InteriorNet: Mega-scale Multi-sensor Photo-realistic Indoor Scenes Dataset

Wenbin Li[1]
wenbin.li@imperial.ac.uk
Sajad Saeedi[1]
s.saeedi@imperial.ac.uk
John McCormac[1]
brendan.mccormac13@imperial.ac.uk
Ronald Clark[1]
ronald.clark@imperial.ac.uk
Dimos Tzoumanikas[1]
dimosthenis.tzoumanikas14@imperial.ac.uk
Qing Ye[2]
zhentou@qunhemail.com
Yuzhong Huang[2]
yuzhongh@usc.edu
Rui Tang[2]
ati@qunhemail.com
Stefan Leutenegger[1]
s.leutenegger@imperial.ac.uk

[1] Department of Computing
Imperial College London
London UK, SW7 2AZ

[2] KooLab, Kujiale.com
Hangzhou China

Abstract

Datasets have gained an enormous amount of popularity in the computer vision community, from training and evaluation of Deep Learning-based methods to benchmarking Simultaneous Localization and Mapping (SLAM). Without a doubt, synthetic imagery bears a vast potential due to scalability in terms of amounts of data obtainable without tedious manual ground truth annotations or measurements. Here, we present a dataset with the aim of providing a higher degree of photo-realism, larger scale, more variability as well as serving a wider range of purposes compared to existing datasets. Our dataset leverages the availability of millions of professional interior designs and millions of production-level furniture and object assets – all coming with fine geometric details and high-resolution texture. We render high-resolution and high frame-rate video sequences following realistic trajectories while supporting various camera types as well as providing inertial measurements. Together with the release of the dataset, we will make executable program of our interactive simulator software as well as our renderer available at `https://interiornetdataset.github.io`. To showcase the usability and uniqueness of our dataset, we show benchmarking results of both sparse and dense SLAM algorithms.

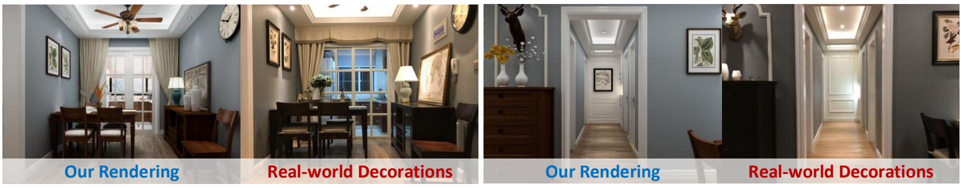

Figure 1: Our rendering vs. real decoration guided by our furniture models and layouts.

1 Introduction

Spatial perception has undergone a true step-change during the past few years, partly thanks to the Deep Learning revolution in Computer Vision, but not only. Spatial perception – core to e.g. a robotic system – involves a plethora or elements that should be addressed jointly: these include the understanding of camera pose relative to the environment, i.e. Simultaneous Localization and Mapping (SLAM), which ideally includes a very dense reconstruction of at least the immediate surrounding environment, allowing the robot to interact with it. As purely geometric understanding is still limiting, the research community has been shifting towards tightly integrating it with semantic understanding, object-level mapping, estimating dynamic content, etc.

In order to benchmark SLAM, as well as to both train and evaluate semantic understanding, datasets of increasing scale have been playing a central role. While we argue that real recorded datasets will remain crucial for the foreseeable future, they suffer from scalability limitations: for instance preparing ground truth scene models is a challenging task, with methods such as laser scanning being a costly and time-consuming process. Thanks to advances in computer graphics and computational power, the use of rendered synthetic models is an obvious choice and recent works such as ICL-NUIM dataset [13] for SLAM and SceneNet RGB-D dataset [17] for semantic labeling are examples of such works. In this paper, we have improved upon current rendering methods and datasets first of all by proposing a versatile and fast rendering framework for a high degree of photo-realism which leverages the availability of *millions of realistic indoor designs* composed of *millions of detailed digital object models*. Importantly, we model realistic lighting and scene *change over time*. Aside from RGB rendering, we can opt for depth images and semantics. Second, we use and synthesize realistic trajectories as ground truth to render at *video frame rate* from various classes of typical motion patterns. Fig. 1 shows samples from a real-world environment and synthesized images generated from models. In summary, our contributions are:

- Our scene database contains around 1M furniture CAD models and 22M interior layouts (Sec. 3), which have been created for real-world production and decoration.
- Our dataset contains two parts: (1) 15k sequences rendered from 10k randomly selected layouts, with 1k images for each sequence; (2) 5M images are rendered from 1.7M randomly selected layouts, with 3 images per layout.
- To simulate realistic scenes that change over time (Sec. 4), we interfaced a physics engine for object movement and provide flexibility to manipulate lighting conditions.
- We introduce a learned algorithm to realistically style random trajectories (Sec. 5).
- We implemented a fast, photo-realistic renderer, called *ExaRenderer* (Sec. 6).
- We implemented a user friendly simulator, called *ViSim* (Sec. 6.3), to assist creating monocular or stereo camera trajectories and synthesize related ground truth.
- We show the usefulness of the dataset by providing SLAM evaluation results (Sec 7).
- We will release our dataset (consisting of the rendered sequences and images) as well as *ExaRenderer*, *ViSim*, and a subset of the 3D models and layouts used for evaluations.

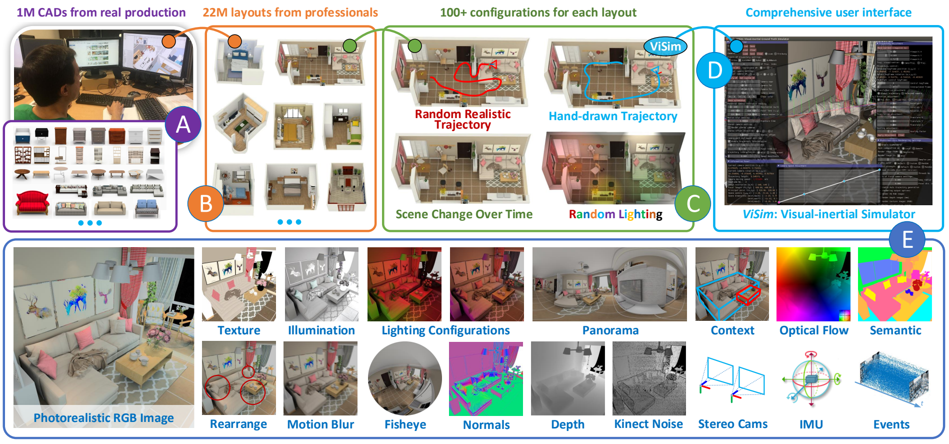

Figure 2: System Overview: an end-to-end pipeline to render an RGB-D-inertial benchmark for large scale interior scene understanding and mapping. (**A**) We collect around 1 million CAD furniture models from world-leading furniture manufacturers. These models have been used in the real-world production. (**B**) Based on those models, around 1,100 professional designers/companies create around 22 million interior layouts. Most of such layouts have been used in real-world decorations. (**C**) For each layout, we generate a number of configurations to represent different lightings and simulate scene change over time in daily life. (**D**) We provide an interactive simulator (*ViSim*) to create the ground truth monocular/stereo trajectories, as well as IMU and event camera data. Trajectories can be set manually, or using random walk and neural network based generation. (**E**) All supported image sequences and ground truth data.

2 Background

Benchmarking algorithms with real-world and synthetic datasets for various computer vision tasks has been playing an important role, especially in the field of robotics. In this section, we review popular datasets used in semantic labeling and SLAM.

In semantic labeling, the two most closely related synthetic datasets are the SUN-CG [28] with photo-realistic rendering [33], and SceneNet RGB-D [17]. SUN-CG also provides realistic hand-designed layouts similar to ours but from Planner 5D [3], a layout tool for amateur interior design. The number of layouts (45K), unique object meshes (2.6K), and rendered images ([33] 500K) are orders of magnitude smaller and without the daily life noise added, when compared to our proposed dataset – and importantly, we focus on rendered full trajectories rather than still images. SceneNet RGB-D [17] provides full trajectories, but at much lower sample rate and of only one type, whereas here we provide a number of base random trajectories, as well as a learned style for realistic jitter. SceneNet RGB-D also contains fewer images (5M) which are of a lower resolution (320×240) and quality, containing circular artefacts from the photon mapping process. The object models in SceneNet RGB-D are from ShapeNet [9] which contains 51K models from public online repositories. Although an excellent resource, the quality of the models and textures from these repositories can be quite variable and missing ground truth metric scales requires automated prediction [26, 27]. Additionally, Zhang *et al.* [32] show that pretraining with a synthetic dataset improves the results of computer vision tasks such as surface normal prediction, semantic segmentation, and object boundary detection. Qi *et al.* [24] use human context to simulate layout of the indoor environment. Here we use only assets of commercial production quality, with accurate metric scales and high quality textures at a much larger scale, all semantically labelled by the manufacturer. The closest related work with real-world data is the ScanNet dataset [11]. It is a dataset consisting of 2.5M frames, containing trajectories of 1.5K indoor scenes. The scenes are manually annotated with a tool designed for use in Amazon Mechanical Turk.

