2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

WorldGen: A Large Scale Generative Simulator

Chahat Deep Singh[1], Riya Kumari[1], Cornelia Fermüller[1], Nitin J. Sanket[2], Yiannis Aloimonos[1]

***Abstract*— In the era of deep learning, data is the critical determining factor in the performance of neural network models. Generating large datasets suffers from various challenges such as scalability, cost efficiency and photorealism. To avoid expensive and strenuous dataset collection and annotations, researchers have inclined towards computer-generated datasets. However, a lack of photorealism and a limited amount of computer-aided data has bounded the accuracy of network predictions.**

To this end, we present WorldGen – an open source framework to automatically generate countless structured and unstructured 3D photorealistic scenes such as city view, object collection, and object fragmentation along with its rich ground truth annotation data. WorldGen being a generative model gives the user full access and control to features such as texture, object structure, motion, camera and lens properties for better generalizability by diminishing the data bias in the network. We demonstrate the effectiveness of WorldGen by evaluating deep optical flow. We hope such a tool can open doors for future research in a myriad of domains related to robotics and computer vision by reducing manual labor and cost for acquiring rich and high-quality data.

I. Introduction

High-quality image data is pivotal for accurate deep learning models. For certain tasks, high-quality data is more important than the choice of neural architecture and hyperparameters. Data collection, cleaning, and annotation have become a nightmare lately sometimes costing millions of dollars. This is mostly due to challenges in data diversity, image quality, pixel-perfect manual annotations, licensing [1], and security concerns. It is exacerbated by the fact that writing instructions for human labelers for data labeling is not a trivial task because it is not always possible to explain concisely the requirements. To this end, we propose a universal framework to generate countless 3D scenes of different environments along with annotated images for various tasks of autonomous cars, drones, and indoor robots. This would enable deep learning frameworks to gather photorealistic data much more efficiently in terms of cost, speed, labor, and variety.

On the other side of the spectrum, synthetic data and simulators today are commonly used but are very limited as they suffer from *scalability* and/or *photorealism*. Researchers and practitioners have extensively used these data as a benchmark to evaluate depth, optical flow, segmentation, etc. because of the *perfect* ground truth annotation. Most of these simulators do not support a realistic camera model or ray-tracing ability to render high-quality photorealistic images. Although, few approaches have proven to narrow down the generalization gap using synthetic data [2], [3] for geometric-based quantities such as depth and optical flow as opposed to texture-centric approaches like semantic segmentation.

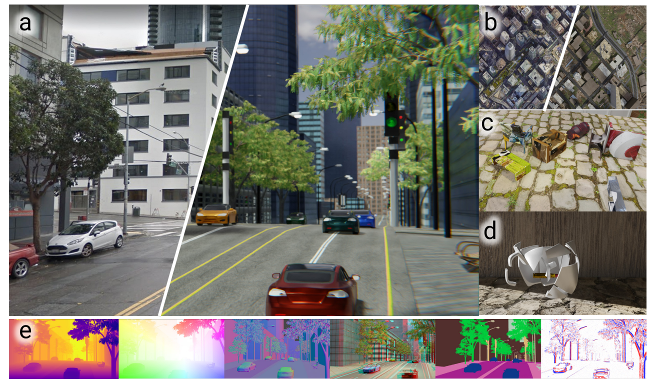

Fig. 1. Generative ability of WorldGen: (a) Comparison between Google Street View (left) and the same street in WorldGen (right), (b) Comparison of Google Maps satellite image vs. WorldGen top view, (c) Collection of 3D objects in motion, (d) Object fragmentation,(e) Annotation from left to right: depth, optical flow, surface normals, stereo anaglyph, image segmentation, event frame. *All the images in this paper are best viewed in color on a computer screen at 200% zoom.*

TABLE I

Comparison of the different simulation environments.

Name	Rendering	GI	Physics	Scaling
UnrealCV [4]	UE4	×	UE4	×
iGibson [5]	PyRender	×	PyBullet	✓
Omnidata [6]	Blender	✓	–	×
Blenderproc [7]	Blender	✓	Bullet	×
AirSim [8]	UE4	×	AirSim	×
CARLA [9]	UE4	×	UE4	×
Kubric [10]	Blender	✓	PyBullet	✓
WorldGen *(Ours)*	Blender	✓	Bullet	✓

* Only Blender rendering engine supports ray-tracing with OptiX denoiser for photorealistic images; other use rasterization. GI: Full Global Illumination Support; Physics Engine Used; Scaling: Scalable to generate large datasets.

To reduce this sim-to-real gap even further, a large amount of synthetic scene generation with wide variability is required. This would require either to manually creating high-quality 3D assets or bringing together these assets in a structured way to form a scene. It becomes clear that a scalable simulation environment becomes a necessity for neural networks to improve the quality of the prediction models [5] and [10] (refer Table I) demonstrated the power of scalable data by providing an in-depth analysis of the prediction using their data for training. In this work, we present a scalable framework to automatically generate structured and unstructured environments for perception and robotics applications without any need for manual effort. Fig. 1 demonstrates the ability of WorldGen to

This work was supported in parts by Northrop Grumman (N00014-17-1-2622, SMA 1540917), Army Research Laboratory under a cooperative agreement with the University of Maryland (ArtIAMAS), Army Cooperative Agreement (W911NF2120076) and NSF Grant (OISE 2020624). Chahat Deep Singh was funded by Ann G. Wylie fellowship at the University of Maryland, College Park.

