Silver: Novel Rendering Engine for Data Hungry Computer Vision Models

Abdulrahman Kerim*
Lancaster University
Lancaster, UK
a.kerim@lancaster.ac.uk

Leandro Soriano Marcolino
Lancaster University
Lancaster, UK
l.marcolino@lancaster.ac.uk

Richard Jiang
Lancaster University
Lancaster, UK
r.jiang2@lancaster.ac.uk

Figure 1: Sample scene generated using *Silver*.

ABSTRACT

Large-scale synthetic data is needed to support the deep learning big-bang that started in the recent decade and influenced almost all scientific fields. Most of the synthetic data generation solutions are task-specific or unscalable while the others are expensive, based on commercial games, or unreliable. In this work, a new rendering engine called *Silver* is presented in detail. Photo-realism, diversity, scalability, and full 3D virtual world generation at run-time are the key aspects of this work. The photo-realism was approached by utilizing the state-of-the-art High Definition Render Pipeline (HDRP) of the Unity game engine. In parallel, the Procedural Content Generation (PCG) concept was employed to create a full 3D virtual world at run-time, while the scalability of the system was attained by taking advantage of the modular approach followed as we built the system from scratch. *Silver* can be used to provide clean, unbiased, and large-scale training and testing data for various computer vision tasks.

CCS CONCEPTS

• **Computing methodologies** → **Rendering**; **Computer vision**.

KEYWORDS

synthetic data, computer vision, computer graphics

1 INTRODUCTION

When a problem can be described by a limited set of rules, computers perform very well and usually surpass human performance in terms of speed and accuracy. However, when the problem is too hard to be formulated, computers fail dramatically. Human reasoning and intelligence are still very far from being understood or formulated. The vast research done in this field still only scratches the surface [4, 14, 15]. Visual perception is one of the most complex tasks that the human brain performs accurately and efficiently. As visual perception is important for human daily activities, it is critical for the machine (agent) to perceive the environment, reason, plan, and then interact to achieve its goal. The machine could be as simple as an Optical Character Recognition (OCR) program or as complex as an autonomous car or robot on another planet. Failure of such agents could lead to deaths or injuries, damage to the environment or properties, failure of missions, and loss of millions of dollars.

The recent great success of Artificial Neural Networks (ANN) in solving complex problems motivated many researchers to apply it to machine perception problems too. In parallel to this, the great advancement in chip design, microelectronics, and the introduction of General-Purpose Graphics Processor Architectures like the General Purpose Graphics Processing Unit (GPGPU), facilitated training deep neural networks with millions of parameters. These deep ANNs with their high degree of non-linearity, help in approximating the real functions or phenomena behind complex vision problems (like semantic segmentation, instance segmentation, and object recognition) in a much more accurate way. Unfortunately, training these deep learning models requires a great amount of data together with their corresponding annotations or ground-truths. Finding, collecting, and annotating suitable data is cumbersome, time-consuming, error-prone, expensive, and subject to privacy issues. Perhaps, the lack of diverse, high quality, and precisely labeled data can be attributed to the previously mentioned reasons. Unfortunately, these factors cause many major data quality issues in the

*Corresponding author.

field of computer vision and they clearly present an obstruction toward the aim of optimal performance computer vision models in practice.

The promising solution seems to be in the leading-edge game engines like Unity [23], Unreal Engine [7], and CryEngine [6]. Leveraging the powerful tools of the Unity game engine, we present a system that creates three-dimensional, photo-realistic virtual worlds, procedurally at run-time. A sample scene generated by our rendering engine is shown in Figure 1. The system currently supports various computer vision tasks such as semantic segmentation, instance segmentation, depth estimation, and others. It allows researchers, with no computer graphics background, to generate large-scale synthetic datasets for training or testing their own computer vision models. Our main contributions can be summarized as follows:

- To the best of our knowledge, *Silver* is the pioneer work to use HDRP of Unity game engine to build a 3D virtual world at run-time for synthetic data generation.
- Unlike other frameworks, *Silver* considers data generation as an online problem which opens the door for using synthetic data with online learning algorithms.
- This is the first work that considers generating a photo-realistic, full-city, procedurally and at run-time.

2 RELATED WORK

With the substantial advancements in deep learning based approaches, we have witnessed unprecedented progress in computer vision. This progress is attributed to the large-scale benchmark datasets that were collected in the past few years [5, 9, 12, 13]. Although the exponential increase in the amount of digital data today makes data collection easier than before, manual labelling of large volumes of examples with high quality and accurate labels still requires too much effort and comes with a tremendous cost. The utilization of synthetic data in the computer vision field has just started recently. Its increase in popularity started to attract many researchers to propose novel algorithms to generate such datasets with their corresponding ground-truths. Although special care needs to be taken to weigh each method's advantages and disadvantages, they seem a promising solution to overcome the lack of suitable data for training supervised learning models. In this section, we summarize the mainstream of synthetic data generation.

Adapting a specific video game to generate synthetic data with its corresponding ground-truth for the task of semantic segmentation was shown by Richter et al. (2016) [17], where the game Grand Theft Auto V was modified for that purpose. At the same time, another work by Shafaei et al. (2016) [21] investigated utilizing photo-realistic video games to generate synthetic data and their corresponding ground truths for image segmentation and depth estimation. Using open source animation movies was another method discussed by Butler et al. (2012) [2] where they were able to obtain an optical flow large-scale dataset, MPI-Sintel, following a systematic and easy process.

Unfortunately, the previous methods present a partial solution for the data generation issue because of the lack of control on the environment elements. At the same time, integrating new elements or behaviours to the scene like including a new 3D model, material or texture is extremely hard or simply unattainable. Another issue is the limitations of the proposed systems to specific computer vision tasks. Although these approaches are based on high quality and rich 3D virtual worlds, the failure of such methods to randomize the scene elements will lead to some clear repetitions (to scene elements) when a large-scale dataset is required to be generated by these methods. Procedural synthetic data generation offers an alternative to the previous solution.