There are many real-world visual localization and mapping datasets. Examples are the TUM RGB-D dataset with 19 different trajectories [29], the EuRoC dataset with 11 visual and inertial sequences [8], the KITTI dataset with large-scale 2D outdoor trajectories [12], and New College dataset with medium scale trajectories. Ground truth for these trajectories is either obtained from GPS or motion capture systems, and for the map from laser scanners (in KITTI and EuRoC only). In the active vision dataset [6], limited control options are provided to control the trajectory in 2D while navigating through the images. These datasets are very useful, but either the number of trajectories or the environments are limited.

Among synthetic visual SLAM datasets, UnrealCV [25], [32], uses the Unreal Engine to render realistic images for labelling and SLAM. The ICL-NUIM dataset [13] provides eight sequences from two models using POV-Ray renderer [4]. These methods require manual work to setup the rendering environment and also the number of the models is limited.

3 Dataset Overview

The proposed pipeline (Fig. 2) provides a large scale furniture model database and interior layouts, plus an end-to-end rendering pipeline to create the proposed RGB-D-inertial data.

3.1 Furniture Models and Interior Layouts

We collected 1,042,632 computer-aided design (CAD) models of furnitures from 42 world-leading manufacturers. Those models have been categorized into 158 main classes and are manually mapped to NYU40 categories [20]. Fig. 3 shows the statistics and sample images of the top 50 categories in our database. Our furniture model database provides several unique features: (1) all object meshes are measured in real-world dimensions and feature the exact same measurements as their real-world counterparts; (2) all object meshes are high resolution (triangles and vertex) and associated with hierarchical semantic labeling information. For instance, a specific sofa model contains 734,198 triangles, every vertex of which is labeled in terms of parts i.e. arms, legs, seat cushion, back foam and frame; (3) all objects are used in real production and will be easily spotted in real-world homes; (4) every object is associated with multiple textures and materials. For example, every chair has different color schemes, covers and accessories. The material information of the models is provided by manufacturers, which is compatible to *VRayStyle* [5]. Material representation is guided by compositing 4 Bidirectional Reflectance Distribution Function (*BRDF*) properties: lambertian, microfacet, dielectric and transmission.

Given such a large-scale object database, 1,078 professional designers working with end customers have created 22,652,123 interior layouts for different scenarios since Oct. 2014. Fig. 3 illustrates the statistics of our layouts which contain 16 types of rooms adopted from the interior design industry. The most representative room types are bedroom, guest room, bathroom, living room and kitchen. Our layouts are diverse in range from studio to 42-room apartments. Most of the layouts are being used in real-world decoration.

3.2 Configurations

For each layout, we generated a number of different configurations to either rearrange the furniture or randomized the lighting conditions (Fig. 5). For the former (Sec. 4.1), we im-

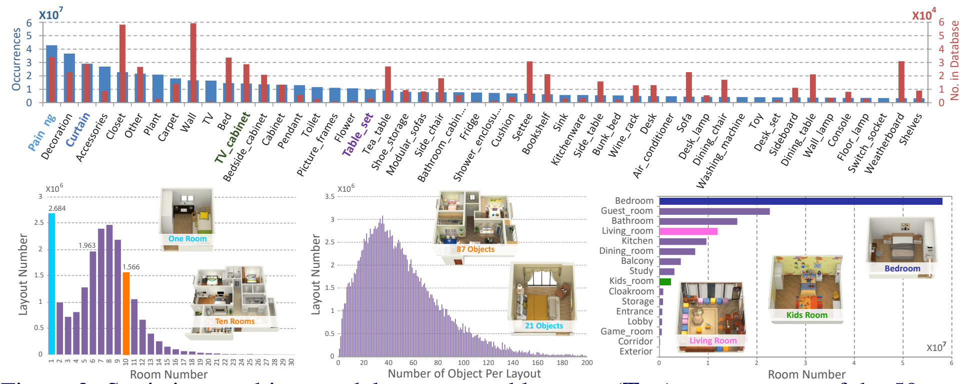

Figure 3: Statistics on object models, rooms and layouts. (**Top**): occurrences of the 50 most common categories of objects from the layouts (*blue*), and number of objects in our database (*red*). (**Bottom Left**): distribution of number of rooms per layout. (**Bottom Middle**): distribution of number of object per layout. (**Bottom Right**): Distribution of room type.

plemented a physics based automatic framework to slightly rearrange the movable objects. For the latter (Sec. 4.2), we support both manual or automatic changes on color and intensity of the default lighting. These setups diversify our layouts to simulate traces of daily life and the changes of natural lightings through a day.

