Dynamic Object Removing SLAM adapting MonoRec

Yulun Zhuang[1], Hao Liu[1], Xingqiao Zhu[1], Janani Peri[2], Vaishnavi Harikumar[2]

***Abstract*— In this paper, we present a new dynamic object removing SLAM method named MonoRec-SLAM. Our method adapts the MaskModule from MonoRec, a dense 3D dynamic environments reconstruction architecture. The adapted MaskModule aims to predict a mask indicating the probability of a pixel belonging to a moving object useing a set of cost volumes that encode geometric priors between frames. The segmented frames are then fed into ORB-SLAM3 to obtain the enhanced frames without dynamic objects. Results are evaluated on the KITTI and TUM datasets and compared with DynaSLAM in terms of Absolute Pose Errors (APE), mask inference and camera pose tracking time. We demonstrated that our method achieves an average APE improvement by 6.03% on KITTI dataset, obtains a more static map of the scenes, and achieves a great balance between real-time capability and dynamic object masking. Additionally, it is 42.61% faster in mask inference and takes 31.35% of the time in camera pose tracking in comparison with DynaSLAM, enabling it to run in real-time (∼10 FPS).**

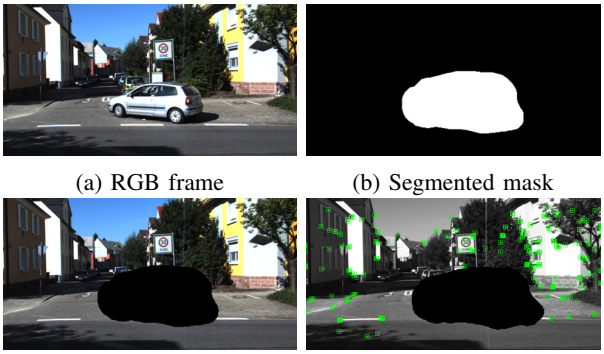
I. INTRODUCTION

SLAM (Simultaneous Localization and Mapping) is a technique used in robotics and computer vision that allows a robot or camera to map an unknown environment while simultaneously estimating its own position within that environment. When visual data, such as images or video is utilized to construct the map and estimate the position of the camera, it is called Visual SLAM. However, traditional visual SLAM systems can struggle in environments with moving objects because they are not static and may not be included in the map. As a consequence, they can only manage small fractions of dynamic content by classifying them as outliers to such static model. Although the static assumption holds for some robotic applications, it limits the applicability of visual SLAM in many relevant cases, such as intelligent autonomous systems operating in populated real-world environments over long periods of time. Dynamic Visual SLAM systems are designed to handle these types of environments by detecting and tracking moving objects separately from the rest of the scene and incorporating this information into the SLAM algorithm to produce a more accurate map of the environment. Many previous works have attempted to mask out dynamic objects in order to improve the efficiency of visual SLAM. [1] [2].

This paper is for the final project of course EECS 568, Mobile Robotics. The open-source code of the project can be found in https://github.com/silvery107/monorec-slam

[1]Yunlun Zhuang, Hao Liu and Xingqiao Zhu are with are with the Department of Robotics, University of Michigan (yulunz, wdliu, xingqiao)@umich.edu

[2]Vaishnavi Harikumar and Janani Peri are with the Department of Aerospace Engineering, University of Michigan (ivaishi, pjanani)@umich.edu



(a) RGB frame (b) Segmented mask

(c) Segmented frame (d) Enhanced keyframe

Fig. 1: Visualization of dynamic object removing

In this paper, we propose a new method called MonoRec-SLAM for dynamic object removing SLAM, which is a combination of Monorec [3] and ORB-SLAM3 [4]. MonoRec-SLAM utilizes the Mask-Module from MonoRec, a dense 3D dynamic environments reconstruction architecture, to predict a mask indicating the probability of a pixel belonging to a moving object. The frames are segmented and then put into ORB-SLAM3 to obtain enhanced frames that exclude dynamic objects. The MonoRec-SLAM method is then tested on two datasets: KITTI and TUM, and then compared with DynaSLAM in terms of Absolute Pose Errors (APE), mask inference, and camera pose tracking time.

The paper is structured as follows: Section II provides a comprehensive overview of related research, Section III presents the methodology in detail, Section IV includes the evaluation and discussion, and Section V concludes the paper.

II. RELATED WORK

A. DynaSLAM

DynaSLAM [1] is a real-time dense SLAM system designed for dynamic environments. It is capable of reconstructing and tracking the 3D geometry of a scene while simultaneously detecting and tracking dynamic objects in the environment.