Procedural Content Generation (PCG) has been proposed as a solution for creating realistic looking environments in relatively short amounts of time, making it easier and cheaper for users to generate virtual worlds from scratch. In its simplest form, a procedural generation framework follows systematic recipes to generate scenes, populations, and actions, based on the given set of instructions. *Silver*, the rendering engine described in this work, is based on this concept.

Under this category, De Souza et al. (2017) [18] investigated the possibility of adapting the PCG concept with Ragdoll physics, random perturbations and muscle weakening to generate a wide range of human actions systematically with their corresponding labels. Another work by Cheung et al. (2016) [3] applied the concept of procedural generation to generate labelled crowd videos. Alternatively, Wrenninge et al. (2018) [24] presented a photorealistic and diverse synthetic dataset that can be generated entirely procedurally. The ability to parameterize the scene generation process and the fact that these parameters are not correlated are the main contributions of [24]. While procedural concept improves diversity, it still limits the scalability of the system and it does not guarantee the photo-realism of the virtual world being generated by it.

An optimal solution to the problem is the one that ensures high realism, diversity, scalability, controllability and most importantly the generalizability of the approach to all computer vision tasks.

3 SYNTHETIC DATASETS AND COMMON LIMITATIONS

The current available computer vision synthetic datasets seem to be covering a wide set of the main computer vision applications like Visual Object Tracking, Semantic Segmentation, Instance Segmentation, Depth Estimation, Action Recognition, and Optical Flow. Some of these datasets provide videos [2, 8, 18, 19] while others [11, 17, 20] provide simply snapshots depending on the application or the task. The resolution of these synthetic datasets is restricted by the method used to generate it. Some of the methods allow only fixed resolutions [16, 20] while others permit generating the dataset

Table 1: Synthetic datasets for computer vision tasks.

Name	CV Tasks	Size	Videos	Resolutions	Other Useful Info
Synscapes [22, 24]	Instance and semantic segmentation Depth Map, 2d and 3d bounding boxes Camera's position and field of view Occlusion and truncation Scene metadata	25K Frames	No	1440X720 2048X1024	Unbiased path tracing for rendering and Monte Carlo-based light transport simulation
Virtual Kitti [8]	Object detection and multi-object tracking Scene-level and instance-level semantic segmentation Optical flow, and depth estimation	21,260 Frames	Yes	1242X375	Unity 5 environments
Synthia [19] SYNTHIA-Rand & SYNTHIA-Seqs	Semantic segmentation Depth Map Car ego-motion	213K Frames 200K Frames	No Yes	960X720	Unity Virtual New York City 13 classes
PHAV [18]	Action recognition Semantic segmentation Instance segmentation Depth map Vertical and horizontal optical flow	6M Frames	Yes	340X256	39,982 videos for more than 1,000 examples for each action of 35 categories
GTA-V [17]	Semantic segmentation	25K Frames	No	1914X1052	Grand Theft Auto V Compatible with CamVid and CityScapes
MPI-Sintel [2]	Optical flow	1628 Frames	Yes	1024X436 (Any)	Effect of resolution on Optical Flow Based on the open-source animated film Sintel
SOMASet [1]	Person reidentification	100K Frames	No	400X200	Makehuman and Blender
LCrowdV [3]	Bounding box, flow estimation, pedestrians count Crowd density, population, lighting conditions Background scene, camera angles Agent personality and noise level	20M Frames	Yes	Any	Unreal game engine Based on procedural modeling and rendering techniques
VIPER [16]	Optical flow, semantic instance segmentation, Object detection and tracking Object-level 3d scene layout Visual odometry	254K Frames	Yes	1920X1080	Grand Theft Auto V Inject a middleware between the game and its supporting graphics library
UBC3V [20]	Single or multiview depth-based pose estimation	6M Frames	No	512X424	Utilize Kinect sensors
PreSIL [11]	Depth information, point clouds Semantic segmentation 2d and 3d labels for object detection	50K Frames	No	1920X1080	LiDAR simulator within Grand Theft Auto V

at various resolutions [2, 3]. Important characteristics of synthetic datasets are as follows:

Diversity: Synthetic datasets can be used for training and testing purposes. For training, diversity is needed to avoid over-fitting to the visual features of the synthetic world. On the other hand, for testing, diversity is required to represent a good proxy of the real world and the possible standard and rare scenarios. The real training and testing data are collected with the aim to be a good proxy of the actual world. Sometimes the space of the problem could be easy to describe and to collect suitable data that represents – approximately – this space. Unfortunately, predicting the problem space for most of the computer vision tasks is extremely hard or simply unfeasible. Thus, datasets with the highest diversity are always expected to improve the overall performance.

Visual Complexity: The real world is full of details that are very hard to be captured even by the state-of-the-art simulators or rendering engines. However, generating synthetic data at a high-level of visual complexity is very important. Moreover, some specific details could be more critical for some computer vision tasks while not for some others. Thus, special care is needed in deciding the level of details the synthetic data should attain and which scene elements are more important than others.

Realism: The ultimate goal of models trained on synthetic data is to be tested on the real-world dataset. However, the most common problem that causes such models to fail or not to perform well is the domain gap (domain shift) problem. Many different approaches are suggested to mitigate this problem. However, improving the photo-realism of synthetic data is, nevertheless, the major approach. It should be noted that realism should not only be limited to photo-realism. However, the concept should be extended to animation, camera motion, object locations in the scene, object frequency, object scales, dynamic objects interactions with the static elements and other dynamic objects, and so on.