[1]Perception and Robotics Group, University of Maryland Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA. Emails: {chahat, rkumari, fer, jyaloimo}@umd.edu.

[2]Robotics Engineering, Worcester Polytechnic Institute, MA 01609, USA. Email: nsanket@wpi.edu

automatically generate structured cities, object datasets and object fragments through code.

A. Related Work

In the last five years, there has been a boom in self-driving car simulators for perception, planning, and control [8], [9], [11]. One of the major drawbacks of such simulators is the limitation in scalability for data generation. Such simulators are shipped with a limited number of scenes (or towns). This is because it requires intensive manual labor to generate these 3D environments for better scalability and generalizability. Scalability in these simulators are present for traffic simulations but not for perception tasks. For instance, CARLA [9] library includes only 40 different building models. Our method relies on 3D structures of buildings from satellite semantics and 3D maps to build countless numbers of building models.

Learning geometric annotations such as an optical flow that relies substantially on motion parameters (rather than textures) have been studied extensively in the past decade. Unlike monocular depth estimation and instance segmentation, the prediction of such geometric quantities through convolutional neural networks generalizes well across different domains. For such predictions, synthetic datasets have widely been used [2], [3], [12], [13] with a limited number of objects and textures. Recent advancements in scalable scene generation environments like Kubric [10] have shown improvements in optical flow prediction due to more variability in data generation.

Another common problem in vision and robotics is segmenting and tracking independently moving objects [14]–[16]. It is a key process to understanding the dynamics of the scene structure for navigation tasks. Applications such as dodging any malicious objects on drones [17], [18], tracking objects using RGB and event cameras in motion [19], [20], and flying through gaps [21] have been well examined. In this paper, we provide a framework to generate countless 3D scenes with texture, structure and lighting variability for the aforementioned applications. We summarize our key contributions next.

B. Key Contributions

- We introduce WorldGen, a generative simulator for creating different environments commonly encountered in real-world robotics and computer vision tasks
- WorldGen is scalable and can generate an infinite amount of photorealistic data with variations to not only object placements, but also object shape, texture, lighting, camera and motion properties.
- WorldGen supports realistic camera distortions such as barrel distortion, chromatic aberration, camera aperture among other computational photographic properties.
- We demonstrate WorldGen's utility on learning more accurate optical flow due to the virtue of better data generation ability.

The paper is organized as follows: Sec. II presents how our WorldGen generative simulator is constructed, its design principles and details. Next, Sec. III presents how WorldGen can be useful for a myriad of robotics and computer vision applications. Finally, we conclude the paper in Sec. IV with parting thoughts for future work.

II. WorldGen Generative Simulator

WorldGen is a high-level open-source python library to generate an unlimited amount of synthetic data. It serves as an ease-of-use platform to generate visual data for simulating self-driving cars, autonomous drones, object segmentation, active vision, motion segmentation, tracking, computational photography and more. Our key contribution is an API to build generative environments and streamline the process of generating synthetic data by lowering the usage difficulty barrier for the researchers and practitioners alike. WorldGen is built around Blender™, a free and open-source 3D creation suite, to generate synthetic data like city maps, a collection of moving objects, and object fragmentation. Currently, our framework utilizes `Bullet` physics engine mainly for object collisions, gravity, friction, and other force fields. Fig. 2 shows as an overview of WorldGen. We build WorldGen around the central concept of scalability and speed. We discuss different parts and a number of details that are used to build WorldGen next.

A. Environment

Our framework is structured in three different stages: (a) Loader: loading assets like objects, materials and textures, lights and/or map information to generate a structured 3D environment, (b) Structural Modification and Animation: this generates structurally different objects by varying texture maps (explained in Sec. II-B), then applies physics and motion models to objects, camera and lights and (c) Rendering: outputs rendered frames and its annotation along with desired metadata (like time, frame number, camera intrinsic, and extrinsic).

Rendering: Our framework utilizes Blender's *Cycles* – a ray-trace based production render engine. To avoid slow renders, the number of samples (or path-traced for each pixel) are usually set to a lower value that produces *graininess* in the output renders. We utilize NVIDIA OptiX™ [22], a recurrent denoising autoencoder to reduce the *graininess* (noise) without a need for more rendering iterations. Furthermore, our framework also utilizes both full Global Illumination (GI) for photo-realistic renders and fast GI approximation for faster renders.

