Practical Persistence Reasoning in Visual SLAM

Zakieh Hashemifar[o] and Karthik Dantu[o]

***Abstract*—Many existing SLAM approaches rely on the assumption of static environments for accurate performance. However, several robot applications require them to traverse repeatedly in semi-static or dynamic environments. There has been some recent research interest in designing persistence filters to reason about persistence in such scenarios. Our goal in this work is to incorporate such persistence reasoning in visual SLAM. To this end, we incorporate persistence filters [1] into ORB-SLAM, a well-known visual SLAM algorithm. We observe that the simple integration of their proposal results in inefficient persistence reasoning. Through a series of modifications, we improve this persistence filtering. Using two locally collected datasets, we demonstrate the utility of such persistence filtering as well as our customizations in ORB-SLAM. Overall, incorporating persistence filtering could result in a significant reduction in map size (about 30% in the best case) and a corresponding reduction in run-time while retaining similar accuracy to methods that use much larger maps.**

I. Introduction

Advances in hardware, software and sensing have resulted in the development of robots capable of executing service applications such as floor cleaning [2], telecommuting [3], and item delivery in public spaces [4]. Many such applications involve repeated traversals in the same urban environment. Simultaneous Localization and Mapping (SLAM) is an extensively researched topic used for robot navigation in such environments. However most existing approaches either rely on the assumption of a static environment or handle particular type of dynamic objects, leading to a poor and inaccurate performance in dynamic settings.

There are two types of dynamics possible within a scene - (i) *Dynamic:* Dynamics in a short timescale that occur during a single robot traversal such as moving cars and people, (ii) *Semi-Static:* Dynamics across longer timescales that can only be detected across multiple visits to the same location such as chairs that have been moved. The main difference between these categories is the possibility of direct observation. The first category could get filtered by the SLAM front-end while the second one requires reasoning over multiple visits to update the map.

Our goal is to improve visual SLAM operation in semi-static environments. [1] is a recent work that models persistence in such environments. This approach neither ignores infrequent moving objects nor requires any training. Unfortunately, there is no practical implementation of their persistence filter in visual SLAM algorithms.

Our goal is to attain efficient persistence filtering in visual SLAM. To this end, we modify ORB-SLAM2 [5], a representative visual SLAM algorithm, to incorporate persistence. The estimated persistence probabilities are used to evaluate the continued availability of individual features. Each verified re-observation or missed detection is used to update the persistence filters. Through this implementation as well as its evaluation, we make the following contributions: (i) We modify ORB-SLAM to incorporate a persistence filter. Using empirical observations, we make several modifications to adapt the persistence filter to visual SLAM, (ii) Using two datasets collected across six separate traversals each, we quantify benefits of such persistence in a realistic setting, (iii) We also validate our implementation on a publicly available dataset [6], (vi) We have open-sourced our customization of ORB-SLAM [7].

II. Related Work

Some approaches try to achieve robustness in dynamic environments by separating dynamic segments from static background. [8] constructs two distinct maps for static and dynamic segments and does localization on the static map. [9] proposes an approach to detect moving objects either by multi-view geometry or deep learning and only uses the features from static part of the map. [10], [11] assign a Hidden Markov Model to each cell of a grid map for occupancy and state transition probability estimation. [12] assume the observed dynamics is caused by hidden processes which are periodic and identified by the Fourier Transform. These approaches are designed for either modeling or removing more frequent dynamics while we are focused on semi-static environments.

Some approaches evaluate the persistence of features in order to retain the long term features in the map. [13] defines the concept of short term and long term memories. The features are transferred from STM to LTM after multiple re-observations and used for global localization. [1] and [14] assign a persistence filter to each feature in order to calculate the persistence probability at any time step using recursive Bayes estimation and based on survival analysis. The features with a low persistence probability are discarded from map. However, these works concentrate on the design of the persistence filter, and do experiments using simulated data or assuming perfect knowledge of data associations. In [6], authors propose a static weighting method for edge points in order to estimate their probability of belonging to static objects.

Furthermore, scoring features and landmarks based on different criteria is another technique for dynamic environment settings. [15] evaluates the utility of features employing the salience, probability of re-observation, and relevance for localization. [16] uses a scoring function based on number of observations or covariance of landmark position and a sampling method for sparsification of the map. [17] analyzes different parameters based on observability and distribution of landmarks for calculating relevance score. In [18], scoring is done based on the times that a landmark has been observed. Unlike our approach, they require an offline map maintenance and summarization phase.

