale (σ) parameters ($scale = 3 : 9 : 1$ $python\ notation$). We plot KL divergence of Gabor filter responses between simulated datasets and real world *CS_train* dataset against filter scales. For both sets ray_trace and $mcpt$, the divergence from reality seems to be decreasing with the scale. That means simulations are more deviated in local statistics than global averages. Moreover, difference between ray_trace and $mcpt$ sets is also decreasing with increasing scales. How do these deviations in input statistics propagate through the training stage of DeepLab?

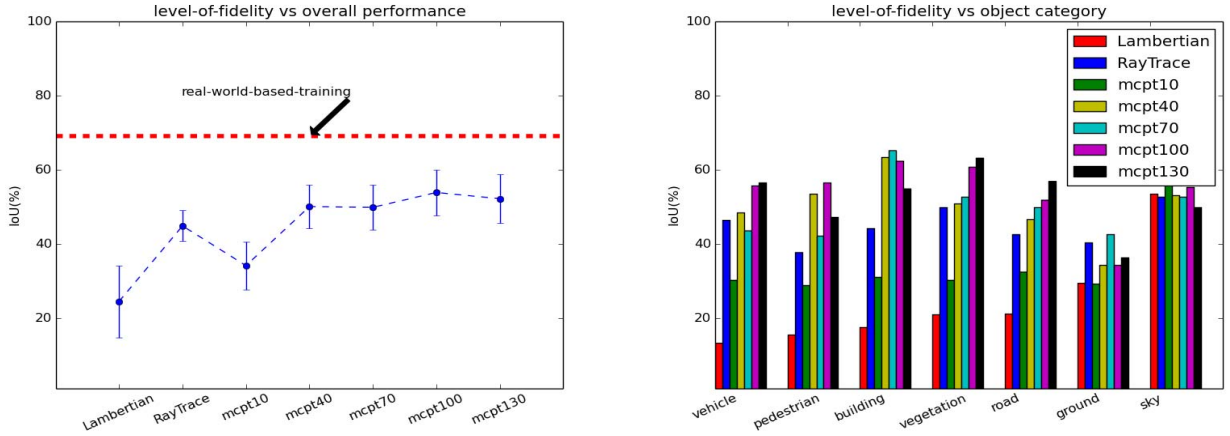
Performance of Virtual-world-based-training: In order to evaluate the role of photorealism and fidelity on the performance of the model trained on virtual-world data, we train the DeepLab independently on our simulated datasets. We then compare the performances of these trained models on a reference real world dataset, i.e. CityScapes validation set (*CS_val*).

As shown in Figure 4a, we plot the corresponding IoU [3] measures to show how the performance of trained model on *CS_val* varies with levels of realism and fidelity in simulated training data. In all configurations, training with simulations seems worse compared to real-world-based-training (red dotted line in the figure). This bias could be due to

[3]To assess performance, we use a standard performance metric, known as the intersection-over-union metric, $IoU = \frac{TP}{TP+FP+FN}$, where TP, FP, and FN are the numbers of true positive, false positive, and false negative pixels, respectively, determined over the whole validation set.

Table 1: Comparing the performance of DeepLab when trained with simulated datasets of different computational-approximation and rendering methods

Training dataset	Validation dataset	mean IoU	vehicle	pedestrian	building	vegetation	road	ground	sky	RenderTime-per-image (in sec)
lambertian	*CS_val*	24.46 ± 9.69	13.19	15.41	17.58	20.99	21.2	29.35	53.52	0.001
ray_trace	*CS_val*	44.81 ± 4.15	46.36	37.72	44.13	49.88	42.58	40.27	52.75	20
mcpt_10	*CS_val*	34.10 ± 6.44	30.19	28.82	31.12	30.22	32.48	29.26	56.65	5
mcpt_40	*CS_val*	50.00 ± 5.88	48.49	53.39	63.34	50.78	46.66	34.27	53.09	34
mcpt_70	*CS_val*	49.82 ± 6.04	43.57	42.17	65.26	52.73	49.85	42.57	52.59	67
mcpt_100	*CS_val*	53.82 ± 6.14	55.8	56.45	62.36	60.83	51.77	34.37	55.21	313
mcpt_130	*CS_val*	52.15 ± 6.58	56.61	47.26	54.98	63.08	56.98	36.31	49.86	547
Variance		± 5.56	± 5.08	± 5.10	± 3.25	± 5.1	± 3.06	± 2.85	± 1.46	NA



(a) mean IoU vs rendering settings

(b) per-class IoU vs rendering

Figure 4: Performance (IoU on *CS_val*) variations due to photorealism and computational-approximation of rendering

sampling diversity (and its lack of closeness to reality) in the virtual world models and computational-approximation in the simulations. We only focus on differences in rendering approximations below.