B. Texture Mapping

WorldGen utilizes UV mapping [23] which is a well-known process in computer graphics that projects a 2D image onto a 3D model's surface. Along with RGB images, open-source textures are often composed of displacement and normal maps in order to *fake* the lighting of *bumps* in the 3D model. These mapping techniques are used to re-detail a simplified or low-poly mesh without increasing the number of vertices, thus reducing the rendering time. By varying the strength of the displacement and normal maps, a structurally different variant of the same object can be rendered without a need to modify the 3D object (See Fig. 3). Thus, we modify these maps with additive white Gaussian noise before applying them to the 3D objects. Such variational texture mapping can contribute to the generation of thousands of *similar* object renders from a small set of object meshes. This can be used to generate large amounts of datasets to predict geometric quantities (or annotations) like optical flow

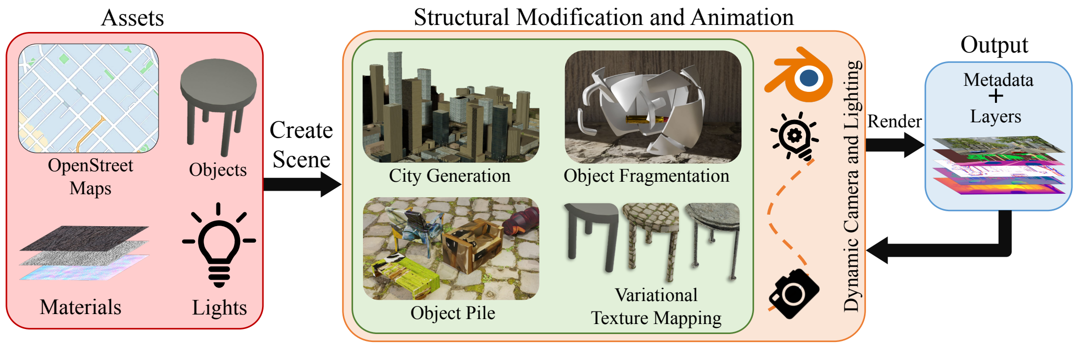

Fig. 2. An overview of WorldGen Framework: (a) Assets: Loads the assets such as maps, objects, materials etc. into WorldGen environment, (b) Structural Modification and Animation: Modifying the texture maps and applying physics and motion models on different objects in the scene, (c) Rendering: Generates rich ground truth data with the desired metadata (time, frame number, camera intrinsic and extrinsic properties).

for better generalizability across different domains. For more details, read Sec. III-A.

Fig. 3. Mapping textures to a round table. Top row: Rendered Output, Bottom row: Sample textures projected on a sphere. (a) Barebone 3D model, (b)-(d) Different Textures applied on (a). *Note: Variational mapping models change the structure of the 3D objects in different renders (notice the legs on the chair). Here, the Gaussian noise in* $(d) > (c) > (b)$.

C. Generative Models

At the time of writing, WorldGen supports three different generative models: *(1)* City Maps *(2)* Object Pile, and *(3)* Object Fragmentation.

1) City Models: Our framework utilizes OpenStreetMaps (OSM), a crowd-sourced project that contains semantic labels and 3D terrain maps from the satellite perspective. Using latitude and longitude as the input, semantic and 3D terrain information is imported into our framework using [24]. Using semantic information like buildings, water bodies, forests, vegetation, roads, highways, pedestrian pathways and railways, a set of relevant assets are imported from open source libraries (see Sec. II-F) along with its appropriate material (see Fig. 4). Furthermore, we can deploy different roof structures such as flat or gable on various buildings. The semantic data from OSM also contains building roof information, allowing our framework to deploy *realistic* roof structures (flat or gable) on the respective buildings. Assets like radio antennas, air-conditioning vents, and chimneys are randomly distributed over the roof surface. For other generative solid objects like roads, 3D meshes are created in the environment using the vertices from the semantic data. Furthermore, lights are imported to the scene with GI support for different types of weather and daytime scenarios to get a photo-realistic visualization (Fig. 5). Using basic

Fig. 4. (a) OpenStreetView, (b) Depth Map, (c) 3D Model View Generated by WorldGen and (d) Final Rendered View

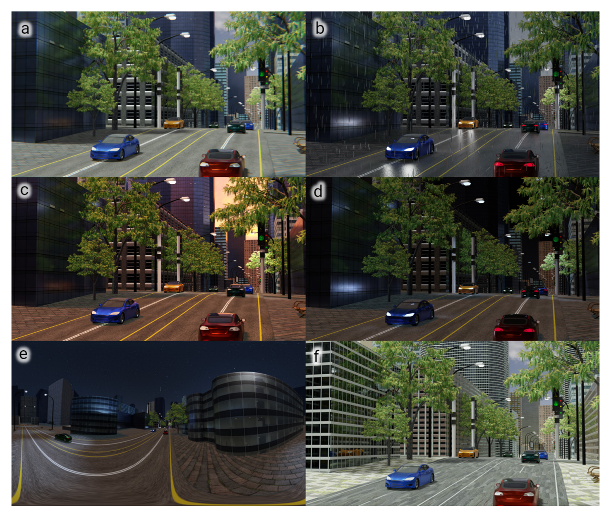

Fig. 5. City environment in different weather and time of the day: (a) Day, (b) Night with rain, (c) Dawn and (d) Night without rain, (e) panoramic view of the city and (f) demonstrates the generative ability of WorldGen by changing the textures of the entire scene while keeping the same structure.

morphological and vector operations, the road intersections are estimated to deploy traffic lights and stop signs near these road crossings. Other commonly available street objects like trees, benches, and street lights are structurally distributed near the pedestrian pathways. Fig. 1 shows the final rendered output of our city environment, complete with realistic distortions present in camera images such as chromatic aberration.