3.3 Rendered Image Sequences and Ground Truth

We propose an end-to-end pipeline to render photo-realistic images and associated ground truth data (Fig. 2 (E)). We created multiple trajectories (Sec. 5) for each layout by varying the combination of linear velocity, angular velocity and trajectory types. Each view of a trajectory consists of both a shutter open and shutter close camera pose (many renderings from poses in-between are averaged to obtain motion blurred renderings). Our images support 640×480 resolution, at 25 Hz, resulting in 1,000 images per trajectory. By using an optimal parameter setup for best image quality, each render takes less than 2 ms on our cluster of 1,300 Nvidia GTX TiTanX GPUs and 2,000 Intel Xeon Phi CPUs. Our pipeline supports different RGB variations (pure texture, illumination and normals), and different types of lens models (panorama, fisheye and depth of field). The rendering speed may vary for rendering variations (a 5,000×2,500 panorama image takes 0.5-1 sec on our cluster to render).

Our rendering pipeline is able to create various per-frame ground truth data. Per-pixel semantic labels e.g. NYU40 (Fig. 4) can be obtained from model settings (provided by manufacturers) and a rendering pass. Per-object semantic context (3D bounding box) is provided to indicate the 3D extent of an object in the scene. Depth is obtained as the Euclidean distance of the ray intersection; and noisy depth is generated by simulating a real *Kinect* mechanism [10]. We also generate instance segmentations and optical flow following [17].

3.4 Dataset Structure and Hosting

Our dataset contains 20M photo-realistic images and many forms of ground truth data on around 1.7M layouts randomly selected from our database. To organize this wealth of data, we separated sequences into 20 different subsets in terms of number of rooms. We also provide a sparse subset that contains a smaller number of images (5M) but at highest diversity. We randomly selected 1.7M layouts, for each of which a random view was rendered in three different configurations, i.e. original layout, random lighting and random rearrange-

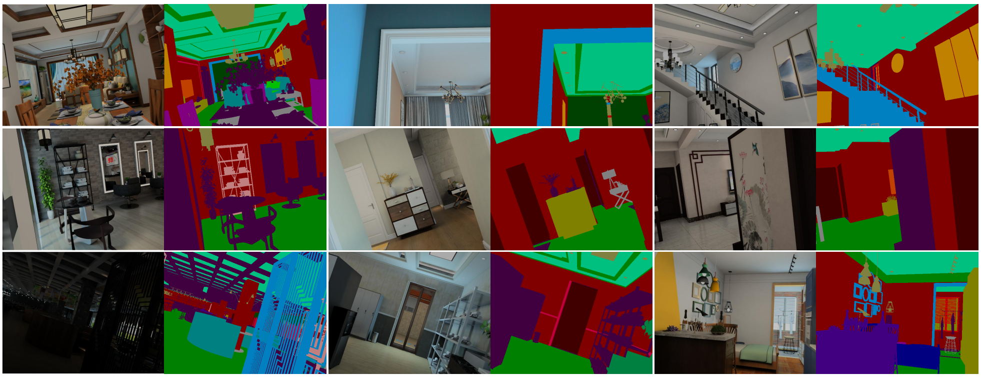
Figure 4: Our rendered images and the associated NYU40 labels.

ment. The parameters for randomly choosing lighting and rearrangement remained the same. The images of each subset mentioned above were randomly divided into 80%, 10% and 10% splits for training, validation and test sets respectively. We also release a subset of CAD models and layouts, as well as the rendering pipeline and the simulator (*ViSim*) for public.

4 Simulating Realistic Scenes

To further bring real-life challenges to the dataset, we added two new features to the models: simulating moving objects as in daily life and simulating variations in lighting.

