Towards a Robust Visual SLAM: Dynamic ORB-SLAM

Xin Luo, Matthew Philippi, Guankun Su, Niankai Yang

I. Introduction

Simultaneous localization and mapping (SLAM), the problem of simultaneously estimating the state of a robot and constructing a map of the environment, is an essential ability for an autonomous robot [1]. While the most accurate solutions for SLAM are usually achieved by using both lidar and visual odometry (e.g., the work in [2]), the best combination of sensors is still an open question for performing SLAM. While the use of lidar enhances the accuracy of the SLAM system, it also raises a cost barrier which constrains the application range of the SLAM system. On the other hand, visual SLAM, which performs SLAM with only camera images, has become an appealing alternative for SLAM due to easy accessibility and the reduced cost of cameras.

Typically, visual SLAM algorithms can be divided into two classes: (i) feature-based methods, and (ii) direct methods [3]. In feature-based methods, a sufficient amount of features are first extracted from the images and matched with successive images for estimating the camera poses. For example, MonoSLAM was proposed in [4] to achieve the estimate of the camera motion and map of 3D unknown structures based on an Extended Kalman Filter (EKF). The same SLAM problem was also solved in [5] with parallel tracking and mapping, which reduced the computational complexity compared to MonoSLAM. Due to the use of feature extraction, feature-based methods are capable of handling large inter-frame displacements. Unfortunately, the step of feature matching in the algorithm is error-prone. For direct methods, since the images from the camera are used directly for camera pose estimations, no prior step for feature matching is required. However, the direct use of images makes direct methods sensitive to the operating environment, and therefore limits their application range.

In this study, we will focus on investigating ORB-SLAM2, a feature-based visual SLAM system. As mentioned above, the performance of the feature-based visual SLAM methods will be, to a large extent, affected by the robustness of the feature matches. If inconsistent feature matches (i.e., outliers) are not rejected before being fed to the back end optimization, both the stability and the optimality of the SLAM performance will be affected [6]. To combat this issue, a large amount of research has been devoted to eliminating outliers from feature matches. Typically, this is achieved by either incorporating outlier detection algorithms (e.g., [7]) or adding robustness to the back end optimization (e.g., [8]).

In this study, we propose two modifications to the original ORB-SLAM2 implementation, which focus on improving outlier detection. The first modification, the dynamic feature filter, aims at incorporating the awareness of the dynamic objects into the outlier detection. This is achieved by regarding the features that have large absolute velocities as dynamic objects and rejecting these features during the pose estimation. The second modification, the dynamic outlier thresholding algorithm, is proposed for addressing the issue of insufficient feature matches when visual SLAM is used in feature-sparse environments. Based on the distribution of all the feature matches from two successive frames, the dynamic outlier thresholding algorithm relaxes the rejection of low quality feature matches to prevent the SLAM system from losing track. To validate the effectiveness of the two proposed modifications, the dynamic feature filter is evaluated on the KITTI dataset [9] and the dynamic outlier thresholding algorithm is evaluated on the Devon Island Rover Navigation dataset [10]. Simulation results show that the proposed modifications are able to improve the localization performance on these two datasets.

The remainder of this report is organized as follows: in Section II, the main characteristics of ORB-SLAM2 are introduced. The dynamic feature filter and its evaluation results on the KITTI dataset are presented in Section III. Section IV introduces the dynamic outlier thresholding algorithm followed by the simulation results on the Devon Island Rover Navigation dataset. Finally, conclusions and future work are discussed in Section V.

II. ORB-SLAM and ORB-SLAM2

In this section a high level introduction of ORB-SLAM2 is given. Since ORB-SLAM2 is an extension of ORB-SLAM [12], the introduction in this section will begin with an explanation of key features of ORB-SLAM.

ORB-SLAM is a feature-based real-time monocular (stereo/RGB-D) SLAM system. As its name implies, ORB-SLAM relies on extracting ORB features, which are a combination of FAST keypoint detector and BRIEF descriptor [13]. Due to the scale and rotation invariant properties of ORB features, ORB-SLAM is able to use these features during its tracking, mapping, re-localization and loop closing tasks. The second feature of ORB-SLAM is the use of covisibility and essential graphs. Both graphs are constructed based on keyframes. Keyframes are frames that are deemed to poses the most key features in the local region. The covisibility graph is used for local mapping, and the essential graph is used for loop closing and global mapping. This two-layer setup makes it possible to maintain a sufficient level of accuracy while reducing the computation complexity of the SLAM algorithm (specifically the computation for global