2) Object Pile: WorldGen's ability to generate a collection of moving objects in space is closest in scope to Kubric [10] and BlenderProc [25]. The key advantage of our framework over Kubric is the flexibility to generate large-scale datasets due to variational texture mapping that can generate different structural variations of the same 3D models. Fig. 1b shows the output of an object pile environment. WorldGen supports a variety of control in physics such as collision shapes using convex hulls, 3D meshes, boxes, cylinders, etc. with varying collision margins, friction, and coefficient of restitution for allowing more control in the data generation process. Furthermore, apart from the standard gravitational force, our framework deploys other force fields such as wind and drag to influence the object's motion which reflects reality much more and would be useful in learning perception modules for realistic scenarios.

3) Object Fragmentation: Another generative model of our framework is to break or *fragment* any 3D mesh into user-defined number of voronoi cell fractures. We utilize [26] to generate the specified number of 3D meshes from the parent mesh. The object fragments into child particles upon contact with another rigid body (active or passive) or any other non-uniform force field. The rendered output is a video sequence from the camera sensors with semantic information of each individual fragment. Fig. 1c shows the simulation environment of the scene where a bullet is colliding with a cup, thereby shattering it into a number of pieces. Such an approach is potentially useful for assessing damage due to collisions.

D. Sensors

WorldGen renders RGB images along with the annotation data such as depth map, optical flow, stereo images, semantic map, and surface normal map (See Fig. 1) which are rendered directly through the Blender Cycles engine. For event camera renders, we utilize a simple event camera model [27], [28] to generate events at a location $\mathbf{x}$ when

$$\| \log\left(\mathcal{I}_t\left(\mathbf{x}\right)\right) - \log\left(\mathcal{I}_{t+\delta t}\left(\mathbf{x}\right)\right) \|_1 \geq \tau \tag{1}$$

where τ is a user-defined threshold, $\mathbf{x}$ is the pixel location and $\mathcal{I}_t$ represents a grayscale image captured at time t. An *event-frame* is generated using the above model along with additive Gaussian noise. Alternatively, a more realistic event model [29], [30] can be adapted to our framework. For smaller integration time in generating event frames, we remap the animation timeline to $\mathbb{N}$ times more frames to give a slow-motion effect to simulate a visual high speed camera from which events are generated.

Multiple such sensors can be spawned in the environment to render outputs from different camera pose at the same time. WorldGen takes 6 degree-of-freedom camera extrinsic parameters to render outputs from different cameras. This setup can be used to simulate a sensor suite like the ones typically found on a self-driving car or an autonomous drone.

Camera Properties: WorldGen supports both dynamic camera trajectories as well as variation in camera intrinsic properties – dynamic focal length, depth of field, camera aperture radius, or variable camera baseline in the case of stereo output to simulate and generate data for previous works like [31], [32]. These properties can be temporally modified by using simple linear interpolation between defined key-frames or Bézier curves. WorldGen uses the same method to dynamically move the solid objects in the scene. Also, due to variability in camera focal length, our framework supports both fisheye and equirectangular projection to render full 360-panorama images (Fig. 5e). Furthermore, WorldGen also supports real-world camera distortions for a more photorealistic render such as lens glare, chromatic aberrations, and barrel distortion. This is particularly useful for handling corner case scenarios which might aid in better sim-to-real generalization.

E. Lighting and Climate Conditions

Currently, WorldGen supports three lighting conditions – midday, sunset and night as well as 4 different weather conditions – rain, cloudy, clear, and fog (Figs. 5a to 5d). Since WorldGen is Global Illumination (GI) enabled, the sunlight diffusion and reflection are photorealistic. Along with GI, it supports fast GI approximation for faster render time with a small loss in photorealism. The diffused sky radiation; the angle, power, and color temperature of the sun, and light sources such as street lights can be temporally varied.

F. Assets

The elegance of WorldGen generative simulator is to use and modify existing 3D meshes, and materials to create new 3D environments. For the object pile and fragmentation scene generation, WorldGen imports objects and materials by reading a text file that contains a list of wavefront `.obj` file path and material `.mtl` file path which makes it easier for the user to import any 3D object mesh into the WorldGen environment. We utilize a myriad of 3D object databases and image textures for incorporating a wide variety of possibilities. Furthermore, users can also import any custom objects/textures along with the ones described below:

ShapeNetCore.v2 [33]: ShapeNetCore.v2 contains about 51,300 unique 3D models across 55 object categories that are manually verified for classification and alignment annotations. We utilize Kubric's cleaned version of ShapeNet objects that fixes issues regarding auto-smoothing and backface culling that are present in the original ShapeNetCore objects.

Google Scanned Objects (GSO) [34]: (CC-BY 4.0 License) GSO contains 1030 scanned objects and their associated metadata totaling $\sim$13GB. Since these models are scanned and not hand crafted in a 3D modeling tool, they realistically reflect real object properties.

