2023 IEEE International Conference on Robotics and Automation (ICRA)
Workshop on Active Methods in Autonomous Navigation
29 May – 2 June, 2023, London, UK

Simulation of Dynamic Environments for SLAM

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

***Abstract*— Simulation engines are widely adopted in robotics. However, they lack either full simulation control, ROS integration, realistic physics, or photorealism. Recently, synthetic data generation and realistic rendering has advanced tasks like target tracking and human pose estimation. However, when focusing on vision applications, there is usually a lack of information like sensor measurements or time continuity. On the other hand, simulations for most robotics tasks are performed in (semi)static environments, with specific sensors and low visual fidelity. To solve this, we introduced in our previous work a fully customizable framework for generating realistic animated dynamic environments (GRADE) [1]. We use GRADE to generate an indoor dynamic environment dataset and then compare multiple SLAM algorithms on different sequences. By doing that, we show how current research over-relies on known benchmarks, failing to generalize. Our tests with refined YOLO and Mask R-CNN models provide further evidence that additional research in dynamic SLAM is necessary. The code, results, and generated data are provided as open-source at https://eliabntt.github.io/grade-rr.**

I. INTRODUCTION

Intelligent robots should be able to perceive and understand the world around them to be autonomous and able to interact with it. However, especially when addressing dynamic environments, it is not possible to experiment with them directly in the real world due to the inherent risk of damaging or hurting people and animals. Therefore, it is crucial to verify them beforehand in simulation.

Gazebo [2] is by far the most popular framework for simulating robots due to its simplicity, reliable physics engine, and tight integration with ROS [3]. Nevertheless, it is not photorealistic, there is a limited variety of assets/worlds that can be loaded without considerable effort, and the simulation engine cannot be fully controlled. For these reasons, various alternatives emerged, e.g. TartanAir [4], AirSim [5], AIHabitat [6], BenchBot [7], and iGibson [8]. However, they all lack either full control of the simulation, ROS integration, realistic physics and appearance, or SIL/HIL capabilities. Additionally, some simulate environments with *only* rigid objects [2], [6] or do not include dynamic assets since this would pose various challenges such as their placement, their management, and their generation. Then, relying on pre-recorded robotic datasets is typically non-trivial due to differences in the form factors of the robots (e.g. placement of the sensors, stiffness of the joints), in the sensor settings (e.g. focal length, FPS, IMU frequency) or in the noise models. Moreover, using a dataset is a passive action which cannot be used when testing active or autonomous methods. For these reasons, although (non-)rigid moving objects are common in real life, a lot of research in robotics still assumes a (semi-)static world. This greatly hinders efforts towards various research topics such as SLAM and navigation in dynamic environments, target tracking, and visual robot learning, thus limiting autonomy in robotics.

For these reasons, we developed a framework 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.

To demonstrate the limitations of current state-of-the-art dynamic SLAM methods, we used GRADE to: i) generate an **indoor** dynamic environment dataset by using only freely available assets and FUEL [9], ii) test popular indoor dynamic SLAM algorithms with some generated sequences to evaluate those and benchmark our work. While testing static sequences demonstrates that the data is usable by said frameworks, evaluating the dynamic ones proves that these methods cannot generalize to data different from the currently used benchmark datasets. We then experiment with different trained models of YOLOv5 [10] and Mask R-CNN [11] and show that not always the best-performing one in terms of precision corresponds to the best ATE result. All of this while focusing on a metric that is often overlooked by the community: the amount of time the SLAM framework is capable of tracking the trajectory, which is an essential indication of the robustness of the considered method and helps to create a contextualization of the results.

II. RELATED WORK

Historically, one of the core robotics problems is mapping an unknown environment. A lot of the current SLAM research still focuses on static environments [12], despite the belief of this being a solved problem, and how to actively explore them [13]. Lately, visual SLAM has gained traction with respect to other methods, with RTABMap [14] and ORB-SLAM [15] which are just two among all the possible frameworks that can be used to perform it. Most of the current perception-based methods are developed in static environments and are expected to fail or degrade in dynamic ones, making them hard to be used in real-world everyday scenarios. Indeed, tracking the camera trajectory of the robot in dynamic environments is a notoriously difficult problem [16]. Nonetheless, research in SLAM for dynamic

[*]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.

worlds has still limited (although increasing) traction, mainly due to difficulties in simulating data and the inherent danger of directly testing an autonomous method in the real world. Many methods addressing dynamic worlds rely on segmentation or optical clues to filter out features of dynamic subjects, and most have no real-time capabilities. Among those, one of the most successful is DynaSLAM [17], which uses Mask R-CNN [11] and multi-view geometry. DynamicVINS [18] employs YOLOv5 [10] to mask the features belonging to dynamic objects. StaticFusion [19] instead relies on pointclouds clustering segmentation to work. Learning-based methods, such as TartanVO [20], propose to learn visual odometry on synthetic and real data to reconstruct the robot trajectory. However, the limited availability of testing sequences and environments makes those fail when applied to different situations or environments, as shown also in [20].