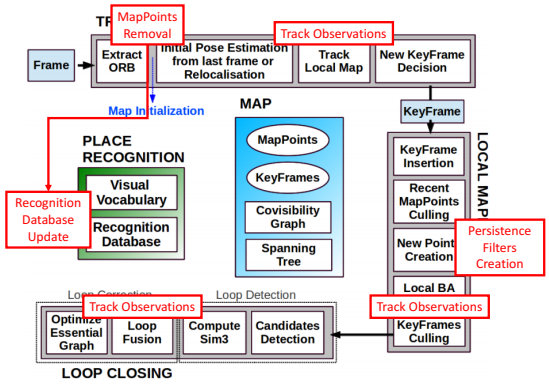

Fig. 1: ORB-SLAM [5] with changes for persistence (red).

Learning unique and practical features is another way of dealing with environmental changes. [19] employ a CNN classifier based on image patches and depth map for selecting map features that are persistent over time. In [20], they learn regressors for detecting reliable keypoints which are resistant to weather and lightening conditions. These approaches employ different criteria for selection of the best landmarks in dynamic settings. They put the localization and mapping output to use for utility estimation of landmarks. However, as far as we can tell, none of them employ the estimated scores to optimize the localization and mapping in any way other than map sparsification.

III. Persistence in Visual SLAM

A. Background on ORB-SLAM and Persistence

1) ORB-SLAM: ORB-SLAM [5] is a state-of-the-art graph-based visual SLAM algorithm where nodes correspond to RGB-D frames and edges correspond to 3D visual transformations. ORB-SLAM labels frames as keyframes if it has a low number of common map points with the previous keyframes. The algorithm runs as three separate modules - Tracking, Local Mapping and Loop Closing. Tracking module provides the first estimation of camera's position using available map points in the current frame. Local Mapping and Loop Closing use keyframes to refine localization and to detect loop closures. Also, there is a Keyframe Culling function within Local Mapping which removes keyframes with too few unique observable map points. Figure 1 shows the block diagram of ORB-SLAM along with our changes for incorporating persistence in red.

2) Persistence Filter: [1] proposes an on-line approach to estimate the persistence probability of features within the map and models their temporal evolution of environment. They employ survival analysis and provide a probabilistic generative model for persistence estimation which admits to a recursive Bayesian behavior. Their closed form solution for evaluating the posterior persistence of each individual feature at each time step is as follows:

$$p(X_t = 1|\zeta_{1:N}) = \frac{p(\zeta_{1:N}|T \geq t)p(T \geq t)}{p(\zeta_{1:N})} \tag{1}$$

where X_t represents the current availability state of the feature, $\zeta_{1:N} \subseteq \{0,1\}^N$ are its corresponding noisy observations (true/false) over time and T is the survival time. This work presents a couple of survival time priors based on the knowledge of the feature's activity pattern. Upon having no such information, general purpose priors are introduced which encode maximum entropy through exponential distribution.

$$\begin{aligned} p_T(t) &= \int_0^\infty p_T(t;\lambda).p_\lambda(\lambda).d\lambda. \\ p_T(t;\lambda) &= \lambda f(\lambda t) \\ f(x) &= e^{-x} \end{aligned} \tag{2}$$

where P_T is the probability density function for a prior distribution over the survival time T. Interested readers are referred to [1] for more details.

B. ORB-SLAM for Semi-Static Environments

1) Persistence in ORB-SLAM: We provide a practical implementation of ORB-SLAM for long term operation in semi-static environments. For this, we use a persistence filter to evaluate the persistence of map points and remove the ones that are no longer visible. We used implementation of persistence estimation from [1] with general purpose survival prior [21] as there is no information about the activity pattern of features. In order to accurately incorporate persistence filter into ORB-SLAM, we have to track observations of map points and propagate the impact of map point removals throughout the algorithm.

Persistence Filter Creation: Whenever a map point is created, a persistence filter is initiated and assigned to it. Upon arrival of every new frame, the persistence probability of all the map points is reviewed for current time stamp and any map point with a probability under a certain threshold is removed. In order to update the persistence filters, we have to track the observations of corresponding map points.