MS-COCO [35]: Microsoft's COCO is a dataset that contains images of complex everyday scenes containing common objects in their natural context containing over 328k images across 91 object classes. We use these relatively lower-resolution images to warp around objects to modify their natural appearance.

AmbientCG [36] is a large public domain resource with CC0 license for Physically Based Rendering (PRB) with 3D objects, materials and High Dynamic Range Images (HDRI) containing photo-scanned materials, displacement maps to $\sim$1 pixel accuracy with about 1760 assets with texture resolution up to $8K$.

Polyhaven [37] is a public library with a CC0 license with 500+ pre-processed HDRI images which we resourced for backgrounds and lighting.

cgbookcase [38] is another CC0 license with 540+ high-resolution PBR textures for common city structures such as different walls, roads, buildings, concrete, etc.

Apart from the aforementioned assets, we also utilized open-source libraries and Blender add-ons for pre-processing and rendering operations: `Blender-OSM`, `MapBox` and `object_fracture_cell` and `bpycv`.

G. Design Principles

1) Modularity: WorldGen is a highly modular framework as it is divided into three different elements as discussed in Sec. II-A. Each of the modules can be treated differently and can be replaced by a different third-party module. For instance, we can extend this work to generate a variety of human motions on a custom background using assets from 3D character rigging resources such as Adobe's Mixamo [39] to generate ground truth data annotations for human pose estimation. We plan to add character-rigged motion scene generation and data annotation with varying character motion to WorldGen.

2) Ease of use: The design philosophy of WorldGen is to make it easier for researchers, educators and practitioners to generate new scenes without needing any core knowledge of computer graphics concepts, modeling, animating, and rendering tools. WorldGen bridges this gap by presenting a simple high-level object-oriented Python API with Blender (and Bullet) working in the background.

3) Open Source: Scene and data generation codes will be released upon acceptance and will be free to use in academia. All the third-party assets used in WorldGen with Apache2.0, MIT and CC0 licenses enable researchers to generate different environments, render annotations and share all the data with the community to ensure reproducibility.

4) Scalability: WorldGen brings generative methods to build new structured environments without any need for manual labor. It can be used to generate environments anywhere between a small town and a large city with a compromise on GPU and access memory.

III. Applications

WorldGen is designed to generate different 3D environments and annotated data for robotics and computer vision applications. It is meant to be a one-stop framework for zero-shot generalization due to its variability in shapes, texture, and dynamic lighting; also minimizing the gap for sim-to-real transfer. It would also serve as a simulator for developing and testing planning and control algorithms due to its flexibility since it can use the current state and annotation as input for the next time stamp. To demonstrate the potential and flexibility of WorldGen, we discuss a set of common and challenging problem statements in robotics and computer vision.

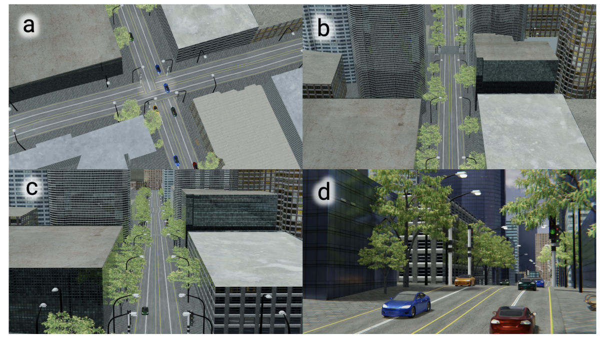

Fig. 6. High resolution views generated by WorldGen from views at different altitudes with dynamic lighting, camera intrinsic, and extrinsic.

A. Improvements in Optical Flow

Optical flow is one of the most fundamental quantities in computer vision and robotics which measures the 2D motion of each pixel between consecutive frames. Contrary to high-level computer vision applications like instance segmentation, ground truth cannot be reliably obtained from real-world data with human annotation. Recent advances in optical flow methods – PWC-Net [40], RAFT [41] and GMFlow [42] all rely on synthetic data like MPI Sintel [13], FlyingChairs [2] and FlyingThings3D [3] for pre-training. Such datasets use synthetic chairs and lack photorealism and realistic 3D motion. Recent advances in dataset generation – AutoFlow [43] learns to render hyperparameters for generating synthetic flow datasets reducing the End Point Error (EPE). AutoFlow lacks 3D motion and photorealism as it utilizes a simple 2D layered model. Having photo-realistic data, Kubric [10] tends to outperform AutoFlow, but only on Sintel *Clean* samples. This is the rendered output that contains shading but no image degradation whereas the *Final* pass includes motion blur, defocus blur and atmospheric effects. Since Kubric lacks both volumetric features (like fog, mist, and rain) and advanced camera features (such as depth of field and motion blur), AutoFlow performs better even without photorealism as images are generated with motion blur and fog models as well as data augmentation for visual effects.