III. Own approach and contributions

As explained thoroughly in GRADE [1], we used our framework to generate a dataset of indoor dynamic sequences autonomously recorded using FUEL [9]. With these evaluations, we want to demonstrate that we can use the data generated in robotic applications by testing state-of-the-art dynamic SLAM methods, and highlight the current limitations of such frameworks. We selected two static SLAM methods, RTABMap [14] and ORB-SLAMv2 [15], to demonstrate that the visual information is not misleading by itself when testing static sequences and that the data is usable for the visual odometry task. Then we picked DynaSLAM, which uses Mask R-CNN to segment dynamic content, DynamicVINS, which instead uses YOLO, StaticFusion, i.e. a non-learning based method that performs clustering on the pointclouds, and TartanVO which, although it is not a proper SLAM system, is a learned visual odometry method developed specifically for challenging scenarios. DynamicVINS was tested in both its VO and VIO variations and with a minor modification to account for possible failures [1].

We used the generated data to train both YOLOv5 and Mask R-CNN, the networks used in [18] and [17]. We used the synthetic data both to train them from scratch and as pre-training step. Using the resulting network weights we evaluate the corresponding SLAM method, i.e. DynaSLAM with Mask R-CNN and Dynamic VINS with YOLOv5, on *fr3/walking* sequences showing contrasting results.

A. SLAM

We select four RGBD sequences from the GRADE dataset [1], in which the robot stays horizontal (H) and four in which the robot is free to move. Each sequence are 60 seconds long marked as static (S), dynamic without flying objects (D), with flying objects (F) and with occlusions of the camera (WO). We perform evaluations of both groundtruth data and with added noise. Depth data was limited both to 3.5 meters, which is a reasonable value when using for example a RealSense D435i, and 5 meters. The added noise to the depth values is based on the model described in [21]. To the RGB data we add random rolling shutter noise ($\mu = 0.015$, $\sigma = 0.006$), and blur following [22]. The IMU drift and noise parameters are taken from [23]. Image data was recorded at 30 Hz, IMU at 240 Hz, and groundtruth pose at 60 Hz. As evaluation metrics, we utilize the ATE RMSE and the amount of time the framework can successfully track the trajectory. The latter is a critical evaluation quantity to be considered. It helps the reader put ATE values in perspective whenever the framework fails due to some featureless frames or occlusions. For consistency, when evaluating DynamicVINS, we considered only experiments in which the first initialization was successful. We first analyze the results with the depth limited at 3.5 meters. We report them on Tab. IV and Tab. III. One can notice that all the methods perform poorly in the majority of the sequences. Focusing on *noisy* experiments, which are more related to reality, we can see that with static sequences most of the methods perform well, except TartanVO and StaticFusion which fail. Furthermore, one should not be misled by the good ATE results of DynaSLAM on dynamic sequences. Indeed, in 5 out of 8 experiments the camera lost track of the trajectory for at least ~ 27 seconds (three times more than 49 s) and performing sometimes worse than the ORB-SLAMv2, i.e. its underlying mapping framework. In general, we can infer that, despite these methods showing compelling results when tested with other datasets, they exhibit several limitations when tested on different data. The fact that the methods perform well with the static sequences demonstrates how it is not a problem of the data used, but it is a problem inherent to the dynamic nature of the environment or the presence of featureless frames. Overall, DynamicVINS seems to be the best-performing method when considering both ATE and the time missing from each experiment. However, despite the help of the IMU, in DH sequence the ATE is over 1.6 meters for just a 60 s sequence. By comparing the tests performed on groundtruth and noisy data one can see that in the majority of the experiments the noisy ones perform slightly worse. However, in general, the results are similar and one can draw the same overall conclusions. Finally, we can compare corresponding sequences with the depth limited to 3.5 and 5 meters using tables Tab. VI and Tab. IV for the ground truth data, and Tab. V and Tab. V for the noisy one. As expected, TartanVO yields equal results, by being a method that works only on visual data. RTABMap shows performance which are greatly degraded in all sequences, in both missing time and ATE. ORB-SLAM and DynaSLAM(VIO) are, for the most part, comparable. We can also notice how DynaSLAM(VO) shows worse performance, indicating the reliance of the VIO counterpart on the IMU. StaticFusion, as shown also in other works like [24], shows degrading performance with increased depth data. DynaSLAM seems overall the most stable, except for the D-noisy sequence which shows high variability.