Track Observations: For tracking observations, we have to go through all modules of ORB-SLAM as shown in Fig. 1. Whenever a decision is made about accepting or rejecting a candidate keypoint match for a map point in the current frame/keyframe, we update the correspondent persistence filter. As discussed, $\zeta_{1:N} \subseteq \{0,1\}^N$ in Eq. 1 represents the noisy observations of corresponding feature. One main challenge for using persistence filter in ORB-SLAM is deciding whether we should assign hit (1) or missed (0) observations to map points. In ORB-SLAM, upon receiving each new visual frame and at different modules, the map points in the spatial locality of current position which are expected to be in the field of view, are projected on the current frame. For each projected map point, a specific neighborhood radius is defined based on the scale invariance region of the point. The ORB keypoints in the respective neighborhood radius are inspected and the one with matching scale level and least descriptor distance is considered a potential match. The final decision about observed map points are made in further optimizations. Prior work on persistence suggests that any map point with no match in this process should be assigned a missed observation and hit otherwise. But from our empirical experimentation, *this approach wrongly removes many persistent map points and*

decreases the localization capability of ORB-SLAM. These details are shown in Section IV.

We observed that the reason for not finding a match for a persistent map point are threefold, 1) Unavailability of expected ORB keypoints in the current frame due to randomness of ORB features, distance between current position and map point position or occlusion, 2) Unexpected higher descriptor distance due to sensitivity of feature descriptors to distance and orientation and 3) Unexpected difference in scale level of corresponding keypoints.

Map Points Removal: If we decide to remove a map point due to low persistence probability, we have to update other required elements relating to that map point. Each map point has an analogous keypoint in the keyframes which observe it and each keypoint contributes to a visual word. Visual words are used by loop detection thread through a visual word dictionary (Recognition Database in Fig. 1). When we remove a map point, we set the corresponding keypoint to unavailable and update the Recognition Database accordingly to avoid picking wrong candidates for loop closure.

2) Persistence+Dist in ORB-SLAM: In order to resolve the issue of missed matches for persistent map points, we add the following criteria to identify missed observations. 1) Since each map point has a reference keyframe representing the agent's position when it was initially created, the distance and orientation difference between the position of reference keyframe and current position should be bounded. The intuition behind these constraints are the sensitivity of keypoint extraction and descriptor calculation to distance and orientation upon observation, 2) Since we have access to depth information of each keypoint, the average received depth value correspondent to pixel locality of projected map point should be in accordance to expected depth for discarding occlusions.

3) Persistent-SLAM: The persistence filter approach requires the current mapping and feature associations to construct the $\zeta_{1:N}$ vector and to estimate the persistence of map points. Our observation is that this need not be one-sided i.e., the mapping process helping the estimation of persistence. We envision techniques that allow the persistence estimation to help the mapping. If a map point is expected to be in the field of view and its persistence probability is high, we do a finer search to find a matching keypoint in the current frame. If there is any keypoint within a few pixels of projected map point with acceptable descriptor distance, we accept it as a match disregarding any scale level difference. It should be noted that we restrict our search to highly persistent map points to avoid potential mis-matches and extra computation overhead.

IV. Evaluation

In this section, we evaluate the incorporated changes into ORB-SLAM algorithm in different steps. We provide results for four different versions of ORB-SLAM:

Vanilla **:** ORB-SLAM algorithm with no change.

Persistence **:** Adding persistence filter defined in [1] to each map point and assigning observations to them solely based on ORB-SLAM feature matching decisions.

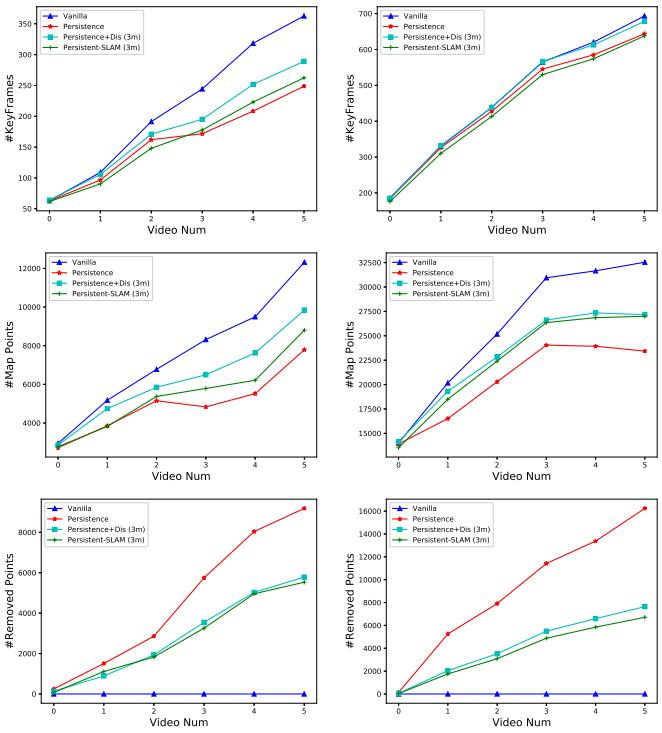

Fig. 2: Map related statistics for DronesLab (left column) and NetworksLab (right column) across days.

