Learning from synthetic data generated with GRADE

Elia Bonetto[*,†] *Student Member, IEEE,* Chenghao Xu[‡,*], and Aamir Ahmad[†,*] *Senior Member, IEEE*

***Abstract*—Recently, synthetic data generation and realistic rendering has advanced tasks like target tracking and human pose estimation. Simulations for most robotics applications are obtained in (semi)static environments, with specific sensors and low visual fidelity. To solve this, we present a fully customizable framework for generating realistic animated dynamic environments (GRADE) for robotics research, first introduced in [1]. GRADE supports full simulation control, ROS integration, realistic physics, while being in an engine that produces high visual fidelity images and ground truth data. We use GRADE to generate a dataset focused on indoor dynamic scenes with people and flying objects. Using this, we evaluate the performance of YOLO and Mask R-CNN on the tasks of segmenting and detecting people. Our results provide evidence that using data generated with GRADE can improve the model performance when used for a pre-training step. We also show that, even training using only synthetic data, can generalize well to real-world images in the same application domain such as the ones from the TUM-RGBD dataset. The code, results, trained models, and the generated data are provided as open-source at https://eliabntt.github.io/grade-rr.**

Fig. 1: An example of one of the generated environments with overlays of the robots used in our experiments.

I. INTRODUCTION

An ideal simulation for developing, testing and validating intelligent robotics systems should have four main characteristics: i) physical realism, ii) photorealism, iii) full controllability, and iv) the ability to simulate dynamic entities.

Addressing all these issues, we developed a solution for Generating Realistic Animated Dynamic Environments — GRADE [1]. GRADE is a flexible, fully controllable, customizable, photorealistic, ROS-integrated framework to simulate and advance robotics research. We employ tools from the computer vision community, such as path-tracing rendering and material reflections, while keeping robotics in our focus. We employ NVIDIA Isaac Sim[1] and the Omniverse[2] suite. With these tools, we sought to solve all of the above issues by i) creating a general pipeline that can produce visually realistic data for general and *custom* robotics research, ii) developing and making available a set of functions, tools, and easy-to-understand examples, that can be easily expanded and adapted, to allow broad adoption and ease the learning curve of the Isaac Sim software. To demonstrate the effectiveness of the method, we: i) generated an **indoor** dynamic environment dataset by using only freely available assets. To demonstrate the visual realism of the simulation, we ii) perform various tests with YOLOv5 [2] and Mask R-CNN [3] by evaluating their performances after training and fine-tuning with our synthetic data. We then evaluate these models with the COCO dataset and some popular dynamic sequences of the TUM RGBD dataset [4] that we manually labelled on the tasks of segmenting and detecting people. With this, we show that pre-training with our synthetic data can outperform the baseline models on the COCO dataset. By evaluating our models with the TUM dataset, we show how our *synthetic* data obtain comparable performance even without *any* fine-tuning on real images.

[*]Max Planck Institute for Intelligent Systems, Tübingen, Germany. firstname.lastname@tuebingen.mpg.de

[†]Institute of Flight Mechanics and Controls, University of Stuttgart, Stuttgart, Germany. firstname.lastname@ifr.uni-stuttgart.de

[‡]Faculty of Mechanical, Maritime and Materials Engineering, Department of Cognitive Robotics, Delft University of Technology, Delft, Netherlands.

The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Elia Bonetto.

[1]https://developer.nvidia.com/isaac-sim

[2]https://www.nvidia.com/en-us/omniverse/

II. RELATED WORK

In this section, we focus on two components: dynamic content of the scene, and the simulation engines.

The majority of the dynamic content on a scene comes from humans and objects. **Humans.** The most widely adopted method to describe a human body pose and shape is SMPL [5]. Using that, any motion, synthetic or real, can be seen as a deformable and untextured mesh. Real-world SMPL fittings are obtainable only in controlled environments, e.g. through a VICON or MOCAP systems [6]. These are limited in the number of subjects, clothing variety, and scenarios. Synthetic data is being used to solve these problems [7], [8]. However, such data does not include either the full camera's state, IMU readings, scene depth, LiDAR data, or offers the possibility to easily extend it after the experiment has been recorded (e.g. with additional cameras or sensors), thus is generally unusable for any robotics application. Indeed, synthetic datasets are usually developed by stitching people over image backgrounds [8], statically placing them in some limited environment [7] and often recorded with static monocular