B. Network models variations

We will consider here four models: S-COCO, S-GRADE, S-GRADE+S-COCO and S-GRADE+COCO. These correspond to different training strategies with a reduced coco dataset (S-COCO), a reduced GRADE dataset (S-GRADE),

COCO, and combination of pretraining and finetuning (S-GRADE+[S-COCO, COCO]). We refer the reader to [1] for additional insights. In general, the best performing models for both YOLO and Mask RCNN when tested on the TUM RGBD labelled data, were obtained with S-GRADE+COCO, followed by COCO (BASELINE in [1]), S-GRADE+S-COCO, S-GRADE, S-COCO.

1) YOLOv5 and Dynamic VINS: We utilize them with Dynamic VINS to evaluate their performance with the TUM *fr3/walking* sequences. The results, presented in Tab. I, are averaged among three runs. The baseline values are not the ones from [18] since we were unable to reproduce them for the *rpy* and *static* sequences. One can easily notice how the models pre-trained on the synthetic data consistently obtain results which are at par or better than the baseline model. Surprisingly, using the model trained on S-GRADE, despite showing the lowest detection performance among the models considered in this test, is the best performing one in two sequences, remarking the fact that more research is necessary on dynamic SLAM. In this case, between the tests, there was no difference in the percentage amount of the successfully tracked trajectory.

Model	S-COCO	S-GRADE	S-GRADE + S-COCO	S-GRADE + COCO	Baseline	% Traj.
w._half	0.064	**0.048**	*0.059*	0.066	0.069	87.81
w._xyz	0.052	0.050	0.049	*0.045*	**0.037**	89.37
w._rpy	0.120	0.224	*0.116*	*0.116*	**0.114**	87.11
w._stat	0.302	**0.199**	0.216	*0.203*	0.218	90.18

TABLE I: ATE [m] and percentage of the tracked trajectory percentage of Dynamic VINS using our models on the *fr3/walking* sequences. In **bold** the best one, in *italics* the second best.

2) Mask R-CNN and DynaSLAM: Also in this case, we evaluate the performance of the trained model with the TUM *fr3/walking* sequences using DynaSLAM. Each result is the average between three experiments, and all are shown in Tab. II. Also in this case, we computed the baseline again, obtaining results which are close to the one reported in [17] with the exception of the *walking_rpy* sequence. However, this was necessary for us to be able to report the percentage amount of the tracked trajectory. We used the trained models that showed the best performance in the segmentation task. Here, thanks to the offline nature of the method and the optimization procedure employed, ATE errors are much lower than the ones obtained from Dynamic VINS (see Tab. I). By analysing the results and taking into consideration both the ATE and the amount of the tracked trajectory, one can see that in all cases there is a benefit to using a detection network which was pre-trained on synthetic data. However, a clear pattern cannot be derived yet, despite both S-GRADE + S-COCO and S-GRADE + COCO showing compelling results.

IV. CONCLUSIONS

In this work, by using data from [1], we have shown how all the tested models methods fail, one way or another, to successfully generalize to our dynamic sequences. However, the majority of them are capable of tracking static trajectories, signifying that the issue is not on the realism or the quality of the data itself but lies possibly on the parameter tuning and, more in general, on the method themselves. We have also shown that the models trained on our synthetic data can bring a performance improvement when a model trained with that is used with the corresponding SLAM framework. However, this also indicates that more research in this regard is needed since there are instances in which networks trained *only* on either S-COCO and S-GRADE perform the best, despite these being the worst-performing models in the detection and segmentation tasks.