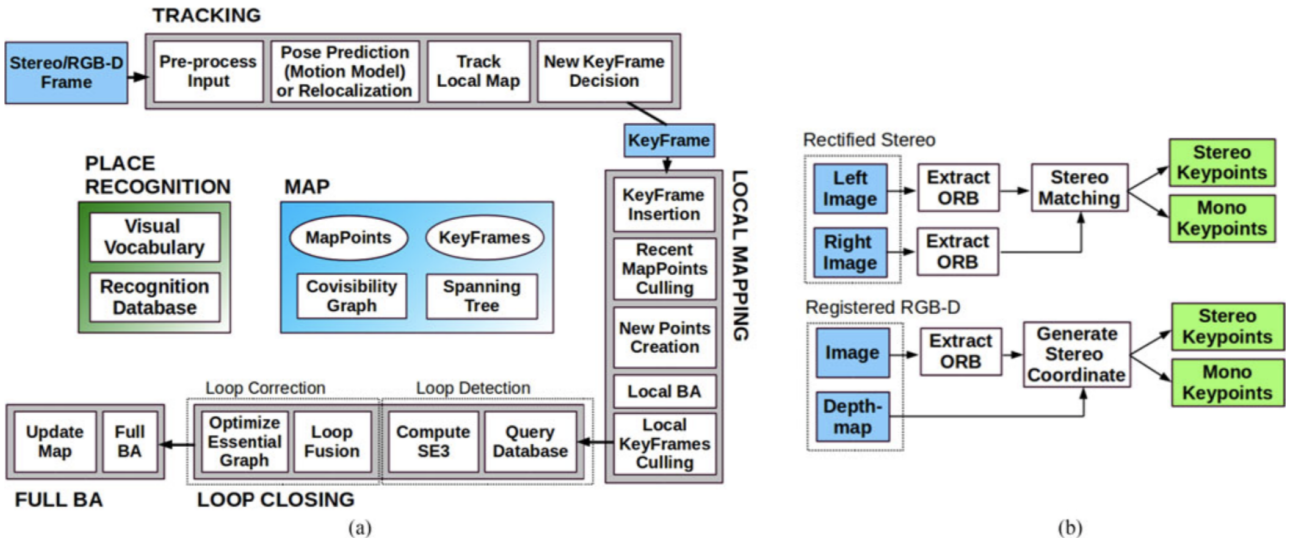

Fig. 1. Schematic of ORB-SLAM2 [11]

optimization). The last key feature of the ORB-SLAM is its survival of fittest approach for processing map points and keyframes. Namely, ORB-SLAM uses generous criteria when deciding to add a new map point or keyframe, while using more restricted criteria for removing an existing map point or keyframe. As a consequence, the main features in the map will be less likely to be omitted by the ORB-SLAM algorithm.

Based on the above three key features, ORB-SLAM processes the data in three threads as shown in Fig. 1(a). The first thread performs tracking by identifying the current pose of the camera based on the last frame. The decision on whether or not a new frame is a keyframe is made in this thread. The main computational burden of the first thread is to process the images from the cameras (i.e., extract the ORB features and pose estimation). The second thread of ORB-SLAM is in charge of constructing the local map by optimizing the covisibility graph. Moreover, the insertion and culling of the map points and keyframes are performed in this thread. The last thread in the ORB-SLAM mainly focuses on global map optimization through loop closures performed via bags-of-words place recognition.

ORB-SLAM2 extends the framework of ORB-SLAM by incorporating the ability to use images from stereo or RGB-D cameras as shown in Fig. 1(b). This is achieved by adding the concept of close and far stereo keypoints (similar points can be computed from the RGB-D camera with transformation). Based on this classification, close point are safely triangulated to provide a reliable depth estimation, while far points are used only for providing the rotation information.

III. Dynamic Feature Filter

In this section, we introduce the Dynamic Feature Filter, which introduces the ability for ORB-SLAM2 to reject dynamic features. Evaluation of the Dynamic Feature Filter is conducted on the KITTI dataset. The KITTI dataset

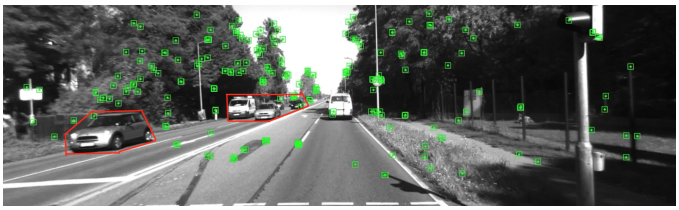

Fig. 2. Dynamic objects in KITTI sequence 04 (Green dots are ORB features; Red blocks highlight the dynamic objects)

is obtained from driving a ground vehicle in Karlsruhe, Germany. It includes stereo camera images, laser scans, high-precision GPS measurements and IMU accelerations from a combined GPS/IMU system. This dataset contains 23 sequences (sequence 00-22) which are real-world traffic situations that ranging from rural areas to inner-city scenes with many static and dynamic objects. In this project, we use only the gray-scale stereo images from sequence 00 to 10 with online benchmarks for testing our algorithm.

A. Dynamic Feature Filter

The problem of dynamic objects is observed during the testing on the KITTI dataset. The tests on KITTI sequence 01 and sequence 04 show that pose estimation error of the robot increases when dynamic objects are present in the environment. This phenomenon is exacerbated as additional dynamic objects are introduced in the environment. For example, at the end of sequence 04, there are multiple incoming vehicles which can be seen in Fig. 2.