The system uses a stereo camera setup and relies on a combination of keyframe-based and direct image alignment methods to estimate camera poses and reconstruct the scene. To handle dynamic objects, DynaSLAM leverages a state-of-the-art object detection and tracking network, which uses a Mask R-CNN architecture trained on the COCO dataset.

DynaSLAM also incorporates a novel probabilistic fusion framework that enables it to fuse 3D reconstructions from multiple viewpoints into a single, consistent model. This

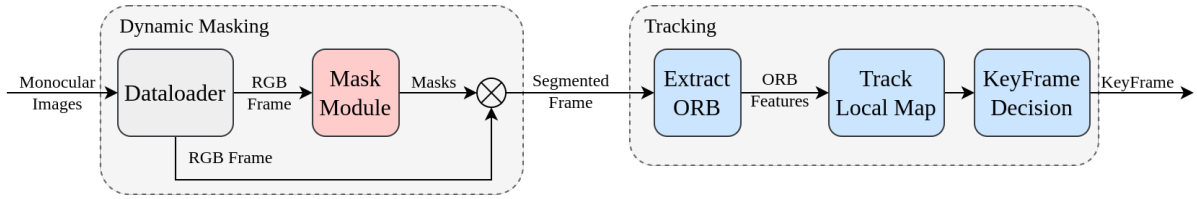


Fig. 2: General pipeline for dynamic object removing in camera pose tracking

allows the system to maintain a coherent and accurate representation of the environment, even when dealing with dynamic objects and camera motion.

Overall, DynaSLAM represents a significant advancement in dense SLAM systems, particularly in its ability to handle dynamic environments. The greatest issue of DynaSLAM is that its operation frequency is not quick enough to support real-time operation.

B. YOLACT

YOLACT [5] is a simple, fully convolutional model for real-time instance segmentation. YOLACT can be trained on only one GPU and is the first real-time (above 30 FPS) approach with around 30 mask mAP on COCO test-dev among various instance segmentation methods tested on COCO. YOLACT produces instance masks by linearly combining the prototypes with the mask coefficients instead of without an explicit feature localization step (e.g., feature re-pooling as Mask R-CNN).

The complex task of instance segmentation is broken into two simpler tasks in YOLACT. In the first task, FCN [6] is used to produced a set of prototype masks, while in the second task, a vector of mask coefficients is predicted after adding an extra head to the object detection. The two parallel tasks are assembled and tested through NMS, and a final mask will be generated for the instances surviving NMS. In this way, YOLACT produces very high-quality masks and and exhibits temporal stability for free.

III. METHODOLOGY

In this work, the MonoRec architecture is used to obtain masks and masked images of the KITTI and TUM datasets that are being tested. These masks are then used to generate masked images. Once the masks and masked images are generated, these are passed into ORB-SLAM3 to map the environment while the dynamic objects are masked. The MonoRec and ORB-SLAM3 architecture is described as follows:

A. MonoRec

MonoRec is a semi-supervised learning-based approach for dense 3D reconstruction of dynamic environments using a single moving camera. The proposed method utilizes a combination of supervised and unsupervised learning to handle the challenges of reconstructing dynamic scenes while accounting for camera motion. The MonoRec approach uses a sequence of consecutive frames, and their corresponding camera poses to generate a dense depth map for a given keyframe. This architecture comprises of two main modules: the MaskModule and the DepthModule. The MaskModule predicts masks for moving objects, which improves depth accuracy and reduces noise in 3D reconstructions while the DepthModule generates a depth map from the masked cost volume. For this project, only the MaskModule was used. Figure 4 shows a flowchart that defines the MaskModule of MonoRec.

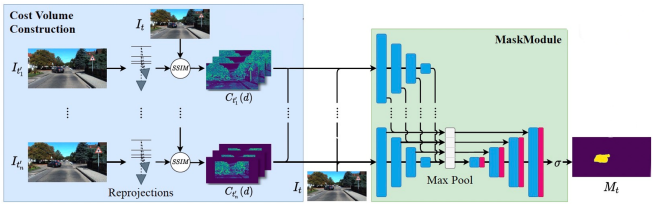


Fig. 4: Architecture of the MaskModule [3]

The MaskModule in MonoRec aims to predict a mask indicating the probability of a pixel belonging to a moving object. To do this, the MaskModule uses a set of cost volumes that encode geometric priors between the keyframe and other frames in the sequence. These cost volumes provide information about inconsistencies in depth estimates due to moving objects. However, geometric priors alone are not enough to accurately predict moving objects, so the MaskModule also uses pre-trained ResNet-18 features to encode semantic priors. The MaskModule is designed as a U-Net architecture with skip connections, and it can be applied to different numbers of frames without the need for retraining.

