Static ORB Extractor for Visual SLAM

1st Chethan Chinder Chandrappa
Electrical and Computer Engineering
University of California Los Angeles
Los Angeles, USA
chinderc@g.ucla.edu

2nd Srinath Naik jmeera
Computer Science
University of California Los Angeles
Los Angeles, USA
srinath@cs.ucla.edu

***Abstract*—Despite significant progress in development of feature based visual SLAM systems, they exhibit poor performance when dynamic objects are present in the scene. This under performance can be attributed to using all the extracted feature points for location and pose estimation irrespective of whether they correspond to static/dynamic objects. In presence of dynamic objects, the calculated projection matrix to map from one frame to the next will be inaccurate due to excessive shift of features, there by leading to poor estimation.**
In this paper, we propose Reprojection Error Map for finding the dynamic mask in the scene using the ORB keypoint correspondences. We describe the three methodologies we have developed for this task and their drawbacks. Finally, we present evaluation of our approaches on dynamic object based scenes in TUM and KITTI datasets.

***Index Terms*—vSLAM, dynamic objects, ORB Keypoints, Reprojection Error Map.**

I. Introduction

SLAM is the core of most of the autonomous robotic systems which move and interact with the environment. Accurate localization and mapping are essentials for safe and efficient functioning of these systems. Over the years, feature based visual SLAM systems[1] were developed, which can be used on plethora of visual SLAM tasks in real time due to their fast performance and easy adaptation. Recently, Deep Learning based visual odometry systems like DeepVO[2], UndeepVO[3], TartanVO[4] were built by training on huge data-sets. However, these are limited to few use cases as they are trained for some specific scenarios and are slow in general. This leaves the feature based SLAM methods remain state of the art till date.

Feature based SLAM methods extract key features in each frame, compute correspondences in subsequent frames and use projective transformation to estimate the relative camera pose and location. Although these methods achieve considerable performance in some of the applications, it still remains a challenge to create SLAM robust to dynamic environments. In dynamic environments, the inherent movements of the objects in the scene will cause an irregular shift in the key features corresponding to dynamic objects, resulting in an inaccurate projection matrix, leading to poor localization. Moreover, SLAM systems build a global map of the environment as time progresses, the map points corresponding to dynamic objects will accumulate, consuming space and increasing time for computations which are critical in real time systems. Also, as the same environment in a particular frame might have different dynamic objects at various times, it is possible that loop closure detection will fail, resulting in improper optimization.

We have tested ORB-SLAM2 on few sequences of TUM dataset which contain dynamic objects. As it can be clearly seen from Fig. 1 and Fig. 2, which depict Absolute Trajectory Error and Relative Pose Error respectively, the estimated trajectory and poses significantly differ from the ground truth. Building a SLAM system robust to dynamic objects will improve tracking accuracy and create better or more representative map of the environment. Our contribution in this paper is two fold. First we experiment with different methodologies to generate a dynamic mask in a dynamic scene. Then we integrate the above system into ORB-SLAM2 front end to extract only static ORB features from each frame. Finally, we evaluate our approach and end-to-end system on dynamic object based scenes.

In the rest of this paper, the structure is as follows. Section II provides details regarding some recent work relevant to this area. Section III provides a detailed explanation of the three methods we have developed as we progressed towards the objective of creating a robust dynamic object detector. Then, section IV provides details of our experimentation, results and potential drawbacks of each of these methods. Finally, section V consists of conclusion and possible directions for improvements and further research.

II. Related Work

Creating SLAM robust to dynamic objects in the scene is an open research problem and the approaches for tackling it can be classified into two. Traditional methods in which pixel motion is estimated from video using geometric constraints such as reprojection, optical flow etc. to decide upon dynamic areas. [5] relies on EpicFlow[6] to get dense pixel matching between frames, then it uses a probabilistic inference to conclude the dynamic nature based on prior state and motion estimation in current frame. This method will have advantage on abrupt movements of camera which are of short time and does not contribute directly to the object motion. However, optical flow methods suffer from variations in brightness