The limitations of most of the observed synthetic datasets shown in Table 1 is the lack of adverse weather conditions and limited times of the day (hence, illumination). In addition to that, the camera behaviours are quite limited, unrealistic, or cinematic. These problems in fact simplify the actual issue, and cause computer vision models trained on these datasets to over-fit to these situations.

4 SILVER FRAMEWORK

Using the Unity game engine, we built a system that is able to generate a full city procedurally, randomly, at run-time and with high degree of photo-realism. The city is populated with humans, cars, trees, birds and so on. The system runs at around 30 Frames Per Second (FPS) on Ubuntu 18.04 with a GeForce GTX 1080 Ti GPU. It consists of four main components as described in Figure 2.

Scene Generator: Starting from the given parameters, *Silver* initially creates the static part of the 3D virtual world. In this part, first the street length and the number of crosses are set at random. Following this, the buildings are created where buildings' locations, types and frequency are set at random. After that, the other scene elements like benches, trash containers and bags, trees, and other elements are created. Once the static part of the scene is completed, the dynamic part of the scene is initiated. Initially, the characters generator retrieves the locations of buildings and benches, and instantiates characters based on the required characters density. The Microsoft Rocketbox Avatar Library [10] is used to define the character avatar, and the animations are selected based on character pose (standing or sitting). Character animations were adopted from Mixamo[1]. In a similar way, the cars are created. However, number of cars and models are selected at random. Additionally, car shader attributes, namely, *Smoothness*, *Metallic* and *BaseColour*, are all randomized at run-time to give different visual appearance even to the same car model. After that, the plates of the cars are selected at random from a set collected manually from the web. The above main processes are summarized in Figure 3.

Camera Controller: The camera unit is made to be independent of the other scene elements to support the scalability of the system. *Silver* includes two different camera types. One simulates a camera placed on an Unmanned Aerial Vehicle (UAV) or drone. While the second imitates a person carrying a camera and recording others. *Cinemachine* was utilized for that reason since it gives unlimited sets of behaviours that enrich the diversity of the generated synthetic data in terms of the camera view angle, transition, and many others.

Weather and Time Unit: The Unity game engine supports creating *Skybox*, which is a 6-sided cube that is rendered behind all scene objects. We deployed the High Dynamic Range Imaging (HDRI) images from HDRIHAVEN[2] to create two realistic sets of skyboxes. One incorporated the day-time skyboxes from which one skybox at random will be chosen if the simulation time is a day-time. While the second includes the night-time skyboxes from which one will be selected at random when the night time is being simulated. For the time simulation, other actions are taken to increase photo-realism and enrich the diversity. For example, the locations of the sun and the moon are changed randomly. The skybox is rotated at random and the sky emission is set at random as well. After that, the street lights are turned on and off based on the time of the day. The main processes involved in simulating the time-of-day are shown in Figure 4.

The weather condition component is essential in simulating the main conditions that could present some challenges for deep learning models. *Silver* currently allows researchers to generate four different weather conditions, namely, sunny, foggy, rainy and snowy. A parameter is used to specify the severeness (slight, very, extreme) of the weather condition. However, the weather is randomized within the chosen severeness level in a range that is still meaningful and reflects the severeness degree. The aim was to add further diversity even for a given weather severeness level.

After that, for rainy and snowy weathers, the Unity particle system is used to simulate both snow and rain drops. The density of these drops per unit volume is controlled by the severeness level. At the same time, the ground shader is changed to simulate water drops collision or to imitate snow accumulation on the ground. Following this, trees animations are changed (speed up or slow down) based in the severeness of the weather being simulated to give a sense of wind interaction with trees. The foggy weather condition is supported by the Unity games engine. Thus, only simple parameters tuning was required to simulate realistic foggy weather condition.

[1] https://www.mixamo.com/
[2] https://hdrihaven.com/

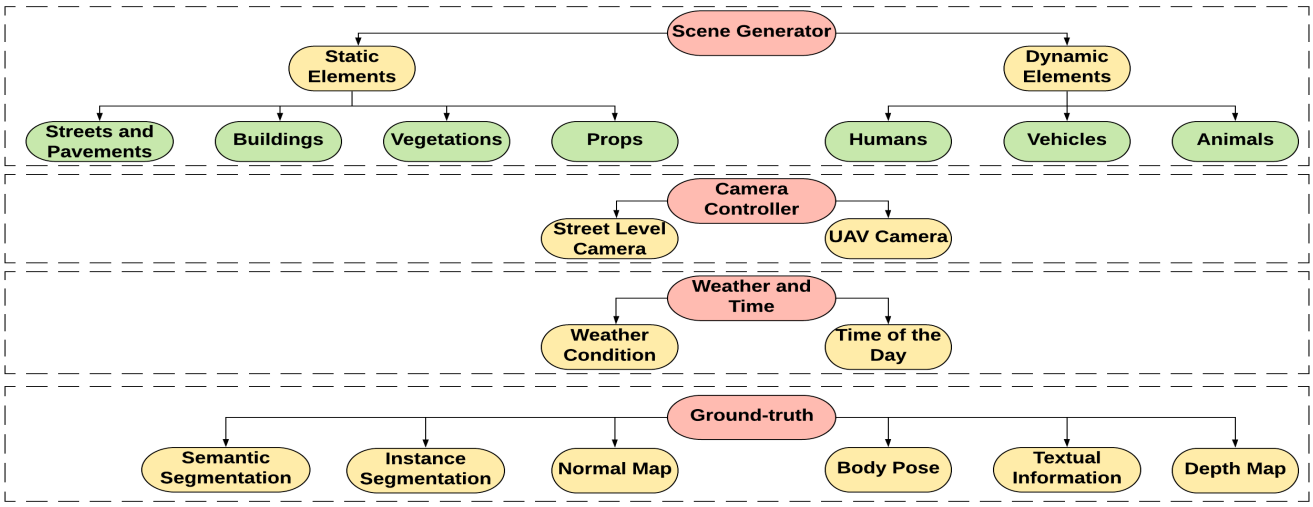


Figure 2: Bird's-eye view of the system architecture.