B. ORB-SLAM3

ORB-SLAM3 is a visual-inertial simultaneous localization and mapping (SLAM) system that can use monocular, stereo, and RGB-D cameras with pinhole and fisheye lens models. The system has two main components:

The tightly integrated visual-inertial SLAM system is a component of ORB-SLAM3 that uses feature-based tracking and mapping to estimate the camera trajectory and map the environment. It also tightly integrates visual and inertial measurements to improve accuracy and robustness. The system relies on Maximum-a-Posteriori (MAP) estimation, even during the IMU initialization phase, which allows it to operate robustly in real-time, in small and large, indoor, and outdoor environments. The visual-inertial SLAM system can use monocular, stereo, and RGB-D cameras with pinhole and fisheye lens models.

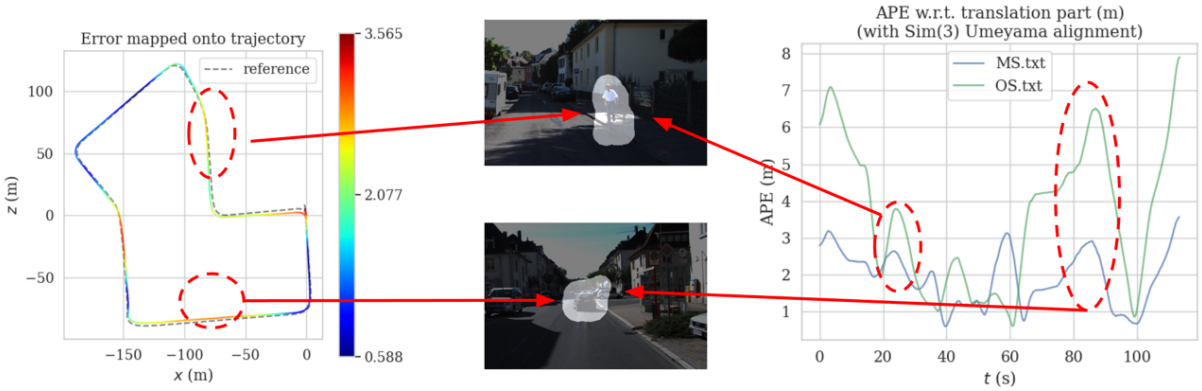


Fig. 3: Visualization of improvements of dynamic object removal in comparison between MonoRec-SLAM and ORB-SLAM3. Left: APE error map of trajectories from MonoRec-SLAM on sequence 07 of KITTI dataset; Right: APE errors of two methods w.r.t. trajectory time.

The multiple map system is a component of ORB-SLAM3 that uses a new place recognition method with improved recall to allow the system to seamlessly merge maps when revisiting mapped areas. The system can reuse all previous information to boost accuracy, even if the observations are widely separated in time or come from a previous mapping session. This allows ORB-SLAM3 to survive long periods of poor visual information and to maintain accuracy over time.

After running the combination of the adapted MaskModule from MonoRec and ORB-SLAM3 which we have introduced as MonoRec-SLAM, a map of the static environment is obtained while the dynamic objects are masked.

The APE or Absolute Pose Error is calculated with respect to the ground truth for MonoRec-SLAM, DynaSLAM, and ORB-SLAM3. APE is calculated as the Euclidean distance between the ground truth position and the estimated position of the camera, and the angle difference between the ground truth orientation and the estimated orientation of the camera. The APE value is calculated for each frame and then averaged over all frames to obtain the final APE value [7].

Figure 4 consists of two graphs that compare the APE for each algorithm to compare their individual performances. The following section discusses the comparison of all three algorithms that advocates MonoRec-SLAM to have the best performance for most cases tested.

IV. EVALUATION AND DISCUSSION

The algorithms discussed in previous sections were evaluated on the following datasets and the results were compared. A brief video presentation about our work can be found in https://youtu.be/h6W1JF9h4wk.

A. Datasets

We evaluated MonoRec-SLAM and DynaSLAM with the KITTI dataset and TUM dataset since they contain moving objects in some sequences which are convenient for us to generate masks. Besides, ground truth trajectories are included in these datasets, which are considered as the baseline for performance evaluation of our method by calculating

(a) KITTI odometry dataset (b) TUM dynamic scenes

Fig. 5: General setup of datasets used for evaluation

Absolute Pose Errors (APE). We only used the monocular data in these datasets.