Since ORB-SLAM2 assumes a static environment, features from dynamic objects will be assumed static, which means that the locations of all map points are not mobile among all frames. Based on this assumption, robot pose is estimated by aligning map points detected in the current frame to those of the previous frame using a rigid body

transformation. Therefore, features from dynamic objects will introduce uncertainty in pose estimation.

To address the issue of dynamic objects, we propose the Dynamic Feature Filter (DFF) to reject features detected from dynamic objects before performing the optimization. Dynamic objects are defined to be objects with nonzero speed in the world frame. Static objects are defined to be objects with zero speed in world frame.

In order to segment dynamic and static object features, we estimate the location of map points detected from the current observation and set a threshold for segmenting. In original ORB-SLAM2, it uses a "constant velocity" model to predict camera pose based on previous frames. After we triangulate matched features from left and right stereo images, their location in the camera coordinate frame is obtained. Based on the camera pose predicted from "constant velocity" model, the locations of matched features in world coordinate are then predicted. Their velocities are then estimated by dividing their location change by the time difference between consecutive observations. Finally, a threshold speed can be chosen based on some prior information to segment these feature matches into dynamic feature or static feature according to their estimated speed in the world frame.

In order to find the appropriate threshold, we plot the histogram of speed for each map point between consecutive observations in Fig. 3. This allows us to see that most features fall into the leftmost two categories, which have a speed ranging from $0 - 20m/s$. However, as we can see from Fig. 3, due to the noise in the velocity estimation, there is no clear division between the dynamic and static objects. This can be attributed to the multiple noise sources in the map point speed estimation process (e.g., the camera pose estimation, the map point's location in the image). Therefore, to remove these noise in the map point speed estimation, we incorporate an averaging filter to smooth the speed estimation. For all map points, their velocity estimates are memorized in a queue for a certain horizon. Thus, newly estimated velocities will be pushed onto the queue. When the queue reaches its capacity, which is set to be the horizon, the filter will pop the oldest velocity in the queue to maintain its capacity. The average speed for map point i is then calculated as

$$||\bar{v_i}|| = ||\frac{v_i^h + v_i^{h-1} + \cdots + v_i^1}{h}||, \tag{1}$$

where h is the horizon, and $||\cdot||$ computes the speed from velocity. To demonstrate the effect of using this moving average strategy, the same histogram of speed for each feature between consecutive observations with the moving average is given in Fig. 4. It can be seen in Fig. 4 that only few map features with large speed are observed, which validates the effectiveness of the moving average strategy.

Based on the computed average speed, map points are then segmented into static and dynamic object sets using the logic in (2). S is the set of static map points, D is the set

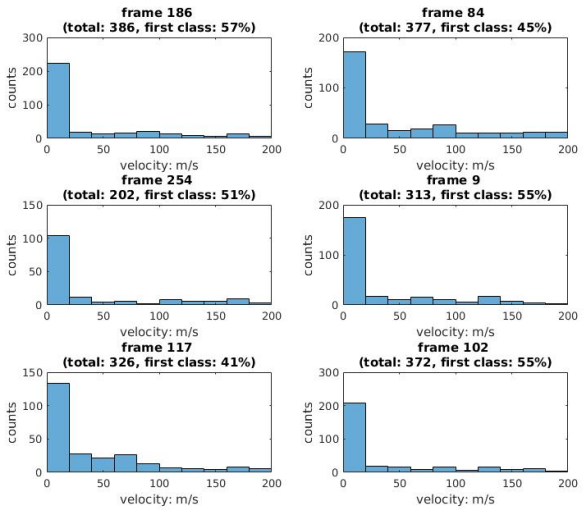

Fig. 3. Histogram of speed of features from randomly selected frames in KITTI sequence 04

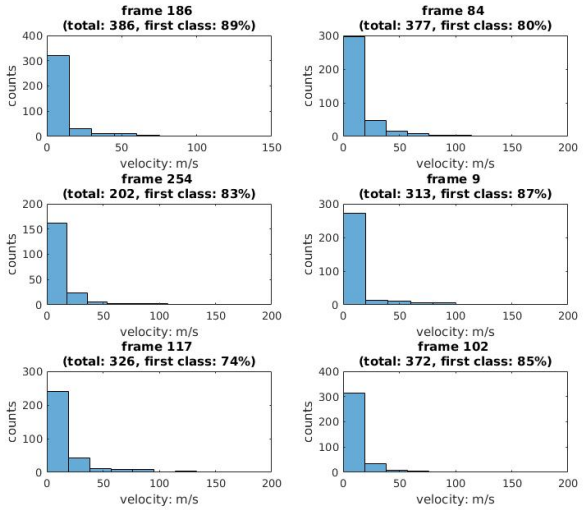

Fig. 4. Histogram of speed of features from randomly selected frames in KITTI sequence 04 with moving average filter

of dynamic map points and $||v_{ref}||$ is the speed threshold.