Impact of Photorealism: The IoU of DeepLab trained with simulated set, *Lambertian* (diffuse reflections only) is worse than the other choices. It is clear that diffuse material assumptions in simulations does not hold in most real world conditions. When we used the *Cook-Torrance* shader that simulates both diffuse and glassy reflections the performance was worse. One possible explanation could be that the deep learned classifier system is not invariant to physical reflections. We therefore deactivated glassy reflections for this study. When we used more photorealistic (*ray_trace* and *mcpt*) sets with global illumination effects, the performance of trained model has been improved drastically (nearly gets double). Thus, it appears that photorealism is important for DCNN performance. But, how does the physical fidelity of these photorealistic effects impact the performance?

Impact of Physics fidelity: To know how physics fidelity in lighting computation impact the trained model, we compare the performance between these datasets (see Table 1). *mcpt_10* seems to be bad compared to rt. however, 10 spp is too less to get a photorealistic image and it suffers with heavy sampling noise. Hence comparing with $mcpt\_10$ may not be fair. By comparing the performance of *ray_trace* and other *mcpt_X* ($X \geq 40$) sets, it appears that physics fidelity seems to be improving the performance a little bit at the cost of high rendering times. For example, on an average, the performance has been improved by 6% when trained with *mcpt* sets instead *ray_trace* set, but at the cost of more than 10 times the rendering time of ray tracer (last column in the table). So, we believe that physics-accuracy of lighting computations of photorealistic effects is also important to some extent. However, it is a trade-off between large computational resources and little performance gains due to physics.

Impact of computational-approximation (spp): We now examine the effect of MC sampling parameter (spp) in MCPT rendering method. Figure 4a is intended to show the variations in IoU due to photorealistic rendering and computational-approximation. The plot seems to be more or less flat for $mcpt$ datasets, especially after $spp = 40$. The performance for $mcpt$ sets varies between IoU $51.8 \pm 5.66\%$, mainly due to rendering noise in the simulations.

From this experiment, we conclude that DCNN's seem to be less sensitive to computational-approximation of physics in simulated training dataset used in our application context and that 40 *spp* were enough for achieving the good performance in our experiments. Hence, accuracy seems to be important to some extent, but, large number of samples (*spp*) may not be necessary. This insight can help us to reduce the time required for rendering the data and be spent at some other means of performance improvement.

Things vs Stuff: Figure 4b shows per-class IoU measures for all rendering settings, while the last row in Table 1 contains variances of per-class IoU measures of training settings with *mcpt_X*, $X \geq 40$. One interesting observation from these numbers is that the computational-approximation does not seem to effecting the system much in classifying the pixels of sky, ground, road (which are considered to be "Stuff[4]" in the semantic segmentation literature(please refer to Things-and-Stuff model[13]), while most of the differences are from the objects such as pedestrian, vehicles, buildings (which are considered to be "Things" [13]). The reason we think is that the things (vehicles, pedestrians etc.) tend to have more diverse complex shapes and textures which may require more detailed CAD models, compared to the stuff (building, ground, ans sky etc.). This observation from object level analysis inspired us to analyze and locate the errors at pixel and region level.

Location of Major errors: We inspect to locate regions where virtual-world-based training is more deviated and not reliable enough. From the analysis in the input image statistics, the simulated data might differ more at high frequency contents due to fidelity. We would like to see how this deviation propagates through DeepLab training.