cameras that take single pictures [7]. Furthermore, many of those are generated without any clothing information [9]. Few datasets, like Cloth3D [10], provide simulated clothed humans with SMPL fittings usable in other simulations. Commercial solutions like RenderPeople[3] or clo|3d[4] exist and would be applicable. However, using such assets limits the possibility of reproduction and re-distributing data. **Dynamic objects.** There are several datasets about objects [11], being those scanned or actual CAD models/meshes. Among those, Google Scanned Objects [11], with its realistically looking household objects, and ShapeNet [12], with its variety, are two good and popular examples of datasets that represent these two categories.

Simulation engines. Gazebo is currently the standard for robotic simulation. High reliable physics and tight integration with ROS are its key advantages. However, the lack of visual realism and customization possibility greatly limits its usability for problems which require more flexibility, personalization, and photorealism. Indeed, alternatives emerged in the latest years, such as [13], [14], [15], [16], [17], [18]. AirSim is one of the first that sought to bridge this gap by working with Unreal Engine. With that, synthetic datasets like TartanAir [19] and AirPose [7] have been developed. [19] is a challenging dataset for visual odometry. However, the generation pipeline and the assets are not publicly available. BenchBot is based on Isaac Sim. However, it has been developed as a benchmarking tool for existing algorithms, and while it is expandable with add-ons those are limited by their own exposed APIs. GRADE [1] is also built directly upon Isaac Sim. However, when compared to BenchBot, it presents a broader focus including both data generation and general robotic testing while exposing how researchers can easily adapt the simulation to their needs, including interactions with objects, visual settings (e.g. fog, time of day) and others.

III. Approach

Using the system introduced in our previous work GRADE [1], we generate an indoor dynamic environment dataset with flying objects and dynamic humans. The details about the GRADE framework and of the simulation management are thoroughly described in [1].

A. Environments

There exist only a few publicly available indoor environment datasets that have realistic lighting and textured meshes. The only *free* viable solution we found is the 3D-Front [20] dataset. The environments are randomized with *ambientCG*[5] textures, and with random light colors and intensity.

B. Dynamic assets

Humans: In our work, we decided to employ human animations from two main datasets. The first one, Cloth3D, comprises various animated sequences with clothed SMPL models. The second one is AMASS's SMPL *unclothed* fittings over the CMU dataset. We randomize the appearance of the assets by using Surreal's SMPL textures, which are, although low-resolution, also freely available. We then load the animation sequence in Blender and export it as a USD file with a tool that we developed. **Objects:** We use two sources of additional objects in our simulation, namely Google Scanned Objects and ShapeNet. This increases variability in the simulation and more difficult scenarios. Those objects are treated as random flying objects. For simplicity, we do *not* restrict those objects to not collide with other parts of the environment.

C. Placement of dynamic assets

To place the humans, we utilize the STLs of the environment and of each animation sequence to check if there is a collision between any given couple of STL meshes. Animated humans are, in this way, randomly placed within the environment. Flying objects are loaded and randomly rigidly animated, without considering any possible collision.

D. Data collection strategy

The number of humans and dynamic objects are randomized once for each experiment, as described in [1]. We simulate a freely moving camera that automatically explores the environments by using a drone model controlled by six independent joints and an active SLAM framework [21]. The initial location of the robot is randomized within the environment. Each experiment lasts 60 seconds, yielding 1800 frames (30 FPS). For each frame we have, among other things, ground-truth instance segmentation and bounding boxes information for each human instance.

IV. Evaluations

A. Human detection

To train YOLO and Mask R-CNN we use both a subset of GRADE, which we call S-GRADE, and the full GRADE dataset (A-GRADE). Images with a high probability of being occluded by flying objects are automatically discarded by using their depth and color information [1]. S-GRADE has 18K frames, of which 16.2K have humans in them and 1.8K are background. The train set has 16K images while the validation set 2K. S-GRADE contains only sequences without flying objects and the additionally generated scenarios (see [1]), with added blur (based on the IMU information), random rolling shutter noise ($\mu = 0.015$, $\sigma = 0.006$), and a fixed exposure time of 0.02 s to the RGB data. A-GRADE consists of all available data, with images containing flying objects and the additional scenarios (incl. the *single* outdoor sequence). To A-GRADE we add noise with the same parameters as in S-GRADE but with a random exposure time between 0 and 0.1 seconds. For A-GRADE we also correct the segmentation masks and bounding boxes to account for the additional blur. This is not necessary in the case of S-GRADE data. A-GRADE is made of 591K images (80/20 train/val split) out of which 362K have humans.