$$\begin{cases} p_i \in S, ||\bar{v_i}|| < ||v_{ref}|| \\ p_i \in D, ||\bar{v_i}|| \geq ||v_{ref}||. \end{cases} \tag{2}$$

B. Simulation Results on KITTI dataset

To validate the effectiveness of the proposed DFF, we conducted testing on sequence 00-10 in the KITTI dataset. For the KITTI dataset, we choose the speed threshold to be $40m/s$ for sequence 01, and $20m/s$ for all others. The horizon for the moving average is 10. When DFF is applied with velocity threshold of $20m/s$, the root mean squared error (RMSE) of translation and rotation for sequence 00 to 10 in the KITTI dataset is listed in Table I. From

TABLE I

RMSE ON KITTI DATASET (DYNAMIC FEATURE FILTER)

	Translation (m)		Rotation (rad)	
	ORB-SLAM2	DFF	ORB-SLAM2	DFF
Seq 00	5.2003	5.1715	0.1236	0.1206
Seq 01	18.6214	17.1409	0.2272	0.1481
Seq 02	8.9950	8.6345	0.1496	0.1481
Seq 03	1.5757	1.2888	0.0864	0.0875
Seq 04	0.5478	0.3762	0.0036	0.0042
Seq 05	1.0672	1.1338	0.0837	0.0800
Seq 06	1.7647	1.5214	0.0253	0.0232
Seq 07	0.7191	0.8106	0.0688	0.0694
Seq 08	6.2716	5.1925	0.1558	0.1464
Seq 09	3.0663	4.0845	0.1105	0.0819
Seq 10	3.2242	3.4769	0.1269	0.1409

Table I, it can be seen that DFF maintains the same accuracy as ORB-SLAM2 when there is no dynamic object in the dataset. Additionally, when dynamic objects are present, DFF outperforms ORB-SLAM2. With sequence 04 as an example (see Fig. 5), in the latter half of the sequence, the translational accuracy of ORB-SLAM2 with DFF (yellow line) is far more accurate than that of ORB-SLAM2 (orange line). This is due to the fact that DFF rejects the dynamic ORB features that are selected from the many cars coming from the opposite direction. However, according to Table I, there are some sequences (e.g., sequence 01 and sequence 09) in which the translation and rotation RMSEs from DFF are larger than those of ORB-SLAM2. The cause of this is mainly due to the estimation of map points' absolute velocity being based on the velocity of the cameras obtained from "constant velocity" model. Thus, to improve the performance of DFF, an accurate estimate of the camera velocity is needed. For example, in sequence 01, the velocity of the car is relatively large, so we adjusted the threshold for DFF from $20m/s$ to $40m/s$. From the results shown in Fig. 6, we can see that DFF is able to achieve better accuracy than ORB-SLAM2. Finally, it should be noted that although both the translational and rotational accuracy can be improved by using DFF, the error does not go to zero. Motion model error in velocity estimation and inaccuracy of measurement data are other potential sources of error.

IV. DYNAMIC OUTLIER THRESHOLDING ALGORITHM

In this section, the Dynamic Outlier Thresholding Algorithm (DOTA) is introduced. This algorithm is an improvement on the original static thresholding greedy nearest neighbor algorithm used by ORB-SLAM2. DOTA is able to improve data association in all environments and tracking robustness in feature sparse environments with minimal changes to computational cost. The performance of this algorithm is evaluated on the Devon Island Rover Navigation dataset.

A. Devon Island Rover Navigation Dataset Background

The Devon Island Rover Navigation dataset tracks an autonomous vehicle in an uninhabited area that consists mostly of rocky terrain with limited distinctive features to