The KITTI dataset [8] is a widely used dataset in mobile robots and autonomous driving. It is made up of hours of traffic scenarios captured using a range of sensor modalities, including high-resolution RGB, grayscale stereo cameras, and a 3D laser scanner. The odometry benchmark of KITTI consists of 22 stereo sequences, saved in lossless png format: 11 sequences (00-10) with ground truth trajectories for training and 11 sequences (11-21) without ground truth for evaluation. We used Sequence 00, 04, 05 and 07 to evaluate MonoRec-SLAM and DynaSLAM and compared the results, as shown in Figure 5a.

The TUM dataset [9] is an RGB-D dataset, providing color and depth images from a Microsoft Kinect sensor along the sensor's ground-truth route. The data was collected at a full frame rate of 30 Hz with a sensor resolution of 640x480. A high-accuracy motion-capture system with eight high-speed tracking cameras (100 Hz) provides the ground-truth trajectory. To test the 2 methods, we use the dynamic object sequence, which is intended to evaluate the robustness of visual SLAM and odometry algorithms to slowly moving dynamic objects, as illustrated in Figure 5b.

B. Results

The overall system shares the same architecture of common visual SLAM as is shown in Figure 4. In our

project, we used ROS to communicate between adapted MaskModule and ORB-SLAM3. The mask node published */camera/img_raw* message including the RGB frame information after being masked and this message is subscribed by the ORB-SLAM3 node to solve the tracking problem. The system is run with ROS Noetic under Ubuntu 20.04 on a laptop with a single NVIDIA RTX 3060 GPU and Intel i7-12700H CPU.

Figure 3 shows how APE changes with time when we were testing with sequence 07 in the KITTI dataset. The right-hand side compares the performance of MonoRec-SLAM and ORB-SLAM3.

TABLE I: Comparison of APE in RMSE. Results for DynaSLAM and ORB-SLAM3 are taken from original papers and verified in our experiments shown in brackets.

Sequence	MonoRec-SLAM	DynaSLAM [1]	ORB-SLAM3 [4]
00	6.71	7.55 (7.77)	**5.33** (7.14)
04	1.39	**0.97** (0.65)	1.62 (1.45)
05	**4.60**	4.60 (6.77)	4.85 (4.84)
07	**2.09**	2.36 (3.66)	2.26 (3.90)

Besides sequence 07, we also ran MonoRec-SLAM on sequences 00, 05, and 07. The RMSE(root mean square error) comparison between MonoRec-SLAM, DynaSLAM and ORB-SLAM3 is shown in Table I.

The comparison of computation time in milliseconds for generating masks and tracking poses between MonoRec-SLAM and DynaSLAM is shown in TableII.

We also tried applying MonoRec-SLAM on TUM dataset. However, the result is not that idea and it continuously loses track of its own position. Figure 6a shows the mask we get with the TUM freiburg 3 dataset.

TABLE II: Comparison of mask inference and tracking time. Results for DynaSLAM are taken from original paper.

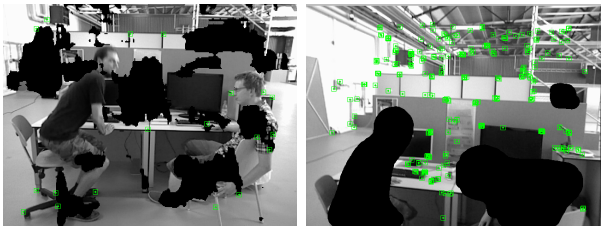
	MonoRec-SLAM	DynaSLAM [1]
Masking (ms)	**83.10**	195.00
Tracking (ms)	**104.62**	333.68
Real-Time	**Yes**	No

C. Discussion

The graph on the left in Figure 3, shows the APE of MonoRec-SLAM mapped onto the ground truth trajectory. The red dotted circles highlighted on this graph show the position on the trajectory, at which our method performs significantly better than ORB-SLAM3. In the right figure in Figure 3, MS stands for MonoRec-SLAM and OS stands for ORB-SLAM3. The red dotted circles on this figure highlight the timestep (around 25 and 85 seconds) at which the APE for MonoRec-SLAM is significantly lower (46% and 63% respectively) than that of ORB-SLAM3. In these instances, the frame consists of moving or dynamic objects which are being masked by the adapted MaskModule, and therefore the error at these points for our method is low, justifying the functionality of the dynamic mask generated from the adapted MaskModule.