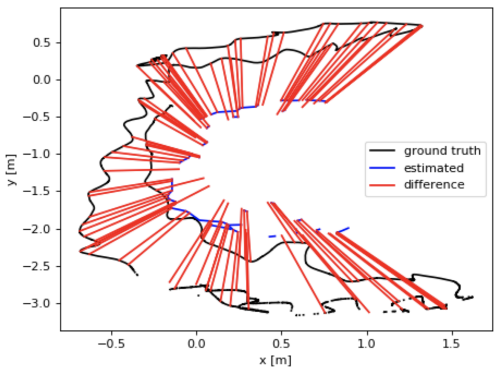

Fig. 1. Absolute Trajectory Error in a dynamic object scenario from TUM dtaset

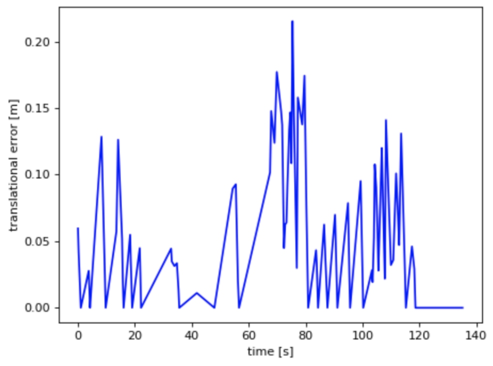

Fig. 2. Relative Pose Error in a dynamic object scenario from TUM dtaset

across the frames.

Recently, we have Deep Learning based methods where pre-trained networks are used to segment the potential dynamic objects(eg: car, human) in the frame and then decide if these regions are actually moving. DS-SLAM[7] adopts SegNet[8] to get pixel wise semantic segmentation followed by moving consistency check between frames to determine dynamic points. Also, it creates a semantic map of the environment. DOT[9] uses Detectron2[10] network for semantic segmentation to generate dynamic object masks. Then, it performs consistency check between frames to decide upon which of these objects are actually moving. Finally, it also creates new masks based on the dynamic object motion estimation which is termed as 'Mask Propagation'. A major limitation of these methods are that they are not flexible and can only detect predetermined objects as dynamic. Moreover, they are slow in practice due to heavy computation.

Most of the above mentioned work integrates into front end of ORB-SLAM2 and report an improved performance on dynamic object scenarios of various datasets.

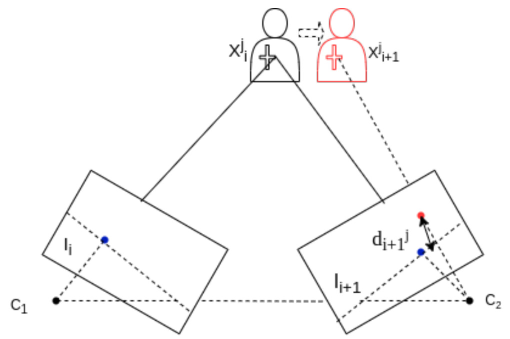

Fig. 3. Depiction of distance between epiline and corresponding keypoint

III. Design Process

A. Definitions

We denote I_i as a frame at an instant i. The temporal index is denoted by i. Each individual keypoint is referred by an index j. The keypoints used in the frame i are denoted by K_i^j. The fundamental matrix calculated between the frame F_i and F_{i+1} keypoints is denoted by $Fmat_i$. Similarly, the homography between the successive frames is denoted by H_i.

B. Method 1: Based on distance between epilines and key-points

In this method, the feature keypoints and descriptors for the interest points are found using the ORB. The correspondences between the successive frames were found by applying the Flann based matcher on respective keypoints and descriptors. Each correspondence contains pair of Keypoints (K_i^j, K_{i+1}^j)

The 3D keypoint X_t^j is mapped on the F_i as K_i^j and on F_{i+1} as K_{i+1}^j. We can leverage the epipolar constraints on these keypoints for finding the dynamic keypoints. As shown in Fig. 3, distance d_{i+1}^j between the epiline l_{i+1}^j on frame F_{i+1}, calculated from corresponding keypoints K_i^j and Fundamental matrix $Fmat_i$, and corresponding keypoint K_{i+1}^j on frame F_{i+1}. If this distance d_j between the corresponding keypoints K_i^j and K_{i+1}^j in the consecutive frames is significant, we can consider the keypoint K_{i+1}^j on frame F_j belongs to dynamic object. We can see in the figure that a change in the position of the keypoint increases the distance d_{i+1}^j between the keypoint K_{i+1}^j and epiline l_{i+1}^j. If the distance is greater than the set threshold τ, it is considered as a keypoint belonging to a dynamic object and marked as an outlier or else an inlier. The mathematical formulation for the above defined epipolar constraint is defined below,

$$K_{i+1}^j \cdot Fmat_i K_i^j > \tau$$

The Inlier set is then used for finding the camera poses $[R|T]$.