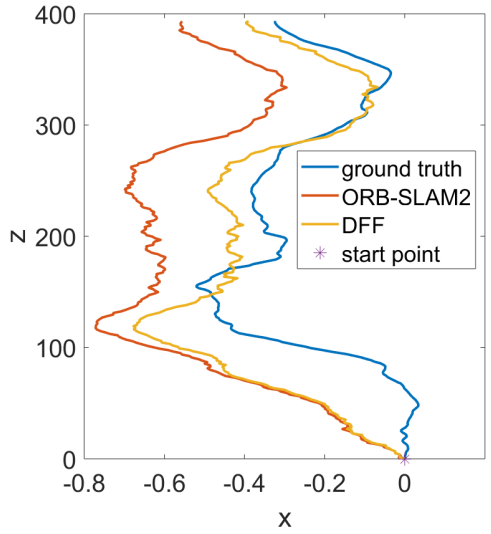

Fig. 5. 2D Path of sequence 04

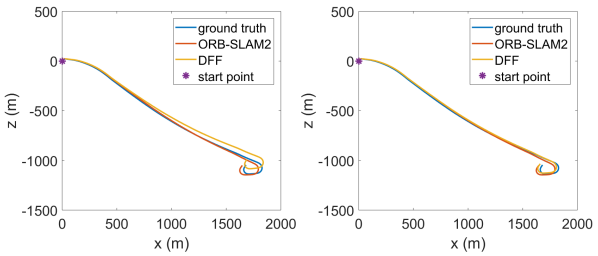

Fig. 6. Sequence 01 with velocity threshold of 20m/s and 40m/s

navigate off. The purpose of this dataset is to simulate an off-world environment similar to Mars or the Moon. This environment poses significant challenges to algorithms that rely on distinct landmarks for mapping and localization. We chose to focus on sequence 00 of the dataset which provides no loop closures. Before running ORB-SLAM2 on a dataset, the ORB parameters must be specified for the ORBextractor class in a file that is read at startup. These parameters consist of `nFeatures`, `scaleFactor`, `nLevels`, `iniThFAST` and `minThFAST`. When sequence 00 is run with the same ORB parameters as those for the KITTI dataset, ORB-SLAM2 loses the track at frame 180 out of 2087 and is unable to regain the track. By tuning the ORB parameters, ORB-SLAM2 is able to extend the track to frame 1683 out of 2087, but was never able to progress through the whole dataset without losing the track. The ORB parameters and resulting trajectories are shown in Section IV-C. The inability for ORB-SLAM2 to hold the track through the whole sequence drives the need for a new Dynamic Outlier Thresholding Algorithm discussed in detail in the following section.

B. Dynamic Outlier Thresholding Algorithm

The problem of insufficient feature matches is observed during the testing on the Devon Island Rover Navigation

dataset. Without significantly increasing both the number of extracted ORB features and the χ^2 threshold, ORB-SLAM2 fails in obtaining a pose estimate through the entire sequence. To complicate matters, there is no global loop closure within the dataset, so any chance for relocalization disappears as soon as tracking is lost. This demonstrates an intrinsically brittle characteristic of the ORB-SLAM2 system in that it relies on bundle adjustment for all localization and mapping operations. If a current frame does not contain any feature that can be matched to previously observed features, there is no possible way to link the frame's state to the remainder of the graph. Thus, by nature of its architecture, to maintain a current state estimate relative to the rest of the map, matches must be found at each frame.

Whether matches can be found depends on several factors. Firstly, it depends on the features that were extracted from the current frame. These features cannot necessarily be controlled deterministicly. There is an upper bound to the number of ORB features that can be extracted in terms of computation cost, and the quality of any ORB features extracted is not guaranteed. There is also dependency on the map points available for matching which is given a priori from the current map. The last major factor is the data association method. In the current ORB-SLAM2 implementation, data association is done via a simple greedy nearest neighbor algorithm with a preset Mahalanobis distance threshold. There are also other match compatibility checks done occasionally, but these are used to reject matches rather than to accept matches. This data association algorithm was found to be a significant contributor to the lack of robustness in ORB-SLAM2 in feature sparse environments. In order to prevent loss of tracking, parameters like the Mahalanobis distance threshold(s) and second best nearest neighbor rejection ratio need to be manually tuned to handle the environment. Even then, the algorithm usually only suffers difficulty in a small subset of frames out of the entire sequence. Frequently, when the Mahalanobis distance threshold is increased, bad data associations are made during frames where the algorithm normally has no difficulty even with tight thresholds. To address this issue a new nearest neighbor algorithm is introduced that produces optimal nearest neighbor matches while dynamically adjusting the Mahalanobis distance threshold and other parameters to improve data association and outlier detection so that the integrity of the graph can be ensured.

1) Optimal Nearest Neighbor Matching: ORB-SLAM2 utilizes a greedy nearest neighbor data association scheme that tracks the two best matches for any given map points. A search is done through the entire set of frame features to be matched and distances are compared with the current map point in consideration. The two best distances are stored. After that, the original algorithm considers the following conditions: (i) the best distance is below the Mahalanobis distance threshold, and (ii) the second best distance is larger than the best distance by a specified ratio, (i.e. there are no other close possible matches).If both conditions are held, then the best match is accepted for that map point. This process is then iterated for every map point that is considered "covisible" for matching purposes. Two problems occur during this process. The first is that the algorithm does not find the globally optimal nearest neighbor match. The order in which the map points are matched is arbitrary, and once a match is accepted, that frame feature is no longer available to be matched with another map point. It is possible that the closest match to a frame feature in a nearest neighbor sense is actually a map point at the end of the array. However, the arbitrary Mahalanobis distance threshold may have dictated that an earlier map point was "good enough" to be matched with it instead. These errors have the potential to cascade through the matching process as poor data associations induce additional poor data associations. Instead of performing data association on a map point by map point basis, DOTA assembles a sparse data association matrix that contains the distance metric for every possible association between map points and frame features. This sparse matrix (vector of distances and indices) is then sorted in ascending order according to the distance metric.