Persistence+Dis **:** Incorporating distance, orientation and occlusion constraints to persistence filtering to minimize false removal of map points.

Persistent-SLAM **:** Using persistence probabilities from *Persistence+Dis* to improve feature matching in ORB-SLAM.

A. Experimental Methodology

We collected two datasets to test the algorithms. As there is some randomness in extracted keypoints, detected feature matches and number of constructed keyframes, we run each algorithm three times and average our results. We vary the distance threshold in *Persistence+Dis* and *Persistent-SLAM* which specifies the admissible distance for assigning missed observations (Section III) with four values – 0.5, 1.0, 3.0 and 5.0 meters. We should notice that this parameter distance threshold has no impact on *Vanilla* and *Persistence* approaches.

B. Datasets

We used a Turtlebot with a Kinect 360 at 30 fps with RGB and depth images using ROS at two different locations each day for six days.

DronesLab: This dataset is collected from a lab environment with an occupancy of about fifteen people. The space is a typical office space with cubicles with fixed desks, cabinets and shelves around the lab, and about twenty chairs that move over the course of a day. Sample images are shown in Figure 3(left).

NetworksLab: This dataset is collected from a larger office with two long aisles. There are many static objects scattered uniformly in this space like fixed boards, cubicles with desks, cabinets. There are also a couple of chairs/boxes that move

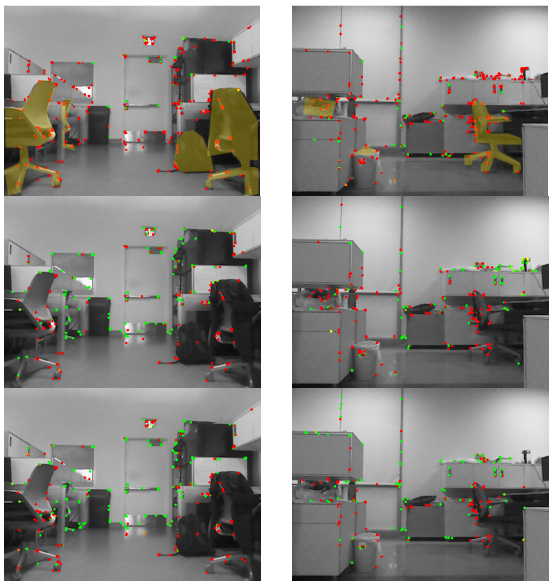

Fig. 3: Available(green) and removed(red) map points of two example keyframes of DronesLab (left) and NetworksLab (right) for three approaches; *Persistence* (top), *Persistence+Dis* (middle), *Persistent-SLAM* (bottom)

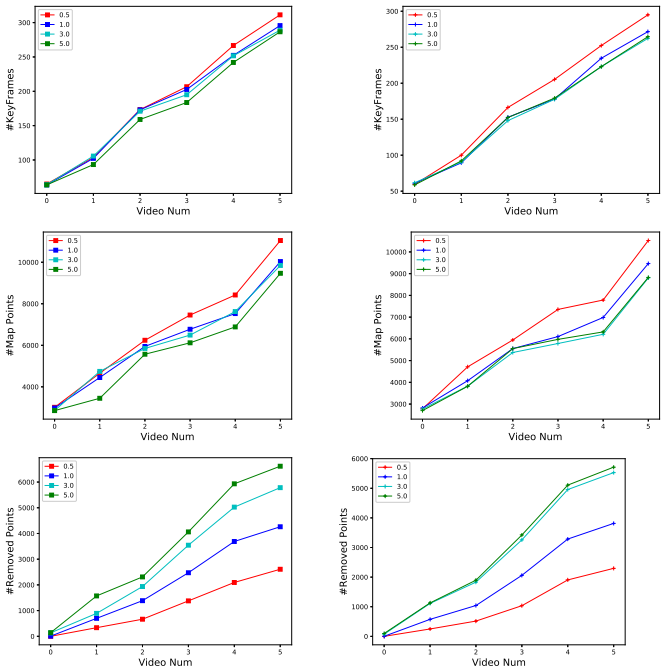

Fig. 4: Map-related statistics in *Persistence+Dis* (left) and *Persistent-SLAM* (right) with different distance thresholds for DronesLab dataset over daily videos.

around over the course of a day. This space represents a less dynamic setting than DronesLab. Sample images are shown in Figure 3(right).