	S-COCO	S-GRADE	S-GRADE + S-COCO	S-GRADE + COCO	Baseline
w._half	0.031	0.034	0.032	**0.028**	*0.030*
	79.24	*89.37*	78.65	89.34	**89.53**
w._xyz	*0.017*	*0.017*	**0.016**	**0.016**	**0.016**
	91.43	86.35	**91.85**	83.93	83.90
w._rpy	**0.034**	0.104	0.060	*0.037*	0.040
	72.99	**80.25**	67.82	*77.79*	75.68
w._stat	0.010	**0.007**	**0.007**	*0.008*	**0.007**
	91.89	*91.61*	**91.89**	77.77	89.06

TABLE II: ATE [m] and percentage of the tracked trajectory of DynaSLAM using our models on the *fr3/walking* sequences. In **bold** the best, in *italics* the second best.

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] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[3] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[4] 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.

[5] 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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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

[11] 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.

Sequence	Dynamic VINS (VIO)		Dynamic VINS (VO)		TartanVO		StaticFusion		DynaSLAM		ORB-SLAMv2		RTABMap	
	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]
FH	0.155	1.03	0.367	0.43	0.582	0.00	0.854	0.00	0.309	1.50	0.386	0.00	0.097	1.97
F	0.886	1.27	1.857	1.37	4.223	0.00	3.992	0.00	0.178	49.67	0.144	44.30	0.125	47.03
DH	1.681	0.43	1.183	3.30	1.234	0.00	1.091	0.00	0.002	57.33	0.005	56.80	0.013	49.13
D	0.707	0.67	1.598	1.63	1.356	0.00	2.278	0.00	0.043	26.93	0.700	11.07	0.405	28.77
WOH	0.491	1.03	0.871	1.23	2.399	0.00	1.826	0.00	0.023	28.53	0.022	27.90	0.101	27.93
WO	1.086	2.63	1.163	3.17	2.473	0.00	2.213	0.00	0.119	55.23	0.171	48.23	0.075	50.70
SH	0.419	0.57	0.069	0.43	2.517	0.00	4.184	0.00	0.016	0.00	0.018	0.00	0.072	0.00
S	0.177	0.57	0.137	0.43	1.308	0.00	3.538	0.00	0.029	0.00	0.026	0.00	0.133	0.00

TABLE III: ATE RMSE [m] and missing time [s] of the tested **noisy** sequences. Each experiment is 60 seconds long. Depth limited to 3.5 m.

Sequence	Dynamic VINS (VIO)		Dynamic VINS (VO)		TartanVO		StaticFusion		DynaSLAM		ORB-SLAMv2		RTABMap	
	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]
FH	0.069	0.67	0.201	0.43	0.551	0.00	0.085	0.00	0.241	3.43	0.149	0.00	0.126	0.00
F	0.647	1.20	1.337	2.93	4.132	0.00	2.866	0.00	0.147	49.23	0.431	39.10	0.115	46.93
DH	8.103	0.43	1.178	8.17	1.259	0.00	1.664	0.00	0.008	57.07	0.005	48.53	0.094	21.63
D	0.188	0.67	1.304	0.83	1.264	0.00	1.212	0.00	0.057	6.33	0.459	0.30	0.492	7.07
WOH	0.239	1.10	1.272	1.00	2.361	0.00	1.980	0.00	0.015	27.70	0.012	27.73	0.042	25.87
WO	0.501	2.20	0.985	3.47	2.380	0.00	2.807	0.00	0.083	55.20	0.163	48.23	0.053	50.60
SH	0.109	0.57	0.023	0.43	2.395	0.00	0.594	0.00	0.016	0.00	0.012	0.00	0.039	0.13
S	0.205	0.57	0.039	0.43	1.205	0.00	7.919	0.00	0.010	0.00	0.011	0.00	0.043	0.00

TABLE IV: ATE RMSE [m] and missing time [s] of the tested sequences **w/o added noise**. Each experiment is 60 seconds long. Depth limited to 3.5 m.

Sequence	Dynamic VINS (VIO)		Dynamic VINS (VO)		TartanVO		StaticFusion		DynaSLAM		ORB-SLAMv2		RTABMap	
	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]
FH	0.179	0.67	0.349	0.43	0.568	0.00	0.059	0.00	0.200	4.50	0.291	0.00	0.086	15.07
F	0.904	1.63	2.315	2.13	4.192	0.00	2.781	0.00	0.189	45.53	0.129	43.00	0.125	50.63
DH	1.749	0.43	2.047	3.40	1.214	0.00	14.938	0.00	0.002	57.40	0.005	56.73	0.030	50.07
D	0.611	0.67	1.616	1.23	1.350	0.00	22.374	0.00	0.109	5.97	0.652	5.13	0.171	41.73
WOH	0.561	1.27	1.550	0.90	2.389	0.00	4.926	0.00	0.025	27.83	0.022	27.87	0.047	33.30
WO	0.962	2.40	1.429	2.40	2.399	0.00	1.418	0.00	0.112	55.23	0.142	48.20	0.041	52.53
SH	0.404	0.57	0.063	0.43	2.537	0.00	2.721	0.00	0.017	0.00	0.017	0.00	0.061	17.13
S	0.199	0.57	0.066	0.43	1.259	0.00	22.282	0.00	0.029	0.00	0.027	0.00	0.220	11.53