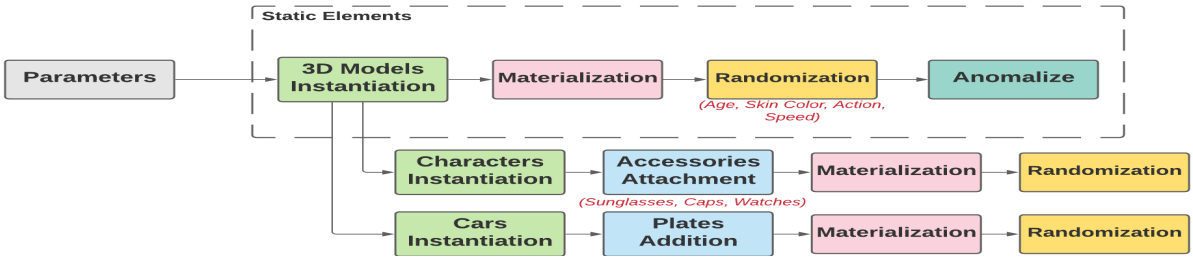


Figure 3: Flowchart describing the scene creation.

The main processes of simulating the four weather conditions are shown in Figure 4.

Ground-truth Extractor: One of the main goals of *Silver* is to generate synthetic data with annotations for different computer vision tasks.

A custom rendering pass and a simple shader are defined to extract the *depth maps* for each frame. The vertex position and texture coordinates are retrieved then converted to screen space. After that, the depth information is linearized and scaled. *Normal maps* are simply extracted using graph shader and a custom pass volume. The world space normal vector is retrieved and encoded as the base colour for the lit shader.

For *instance segmentation*, the objects being created in the scene are retrieved and for each object its mesh is drawn again for each frame.

For *semantic segmentation*, however, each category (class) of objects are given a different tag at the creation process. Then, all objects with one tag are given the same colour. It should be noted that Unity supports around 10K tags. Thus, the number of classes can reach 10K which is much larger than the number of semantic classes in most of the datasets (order of 10).

For the *human body pose* information, the avatar is accessed at each frame and the main 14 joints coordinates are provided. The world space locations of these joints are transformed to screen space locations and saved for each character.

5 SYSTEM EVALUATION

The system was evaluated both qualitatively and quantitatively. In Section 5.1 the quality and the variation of the generated images are discussed. On the other hand, time complexity and memory utilization are addressed in Section 5.2.

5.1 Diversity and Photo-realism

To avoid over-fitting to the visual features of the synthetic world, the content of the environment is diversified by including a wide set of 3D models, textures, animations, weather conditions, illumination and lighting, and recording set-ups. A sample of the key variations *Silver* currently supports is illustrated in Table 2. In parallel to that, example frames from the supported weather conditions and times-of-the-day are shown in Figure 5.

Silver utilizes HDRP of Unity game engine to develop a photo-realistic 3D virtual world for synthetic data generation. HDRP is based on the Scriptable Rendering Pipeline (SRP). Generally, it is intended for high visual fidelity applications. The other key feature of HDRP is the Physical Light Unit (PLU) that relies on

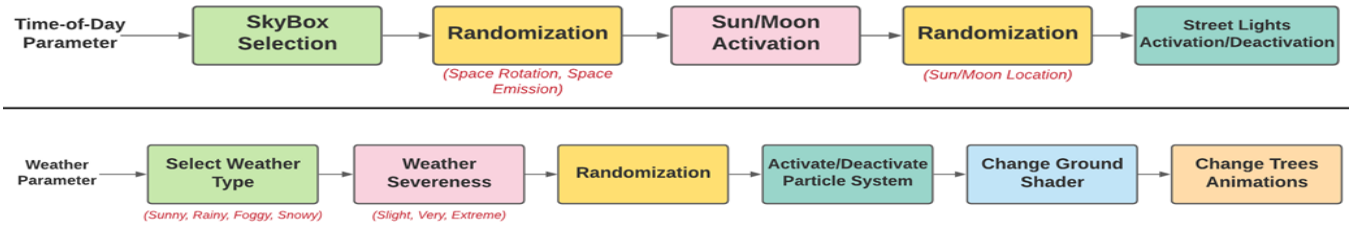


Figure 4: Time-of-day and weather simulation flow charts.

Table 2: Variations for some of the main attributes the proposed engine supports.

Attribute	Variations	Attribute	Variations	Attribute	Variations	Attribute	Variations
People	Gender - Age Colour - Ethnicity Height - Speed Actions	Building	Shops Condominiums	Weather	Normal Rainy Snowy Foggy	Human Accessories	Glasses Hats Hand watches
Car	Model Colour Reflectiveness Plates	Street	Length Traffic Lights Cross Locations Cross Numbers	Time of the Day	Day time Night	Vegetation	Trees Plants
Sky	Sun/Moon/Stars Location Stars Contrast Clouds Location	Lighting	Traffic Lights Street Lights	Prop	Waste Bin Waste Bags Benches	Animal	Birds

real-life lighting measurable values. All of these attributes together contribute to the final photo-realistic rendering that is shown in Figure 6.