We took a random subset of the COCO dataset, which we call **S-COCO**, counting 1256 training and 120 validation

[3] https://renderpeople.com/
[4] https://www.clo3d.com/
[5] https://ambientcg.com/

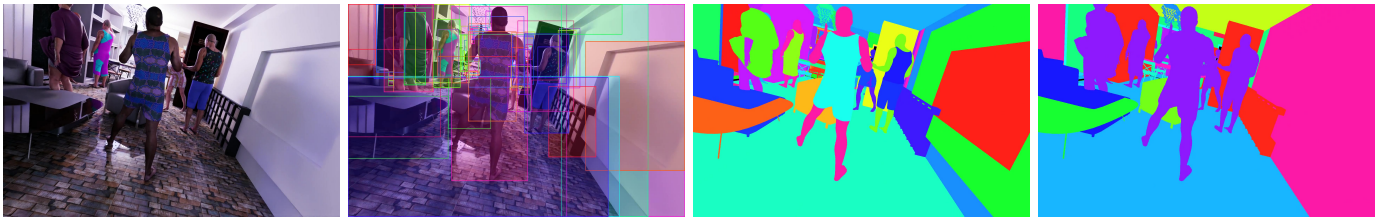

Fig. 2: An example of the generated data following the pipeline described in Sec. III, using one of the 3D-Front environments and dynamic humans. From the left, we can see the rendered RGB image, 2D bounding boxes, semantic instances, and semantic segmentation. Best viewed in color.

images, totalling for $\sim 2\%$ and $\sim 4\%$ of COCO, used to understand how the networks perform with limited real data. For evaluation, we use both the subset of COCO which contains humans in the frame (called COCO from now on), and the *fr3_walking* sequences of the TUM dataset. Those sequences are related to a scenario that has a high similarity to our dataset, thus making them a good testing metric. Our BASELINEs are the models of networks *pre-trained* with the COCO dataset. We evaluate the performance with the COCO standard metric (mAP@[.5, .95], AP in this work) and the PASCAL VOC's metric (mAP@.5, AP50 in this work). Both networks are trained from scratch with S-GRADE and A-GRADE, without using any pre-trained weight or real-world data, and S-COCO. We then fine-tune S-GRADE and A-GRADE by using both S-COCO and COCO using their *corresponding* validation sets, unlike [8].

1) YOLOv5: was trained for a maximum of 300 epochs with random initial weights and no hyperparameters tuning. The results can be seen in Tab. I. As expected, when we evaluate the performance on COCO with the network trained from scratch using S-GRADE we obtain much lower precision when compared even to the model trained only on S-COCO. However, when the model trained on S-GRADE is then fine-tuned on S-COCO, we see an increased AP for both metrics of around 6%. Interestingly, when tested with the TUM data, we can see how S-GRADE performs similarly to the S-COCO, resulting in a $\sim 5\%$ lower AP50 but in a 6% higher AP. Once fine-tuned, AP increases $\sim 6\%$ and AP50 $\sim 7\%$ compared to S-COCO when tested against COCO itself, and 8% in AP50 and 12% in AP when tested with TUM. In both cases, when fine-tuned on the full COCO, the performances overcome the ones of the original pre-trained network. This is more noticeable when considering the COCO dataset ($\sim 5\%$). Comparing now A-GRADE and S-GRADE we can notice how YOLO overfit on the task of indoor human detection. Indeed, while networks (pre)trained on A-GRADE perform better when evaluated on the TUM dataset, they exhibit comparable or worse performance when tested with COCO. In our opinion, this may also be linked to the huge amount of specialized data of A-GRADE. This is also suggested by the tests we performed using the 50th training epoch checkpoint, identified as E50 in Tab. I. We can see how A-GRADE-E50 performs consistently better than A-GRADE, S-GRADE and S-GRADE-E50 in all metrics and dataset. Moreover, A-GRADE-E50, when tested on TUM data, performs better than models trained from scratch on both synthetic and real data, as well as models fine-tuned on S-COCO. However, using this checkpoint as pre-training starting point, yields performance improvements only when used with S-COCO.