Because the sparse matrix vector is sorted, the first entry in this set of sparse entries is known to be the first globally optimal nearest neighbor match. DOTA iteratively extracts the distance and indices of the first entry in the sparse matrix vector, and if either the distance is below the distance threshold, or a minimum number of matches has not yet been met, stores that match as a globally optimal match. For every match made, all other data association matrix entries containing either the same map point or frame feature index are eliminated. The first entry in the sparse matrix vector is then again the globally optimal nearest neighbor match, and the process repeats until either the matrix has zero entries or the next match fails to meet the match criteria (minimum number of matches or distance threshold). By construction, if the next match fails, the remainder of the potential matches in the matrix will also fail. This process can be visualized either at a high level as a set of operations performed on a full matrix with sparse entries (see Fig. 7) or directly as a set of operations on a vector of arrays that represent this matrix in heap memory (see Fig. 8). All distance comparisons occur in a single sorting step, redundant matrix entries are eliminated at every match step, matching is terminated as soon as the algorithm determines further matches meeting match criteria cannot be made, and no time is spent searching empty entries of a dense matrix. All of these factors together allow the algorithm to find globally optimal matches without significantly impacting execution time. DOTA was measured side by side against the original implementation and no difference above noise could be seen in mean or maximum frame times. This new approach to nearest neighbor search has supplanted all previous data association implementations in ORB-SLAM2 and provides the extra data association robustness needed to implement the second innovation of DOTA.

2) Dynamic Outlier Threshold: As noted previously, ORB-SLAM2 is intrinsically brittle without regular loop closures due to the need for ORB feature matches to previously observed map points every frame to maintain tracking.

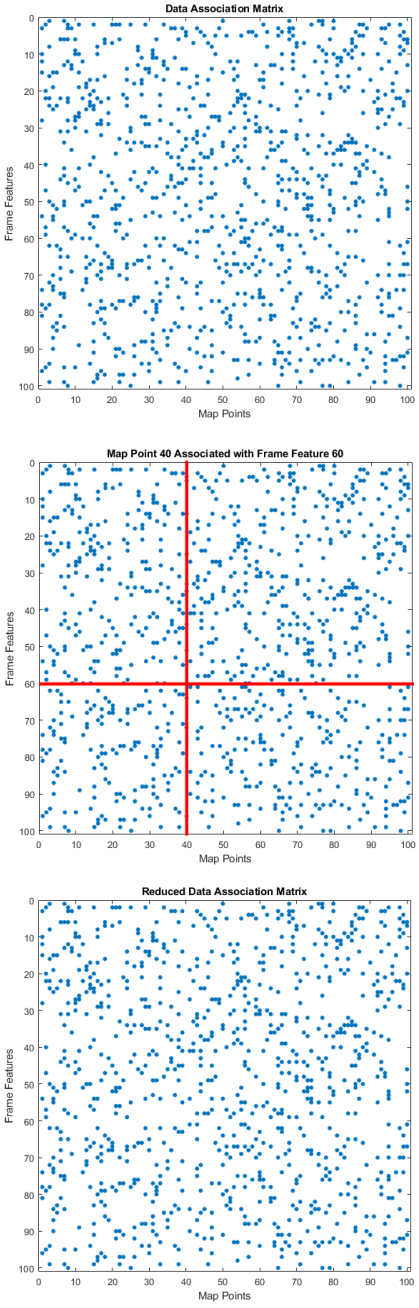

Fig. 7. Sparse matrix representation of Optimal Nearest Neighbor Matching

Whether a sufficient number of matches are detected to adequately compute a new pose estimate is never guaranteed under the previous form of ORB-SLAM2. It is also subject to both arbitrary user-defined thresholds and a multitude of other checks that can each disqualify potential matches.

The initial hypothesis of this investigation was that given a sufficient number of ORB features extracted from each frame, there should be at least a small subset of ORB features that either represent matches or very close matches to some of the previously observed map points. Even if there existed only very poor matches out of the ORB features extracted, accepting the minimum number to compute a pose update

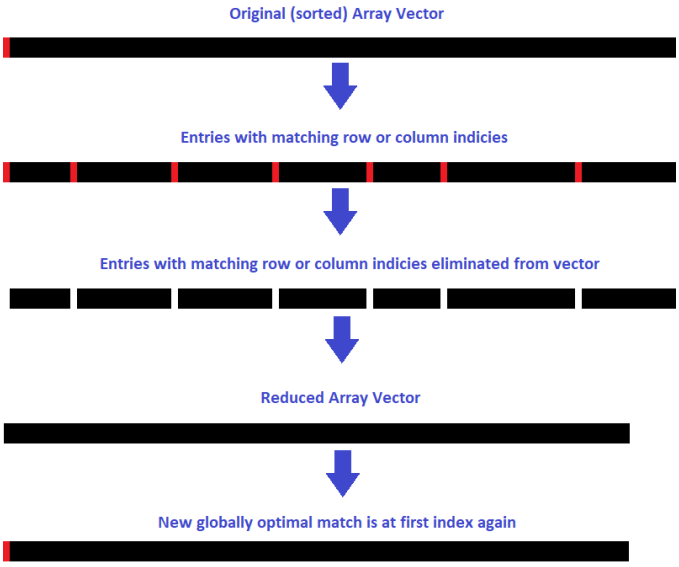

Fig. 8. Array vector representation of Optimal Nearest Neighbor Matching

ensured that there was at least a very weak link in the graph that maintained its integrity that could subsequently be strengthened/fixed through either loop closure or reobservation. More often than not though, loss of tracking was not necessarily caused by poor available matches but rather a combination of bad matches being accepted and good matches being rejected during the frame that tracking was lost. Thus, ideally, the algorithm would use a tight distance threshold to reject bad matches, when matches are abundant, to ensure accuracy and use a loose distance threshold when matches are sparse to ensure robustness to noisy camera data and sudden pose changes. This behavior can be described as a function that dynamically adjusts the effective distance threshold value according to the distribution of feature match quality between two sets of data