4.1 Simulating Scene Change Over Time

As one of the main contributions, we provide certain level of flexibility to automatically configure the arrangement of the furniture objects in order to simulate the traces of daily life. Each object in our database is assigned a convex collision hull, as well as a mass (depending on different objects, in range from 0.05 to 43.3 kg) and a friction coefficient (depending on different surface materials, in range from 0.08 to 0.27) provided by the manufacturers. All those objects are then manually labeled as either *Movable* or *Unmovable*. The former may contain some objects moved on a daily basis, such as mugs, chairs, decorations and shoes etc. The latter may be some big furnitures barely moved, such as fridge, TV, bed and bookshelf etc. We further use an open-source physics engine, Project Chrono [1] to dynamically rearrange the furniture objects within a layout. For each configuration, we randomly select 5% to 45% movable objects within a layout. We offset the center of gravity slightly to the bottom of the object mesh, in order to avoid tipping over. For each of selected object, we apply a random acceleration (in range from 0.5 to 2 ms^{-2}) onto the geometric/mass center of the mesh along a random direction but parallel to the ground. The system takes 10 seconds to allow objects to settle to a physically realistic configuration. In most of the cases, the objects would move less than 2 meter from the original position.

4.2 Simulating Lighting

As another main contribution, we provide a flexibility to manipulate the lighting. A default lighting setup (*light.xml*) is provided for each layout. Those lighting setups include information of lighting type (SunLight, SpotLight and AreaLight), energy, maximum distance (like *PBRT*-v3 [22]), position, direction and brightness, which are provided by manufacturers and have been aligned and fixed to specific objects with lighting sources e.g. lamps. Our simulator is able to assist configuring actual RGB value, temperature and brightness of a specific

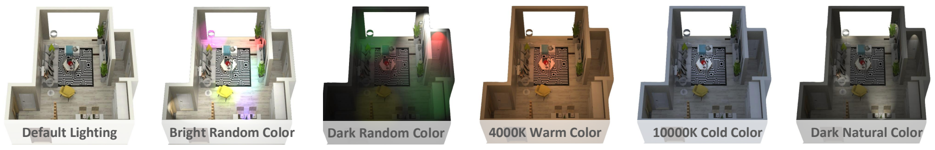

Figure 5: Lighting setup: natural and random color collection, brightness and temperature.

light, or turning it on/off. We also support automatically generating different combinations of lighting setups (Fig. 5).

5 Trajectory Generation

Our dataset provides random trajectories through the scenes. We generated three different types of trajectories, (**type-1**): two-body random [17] with height constraints; (**type-2**): hand-held; and (**type-3**): look-forward. For each one we have two parameters which can be selected to change the overall speed and angular velocity of the motion. To prevent overly smooth unrealistic trajectories, we augment each of these base trajectories with a learned style model which produces realistic appearing camera jitter.

5.1 Base Random Trajectories

Our random trajectories follow the basic two-body random trajectory system described in [17]. It simulates two physical bodies randomly moving around and colliding within the indoor free-space. One defines the camera position, and the other the look-at point. To produce more realistic views (as from a person), the minimum and maximum height is limited to be between 1 m and 2 m. We also used a slightly randomized up direction with a range from 0° to 5° difference out of gravity direction. We improve upon this model by creating two additional modes of trajectory generation.

The first one is designed to simulate a bias in hand-held cameras that tend to look downwards, by restricting the look-at point to be lower than the camera position. The second is designed to provide translational motion along the view direction, a motion type not frequently encountered in the basic two-body system. To achieve this, and give a smooth trajectory, a target point is set in the direction of the velocity from the camera motion, and a proportional force is applied to the current look-at point to smoothly correct the displacement to the current velocity vector. We also bias the random forces to be in the hemisphere of the velocity vector, to create a smoother look-at target. We also systematically varied the trajectory parameters. Unlike in [17], where a single set of parameters governed every trajectory, we picked random uniform velocity multiplier from 0.5-5.0 and a random angular velocity scalar from 0.5-3.0 for each trajectory. This feature provides more variety amongst the trajectories.

5.2 Realistic Trajectories

An important but neglected factor in existing synthetic datasets for scene understanding is the realism of the trajectory models. Previous work on automated trajectories have used simple and smooth trajectories [17]. Although these trajectories adequately explore the extents of the scene, they do not capture the nuances of real camera motion which can have a significant effect on the visual data. For example, the repetitive impulses in acceleration incurred by the steps of a walking cameraman inevitably adds a significant amount of motion blur to the sequence – especially in low light indoor conditions. In other papers [14], motion-capture trajectories have been captured to solve this – but such an approach is limited to small datasets where it is feasible to manually capture such trajectories. In this paper, we therefore use a data-driven method to automatically synthesize millions of realistic camera trajectories which satisfy the constraints of the scenes. In particular we harness the recently proposed *Wavenet* architecture [21]. The model takes as input a window of previous velocities and outputs the next velocity – in this manner, synthesizing a trajectory of arbitrary length . To ensure that the synthesized trajectory does not collide with objects in the scene, we apply the force model from the preceding Section to the velocities to avoid collisions, and to constrain the overall motion to that of the base random trajectory. We trained the generative model on trajectories from [28] and [29] and synthesized trajectories on the fly during rendering.