C. Experiments

1) ***SLAM Experiment****:* In this experiment, we run each ORB-SLAM version over all videos for both datasets in order and measure the number of keyframes and map points. Unless

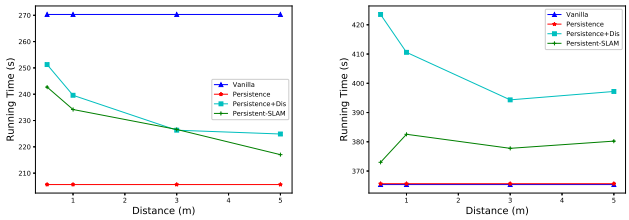

Fig. 5: Runtime for DronesLab (left) and NetworksLab (right)

otherwise noted, most discussions are regarding map size after running each algorithm through all datasets from Figure 2.

From Figure 2, the *Vanilla* algorithm keeps the most number of keyframes (top row) and most number of map points in the map (second row). These are direct consequences of not removing map points. The figure also shows the increasing number of removed map points through time in *Persistence* , *Persistence+Dis* , and *Persistent-SLAM* algorithms, and a corresponding decrease in the number of keyframes compared to *Vanilla* approach. Overall, we see a decrease of about 30% in number of map points and keyframes for the DronesLab dataset, and about 20% in number of map points and 10% in keyframes for the NetworksLab dataset between *Vanilla* and *Persistent-SLAM* algorithms. We will make several, more detailed observations that explain these results.

Our first observation is that directly incorporating a persistence filter into a visual SLAM algorithm (*Persistence*), is inadequate. While it removes a large number of map points during repeated runs (Figure 2(bottom row)), it also removes map points that are actually persistent. The same is observed to a lesser extent when we add the distance constraints (*Persistence+Dis*). This is visualized in Figure 3 where we highlight retained map points (green) and removed map points (red). We have highlighted the moved objects over the course of 6 days. As you can see, *Persistence* algorithm removes most of the map points including ones on the floor, on static objects like cabinets and white boards along with ones on the chairs. The *Persistent-SLAM* implementation (Figure 3(bottom)) retains most static map points and only removes the ones on moved objects such as chairs, bag, contents of trash can etc.

A second observation is that the reduction of total keyframes created between *Vanilla* and *Persistent-SLAM* algorithms is much smaller in the NetworksLab dataset in comparison with the DronesLab dataset. We believe that this is from the inherent dynamics observed in the environment. DronesLab, with larger occupation, saw more change in the environment through the six days in comparison to the NetworksLab. This observation should be intuitive as well - if there was no change in the scenes across multiple days, we should expect our persistence filter to observe that all map points are persistent, and not remove any of them. We believe that this observation demonstrates that our *Persistent-SLAM* , is able to correctly capture the persistent features and reflect the dynamics accurately in the final map.

Figure 4 shows the three metrics of interest for four settings of the distance parameter - .5m, 1m, 2m, and 5m on DronesLab. The left and right columns show results from

	Map Size (KB)				#Map Points				F1 Score			
Method	0.5	1	3	5	0.5	1	3	5	0.5	1	3	5
Vanilla	7063.3	7063.3	7063.3	7063.3	4873	4873	4873	4873	**0.95**	**0.95**	**0.95**	**0.95**
Persistence	6834.4	6834.4	6834.4	6834.4	**1430**	**1430**	**1430**	**1430**	0.74	0.74	0.74	0.74
Persistence+Dis	6748.7	6719.5	**6291.2**	6899.9	2416.6	2245.3	1720.6	2107.3	0.95	0.91	0.7	0.81
Persistent-SLAM	**6026.0**	**6212.9**	6295.8	**6021.2**	2397	1999.6	1732.6	1782.3	0.95	0.9	0.83	0.93

Fig. 6: Localization results for DronesLab dataset using the constructed map from first two videos

	Map Size (MB)				#Map Points				F1 Score			
Method	0.5	1	3	5	0.5	1	3	5	0.5	1	3	5
Vanilla	21.92	21.92	21.92	21.92	20197	20197	20197	20197	**0.95**	**0.95**	**0.95**	**0.95**
Persistence	21.67	21.67	21.67	21.67	**9319**	**9319**	**9319**	**9319**	0.75	0.75	0.75	0.75
Persistence+Dis	21.899	21.650	21.597	21.734	16486	13950	13256	12644	0.98	0.98	0.96	0.93
Persistent-SLAM	**20.21**	**20.79**	**20.38**	**20.69**	15810	14964	12110	12831	0.95	0.97	0.96	0.92