Utilizing HDRP for the purpose of synthetic data generation is one of the key contributions of this work. As shown in Figure 7 the synthetic data generators PHAV and Synthia generate less photo-realistic synthetic data as compared to *Silver* and Synscapes. Non-realistic weather simulation, low polygon 3D models, and unrealistic characters and vehicles are crucial issues behind the lack of photo-realism. An addition to this, the core problem is utilizing a non-physically based rendering pipeline which makes the interaction of the light with the materials clearly unrealistic. On the other hand, the main advantage of our system in comparison with Synscapes is the capability of *Silver* to simulate both different weather conditions and diverse night-times. Additionally, *Silver* diversifies buildings types, characters accessories, animations, age and skin color.

Table 3 details some key features of *Silver* as compared to other synthetic data generation systems. At the same time, it shows that these systems present a partial solution to the problem. Most of them are task specific, achieve diversity but neglect photo-realism or vice versa. For these reasons, *Silver* is developed to present a good solution for the synthetic data generation task.

5.2 Scalability

One main goal of this work is to make the system scalable and extendable to a range of different problems. In other words, adding new 3D models for human characters, buildings, trees, cars, or new animations, or textures, or camera setup should be straightforward. At the same time, extending the system to support new computer vision tasks shall not require major modifications to the other system components. This was achieved by following a modular approach in building *Silver* where each component is independent of the unrelated ones.

To dissect the complexity of the system, the following six components were analysed: Triangles, Vertices and Batches Count, CPU Time, FPS, and SetPass Calls. The first three represent the total number of triangles, vertices and batches (static, dynamic, and instance types) Unity processes for a single frame. CPU Time illustrates the time required to process and render a single frame. FPS, however, provides the number of frames rendered per second. The higher the value, the smoother the system will run. On the other hand, SetPass Calls shows how many times Unity changes the shader pass as it renders one frame.

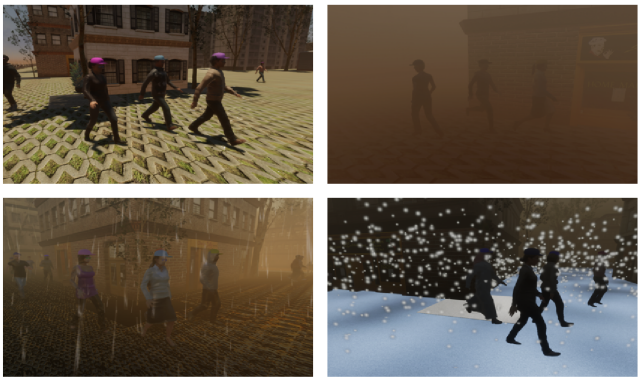
Figure 8 shows the average and standard deviation values for these six metrics taken for 5 iterations. Scalability was studied for four crowdedness levels, namely Few (N≤20), Moderate (N∼50), High (N∼100), and Extreme (N∼300). N represents the number of generated pedestrians. Figure 8 clearly shows that the number of Triangles, Vertices, Batches, and SetPass Calls all are independent of the system load, i.e. generating more characters. FPS decreased and CPU time increased as expected since more characters were generated.

5.3 Sample Dataset

Our system automatically provides ground-truth for depth estimation, normal map, semantic segmentation, instance segmentation and body pose estimation. Figure 9 shows one RGB frame with its corresponding ground-truths for the mentioned computer vision tasks. In addition to other textual information describing the time of the day, the weather condition, and other useful information.

Table 3: A comparison between *Silver* features and three state-of-the-art synthetic data generators, namely, Synscapes [22, 24], Synthia [19], and PHAV [18].

Simulator	Computer Vision Tasks					Weather Conditions				Times-of-Day		Public Availability	Photo-realism
	Semantic Segmentation	Instance Segmentation	Depth Estimation	Surface Normals Estimation	Pose Estimation	Normal	Rainy	Foggy	Snowy	Day-time	Night-time	/	/
Synscapes [22, 24]	✓	✓	✓	-	-	✓	-	-	-	✓	-	-	✓
Synthia [19]	✓	-	✓	-	-	✓	✓	-	✓	✓	✓	-	-
PHAV [18]	✓	✓	✓	-	✓	✓	✓	✓	-	✓	✓	-	-
Silver (ours)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓



(a) Different weather conditions

(b) Different times-of-the-day

Figure 5: Sample frames from sequences rendered at different weather conditions (a) and times-of-the-day (b).

To show one particular usage of *Silver*, a sample dataset was generated and released publicly. The dataset contains sample sequences spanning the main supported attributes. The ability to generate different scenes and the process of sampling models, materials, and animations are thought to minimize the visual similarity even among a large number of sequences.

6 DISCUSSION

In the scope of applying synthetic data for computer vision tasks, the data diversity is of a central concern. *Silver* deploys uniform distribution to set colors, select 3D models, choose animations or to configure other scene elements. In reality this may be considered as a strong assumption. To address this, we are planning to consider the real distribution of scene elements and their associated attributes as observed in the real world. At the same time, we would like to make these distributions conditioned on the type of weather condition and other related aspects of the environment. For example, in a winter environment, it is more likely to observe people wearing dark color clothes and so on.

In this work, a qualitative overview of the system was shown. However, an open question is still to prove the usability of the generated data for training and/or evaluation purposes. As a future work, we plan to answer this question considering some major computer vision tasks such as semantic segmentation and depth estimation. In parallel, we aim to extend the system to other computer vision tasks and improve the photo-realism and diversity even further. We believe that utilizing *Silver* to generate photo-realistic, large-scale, and diverse training data will help computer vision models to train better and to achieve better performance on real world data.

7 CONCLUSIONS

In this paper, we presented a tool called *Silver*, that allows researchers to generate 3D photo-realistic virtual worlds procedurally and at run-time. No computer graphics knowledge is required, no privacy issues involved, and no manual annotation is needed. The system is released for research usage and researchers are welcomed to adopt it to their own needs. A sample dataset was generated to show the capability of the system.