From Table I, we can observe that our method has the best performance, compared to both DynaSLAM and ORB-SLAM3 for sequences 05 and 07 with an average APE improvement by 6.03%. While for sequence 00, ORB-SLAM3 seems to have the best performance, our method still performs better than DynaSLAM. A potential reason why ORB-SLAM3 outperformed both MonoRec-SLAM and DynaSLAM on sequence 00 is that sequence 00 is a long-duration dataset with multiple loop closure, which already provides enough features for mapping and localization. In this case, masking might just reduce the features it detects and reduce the accuracy of the final result.

From Table II, it is evident that our method is 42.61% faster in mask inference and takes 31.35% the time in camera pose tracking in comparison with DynaSLAM, enabling it to perform SLAM in real time (∼10 FPS).



(a) MonoRec segmentation (b) DynaSLAM segmentation

Fig. 6: Failure cases on the TUM dataset

Regarding our failure on the TUM dataset, we have proposed two possible reasons. By comparing Figure 6a and Figure 6b, instead of masking the two moving people in the scene, the adapted MaskModule is randomly masking the background. One possible reason is that the two people are close to the camera and covered a large area of the scene, they might be considered as the background, and the real background is considered as a moving object, leading to the random mask on the background. Another possible reason is that the two people are moving too slowly to be recognized by the adapted MaskModule.

D. Future Work

Considering the failure in evaluation on the TUM dataset, one desired future work would be revising the design and implementation of MonoRec to increase the robustness of the adapted MaskModule.

(a) Before (b) After

Fig. 7: Background inpainting on KITTI dataset

Besides, we plan to finalize the background inpainting function. Before the paper submission, we deployed the

learning-based Flow-edge Guided Video Completion [10] algorithm for background inpainting of segmented frames where Figure 7 shows its performance. Figure 7a shows the figure before inpainting and Figure 7b shows the figure after segmentation and inpainting. It works well with a high-quality mask, but when dynamic objects are not ideally segmented, the inpainting technique will distorted the frame. At the same time, the function increases the runtime to a large extent that, we can no longer run the overall system in real-time with inpainting. In the future, we will optimize the algorithm and take the advantages of Homography transform techniques for inpainting alignments [11] for more accurate inpainting and faster runtime.

V. CONCLUSION

We input RGB frames into the adapted MaskModule and generated masked frames, and then obtained the enhanced keyframes by ORB-SLAM3 module tracking the masked segmented frames. We evaluated our method on both KITTI and TUM datasets, and the results were compared with DynaSLAM. The dynamic objects are masked in real time successfully on the KITTI dataset, while it is failed on the TUM dataset. By evaluating the APE, we demonstrated that MonoRec-SLAM performs better than DynaSLAM in most sequences. What's more, we also demonstrated that our method runs faster than DynaSLAM and supports real-time operation by comparing the mask inference, tracking time, and real-time operation. We concluded that our method can achieve in most cases a higher accuracy on the KITTI dataset, obtains a more static map of scenes, and achieves a great balance between real-time capability and dynamic object masking.

REFERENCES

[1] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "Dynaslam: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, 2018.

[2] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, "Ds-slam: A semantic visual slam towards dynamic environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1168–1174.

[3] F. Wimbauer, N. Yang, L. Von Stumberg, N. Zeller, and D. Cremers, "Monorec: Semi-supervised dense reconstruction in dynamic environments from a single moving camera," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6112–6122.

[4] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[5] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: Real-time instance segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9157–9166.

[6] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE*, no. 4, 2017.

[7] J. Wang, X. Chen, J. Yan, and J. Liu, "Research on underwater complex scene slam algorithm based on image enhancement," *Journal of Physics: Conference Series*, vol. 1797, no. 1, p. 012020, 2021.

[8] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[9] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 573–580.

[10] C. Gao, A. Saraf, J.-B. Huang, and J. Kopf, "Flow-edge guided video completion," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*. Springer, 2020, pp. 713–729.

[11] M. Granados, K. I. Kim, J. Tompkin, J. Kautz, and C. Theobalt, "Background inpainting for videos with dynamic objects and a free-moving camera," in *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part I 12*. Springer, 2012, pp. 682–695.

[12] 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.

[13] H. Liu, R. Soto, F. Xiao, and J. L. Yong, "Yolactedge: Real-time instance segmentation on the edge," in *International Conference on Robotics and Automation*, 2021.