Fig. 7: Localization results for NetworksLab dataset using the constructed map from first two videos

the *Persistence+Dis* and *Persistent-SLAM* respectively. Since this distance threshold specifies the maximum admissible distance between current pose and the position of the reference keyframe where map point was created to determine if a map point was actually missed, larger distance thresholds seemingly cause assignment of more missed observations resulting in removal of a larger number of map points. Hence the number of available map points decrease with distance threshold. More map point removals increase keyframe removals which leads to lower number of keyframes in larger distance thresholds. Lower number of available map point and keyframes generates lower sized maps.*Persistent-SLAM* approach provides us with lower number of keyframes and map size compared to *Persistence+Dis* . This is due to the finer search for highly persistent map points which cause additional feature matching among visual frames. Better feature matching lowers creation of unnecessary keyframes and the map size decreases accordingly. The map statistics of NetworksLab for different distance thresholds follow the same pattern as DronesLab. Larger distance thresholds increase map point removals. But there is not much difference among the number of keyframes for different distance thresholds which is due to characteristics of the environment as stated before.

Figure 5 shows the cumulative running time of all ORB-SLAM threads for all approaches and different distance thresholds. For DronesLab, *Vanilla* approach owns the longest run time and *Persistence* approach has the largest decrease in processing time due to higher map point and keyframe removals which decrease the feature matching time. However for NetworksLab, since there are many stationary objects distributed uniformly over the place and less dynamics, fewer keyframe removals happen and the overhead of processing for updating required persistence filters keeps the processing time of all approaches on par or higher than *Vanilla* . The running time of *Persistent-SLAM* and *Persistence+Dis* usually decrease with larger distance thresholds through removing more map points and keyframes. *Persistent-SLAM* saves more running time than *Persistence+Dis* approach by creating less number of keyframes and spending less time in keyframe processing. Finally, we observe that the processing time of 5m threshold is higher than 3m for NetworksLab unexpectedly. In environments with little dynamics, more map point removals could increase the probability of removing persistent map points which may lead to the construction of additional keyframes in future visits. Our analysis shows that the number of total

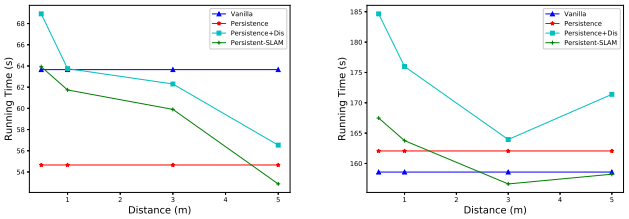

Fig. 8: Run time for localization for DronesLab (left) and NetworksLab(right)

constructed keyframes in *Persistence* approaches is higher than *Vanilla* which is not true for the DronesLab dataset.

2) ***Localization Experiment****:* In this experiment, we let each algorithm create a map of the environment using the first two collected videos. For the following four videos, we prohibit addition of any map point and only do localization as well as map point removal. This experiment is intended to highlight a benefit of persistence, which is to capture the relevant features but remove irrelevant ones. If our persistence setup is working well, the localization accuracy will remain high after several runs. We evaluate three metrics in this experiment - Map size, number of map points and F1 score which indicates the percentage of frames localized correctly.

Table 6 shows the localization and mapping results on DronesLab for all methods and distance thresholds. While the *Persistence* approach has the least number of map points, it has the least F1 score. This behavior is due to incorrect removal of many persistent map points resulting in several false negative localizations. However, *Persistent-SLAM* shows accuracy close to *Vanilla* with fewer number of map points (50% lesser for 0.5m distance threshold) and much higher than *Persistence* approach even for larger distance thresholds. Unlike our expectation of lower F1 score for higher distance thresholds due to larger number of map point removals, the F1 score of *Persistent-SLAM* approach for distance of 5m is relatively close to lower distance thresholds. One reason for this could be the inherent randomness in keyframe construction. This was the reason we ran each algorithm thrice and averaged the results, but its effects could still be persistent. This randomness may affect the map point removals, keyframe removals and feature matching among frames. On finer inspection, our results indicate that 86 frames have been successfully localized only once in the three runs for distance threshold of 5m. This number is 26 for distance of 0.5m. An interesting observation is that the *Persistence* approach has a larger map size than