	COCO		TUM	
	AP50	AP	AP50	AP
BASELINE	0.753	0.492	0.916	0.722
S-COCO	0.492	0.242	0.661	0.365
S-GRADE	0.206	0.109	0.616	0.425
S-GRADE-E50	0.234	0.116	0.683	0.431
A-GRADE	0.176	0.093	0.637	0.459
A-GRADE-E50	0.282	0.154	0.798	0.613
S-GRADE + S-COCO	0.561	0.302	0.744	0.488
A-GRADE + S-COCO	0.540	0.299	0.762	0.514
A-GRADE-E50 + S-COCO	0.558	0.314	0.808	0.565
S-GRADE + COCO	0.801	0.544	0.931	0.778
A-GRADE + COCO	0.797	0.542	0.932	0.786
A-GRADE-E50 + COCO	0.797	0.543	0.932	0.777

TABLE I: YOLOv5 bounding box evaluation results.

2) Mask R-CNN: We use the *detectron2* implementation of Mask R-CNN, using a 3x training schedule[6] and a ResNet50 backbone. We used the default steps (210K and 250K) and maximum iterations (270K) parameters when training A-GRADE and COCO, while reducing them respectively to 60K, 80K and 90K when training S-GRADE and to 80K, 108K and 120K for S-COCO. We evaluate the models every 2K iterations and save the best one by comparing the AP50 metric on the given task. Due to the size of A-GRADE, we opted to evaluate the model trained from scratch on this data every 3k iterations and that this may be a sub-optimal training schedule considering the size of the dataset. Note that by default, and differently from YOLO, Mask R-CNN does not use images without segmentation targets. We then test both the bounding box and the instance segmentation accuracy using those models with a confidence threshold of both 0.70. The results, depicted in Tab. II and Tab. III, allow us to make similar considerations to the ones that we have done above. The main difference is that, in this case, we are not able to surpass the baseline results. However, we argue that this may be related to the training and the evaluation schedule, which greatly impacts the results on this network[7]. These specify the frequencies of the evaluations on the validation set, the learning rate value and its decay, and are tied to both the size of the dataset and the number of GPUs used. Indeed, when we trained from scratch the model with the COCO data, C-BASELINE in the table, we obtained lower performance with respect to both the released baseline model and our fine-tuned models, successfully showing the usefulness of our synthetic data.

[6] https://github.com/facebookresearch/detectron2/blob/main/configs/Misc/scratch_mask_rcnn_R_50_FPN_3x_gn.yaml

[7] https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md

	COCO		TUM	
	AP	AP50	AP	AP50
BASELINE	0.495	0.716	0.716	0.886
S-COCO	0.161	0.340	0.250	0.526
S-GRADE	0.064	0.128	0.312	0.563
A-GRADE	0.115	0.202	0.502	0.727
S-GRADE + S-COCO	0.232	0.428	0.412	0.708
A-GRADE + S-COCO	0.262	0.450	0.489	0.736
S-GRADE + COCO	0.474	0.693	0.679	0.858
A-GRADE + COCO	0.489	0.714	0.696	0.869
C-BASELINE	0.471	0.693	0.653	0.829

TABLE II: Mask R-CNN bounding boxes evaluation results. Thr. **0.7**

	COCO		TUM	
	AP	AP50	AP	AP50
BASELINE	0.432	0.705	0.674	0.887
S-COCO	0.155	0.351	0.231	0.543
S-GRADE	0.043	0.100	0.264	0.509
A-GRADE	0.088	0.178	0.408	0.709
S-GRADE + S-COCO	0.195	0.401	0.374	0.665
A-GRADE + S-COCO	0.231	0.460	0.449	0.758
S-GRADE + COCO	0.415	0.682	0.611	0.858
A-GRADE + COCO	0.430	0.710	0.638	0.869
C-BASELINE	0.410	0.681	0.584	0.838

TABLE III: Mask R-CNN instance segmentation evaluation results. Thr. **0.7**

Another difference is that, when using A-GRADE, we are consistently better than the corresponding model (pre)trained on S-GRADE in both datasets. In the tasks of people detection A-GRADE shows similar performance to S-COCO on the COCO dataset and way better results on the TUM dataset. Indeed, the performance is $\sim 2\%$ better on COCO and $3-5\%$ better on TUM when compared to C-BASELINE.