$$p_{th} = w_0(\xi)p_0, \tag{3}$$

where p_{th} is the dynamic threshold, $w_0(\xi)$ is the adaptive factor function that varies as a function of ξ, and p_0 is the static threshold value (function gain). ξ represents the distribution of feature match quality between two sets of data. The reason why a motion model was not incorporated in this investigation is twofold. First, it subjects ORB-SLAM2 to the requirement that control data must be available to the algorithm as well as camera data, and a motion model must be specific to every new application. Second, control data is not available on many datasets like KITTI, so this renders the algorithm irrelevant for many SLAM comparisons and applications that posses only sensor data.

To accomplish this dynamic outlier thresholding, a very simple scheme is introduced. When ORB-SLAM2 cannot recover an adequate pose transformation from its constant velocity model, it attempts a bag-of-words search from the reference frame pose. If this fails, the algorithm cannot update the graph, and it loses the track. Later, the local map

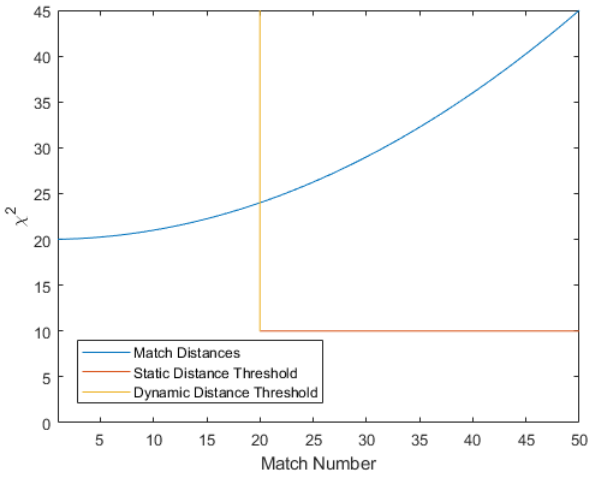

Fig. 9. Dynamic Threshold Function

is then tracked from this estimation, and if this fails during a search by projection, the algorithm loses track. During the bag-of-words search, the first 15 globally optimal matches are accepted before thresholding begins. Later, during the local map tracking, the first 30 globally optimal matches are accepted before thresholding begins. If more than the minimum number of matches can be found through thresholding alone, then the number of matches accepted is governed by the distance threshold. Thus, this process is robust when a difficult frame is encountered, yet completely transparent when available matches are in abundance. This means the user can set a tight distance threshold to ensure accurate transforms where possible without the worry of inducing a loss of track scenario when there is sufficient information to continue to localize the vehicle. This process can easily flag when multiple frames have had matches forced to inform any higher level logic that a loop closure must be performed to ensure continuing accuracy. Meanwhile it at least provides a rough approximation of the current vehicle state that can be used to help it return to an explored area. This dynamic threshold can be visualized in Fig. 9.

C. Simulation Results on the Devon Island Rover Navigation dataset

DOTA shows significant improvements on the Devon Island Rover Navigation dataset for any given set of algorithm settings. For all tested settings, ORB-SLAM2 in its original form is incapable of tracking through the complete sequence. The track is lost, often very near the start, and it is never recovered for the remainder of the sequence. When settings are tuned to permit localization and mapping as far into the sequence as possible, the state estimate obtained is quite poor, and execution speed drops precipitously with an increasing number of ORB features extracted. DOTA on the other hand always finds an approximate state estimate, even when that state estimate has significant error. It should be noted that this error is usually less than the error seen in the original ORB-SLAM2 implementation given equivalent

TABLE II
ORB PARAMETERS

	nFeatures	scaleFactor	nLevels	iniThFAST	minThFAST
KITTI	2000	1.2	8	20	7
Tuned	5000	1.2	8	8	3

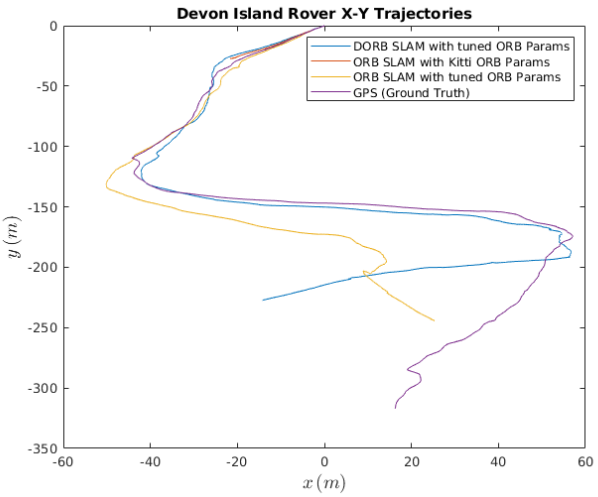

Fig. 10. Devon Island X-Y Trajectories

settings. Many different settings were tested, but the two most significant set of parameters are shown in Table II. The KITTI settings represent a control set while the tuned settings are the those which were found to track the furthest through sequence 00 of the Devon Island Rover Navigation dataset.