C. Method 2: Based on Motion Vector Clustering

Here, we find a good set of keypoints K_i^j in the initial frame F_i using Harris corner detector. These keypoints were

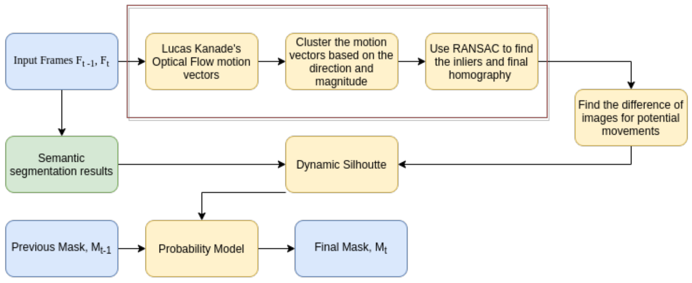
Input Frames F_{t-1}, F_t
Lucas Kanade's Optical Flow motion vectors
Cluster the motion vectors based on the direction and magnitude
Use RANSAC to find the inliers and final homography
Find the difference of images for potential movements
Semantic segmentation results
Dynamic Silhoutte
Previous Mask, M_{t-1}
Probability Model
Final Mask, M_t

Fig. 4. Pipeline for finding the dynamic mask using Motion vector clustering

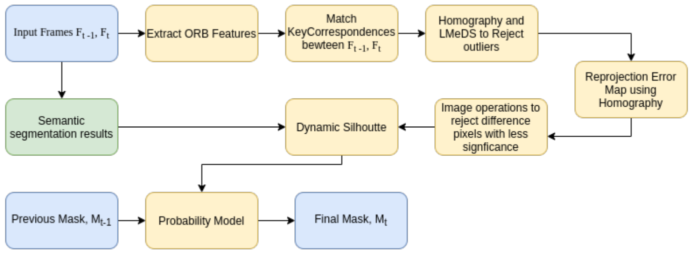
Input Frames F_{t-1}, F_t
Extract ORB Features
Match KeyCorrespondences bewteen F_{t-1}, F_t
Homography and LMeDS to Reject outliers
Reprojection Error Map using Homography
Image operations to reject difference pixels with less signficance
Semantic segmentation results
Dynamic Silhoutte
Previous Mask, M_{t-1}
Probability Model
Final Mask, M_t

Fig. 5. Pipeline for finding the dynamic Mask using Reprojection Error Map

tracked in the successive frames using the Iterative Lucas-Kanade optical flow method. In optical flow, an assumption of brightness consistency across the temporal axis is made. The following equation represents the optical flow equation applied on successive frames.

$$\nabla I \cdot \vec{V} = -I_t$$

Here, $\vec{V}$ represents the motion flow in second frame, ∇I represents the spatial intensity gradient, and I_t is Intensity gradient along the temporal axis. We have two unknowns (V_x, V_y) and one equation, hence need more equations to solve this problem. To solve this ambiguity, Lucas Kanade algorithm assumes that the local neighbourhood of a pixel have the same optical flow. After getting equations for the local neighborhood of the pixels, least squares can be used to solve the flow vectors and where the keypoints are in the successive frames. Likewise we can track all the keypoints in successive frames.

After tracking these keypoints in the successive frames, a new set of keypoint correspondences (K_i^j, K_{i+1}^j) were found which are present in both frames. As described in the Fig. 4, Motion vectors $\vec{v_i^j}$ are calculated using theese pair of keypoint correspondences K_i^j, K_{i+1}^j. Each motion vector consist of direction and magnitude. If camera is moving, we can see that most of the motion vectors have the similar direction and magnitude of the motion vector depends on the distance of the object from the camera for a rigid body transformation. After getting the set of motion vectors $\{ \vec{v_i^j} \}$ between correspondences, we will find the clusters among the motion vectors which are similar. We used K-means clustering to find the similarity of motion vectors, where K is set to the number of dynamic objects in the scene after panoptic segmentation. We will randomly pick the clusters and use of the keypoints related to the motion vectors in the cluster to find the homography H_i^j. After repeating the above steps for some iterations, from the set of homographies H_i^j which are obtained from random clusters, we will find the best homography, inliers and outliers using RANSAC method to find the best homography which minimizes reprojection error between keypoints.

Next, we will apply this homography on the first frame F_i to get the projected frame F_i'. Then we take the difference between the F_i' and F_{i+1} to obtain the difference image D_i. We applied the opening operation on D_i to reject the weak edges and get the strong differences candidates. We can then use the semantic segmentation results and match them with the difference candidates to find the potential dynamic regions S_i. Flood fill algorithm is used to get dynamic mask by using the difference candidates as the seed points. We maintain a probability model to get the robust dynamic object foreground M_i over the frames. We initialize this probability map with 0.5 for all the pixels initially.

$$P_{i+1} = (1-\alpha)P_i + \alpha S_i$$

$$M_{i+1}(x,y) = P_{i+1}(x,y) > \tau_p$$

Above equations represent the probability model for finding the final dynamic mask. We give the weight of α to new potential dynamic region and $1-\alpha$ to the previous probability map. Using the equation, we calculate the probability map at time step $i+1$. Thresholding the probability map by τ_p will give us the dynamic mask M_i. We can then use this inverted mask to get the static ORB keypoints.