In Tab. IV and Tab. V we report the results of the same models with a threshold value of 0.05 as done in [8]. Although they report only bounding boxes results and use a slightly different network model, we can still draw some conclusions. Indeed, notice how our training results on S-COCO and COCO (C-BASELINE on the tables) are comparable to theirs in terms of AP and AP50. The differences are most probably linked to the difference between the dataset size when comparing S-COCO and the number of training steps when considering COCO. However, we can see that both S-GRADE and A-GRADE greatly outperform PeopleSansPeople synthetic data, obtaining a remarkable +3-8% in the small version of the dataset and a +7-12%, despite the fact that our data is solely focused on indoor environments. This, with much shorter training procedures, i.e. 270K iterations for A-GRADE as opposed to 4M of the biggest synthetic set of [8]. Furthermore, while it is true that their increment when using a subset of COCO as fine-tuning dataset is noticeable (around 30%), and much greater than ours, the improvement they exhibit when using both the full dataset and full COCO is just 0.7%, almost half of our 1.3%. However, in addition to the longer training procedure, we must also account for the fact that the validation set used in [8] is the *full* validation set. Thus, their saved model, i.e. the best-performing model on the validation set on the given metric, is saved according to the full COCO validation set, while ours are saved based on the reduced validation set of just 120 images.

	COCO		TUM	
	AP	AP50	AP	AP50
BASELINE	0.556	0.841	0.736	0.920
S-COCO	0.195	0.439	0.282	0.610
S-GRADE	0.077	0.167	0.343	0.637
A-GRADE	0.140	0.269	0.531	0.784
S-GRADE + S-COCO	0.265	0.518	0.432	0.748
A-GRADE + S-COCO	0.303	0.560	0.515	0.788
S-GRADE + COCO	0.539	0.833	0.713	0.916
A-GRADE + COCO	0.550	0.843	0.728	0.916
C-BASELINE	0.537	0.829	0.692	0.898

TABLE IV: Mask R-CNN bounding boxes evaluation results. Thr. **0.05**

	COCO		TUM	
	AP	AP50	AP	AP50
BASELINE	0.479	0.817	0.692	0.922
S-COCO	0.168	0.392	0.241	0.568
S-GRADE	0.048	0.117	0.283	0.561
A-GRADE	0.100	0.214	0.425	0.749
S-GRADE + S-COCO	0.216	0.465	0.387	0.694
A-GRADE + S-COCO	0.247	0.515	0.458	0.780
S-GRADE + COCO	0.467	0.805	0.633	0.905
A-GRADE + COCO	0.476	0.813	0.660	0.908
C-BASELINE	0.461	0.801	0.611	0.890

TABLE V: Mask R-CNN instance segmentation evaluation results. Thr. **0.05**

3) Considerations: Testing over COCO is, in our opinion, not fair since we lack crowded scenes, outdoor scenarios with humans placed in the background, and diversified clothing in the assets we use (i.e. we do not have humans wearing ski suits or helmets). Thus, when testing against COCO, we are using a model trained on an indoor dataset to evaluate its performance on not comparable data. Indeed, we see how our synthetic data generalize well to the real world if we consider the TUM dataset, which instead consists of sequences more related to the one that we generate.

V. CONCLUSIONS

In this work, we presented a novel framework, named GRADE, to simulate multiple robots in realistically looking dynamic environments. GRADE is a flexible system that covers all the steps necessary to do that, from the generation of the single assets to fine simulation management, from placement of said assets to post-processing of the data. With GRADE we generated a dataset of indoor dynamic environments and used that to i) show how our synthetic data alone can be used for training a good indoor human detection model, and ii) to improve the performance of both YOLO and Mask R-CNN when used for pre-training. This holds even though the current quality of the assets is not optimal due to the choice of using only freely available ones. We believe that adopting commercial solutions for environments and/or dynamic humans will greatly increase the quality of the generated data. Finally, we demonstrate how our data is, in principle, better than the one introduced by PeopleSansPeople, both when used as pre-training data and when adopted as-is, despite being focused solely on indoor scenarios. This is by yielding performance improvements that range between 1.3% to 12% while using up to 10 times shorter training procedures. Finally, all our work is available as open source and based solely on open-source assets.