Dataset	Method	RMSE of Translational Error (m)				Map Size (MB)			
		0.5	1	3	5	0.5	1	3	5
low dynamic fr3/sitting-halfsphere	Static Weighting [6]	0.039	0.039	0.039	0.039	-	-	-	-
	Vanilla	0.041	0.041	0.041	0.041	10.14	10.14	10.14	10.14
	Persistence	0.039	0.039	0.039	0.039	**5.0**	**5.0**	**5.0**	**5.0**
	Persistence+Dis	**0.02**	**0.019**	**0.019**	**0.019**	7.43	6.72	7.48	6.44
	Persistent-SLAM	0.034	0.024	0.021	0.021	7.58	6.92	7.08	7.05
high dynamic fr3/walking-halfsphere	Static Weighting [6]	0.053	0.053	0.053	0.053	-	-	-	-
	Vanilla	0.369	0.369	0.369	0.369	15.26	15.26	15.26	15.26
	Persistence	0.271	0.271	0.271	0.271	**6.7**	**6.7**	**6.7**	**6.7**
	Persistence+Dis	**0.27**	**0.177**	**0.242**	**0.266**	12.36	10.68	11.0	11.72
	Persistent-SLAM	0.281	0.221	0.38	**0.266**	12.2	12.1	13.43	12.92

Fig. 9: Localization results for TUM RGB-D dataset

Method	Localization Error(RMSE(m))				Map Size (MB)				# Unlocalized Frames			
	0.5	1	3	5	0.5	1	3	5	0.5	1	3	5
Vanilla	**0.11**	0.11	0.11	**0.11**	6.63	6.63	6.63	6.63	449	**449**	449	449
Persistence	0.14	0.14	0.14	0.14	**4.43**	**4.43**	**4.43**	**4.43**	638	638	638	638
Persistence+Dis	0.11	0.12	**0.1**	0.13	6.28	6.51	6.46	6.31	451.3	451	**441**	**445.6**
Persistent-SLAM	0.13	**0.11**	0.11	0.13	5.79	5.77	5.48	5.65	**446.6**	449.3	447	451.6

Fig. 10: Localization results for Kidnapped Robot dataset

Persistent-SLAM in this experiment. The reason is that no keyframe culling is allowed when the localization mode is activated. But since *Persistent-SLAM* creates less number of keyframes due to a finer search for highly persistent map points, it has a smaller map size.

Figure 8(a) shows the running time of all threads of ORB-SLAM for all approaches in this experiment on DronesLab. Although the *Persistence+Dis* is showing a comparable performance to *Persistent-SLAM* wrt F1 score, the map sizes and running time of *Persistent-SLAM* are much lower than *Persistence+Dis* approach. The reason is that the *Persistent-SLAM* creates lower number of keyframes which leads to lower computation time for keyframe processing and feature matching between current frame and spatially local keyframes. Similarly, larger distance thresholds result in larger number of map point removals which in turn result in less time spent in feature matching. Therefore, larger distance thresholds result in lower running times. Figure 8(b) shows the accumulated running time for all approaches on NetworksLab. The *Vanilla* version seems to have the least running time unlike DronesLab dataset where the *Persistence* approach had the lowest running time. We again attribute this to the limited dynamics in this dataset. Having same number of keyframes as *Vanilla* approach makes the processing time of other approaches comparable to it with additional overheads to maintain persistence. Table 7 shows the localization results for all approaches for NetworksLab over different distance thresholds. The results almost follow the same pattern as DronesLab. Even with a distance threshold of 5m, *Persistence+Dis* and *Persistent-SLAM* approaches reach an F1 score of more than 0.91 which is better than the *Persistence* approach and they only retain about 65% of map points compared to *Vanilla* version.

3) ***Comparison to Static Point Weighting [6]:*** [6] employs static weighting to indicate the probability of being part of the static environment and update their weights across time. To the best of our knowledge, this is the most recent approach applicable to semi-static environments which also provide their results on public datasets. We run our approach on two public dynamic sequences of TUM RGB-D dataset [22] and measure the tracking error of both methods.

Table 9 shows the localization error and map size of our proposed method and [6]. We tested two different kinds of sequences, with low and high dynamics. In low dynamics, *Persistence+Dis* and *Persistent-SLAM* experience the lowest error most of the times which shows their higher capability for keeping static features in the map. The map size of *Persistence+Dis* and *Persistent-SLAM* are in the same order for this dataset due to low dynamics. For high dynamics, [6] has lower error. The reason is that our proposed method is well fitted for semi-static environments, but this sequence includes rapid movements of people. In this sequence, *Persistence* approach shows comparable results to *Persistence+Dis* and *Persistent-SLAM* which stems from removing too many features and having low probability of wrong feature matching. Also, due to higher rate of dynamics, there is more fluctuation in the localization accuracy of *Persistence+Dis* and *Persistent-SLAM* over different distance thresholds.