D. *Method 3: Based on Reprojection Error Map*

In this, the ORB keypoints and descriptors are extracted and matched to get the correspondences between the two frames as explained in Method 1. After getting the correspondences, we will use the least median squared errors(LMeDs) for finding the best homography. The LMeDs is used because of its robustness to outliers. It gives us the minimum value for the median squared errors of homography over the sampled correspondences found.

$$H_i = \min \operatorname{med}_{1,2..i\leq p} H_i(K_t, K_{t+1})$$

We use this homography matrix to find the projection F_i' of F_i on F_{i+1}. Then, we find the reprojection error map by subtracting these two images. The following steps of finding the difference of images and curating the potential dynamic candidates using the probability model is same as described in Method 2. Pipeline for the third method is shown in Fig. 5.

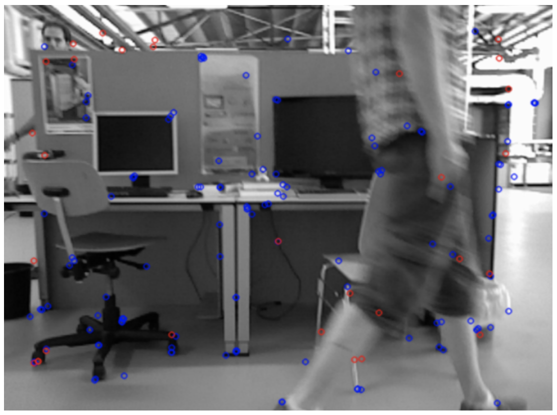

Fig. 6. Bad categorization of static and dynamic points due sudden camera movements

IV. Experiments and Results

A. Distance between epilines and keypoints

In Method 1, we have set the threshold τ as 3 pixels considering the central pixel as keypoint in frame F_{i+1} and farthest pixel is a corner pixel in the 5x5 neighborhood. We have run our experiment on a dynamic scene video from TUM dataset where the output represents static keypoints in blue and dynamic keypoints in red as shown in Fig. 6. We observed that the threshold τ varies depending on the depth of keypoint from the camera. It is evident that, if the object is far from the camera, then the distance between keypoints and epilines reduces. Therefore, we concluded that depth information is essential for efficiently setting the threshold for each of the keypoint. However, our depth data from the given dataset also had some noise. This method has the drawback that in case of sudden changes in camera movement, it will result in bad correspondences because of blur in the motion, as shown in Fig. 6.

B. Motion Vector Clustering

For the optical flow based method, we set to extract 100 corners, which are then tracked in the successive frames. If the tracked corners fall below 50 percent, again new 100 corners are extracted from the frame and tracked. These tracked keypoints are then used for finding the homography using the minimum projection error. We used Detectron2[10] network for the panoptic segmentation. The number of dynamic classes are limited to 5, assuming maximum 4 dynamic classes can be significant in the frame. We found out from experiments that setting 5 classes is working best for clustering the motion vectors which can compensate for both depth problems as well as dynamic objects. The visualisation at various parts of the pipeline are shown in Fig. 7, Fig. 8 and Fig. 9.

For the probability model, we have set the alpha to be 0.75 [9]. We can observe that some of the static regions are present even after applying probability model because the motion vectors also depend on depth of the key correspondences.

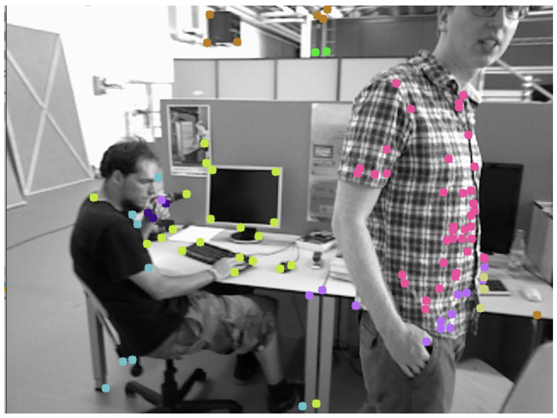

Fig. 7. Motion vector clustering for optical flow vectors on the successive images