We compare and analyze optical flow predictions using RAFT on Sintel *Clean* and *Final* pass which are trained on different datasets. Table II shows the EPE in optical flow. Note that WorldGen causes a significant increase in optical flow accuracy as compared to FlyingChairs. Furthermore, WorldGen outperforms AutoFlow due to the difference in photorealism. Note that WorldGen slightly improves over Kubric. We speculate one of the reasons to be WorldGen rendering multiple instances of the same object configuration by warping different textures on the objects, ensuring the network avoids over-fitting to object textures, and thereby learning the geometrical/motion properties of the scene. Since WorldGen supports dynamic lighting, depth of field, and motion blur modeling, we see a significant improvement over Kubric in the *Final* pass. Note that AutoFlow performs better in *Final* pass (although only slightly) because the hyperparameters of AutoFlow have been learned to optimize the performance on the Sintel dataset which gives AutoFlow an unfair advantage.

TABLE II
OPTICAL FLOW EPE COMPARISON OF TRAINING RAFT [41] ON DIFFERENT DATASETS. LOWER IS BETTER.

Dataset	Dimensionality	Parameters	Sintel Clean	Sintel Final
FlyingChairs [2]	2D	Manual	2.27	3.76
Kubric [10]	3D	Manual	1.89	3.02
AutoFlow [43]	2D	Learned	2.08	**2.75**
WorldGen (Ours)	3D	Manual	**1.86**	2.87

B. Computational Photography

Computational photography is one of the key areas that require a large collection of image data for the prediction of depth maps, denoising image data, etc. Most of the current photo-realistic synthetic data generators in robotics and computer vision do not explore and support many features that are offered by Blender, Unity, or Unreal Engines such as volumetric effects and variability in sensor sizes, camera lens, and rolling shutter sensors. Since WorldGen supports generating data for different light conditions, different *bokeh* blur due to depth of field effect, motion blur, variable focal lens and sensor size along with *Albedo*, *Clean* and *Final* renders, it can be used to generate data for the prediction of various computational photography applications such as HDR+ datasets [44], depth from defocus [45], [46], recovering image from a rolling shutter blur [47], [48], extracting a video sequence from a single blurred frame [49] and for learning synthesis motion blur from a pair of images [50]. Furthermore, this work can be utilized to generate datasets for learning depth estimators using a coded aperture [51], [52] camera. We intend to add a toolbox to generate coded *blurred* frames in the WorldGen environment after release.

C. View Synthesis using Neural Radiance Fields

Neural volume rendering has exploded in the last two years [53]. One of the major challenges in [53] is to synthesize in the case of dynamic relighting. These methods underperform with views ranging from satellite level capturing the entire city to the ground level, majorly due to large camera displacement and dynamic lighting. Recent advances in multi-scale scene rendering [54] and dynamic irradiance view synthesis [55] have adapted to work in such adverse conditions. In [55], the authors utilize multi-date images due to their significant changes in appearance, mainly due to varying shadows and transient objects like cars and vegetation using the WorldView-3 dataset. WorldGen can generate thousands of 3D satellite views with dynamic lighting at a higher resolution than the satellite images with known camera intrinsic and extrinsic for more robust view synthesis in satellite images with larger displacement in different lighting and weather conditions (See Fig. 6). Also, this can be extended towards learning high-level planners for aerial cinematography [56].

D. Active and Interactive Perception

Not only is WorldGen an automatic 3D scene and dataset generation tool but it also serves as a framework for active and interactive perception applications. Robots such as quadrotors are often required to use the current prediction and annotations as input to the next state for its solution and testing. Currently, there is no test bench to verify the robotics algorithms that rely on active and interactive perception approaches [57], [58]. WorldGen will serve as a benchmark pseudo-simulator to validate the performance of such algorithms for better reproducibility. We hope WorldGen will become the OpenAI Gym [59] for Active and Interactive perception.

E. Generating Real World Traffic

The problem of generating *realistic* traffic scenes automatically is a challenging task. Existing methods typically are driven by planning and reinforcement algorithms that require current and previous traffic states, traffic signal conditions, and road/pathways vertices [60]. Since WorldGen has the ability to export these parameters, neural autoregressive models such as [61] can be deployed for realistic traffic distribution. The recent advances in neural fields such as Panoptic Neural Fields [62] have pushed the boundary to detect traffic in a 3D representation, allowing us to perform 3D scene editing. Using such methods, WorldGen can also import real-life traffic into its model for generating photorealistic real-world city traffic scenarios.

F. Human Pose Estimation

In recent years, 3D human pose estimation has made great improvements which have been driven by large-scale datasets. However, these datasets lack accuracy in the pose as it mostly annotated by humans from open-world datasets [63] that suffer from human labeling accuracy, mainly due to occlusions. On the contrary, motion capture systems have helped to generate accurate human pose data but they lack an open world environment due to motion capture space restrictions [64]. Lastly, the simulation environment for dataset generation lacks photorealism [65]. We plan to develop another generative environment with WorldGen by utilizing Mixamo [39], an open platform with a variety of character rig models and animations; and warping different 3D materials to generate a large of photorealistic character animation even in the backgrounds such as small towns and city. This will push the boundaries for monocular human pose estimation and open new avenues for special sensors such as event cameras.