References

[1] E. Bonetto, C. Xu, and A. Ahmad, "GRADE: Generating realistic animated dynamic environments for robotics research," *arXiv preprint arXiv:2303.04466*, 2023.

[2] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, K. Michael, TaoXie, J. Fang, imyhxy, Lorna, Z. Yifu, C. Wong, A. V, D. Montes, Z. Wang, C. Fati, J. Nadar, Laughing, UnglvKitDe, V. Sonck, tkianai, yxNONG, P. Skalski, A. Hogan, D. Nair, M. Strobel, and M. Jain, "ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation," Nov. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7347926

[3] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[4] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[5] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "Smpl: A skinned multi-person linear model," *ACM Trans. Graph.*, vol. 34, no. 6, nov 2015. [Online]. Available: https://doi.org/10.1145/2816795.2818013

[6] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black, "AMASS: Archive of motion capture as surface shapes," in *International Conference on Computer Vision*, Oct. 2019, pp. 5442–5451.

[7] N. Saini, E. Bonetto, E. Price, A. Ahmad, and M. J. Black, "Airpose: Multi-view fusion network for aerial 3d human pose and shape estimation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4805–4812, 2022.

[8] S. E. Ebadi, S. Dhakad, S. Vishwakarma, C. Wang, Y.-C. Jhang, M. Chociej, A. Crespi, A. Thaman, and S. Ganguly, "Psp-hdri+: A synthetic dataset generator for pre-training of human-centric computer vision models," in *First Workshop on Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022*, 2022.

[9] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid, "Learning from synthetic humans," in *CVPR*, 2017.

[10] H. Bertiche, M. Madadi, and S. Escalera, "Cloth3d: Clothed 3d humans," in *European Conference on Computer Vision*. Springer, 2020, pp. 344–359.

[11] L. Downs, A. Francis, N. Koenig, B. Kinman, R. Hickman, K. Reymann, T. B. McHugh, and V. Vanhoucke, "Google scanned objects: A high-quality dataset of 3d scanned household items," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 2553–2560.

[12] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.

[13] B. Talbot, D. Hall, H. Zhang, S. R. Bista, R. Smith, F. Dayoub, and N. Sünderhauf, "Benchbot: Evaluating robotics research in photorealistic 3d simulation and on real robots," 2020.

[14] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017. [Online]. Available: https://arxiv.org/abs/1705.05065

[15] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, M. Deitke, K. Ehsani, D. Gordon, Y. Zhu, A. Kembhavi, A. Gupta, and A. Farhadi, "AI2-THOR: An Interactive 3D Environment for Visual AI," *ArXiv*, vol. abs/1712.05474, 2017.

[16] B. Shen, F. Xia, C. Li, R. Martín-Martín, L. Fan, G. Wang, C. Pérez-D'Arpino, S. Buch, S. Srivastava, L. P. Tchapmi, M. E. Tchapmi, K. Vainio, J. Wong, L. Fei-Fei, and S. Savarese, "igibson 1.0: a simulation environment for interactive tasks in large realistic scenes," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, p. accepted.

[17] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[18] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem, "Sim4cv: A photo-realistic simulator for computer vision applications," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 902–919, Mar. 2018. [Online]. Available: https://doi.org/10.1007/s11263-018-1073-7

[19] W. Wang, D. Zhu, X. Wang, Y. Hu, Y. Qiu, C. Wang, Y. Hu, A. Kapoor, and S. Scherer, "Tartanair: A dataset to push the limits of visual slam," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[20] H. Fu, B. Cai, L. Gao, L.-X. Zhang, J. Wang, C. Li, Q. Zeng, C. Sun, R. Jia, B. Zhao *et al.*, "3d-front: 3d furnished rooms with layouts and semantics," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 933–10 942.

[21] B. Zhou, Y. Zhang, X. Chen, and S. Shen, "Fuel: Fast uav exploration using incremental frontier structure and hierarchical planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 779–786, 2021.