We approach this by analyzing the performance at object boundaries with the help of trimaps [16]. Trimaps are masks with the pixels that are located within a narrow band of object boundaries, as shown in Figure 5a. We create trimaps of varying pixel-widths for all images of *CS_val*, and compute IoU only at those pixels in the white band region. Figure 5b show how performance changes from boundaries to more global spatial contexts. One can observe that the performance deviates more near boundaries (lower values of trimap's pixel-widths) than that over entire image space. We postulate that this may be due to the same reason that the higher frequency contents more deviate from reality in simulated sets. The difference between simulated and real sets can be explained by the fact that the object boundaries and shadows in virtual worlds are quite sharp while real world boundaries have effects of color bleeding and penumbra (due to sensor effects). So, modeling sensor and lens effects in simulations may be important to reduce the statistical deviations in pixel-level tasks such as semantic labeling.

Transfer learning and Data combinations: The performance of virtual-world-based-training was always biased, compared to the setting of real world training (see the difference between blue curve and red line in Figure 4a). Given the insights from our experiments on effect of computational-approximation, a possible explanation for performance bias could be the level of diversity in the virtual world models to capture real world geometric variations. Recall that we only modelled Manhattan-like geometric variations with limited number of CAD shapes in virtual world. Level-of-diversity that a dataset with limited samples captures, is a question, even to real world datasets. For instance, the model trained on *CS_train* is achieving the performance levels of 69.54% on *CS_val* (which is from same domain), while a performance degradation of 16% has been observed when it is tested on another validation set, $CamVid\_val$ [2] (captured in a geographical location different from *CS_val*).

To see how far we can reach with only simulated data, we increase the diversity of the simulated data by augmenting our data *mcpt_40* with 5000 samples from another simulated data set, *Synthia* [24]. *Synthia* is a recent large scale simulated data generated and used in the work [24]. Surprisingly, the DCNN when trained on this integrated dataset (of 10,000 images) has produced a model with more generalization capabilities, than the one trained on *CS_train* (see Table 2). This model consistently performs on the both *CS_val* (68%) and CamVid_val (69%), while the other model (trained on *CS_train*) seems to overfit *CS_val* (67%) and biased on CamVid_val (54%). This shows that data diversity improves the generalization capabilities of vision models. For the sake of fair comparison, we also provided the performance of the model trained of total real world data available (*CS_train*+*CamVid_train*) on both validation sets (see the table).

Like many works [28, 30, 10, 7, 24, 23] already reported, we also found that real world samples added to the simulated data can correct the performance bias of the trained model. In this set of experiments, we add 10% real world data samples of the training data from the target domain (*CityScapes* or *CamVid*) and retrain the system. The quantitative results are shown in Table 2. These IoU values are on par with (sometimes better than) the levels that are achieved with full training data. This can significantly reduce the number of real world samples needed at training/development phase.

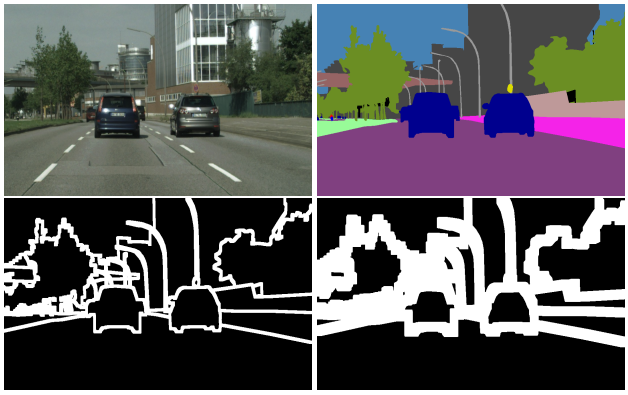
5. Discussion and Conclusion

In this work, we addressed the space of virtual-to-reality gap in the context of semantic segmentation in traffic

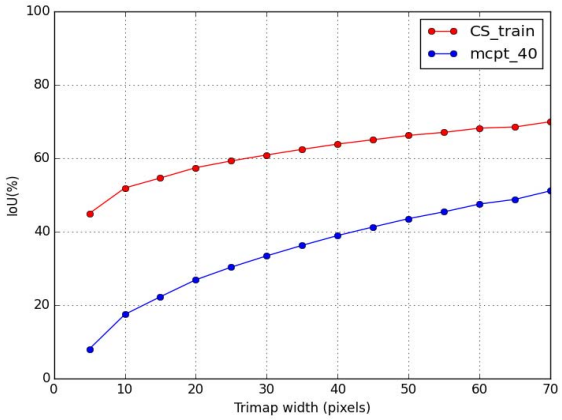
[4]The objects that has no limited spatial extents are considered as *Stuff*, such as road, ground, sky etc., while *Things* represent the objects whith limited size, such as pedestrians and vehicles.