IV. CONCLUSION

We introduce WorldGen, a modular, generative, open-source Python API as a large scale generative simulator with a myriad of scenes. We can generate accurate ground truth labels for optical flow, depth, surface normals, depth cameras, semantic maps and event data for driving scenes, moving object pile and object fragmentation. WorldGen also simulates camera properties, motion properties for a photorealistic perception-centric simulation and data generation. Not only can WorldGen change objects, lighting and scenes dynamically, it can also edit objects dynamically aiding in generation of data to better enable generalization of neural networks such as optical flow as shown in our experiments. Currently, sensors such as SONAR and LIDAR are missing from WorldGen along with behaviour modelling for self-driving cars, rigging of human characters is not implemented and we see this as a potential direction for future work.

REFERENCES

[1] Yuki M Asano, Christian Rupprecht, Andrew Zisserman, and Andrea Vedaldi. Pass: An imagenet replacement for self-supervised pretraining without humans. *arXiv preprint arXiv:2109.13228*, 2021.

[2] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[3] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016.

[4] Weichao Qiu and Alan Yuille. Unrealcv: Connecting computer vision to unreal engine. In *European Conference on Computer Vision*, pages 909–916. Springer, 2016.

[5] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, et al. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *arXiv preprint arXiv:2108.03272*, 2021.

[6] Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10786–10796, 2021.

[7] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, and Harinandan Katam. Blenderproc. *arXiv preprint arXiv:1911.01911*, 2019.

[8] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer, 2018.

[9] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[10] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3749–3761, 2022.

[11] Panpan Cai, Yiyuan Lee, Yuanfu Luo, and David Hsu. Summit: A simulator for urban driving in massive mixed traffic. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4023–4029. IEEE, 2020.

[12] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1):7–42, 2002.

[13] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, October 2012.

[14] Anton Mitrokhin, Cornelia Fermüller, Chethan Parameshwara, and Yiannis Aloimonos. Event-based moving object detection and tracking. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[15] Anton Mitrokhin, Chengxi Ye, Cornelia Fermüller, Yiannis Aloimonos, and Tobi Delbruck. Ev-imo: Motion segmentation dataset and learning pipeline for event cameras. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6105–6112. IEEE, 2019.

[16] Francisco Barranco, Cornelia Fermüller, and Eduardo Ros. Real-time clustering and multi-target tracking using event-based sensors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5764–5769. IEEE, 2018.

[17] Nitin J Sanket, Chethan M Parameshwara, Chahat Deep Singh, Ashwin V Kuruttukulam, C Fermuller, Davide Scaramuzza, and Yiannis Aloimonos. Evdodge: Embodied ai for high-speed dodging on a quadrotor using event cameras. *arXiv preprint arXiv:1906.02919*, pages 31–45, 2019.

[18] Davide Falanga, Kevin Kleber, and Davide Scaramuzza. Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics*, 5(40):eaaz9712, 2020.

[19] Chethan M Parameshwara, Nitin J Sanket, Chahat Deep Singh, Cornelia Fermüller, and Yiannis Aloimonos. 0-mms: Zero-shot multi-motion segmentation with a monocular event camera. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9594–9600. IEEE, 2021.

[20] Andreas Wedel, Annemarie Meißner, Clemens Rabe, Uwe Franke, and Daniel Cremers. Detection and segmentation of independently moving objects from dense scene flow. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 14–27. Springer, 2009.

[21] Nitin J Sanket, Chahat Deep Singh, Kanishka Ganguly, Cornelia Fermüller, and Yiannis Aloimonos. Gapflyt: Active vision based minimalist structure-less gap detection for quadrotor flight. *IEEE Robotics and Automation Letters*, 3(4):2799–2806, 2018.

[22] Chakravarty R Alla Chaitanya, Anton S Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics (TOG)*, 36(4):1–12, 2017.

[23] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM transactions on graphics (TOG)*, 21(3):362–371, 2002.

[24] Prochitecture. Blender-osm: Openstreetmap and terrain for blender.

[25] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, and Harinandan Katam. Blenderproc. *arXiv preprint arXiv:1911.01911*, 2019.

[26] Blender Cell Fracture. https://docs.blender.org/manual/en/latest/addons/object/cell_fracture.html.

[27] Nitin J Sanket, Chahat Deep Singh, Chethan M Parameshwara, Cornelia Fermüller, Guido CHE de Croon, and Yiannis Aloimonos. EVPropNet: Detecting Drones By Finding Propellers For Mid-Air Landing And Following. In *Robotics: Science and systems (RSS) conference 2021*. Robotics: Science and Systems, 2021.

[28] Nitin J Sanket, Chethan M Parameshwara, Chahat Deep Singh, Ashwin V Kuruttukulam, Cornelia Fermüller, Davide Scaramuzza, and Yiannis Aloimonos. Evdodgenet: Deep dynamic obstacle dodging with event cameras. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10651–10657. IEEE, 2020.

[29] Y Hu, S C Liu, and T Delbruck. v2e: From video frames to realistic DVS events. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2021.

[30] Daniel Gehrig, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. Video to events: Recycling video datasets for event cameras. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, June 2020.

[31] Nitin J Sanket, Chahat Deep Singh, Varun Asthana, Cornelia Fermüller, and Yiannis Aloimonos. Morpheyes: Variable baseline stereo for quadrotor navigation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 413–419. IEEE, 2021.