Fig. 10 shows the resulting horizontal trajectories and Fig. 11 shows the resulting vertical trajectories with the different ORB parameters and DOTA tested. When using DOTA and tuned ORB parameters, ORB-SLAM2 is able to track through the entire sequence and tracks the ground truth trajectory reliably for far longer than original ORB-SLAM2 implementation.

V. CONCLUSIONS AND FUTURE WORK

In this report, a Dynamic Feature Filter (DFF) and a Dynamic Outlier Thresholding Algorithm (DOTA) are proposed to address the outlier detection problem in the ORB-SLAM2 system when performed in different environments. To tackle the problem of low quality feature matches from dynamic objects, DFF estimates the absolute speed of the objects in the world frame based on the ego-motion of the camera (from a constant velocity model). The objects with large absolute speed are rejected by DFF since they are more likely to be dynamic objects that can produce low quality feature matches.

To handle the issue of insufficient feature matches in feature-sparse environment, DOTA is presented. With adaptive thresholding, DOTA is able to guarantee that enough feature matches will survive the outlier detection to perform pose estimation. If the quality of the feature matches are

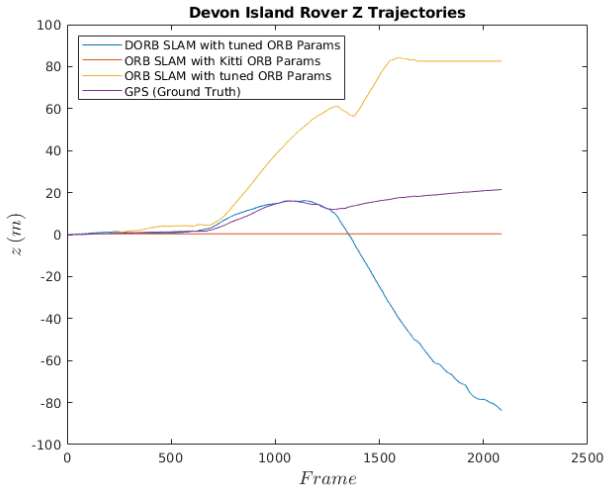

Fig. 11. Devon Island Z Trajectories

good, DOTA will not raise the threshold and quality of the feature matches used in the optimization is maintained. Additionally, an optimal nearest neighbor matching is implemented in DOTA to ensure optimality of the feature matching process. This is achieved by determining feature matching after searching through all the feature matching possibilities.

Simulation results on KITTI dataset (for DFF) and on Devon Island Rover Navigation dataset (for DOTA) verify the effectiveness of the proposed algorithms. It is shown that DFF is able to outperform ORB-SLAM2 for most sequences in the KITTI dataset, and substantial improvement is observed on sequences which have more dynamic objects. For DOTA, the algorithm is able to successfully obtain an estimation through the whole sequence. Future work will be focused on: (i) incorporating camera velocity estimation into DFF; (ii) combining DFF and DOTA as the dynamic ORB-SLAM algorithm and conducting extensive testing on different datasets to validate its effectiveness. All the code used in this study is publicly available at the following github link: **https://github.com/FeliksBrant/Dynamic_ORB-SLAM**. (A backup of the code is available at the following gitlab link: https://gitlab.eecs.umich.edu/rob530team6/orb-slam2.)

REFERENCES

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[2] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2174–2181.

[3] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 15–22.

[4] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 6, pp. 1052–1067, 2007.

[5] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society, 2007, pp. 1–10.

[6] P. F. Alcantarilla, J. J. Yebes, J. Almazán, and L. M. Bergasa, "On combining visual slam and dense scene flow to increase the robustness of localization and mapping in dynamic environments," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 1290–1297.

[7] M. Derome, A. Plyer, M. Sanfourche, and G. L. Besnerais, "Moving object detection in real-time using stereo from a mobile platform," *Unmanned Systems*, vol. 3, no. 04, pp. 253–266, 2015.

[8] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *2013 IEEE International Conference on Robotics and Automation*. Citeseer, 2013, pp. 62–69.

[9] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[10] P. Furgale, P. Carle, J. Enright, and T. D. Barfoot, "The devon island rover navigation dataset," *The International Journal of Robotics Research*, vol. 31, no. 6, pp. 707–713, 2012.

[11] 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.

[12] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[13] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," 2011.