6 *ExaRender*: Photo-realistic RGB Renderer

We have developed a fast photo-realistic renderer on top of *Embree* [30], an open-source collection of ray-tracing kernels for x86 CPUs. We also have an equivalent variation supporting GPU acceleration to make full use of our GPU and CPU clusters. Our renderer, coined *ExaRender*, supports a common subset of *Path Tracing* operations of leading commercial renderers, and provides flexible APIs to extensively customize the workflow.

6.1 Path Tracing

We use the well-known *Path Tracing* [23] approach to deliver high quality image rendering. Path tracing is a *Monte Carlo* method that approximates realistic Global Illumination (*GI*) for rendering images. It simulates many real-world effects such as soft shadows, depth of field, motion blur and indirect lighting. We also support common color bleeding from diffuse surfaces and caustics. Comparing to other popular *GI* approaches, e.g. *Photon Mapping* [15], *Path Tracing* provides a more realistic caustic effect, requires less memory footprint for large-scale rendering, and is efficient to support dynamic scenes.

6.2 Sensors Simulation: RGB-D Camera, IMU and Event Camera

Our renderer supports the pinhole camera model and several lens models such as perspective, depth of field, fisheye and panorama. We used a fixed resolution at 640$\times$480 pixels, and a fixed focus lens at 600 pixels for all the RGB images. We used 5,000$\times$2,500 resolution for panorama and 600$\times$600 for fisheye images. Our renderer simulates camera motion blur by following the method from [17]; the system takes into account all incoming rays throughout the shutter opening time interval, then integrates the irradiance during rendering.

For inertial measurements, we fit a cubic B-spline to given control poses (described above), to represent position and orientation, the latter parameterized as a rotation vector. The continuous-time representation lends itself ideally to obtain IMU readings by computing time derivatives at an arbitrary sample rate. Specifically, position (expressed in the coordinates of the static world) is differentiated twice w.r.t. time, then acceleration due to gravity is added, and finally the vector is rotated into the frame of reference of the IMU to generate ground truth accelerometer measurements; ground truth rotation rate measurements are obtained from the rotation vector time derivative, where the relationship between the two is given e.g. in [2]. We provide IMU readings at 800 Hz (ground truth, or optionally noisy).

Our rendering framework can also be used to generate output of novel camera designs such as event cameras, e.g. [16]. We can achieve this by rendering at lower resolution but

No	Length(m)	(v,ω, type)	ATE(m)	Description
1	21.93	(1,1,1)	0.0428	a sample model
2	22.19	(1,1,1)	0.0352	.. with different lighting
3	21.84	(1,1,1)	0.0515	.. with objects displaced
4	13.88	(9,9,1)	0.1701	16% tracked
5	20.83	(5,6,1)	0.0454	39% tracked
6	17.46	(1,5,1)	0.0172	type-1
7	22.67	(1,1,2)	0.0193	type-2
8	4.79	(1,1,3)	0.3840	type-3, 11% tracked

Table 1: Absolute trajectory error (ATE) for sample sequences when running ORBSLAM2.0. For trajectories, v and w are maximum position and angular velocities in metric units, with the type explained in Sec. 5: (type-1): two-body random; (type-2): hand-held; and (type-3): look-forward.

very high frame rate; then, we assess per-pixel brightness change over time and output events with interpolated timestamps whenever the user-settable intensity threshold is crossed.

Our dataset contains around 20M images which required significant computational power for rendering. To achieve this, our renderer is implemented to be compatible for both CPU and GPU platforms, as well as to support dynamic render distribution onto multiple servers. In this context, we rendered the images on a cluster of 1,300 Nvidia GTX TitanX GPUs and 2,000 Intel Xeon Phi CPUs for around 4 days. We applied 256 samples per pixel (*SPP*) – the most important parameter to trade-off between image quality and rendering speed.