[32] Pablo Pueyo, Eduardo Montijano, Ana C Murillo, and Mac Schwager. Cinempc: Controlling camera intrinsics and extrinsics for autonomous cinematography. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4058–4064. IEEE, 2022.

[33] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[34] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. *arXiv preprint arXiv:2204.11918*, 2022.

[35] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[36] ambientCG. https://ambientcg.com.

[37] Polyhaven. https://polyhaven.com.

[38] cgbookcase. https://cgbookcase.com.

[39] Sue Blackman. Rigging with mixamo. In *Unity for Absolute Beginners*, pages 565–573. Springer, 2014.

[40] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018.

[41] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.

[42] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. Gmflow: Learning optical flow via global matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8121–8130, 2022.

[43] Deqing Sun, Daniel Vlasic, Charles Herrmann, Varun Jampani, Michael Krainin, Huiwen Chang, Ramin Zabih, William T Freeman,

and Ce Liu. Autoflow: Learning a better training set for optical flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10093–10102, 2021.

[44] Michaël Gharbi, Jiawen Chen, Jonathan T Barron, Samuel W Hasinoff, and Frédo Durand. Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)*, 36(4):118, 2017.

[45] Marcela Carvalho, Bertrand Le Saux, Pauline Trouvé-Peloux, Andrés Almansa, and Frédéric Champagnat. Deep depth from defocus: how can defocus blur improve 3d estimation using dense neural networks? In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[46] Shir Gur and Lior Wolf. Single image depth estimation trained via depth from defocus cues. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7683–7692, 2019.

[47] Mingdeng Cao, Zhihang Zhong, Jiahao Wang, Yinqiang Zheng, and Yujiu Yang. Learning adaptive warping for real-world rolling shutter correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17785–17793, 2022.

[48] Zhixiang Wang, Xiang Ji, Jia-Bin Huang, Shin'ichi Satoh, Xiao Zhou, and Yinqiang Zheng. Neural global shutter: Learn to restore video from a rolling shutter camera with global reset feature. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17794–17803, 2022.

[49] Meiguang Jin, Givi Meishvili, and Paolo Favaro. Learning to extract a video sequence from a single motion-blurred image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6334–6342, 2018.

[50] Tim Brooks and Jonathan T Barron. Learning to synthesize motion blur. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6840–6848, 2019.

[51] Anat Levin, Rob Fergus, Frédo Durand, and William T Freeman. Image and depth from a conventional camera with a coded aperture. *ACM transactions on graphics (TOG)*, 26(3):70–es, 2007.

[52] Jorge Bacca, Tatiana Gelvez-Barrera, and Henry Arguello. Deep coded aperture design: An end-to-end approach for computational imaging tasks. *IEEE Transactions on Computational Imaging*, 7:1148–1160, 2021.

[53] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[54] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In *The European Conference on Computer Vision (ECCV)*, 2022.

[55] Roger Marí, Gabriele Facciolo, and Thibaud Ehret. Sat-NeRF: Learning multi-view satellite photogrammetry with transient objects and shadow modeling using RPC cameras. *arXiv preprint arXiv:2203.08896*, 2022.

[56] Ioannis Mademlis, Vasileios Mygdalis, Nikos Nikolaidis, Maurizio Montagnuolo, Fulvio Negro, Alberto Messina, and Ioannis Pitas. High-level multiple-uav cinematography tools for covering outdoor events. *IEEE Transactions on Broadcasting*, 65(3):627–635, 2019.

[57] Nitin J Sanket, Chahat Deep Singh, Kanishka Ganguly, Cornelia Fermüller, and Yiannis Aloimonos. Gapflyt: Active vision based minimalist structure-less gap detection for quadrotor flight. *IEEE Robotics and Automation Letters*, 3(4):2799–2806, 2018.

[58] Chahat Deep Singh, Nitin J Sanket, Chethan M Parameshwara, Cornelia Fermüller, and Yiannis Aloimonos. Nudgeseg: Zero-shot object segmentation by repeated physical interaction. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2714–2712. IEEE, 2021.

[59] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[60] Haitao Yuan and Guoliang Li. A survey of traffic prediction: from spatio-temporal data to intelligent transportation. *Data Science and Engineering*, 6(1):63–85, 2021.

[61] Shuhan Tan, Kelvin Wong, Shenlong Wang, Sivabalan Manivasagam, Mengye Ren, and Raquel Urtasun. Scenegen: Learning to generate realistic traffic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 892–901, 2021.

[62] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic Neural Fields: A Semantic Object-Aware Neural Scene Representation. In *CVPR*, 2022.

[63] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele.

[64] Gregory Rogez and Cordelia Schmid. Mocap-guided data augmentation for 3d pose estimation in the wild. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[65] Salehe Erfanian Ebadi, You-Cyuan Jhang, Alex Zook, Saurav Dhakad, Adam Crespi, Pete Parisi, Steven Borkman, Jonathan Hogins, and Sujoy Ganguly. Peoplesanspeople: A synthetic data generator for human-centric computer vision. *arXiv preprint arXiv:2112.09290*, 2021.