TABLE V: ATE RMSE [m] and missing time [s] of the tested **noisy** sequences. Each experiment is 60 seconds long. Depth limited to 5 m.

Sequence	Dynamic VINS (VIO)		Dynamic VINS (VO)		TartanVO		StaticFusion		DynaSLAM		ORB-SLAMv2		RTABMap	
	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]	ATE [m]	Missing Time [s]
FH	0.073	0.67	0.259	0.43	0.551	0.00	0.059	0.00	0.221	0.30	0.199	0.00	0.229	7.53
F	1.814	1.40	1.362	1.77	4.132	0.00	2.781	0.00	0.228	45.10	0.512	37.13	0.129	48.33
DH	7.492	0.43	1.811	7.50	1.259	0.00	14.938	0.00	0.008	54.40	0.009	49.53	0.108	48.33
D	0.201	0.67	0.738	0.43	1.264	0.00	22.374	0.07	0.038	4.33	0.275	0.53	0.154	23.40
WOH	0.228	1.10	1.256	2.57	2.361	0.00	4.926	0.00	0.012	27.70	0.015	27.70	0.053	30.70
WO	0.679	2.43	1.031	3.10	2.473	0.00	1.418	0.00	0.107	55.10	0.138	48.20	0.025	51.53
SH	0.121	0.57	0.023	0.43	2.395	0.00	2.721	0.07	0.012	0.00	0.011	0.00	0.019	10.60
S	0.226	0.57	0.035	0.43	1.205	0.00	22.282	0.00	0.011	0.00	0.014	0.00	0.018	12.60

TABLE VI: ATE RMSE [m] and missing time [s] of the tested sequences **w/o added noise**. Each experiment is 60 seconds long. Depth limited to 5 m.

[12] I. Abaspur Kazerouni, L. Fitzgerald, G. Dooly, and D. Toal, "A survey of state-of-the-art on visual slam," *Expert Systems with Applications*, vol. 205, p. 117734, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422010156

[13] J. A. Placed, J. Strader, H. Carrillo, N. Atanasov, V. Indelman, L. Carlone, and J. A. Castellanos, "A survey on active simultaneous localization and mapping: State of the art and new frontiers," *IEEE Transactions on Robotics (T-RO)*, 2023.

[14] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21831

[15] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[16] M. R. U. Saputra, A. Markham, and N. Trigoni, "Visual slam and structure from motion in dynamic environments: A survey," *ACM Comput. Surv.*, vol. 51, no. 2, feb 2018. [Online]. Available: https://doi.org/10.1145/3177853

[17] B. Bescos, J. Fácil, J. Civera, and J. Neira, "DynaSLAM: Tracking, mapping and inpainting in dynamic environments," *IEEE RA-L*, 2018.

[18] J. Liu, X. Li, Y. Liu, and H. Chen, "Rgb-d inertial odometry for a resource-restricted robot in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9573–9580, 2022.

[19] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers, "Staticfusion: Background reconstruction for dense rgb-d slam in dynamic environments," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3849–3856.

[20] W. Wang, Y. Hu, and S. Scherer, "Tartanvo: A generalizable learning-based vo," in *Conference on Robot Learning (CoRL)*, 2020.

[21] "BKMs Tuning RealSense D4xx Cam," https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/BKMs_Tuning_RealSense_D4xx_Cam.pdf, [Accessed 23-Feb-2023].

[22] S. Zhang, A. Zhen, and R. L. Stevenson, "A dataset for deep image deblurring aided by inertial sensor data," *Fast track article for IS&T International Symposium on Electronic Imaging 2020: Computational Imaging XVIII proceedings.*, pp. 379–1–379–6(6), 2020.

[23] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Cham: Springer International Publishing, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-26054-9_23

[24] M. Runz, M. Buffier, and L. Agapito, "Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects," in *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2018, pp. 10–20.