Table 2: Data combinations and Generality of the system

Training	Validation	Global	vehicle	pedestrian	building	vegetation	road	ground	sky
CS_train	*CS_val*	67.54	58.92	57.04	72.92	63.7	68.79	63.78	87.64
CS_train	*CamVid_val*	54.29	47.33	42.58	54.84	68.67	45.52	51.25	69.89
mcpt_40	*CS_val*	50	48.49	53.39	63.34	50.78	46.66	34.27	53.09
mcpt_40	*CamVid_val*	39.37	52.97	28.14	34.26	35.18	42.73	19.47	62.85
mcpt_40 + 10% *CS_train*	*CS_val*	67.21	60.1	65.95	51.85	66.68	73.41	71.61	80.91
mcpt_40 + 10% *CamVid_train*	*CamVid_val*	68.93	51.44	60.12	71.22	66.67	65.03	77.91	90.12
CS_train + *CamVid_train*	*CS_val*	69.52	62.03	59.56	77.16	67	72.16	59.33	89.42
CS_train + *CamVid_train*	*CamVid_val*	75.18	70.42	79.25	86.67	66.75	79.53	74.79	68.87
mcpt_40 + *Synthia*	*CS_val*	68.81	68.36	66.39	64.21	72.44	67.08	68.55	74.7
mcpt_40 + *Synthia*	*CamVid_val*	69.05	65.27	67.47	72.76	66.94	66.82	68.22	75.89



(a) Input image and corresponding labels and trimaps of 10 and 30 pixel-width



(b) IoU vs Trimap width

Figure 5: Major erroneous locations: Major errors are located around object boundaries

scenes. We developed a set of required tools for stochastic virtual world generation and scene rendering, that could be used as a systematic experimental bench for the context of urban scene understanding. We also empirically validated the virtual-world-based-training and provided several insights about the impact of photorealism and fidelity on DCNNs. Although it seems to be essential to have all photorealistic effects in the data, DCNNs are not too sensitive towards physics accuracy of the effects. Hence, apparently extreme levels of photorealism may not be necessary. High frequency contents (such as image boundaries), smaller objects (such as pedestrians and vehicles etc.) are relatively more deviated compared to other stuff. However, one can take a special care at these effects at the cost of time. The major role in performance bias is from level-of-diversity of the virtual world model to capture real world variations. In this aspect, virtual data suffers with a dataset shift problem just like any other real world dataset. However, this shift can be corrected by adding a few real world samples to training data. In our experiments just 10% real world dataset was enough to reach the levels of full real world training. This significantly reduces the number of real world samples required at development phase. A large scale diverse virtual dataset (ours+Synthia) seems to be good enough to produce the models with more generalization compared to currently available real world benchmarks in the field (CityScapes+CamVid).

However, transferability of our insights depends on many factors and variables in the experimental pipeline that include: (i) Generative models (G) and Simulation methods used, (ii) Rendering techniques (P) used to generate observations, (iii) Real world data (D), (iv) Algorithm or System being Evaluated (S), (v) Criterion Function or metric for evaluation (L). Thus, (G, P, D, S, L) form a joint space of consideration for the experiment. Our choices in this experimentation can be seen as a specific example from this space. In the data driven learning paradigm, the system S is learned from a set of examples D and little consideration is given for the nature of G. Moreover, preprocessing and data augmentation techniques implicitly use the nature of G and/or P (e.g. rescaling, whitening, application of geometric distortions to generate virtual samples) to augment D during the training process.

Acknowledgements

This work was supported by the German Federal Ministry of Education and Research (BMBF) in the projects, 01GQ0840 and 01GQ0841 (BFNT Frankfurt).

References

[1] http://www.blender.org/.

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

[3] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *Computer Vision–ECCV 2012*, pages 611–625. Springer, 2012.