4) ***Kidnapped Robot:*** This section shows the performance of our different approaches for kidnapped robot problem. For this purpose, we used the *360-kidnap sequence of TUM RGB-D dataset. This dataset doesn't include any dynamics.* Table 10 shows the statistics of all approaches for kidnapped robot problem. *Persistence* approach has the highest number of unlocalized frames which stems from too frequent wrong map point removal even in a static environment. The performance of *Persistence+Dis* and *Persistent-SLAM* is close to *Vanilla* version, but with lower sized map especially for *Persistent-SLAM* approach. The localization accuracy of *Persistence+Dis* and *Persistent-SLAM* is usually a little lower than *Vanilla* version due to removal of some map points.

V. Conclusion

We demonstrate the first implementation of persistence filters in Visual SLAM. First, we show that directly incorporating persistence filters is not sufficient. Through a series of modifications, we were able to customize persistence filtering for visual SLAM. Using two datasets, and through two experiments, we demonstrate the benefits of persistence filtering in visual SLAM by 30% map size reduction and run time improvements. Using publicly available datasets, we also compare our implementation with static weighting, and show that our implementation works better in some scenarios.

REFERENCES

[1] D. M. Rosen, J. Mason, and J. J. Leonard, "Towards lifelong feature-based mapping in semi-static environments," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1063–1070.

[2] *Discovery Robotics*, 2020 (accessed February 20, 2020). [Online]. Available: https://discoveryrobotics.com/

[3] *Double Robotics*, 2020 (accessed February 20, 2020). [Online]. Available: https://doublerobotics.com/

[4] *Savioke*, 2020 (accessed February 20, 2020). [Online]. Available: https://savioke.com/

[5] 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.

[6] S. Li and D. Lee, "Rgb-d slam in dynamic environments using static point weighting," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2263–2270, 2017.

[7] Z. Hashemifar, *Persistent ORB-SLAM2*, 2020 (accessed February 20, 2020). [Online]. Available: https://github.com/droneslab/persistent_orbslam

[8] D. F. Wolf and G. S. Sukhatme, "Mobile robot simultaneous localization and mapping in dynamic environments," *Autonomous Robots*, vol. 19, no. 1, pp. 53–65, 2005.

[9] 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.

[10] G. D. Tipaldi, D. Meyer-Delius, and W. Burgard, "Lifelong localization in changing environments," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1662–1678, 2013.

[11] D. Meyer-Delius, M. Beinhofer, and W. Burgard, "Occupancy grid models for robot mapping in changing environments," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[12] T. Krajník, J. P. Fentanes, J. M. Santos, and T. Duckett, "Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 964–977, 2017.

[13] F. Dayoub, G. Cielniak, and T. Duckett, "Long-term experiments with an adaptive spherical view representation for navigation in changing environments," *Robotics and Autonomous Systems*, vol. 59, no. 5, pp. 285–295, 2011.

[14] F. Nobre, C. Heckman, P. Ozog, R. W. Wolcott, and J. M. Walls, "Online probabilistic change detection in feature-based maps," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 1–9.

[15] S. Hochdorfer, H. Neumann, and C. Schlegel, "Landmark Rating and Selection for SLAM in Dynamic Environments," in *Intelligent Autonomous Systems 13*. Springer, 2016, pp. 401–414.

[16] M. Dymczyk, S. Lynen, T. Cieslewski, M. Bosse, R. Siegwart, and P. Furgale, "The gist of maps - summarizing experience for lifelong localization," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 2767–2773.

[17] J. S. Berrio, J. Ward, S. Worrall, and E. Nebot, "Identifying robust landmarks in feature-based maps," pp. 1166–1172, 2019.

[18] P. Mühlfellner, M. Bürki, M. Bosse, W. Derendarz, R. Philippsen, and P. Furgale, "Summary maps for lifelong visual localization," *Journal of Field Robotics*, vol. 33, no. 5, pp. 561–590, 2016.

[19] M. Dymczyk, E. Stumm, J. Nieto, R. Siegwart, and I. Gilitschenski, "Will it last? learning stable features for long-term visual localization," in *2016 Fourth International Conference on 3D Vision (3DV)*, Oct 2016, pp. 572–581.

[20] Y. Verdie, K. Yi, P. Fua, and V. Lepetit, "Tilde: a temporally invariant learned detector," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5279–5288.

[21] *David Rosen*, 2020 (accessed February 20, 2020). [Online]. Available: https://github.com/david-m-rosen/persistence_filter

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