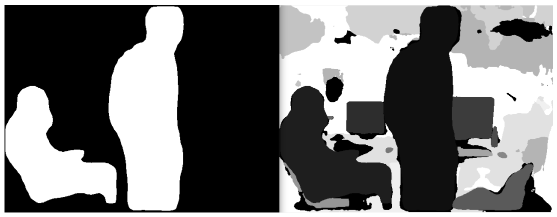

Fig. 8. Potential dynamic mask after matching with segmentation results

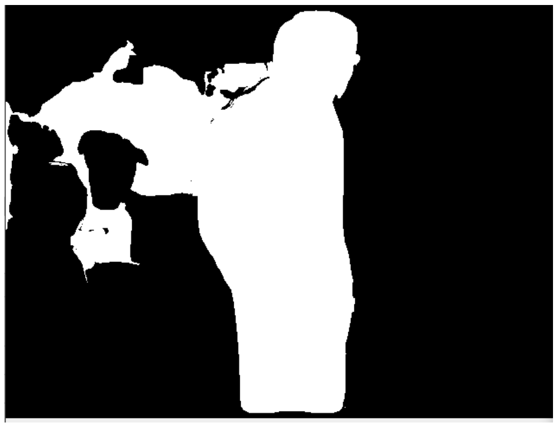

Fig. 9. Noisy dynamic region after probability update using motion vector clustering method

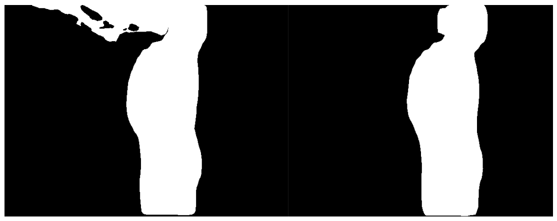

Fig. 10. Removal False positive dynamic regions using probability model from Reprojection error map method

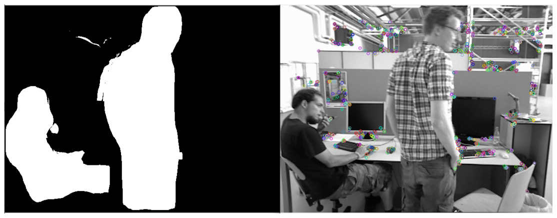

Fig. 11. Results of static ORB keypoints after using dynamic mask from Reprojection error map method.

C. Reprojection Error Map

For this, we extracted 1000 ORB keypoints and used Flann based matcher with minimum matches to be 50. For Lowe's ratio[11], we used min ratio as 0.75 for filtering the bad correspondences. Similar to Method 2, we initialized probability map with 0.5 for all pixel locations. We kept α equal to 0.75 for the probability model. The results of these experiments are shown in Fig. 10 and Fig. 11.

We tried to calculate quantitative metrics like F1 score to measure performance of each of these methods, but are limited due to lack of existing ground truth data for segmented dynamic objects in videos.

V. Conclusion & Future Work

Overall, we have proposed an out of the box dynamic object detection framework which can be integrated into front end of SLAM to filter out keypoints corresponding to dynamic objects. This helps to create SLAM systems robust to dynamic environments. Our framework is independent and can be used on other tasks where dynamic objects are needed from videos, such as video tracking. We also realize potential research frontiers where further improvement can be achieved by tracking the dynamic objects and using their motion models to better estimate the camera pose and location.

Acknowledgment

Thanks to all the fellow classmates of ECE209AS for their feedback and suggestions on the project.

References

[1] Raul Mur-Artal and Juan D Tardós. "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras". In: *IEEE transactions on robotics* 33.5 (2017), pp. 1255–1262.

[2] Sen Wang et al. "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 2043–2050.

[3] Ruihao Li et al. "Undeepvo: Monocular visual odometry through unsupervised deep learning". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 7286–7291.

[4] Wenshan Wang, Yaoyu Hu, and Sebastian Scherer. "TartanVO: A Generalizable Learning-based VO". In: *arXiv preprint arXiv:2011.00359* (2020).

[5] Yuxiang Sun, Ming Liu, and Max Q-H Meng. "Motion removal for reliable RGB-D SLAM in dynamic environments". In: *Robotics and Autonomous Systems* 108 (2018), pp. 115–128.

[6] Jerome Revaud et al. "Epicflow: Edge-preserving interpolation of correspondences for optical flow". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1164–1172.

[7] Chao Yu et al. "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.

[8] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495.

[9] Irene Ballester et al. "DOT: dynamic object tracking for visual SLAM". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 11705–11711.

[10] Yuxin Wu et al. "Detectron2. 2019". In: *URL https://github. com/facebookresearch/detectron2* 2.3 (2019).

[11] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.