This work was not meant yet to provide a proof of the usability of the synthetic data the system is able to generate. However, it was dedicated to representing the system, its main features, design decisions and reasons behind them. As a future work, we are planning to study the usability of our generated synthetic data for different computer vision tasks. *Silver* can be utilized to generate full datasets with rare and challenging attributes. The automatic annotation process ensures accuracy. At the same time, it provides unbiased ground truth since no human is involved in the annotation process.

The source code of the complete system and a sample dataset are both available at https://github.com/lsmcolab/Silver. At the same time, a video displaying a sample 3D virtual world generated by *Silver* is provided at https://youtu.be/Ktlx5bgJLXE.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their constructive feedback that helped to improve the work. We thank Washington L. S. Ramos and Samuel Hardy for proofreading the paper.

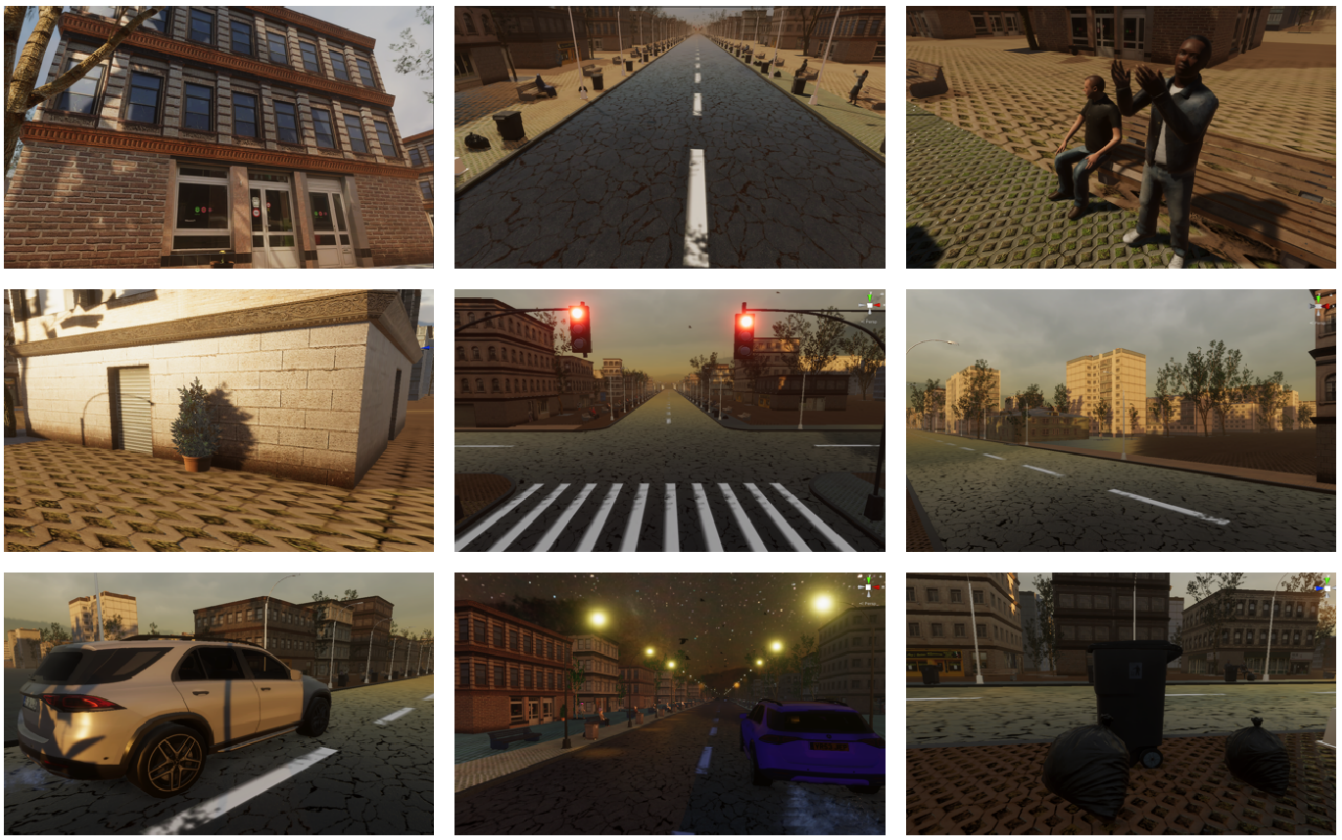


Figure 6: Photo-realism of *Silver*

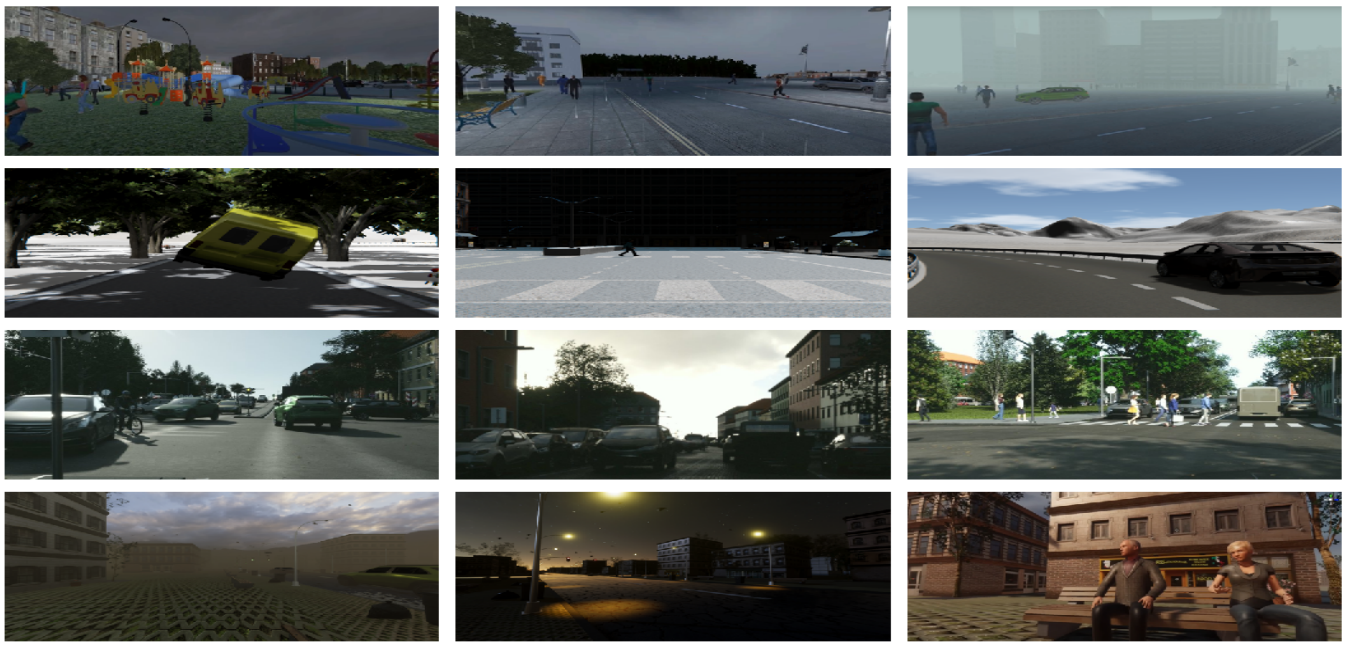


Figure 7: Visual comparison between *Silver* and three synthetic data generators in terms of photo-realism. First three rows show PHAV [18], Synthia [19], and Synscapes [22, 24], respectively, while final row presents our system *Silver*.

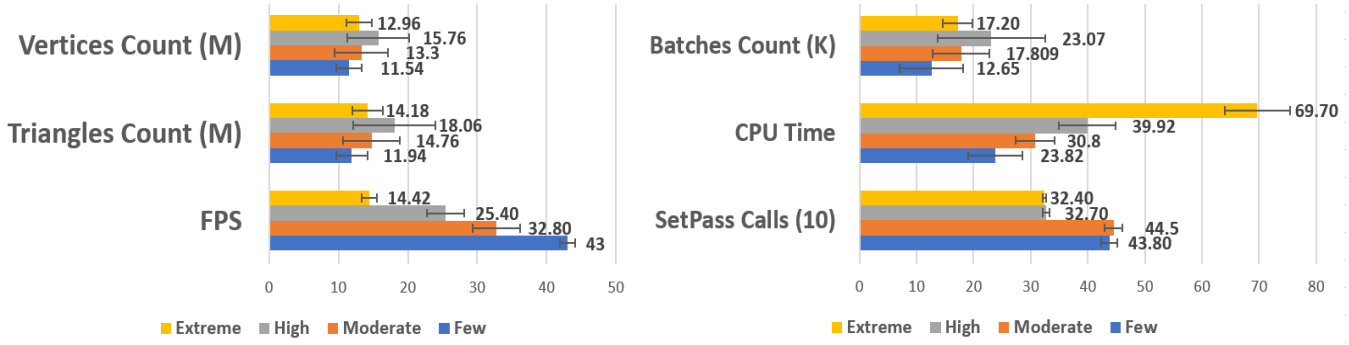


Figure 8: System complexity when varying number of characters.