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[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. *arXiv preprint arXiv:1604.01685*, 2016.

[6] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015.

[7] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. *arXiv preprint arXiv:1605.06457*, 2016.

[8] W. Ge and R. T. Collins. Marked point processes for crowd counting. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2913–2920. IEEE, 2009.

[9] P. Guan. *Virtual Human Bodies with Clothing and Hair: From Images to Animation*. PhD thesis, Brown University, Department of Computer Science, Dec. 2012.

[10] V. Haltakov, C. Unger, and S. Ilic. Framework for generation of synthetic ground truth data for driver assistance applications. In *German Conference on Pattern Recognition*, pages 323–332. Springer, 2013.

[11] R. Haralick. Performance characterization in computer vision. 60(2):245–249, September 1994.

[12] R. M. Haralick. Methodology for experimental computer vision. In *Computer Vision and Pattern Recognition, 1989. Proceedings CVPR'89., IEEE Computer Society Conference on*, pages 437–438. IEEE, 1989.

[13] G. Heitz and D. Koller. Learning spatial context: Using stuff to find things. In *European conference on computer vision*, pages 30–43. Springer, 2008.

[14] M. Jacobsen. *Point process theory and applications*. Springer, 2006.

[15] J. T. Kajiya. The rendering equation. In *ACM Siggraph Computer Graphics*, volume 20, pages 143–150. ACM, 1986.

[16] P. Kohli, P. H. Torr, et al. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.

[17] F. Lafarge, G. Gimel'Farb, and X. Descombes. Geometric feature extraction by a multimarked point process. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1597–1609, 2010.

[18] S. Meister and D. Kondermann. Real versus realistically rendered scenes for optical flow evaluation. In *Electronic Media Technology (CEMT), 2011 14th ITG Conference on*, pages 1–6. IEEE, 2011.

[19] S. G. Narasimhan and S. K. Nayar. Vision and the atmosphere. *International Journal of Computer Vision*, 48(3):233–254, 2002.

[20] M. Oren and S. K. Nayar. Generalization of the lambertian model and implications for machine vision. *International Journal of Computer Vision*, 14(3):227–251, 1995.

[21] Y. I. Parish and P. Müller. Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 301–308. ACM, 2001.

[22] M. Pharr and G. Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2004.

[23] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. *arXiv preprint arXiv:1608.02192*, 2016.

[24] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3234–3243, 2016.

[25] A. Shafaei, J. J. Little, and M. Schmidt. Play and learn: Using video games to train computer vision models. *arXiv preprint arXiv:1608.01745*, 2016.

[26] R. M. Smelik, T. Tutenel, R. Bidarra, and B. Benes. A survey on procedural modelling for virtual worlds. In *Computer Graphics Forum*, volume 33, pages 31–50. Wiley Online Library, 2014.

[27] T. Vaudrey, C. Rabe, R. Klette, and J. Milburn. Differences between stereo and motion behaviour on synthetic and real-world stereo sequences. In *Image and Vision Computing New Zealand, 2008. IVCNZ 2008. 23rd International Conference*, pages 1–6. IEEE, 2008.

[28] D. Vázquez, A. López, D. Ponsa, and J. Marin. Cool world: domain adaptation of virtual and real worlds for human detection using active learning. In *Advances in Neural Information Processing Systems–Workshop on Domain Adaptation: Theory and Applications*, 2011.

[29] D. Vazquez, A. M. Lopez, J. Marin, D. Ponsa, and D. Geroimo. Virtual and real world adaptation for pedestrian detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(4):797–809, 2014.

[30] D. Vázquez, A. M. López, and D. Ponsa. Unsupervised domain adaptation of virtual and real worlds for pedestrian detection. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3492–3495. IEEE, 2012.

[31] V. Veeravasarapu, C. Rothkopf, and V. Ramesh. Model-driven simulations for deep convolutional neural networks. *arXiv preprint arXiv:1605.09582*, 2016.

[32] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shape modeling. In *Proc. CVPR, to appear*, volume 1, page 3, 2015.

[33] J. Xu, S. Ramos, D. Vázquez, and A. M. López. Domain adaptation of deformable part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(12):2367–2380, 2014.