6.3 *ViSim*: An Interactive Simulator

We implemented an interactive simulator, coined *ViSim*, (Fig. 2 (D)) to assist creating monocular or stereo camera trajectories, ground truth IMU readings and events given a layout from our database or from any other 3D scenes with *obj* format e.g. SUNCG [28] and SceneNet RGB-D [17] etc. The simulator provides a user-friendly interface while adding our simulation functionality behind. It conntains a *Design View* and a collection of *Parameter Options* to configure camera calibration, distortion coefficients, stereo view, frame rate and travel time, etc. We also support exporting the camera trajectory into other formats e.g. SLAMBench [19], SLAMBench2 [7], EuRoC [8], Freiburg [29] and ICL-NUIM [13], and also renderer formats such as *OppositeRenderer* (SceneNet RGB-D) [17].

7 Evaluation on Simultaneous Localization and Mapping

To verify the quality of the dataset, we have selected several sequences of images of different trajectory types to run *ORBSLAM2.0* [18] and *ElasticFusion* [31]. The verification test uses the RGB-D mode of *ORBSLAM2.0*, using the default configuration parameters. For each sequence, the absolute trajectory error (*ATE*) is calculated [29].

Fig. 6 (**Left**) shows the ground truth and estimated trajectories for a randomly chosen type-1 trajectory with a randomly chosen model, in three different scenarios: a regular scene (**Left Top image**), the same scene with different lighting (**Left Middle image**), and the same scene but with objects displaced (**Left Bottom image**). Trajectory estimation errors for these cases are given in Table 1, rows 1, 2, and 3 respectively. Note that the dense reconstruction is generated by the *ElasticFusion* [31] algorithm. In the same table, rows 4 and 5 show two other type-1 trajectories with very high position and angular velocities (v and ω respectively). *ORBSLAM2.0* is not able to track all frames, indicating the challenge that the trajectories provide. Rows 5, 6, and 7 show ATE for other types of trajectories with different difficulty levels.

Additionally, another random trajectory (length of 24.91 m) was used with several random models. The generated sequences were used to evaluate ORBSLAM2.0. The average

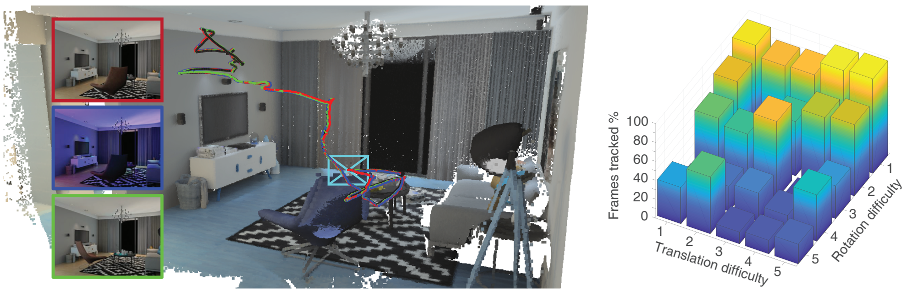

Figure 6: (**Left**) SLAM under change in lighting and object rearrangement. Estimated trajectories along with the ground truth trajectory are shown overlaid on the dense reconstruction (**Right**) Percentage of the frames tracked for 50 trajectories, running *ORBSLAM2.0*.

ATE was 0.0345 m, with standard deviation of 0.02 m amongst different scenes. This indicates that the variation in the models affects the results. To further demonstrate the challenge levels in trajectories, Fig. 6 (**Right**) shows the percentage of the tracked frames by *ORB-SLAM2.0* for 50 different trajectories with different difficulty levels based on maximum position and angular velocities. As the difficulty level increases, the percentage of the tracked frames drops.

8 Conclusions

We have presented a very large synthetic dataset of indoor video sequences that accesses millions of interior design layouts, furniture and object models which were all professionally designed to a highest specification. We then provide variability in terms of lighting and object rearrangement to further devise our scenes and simulate the environment of daily life. As a result, we obtain highly photo-realistic footage at a high frame-rate. Furthermore, a large variety of different trajectory types was synthesized, as we believe the temporal aspect should be given closer attention. We demonstrate the usefulness of our dataset by evaluating SLAM algorithms.

In this work, we configured lighting and scene changes in a random fashion due to lack of real-world ground truth for lighting and scene changes. Also, the scene rearrangement was obtained via a physics engine accessing physical parameters e.g. mass, size, friction coefficient etc. Alternatively, a data-driven approach could be used – which we leave as future work.

9 Acknowledgements

We would like to thank `Kujiale.com` for providing their database of production furniture models and layouts, as well as access to their GPU/CPU clusters. We also thank the Kujiale artists and other professionals for their great efforts into editing and labelling millions of models and scenes. We also highly appreciate the comments and technical support from *Kujiale Rendering Group*, as well as helpful discussions and comments from Prof. Andrew Davison and other members of *Robot Vision Group* of Imperial College London. This research is supported by the EPSRC grants PAMELA EP/K008730/1, Aerial ABM EP/N018494/1, and Imperial College London.