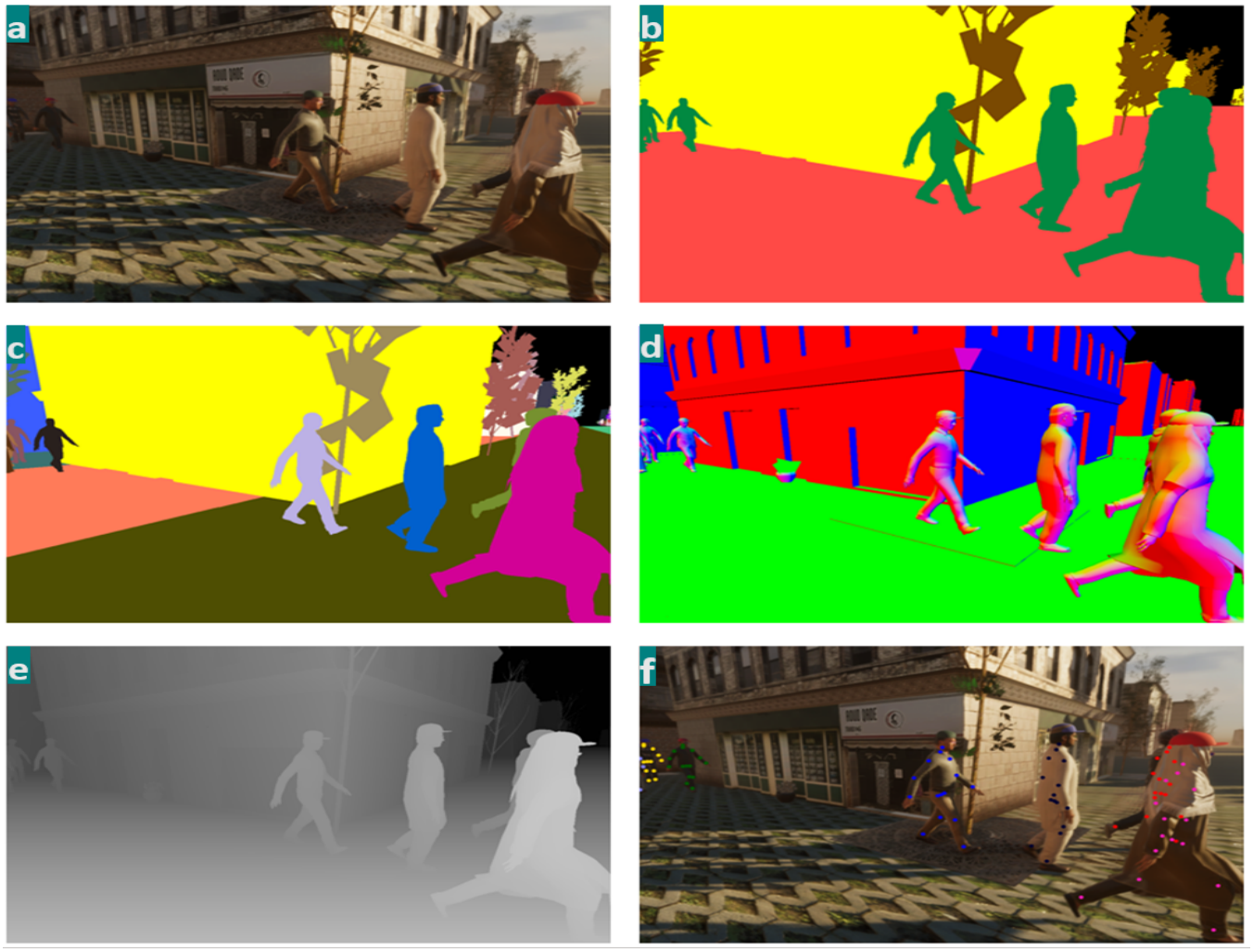


Figure 9: *Silver* provides RGB frames (a) with pixel-level accurate ground-truths for semantic segmentation (b), instance segmentation (c), normal maps (d), depth maps (e), and human body pose information (f).

REFERENCES

[1] Igor Barros Barbosa, Marco Cristani, Barbara Caputo, Aleksander Rognhaugen, and Theoharis Theoharis. 2018. Looking Beyond Appearances:Synthetic Training Data for Deep CNNs in Re-identification. *Computer Vision and Image Understanding* 167 (2018).

[2] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. 2012. A Naturalistic Open Source Movie for Optical Flow Evaluation. In *European conference on computer vision*.

[3] Ernest Cheung, Tsan Kwong Wong, Aniket Bera, Xiaogang Wang, and Dinesh Manocha. 2016. LCrowdV: Generating Labeled Videos for Simulation-Based Crowd Behavior Learning. In *European Conference on Computer Vision*.

[4] Roberto Colom, Sherif Karama, Rex E Jung, and Richard J Haier. 2010. Human intelligence and brain networks. *Dialogues in clinical neuroscience* 12, 4 (2010).

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *2009 IEEE conference on computer vision and pattern recognition*.

[6] Cry Engine. [n.d.]. https://www.cryengine.com/. Online; accessed: 2021-01-20.

[7] Unreal Engine. [n.d.]. https://www.unrealengine.com/en-US/. Online; accessed: 2021-01-20.

[8] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. 2016. Virtual Worlds as Proxy for Multi-Object Tracking Analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[9] Yuying Ge, Ruimao Zhang, Xiaogang Wang, Xiaoou Tang, and Ping Luo. 2019. DeepFashion2: A Versatile Benchmark for Detection, Pose Estimation, Segmentation and Re-Identification of Clothing Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[10] Mar Gonzalez-Franco, Ofek, et al. 2020. The Rocketbox Library and the Utility of Freely Available Rigged Avatars. *Frontiers in virtual reality* 1, article 561558 (2020).

[11] Braden Hurl, Krzysztof Czarnecki, and Steven Waslander. 2019. Precise Synthetic Image and LiDAR (PreSIL) Dataset for Autonomous Vehicle Perception. In *IEEE Intelligent Vehicles Symposium*.

[12] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning Multiple Layers of Features from Tiny Images. *University of Toronto, Tech. Rep.* (2009).

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *European conference on computer vision*.

[14] John Lisman. 2015. The Challenge of Understanding the Brain: Where We Stand in 2015. *Neuron* 86, 4 (2015).

[15] David Papo. 2015. How can we study reasoning in the brain? *Frontiers in human neuroscience* 9 (2015).

[16] Stephan R Richter, Zeeshan Hayder, and Vladlen Koltun. 2017. Playing for Benchmarks. In *Proceedings of the IEEE International Conference on Computer Vision*.

[17] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. 2016. Playing for Data: Ground Truth from Computer Games. In *European conference on computer vision*.

[18] Cesar Roberto de Souza, Adrien Gaidon, Yohann Cabon, and Antonio Manuel Lopez. 2017. Procedural Generation of Videos to Train Deep Action Recognition Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[19] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. 2016. 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*.

[20] Alireza Shafaei and James J Little. 2016. Real-Time Human Motion Capture with Multiple Depth Cameras. In *2016 13th Conference on Computer and Robot Vision*.

[21] Alireza Shafaei, James J Little, and Mark Schmidt. 2016. Play and Learn: Using Video Games to Train Computer Vision Models. *arXiv:1608.01745* (2016).

[22] Apostolia Tsirikoglou, Joel Kronander, Magnus Wrenninge, and Jonas Unger. 2017. Procedural Modeling and Physically Based Rendering for Synthetic Data Generation in Automotive Applications. *arXiv:1710.06270* (2017).

[23] Unity. [n.d.]. https://unity.com/. Online; accessed: 2021-01-20.

[24] Magnus Wrenninge and Jonas Unger. 2018. Synscapes: A Photorealistic Synthetic Dataset for Street Scene Parsing. *arXiv:1810.08705* (2018).