References

[1] Project Chrono. https://projectchrono.org/.

[2] Kindr Library. https://docs.leggedrobotics.com/kindr/cheatsheet_latest.pdf.

[3] Planner 5D. https://planner5d.com/.

[4] POV-Ray. http://www.povray.org/.

[5] VRay. https://www.chaosgroup.com/.

[6] P. Ammirato, P. Poirson, E. Park, J. Košecká, and A. C. Berg. A dataset for developing and benchmarking active vision. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1378–1385, 2017.

[7] Bruno Bodin, Harry Wagstaff, Sajad Saeedi, Luigi Nardi, Emanuele Vespa, John H Mayer, Andy Nisbet, Mikel Lujan, Steve Furber, Andrew J Davison, Paul H.J. Kelly, and Michael O'Boyle. SLAMBench2: Multi-objective head-to-head benchmarking for visual SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3637–3644, 2018.

[8] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The EuRoC micro aerial vehicle datasets. *International Journal of Robotics Research (IJRR)*, 35(10):1157–1163, 2016.

[9] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[10] Benjamin Choo, Michael Landau, Michael DeVore, and Peter A Beling. Statistical analysis-based error models for the Microsoft Kinect$^{\text{TM}}$ depth sensor. *Sensors*, 14(9): 17430–17450, 2014.

[11] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017.

[12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2012.

[13] A. Handa, T. Whelan, J.B. McDonald, and A.J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1524–1531, 2014.

[14] A. Handa, V. Patraucean, S. Stent, and R. Cipolla. SceneNet: An annotated model generator for indoor scene understanding. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5737–5743, 2016.

[15] Henrik Wann Jensen, Per H Christensen, Toshiaki Kato, and Frank Suykens. A practical guide to global illumination using photon mapping. *SIGGRAPH 2002 Course Notes CD-ROM*, 2002.

[16] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128x128 120 db 15us latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, 43(2):566–576, 2008.

[17] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J Davison. SceneNet RGB-D: 5M photorealistic images of synthetic indoor trajectories with ground truth. In *International Conference on Computer Vision (ICCV)*, pages 2697–2706.

[18] R. Mur-Artal and J. D. TardÃşs. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[19] Luigi Nardi, Bruno Bodin, M. Zeeshan Zia, John Mawer, Andy Nisbet, Paul H. J. Kelly, Andrew J. Davison, Mikel Luján, Michael F. P. O'Boyle, Graham Riley, Nigel Topham, and Steve Furber. Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5783 – 5790, 2015.

[20] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *European Conference on Computer Vision (ECCV)*, pages 746–760, 2012.

[21] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[22] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[23] Timothy J Purcell, Ian Buck, William R Mark, and Pat Hanrahan. Ray tracing on programmable graphics hardware. 21(3):703–712, 2002.

[24] Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. Human-centric indoor scene synthesis using stochastic grammar. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5899–5908, 2018.

[25] Weichao Qiu and Alan Yuille. UnrealCV: Connecting computer vision to Unreal Engine. *arXiv preprint arXiv:1609.01326*, 2016.

[26] Manolis Savva, Angel X. Chang, Gilbert Bernstein, Christopher D. Manning, and Pat Hanrahan. On being the right scale: Sizing large collections of 3D models. In *SIGGRAPH Asia 2014 Indoor Scene Understanding Where Graphics Meets Vision*, .

[27] Manolis Savva, Angel X. Chang, and Pat Hanrahan. Semantically-enriched 3D models for common-sense knowledge. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops, 2015*, .

[28] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[29] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. pages 573–580, 2012.

[30] Ingo Wald, Sven Woop, Carsten Benthin, Gregory S Johnson, and Manfred Ernst. Embree: a kernel framework for efficient CPU ray tracing. *ACMTOG*, 33(4):143, 2014.

[31] Thomas Whelan, Stefan Leutenegger, Renato F Salas-Moreno, Ben Glocker, and Andrew J Davison. ElasticFusion: Dense SLAM without a pose graph. In *Robotics: Science and Systems (RSS)*, 2015.

[32] Yi Zhang, Weichao Qiu, Qi Chen, Xiaolin Hu, and Alan Yuille. Unrealstereo: A synthetic dataset for analyzing stereo vision. *arXiv preprint arXiv:1612.04647*, 2016.

[33] Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5057 – 5065, 2017.