Fast-SeqSLAM: A Fast Appearance Based Place Recognition Algorithm

Sayem Mohammad Siam and Hong Zhang

Abstract— **Loop closure detection or place recognition is a fundamental problem in robot simultaneous localization and mapping (SLAM). SeqSLAM is considered to be one of the most successful algorithms for loop closure detection as it has been demonstrated to be able to handle significant environmental condition changes including those due to illumination, weather, and time of the day. However, SeqSLAM relies heavily on exhaustive sequence matching, a computationally expensive process that prevents the algorithm from being used in dealing with large maps. In this paper, we propose Fast-SeqSLAM, an efficient version of SeqSLAM. Fast-SeqSLAM has a much reduced time complexity without degrading the accuracy, and this is achieved by using an approximate nearest neighbor (ANN) algorithm to match the current image with those in the robot map and extending the idea of SeqSLAM to greedily search a sequence of images that best match with the current sequence. We demonstrate the effectiveness of our Fast-SeqSLAM algorithm in appearance based loop closure detection.**

I. INTRODUCTION

The computation task of building or updating a map and simultaneously localizing the robot in the map while a robot is exploring an unknown environment is commonly referred to as simultaneous localization and mapping (SLAM). In a SLAM system, robots need to recognize previously visited locations, i.e., detect loop closures. Correctly recognizing previously visited locations, a SLAM algorithm is able to optimize the pose constraints between the nodes, overcome incremental pose drift [1], [2] and build a correct map.

For long term and large scale SLAM operation, it is important to recognize locations in changing environments over the course of days, nights and seasonal changes. FAB-MAP 2.0 [3], a successful place recognition algorithm based on single image matching and Bayes filtering, demonstrated its operation over a route of 1000 km. However, a single location may look completely different at different times due to illumination and weather changes. As a result, it becomes difficult to match two images of the same location, and algorithms like FAB-MAP which are based on single image matching, have to resort to filtering techniques or will fail quite easily [4].

Milford and Wyeth [4] proposed a sequence based place recognition algorithm, known as SeqSLAM, which finds a sequence of images in the robot map, rather than a single image, that are the best match for the current image sequence being captured by a moving robot. SeqSLAM has demonstrated better performance in recognizing places that underwent severe appearance changes than the other successful SLAM algorithms like FAB-MAP 2.0 [5]. Most recently, Milford and Shen [5] have demonstrated significant improvement of the viewpoint invariance of SeqSLAM by generating synthetic viewpoints using the state-of-the-art deep learning techniques [6]. However, SeqSLAM, which is based on exhaustive sequence search, is computationally costly when the number of images or nodes in the map is large. This is unfortunate as a robot needs to perform all the computation for localization including loop closure detection online in a SLAM system.

The key motivation for our work is to develop a place recognition algorithm for a large scale and long term operation, based on sequence matching like SeqSLAM but computationally efficient. To achieve a high computational efficiency, we first store the image descriptors of a map in a tree structure. Then we use an approximate nearest neighbor (ANN) algorithm [7] to search for N best nearest neighbor images for a current view where N is constant and much smaller than the number of images in the map, n. In addition, rather than computing the distance between all possible image pairs as in SeqSLAM, for each image, we only compute its distance to its N nearest neighbors and store the information in a sparse difference matrix. In addition, we modify the sequence matching algorithm in SeqSLAM so that we can still find the best match for a current sequence in a sparse difference matrix. Specifically, in our modified sequence matching algorithm, we do not search exhaustively as in the original SeqSLAM algorithm; rather we greedily search only from those locations in the map where the best K matches for the current view of the robot are found, among the N total matches, where K is smaller than N. Velocity information of the robot is then used to determine where to search next in the map. Due to the efficiency in the construction of the difference matrix and the greedy sequence search algorithm, the computational complexity of our algorithm is $O(n \log n)$ for finding all loop closure nodes between any two traversals of an environment while SeqSLAM has a computational complexity of $O(n^2)$. Most importantly, when the accuracy of our algorithm is compared with that of SeqSLAM experimentally the computational efficiency of our proposed algorithm does not come at the expense of accuracy.

The rest of the paper is organized as follows. In Section II, we summarize relevant prior research in place recognition. Section III describes our algorithm, Fast-SeqSLAM for loop-closure detection. Our experimental design and results are presented in Section IV focusing on computational complex-

*This work was supported by the Natural Sciences and Engineering Research Council (NSERC) through the NSERC Canadian Field Robotics Network (NCFRN)

Sayem Mohammad Siam and Hong Zhang are with the Department of Computing Science, University of Alberta, Edmonton, AB, Canada siam,hzhang@ualberta.ca

ity and the accuracy of our algorithm. We conclude the work and discuss future research in Section V.

II. Related Work

Condition invariance and computational efficiency are two key prerequisites for large scale and long term SLAM operations. Our work is motivated by a desire to build a place recognition algorithm which is suitable for a long-term and large-scale SLAM system.

Bag-of-Words, a popular technique initially developed for efficient text retrieval and later extended to image matching [8], has been extensively used for place recognition [9], [10], [11]. In a typical place recognition system that uses the BoW approach, scale-invariant local image descriptors, such as SIFT [12] and SURF [13] keypoints, are extracted from the images and these keypoints are vector-quantized to serve as words in the BoW technique. A very successful algorithm, FAB-MAP 2.0 [3], which has demonstrated in its operation for place recognition along a route more than 1000 km in length, employs the Bag-of-Words technique. In this algorithm, only the current view of a single image is used to vote for map images, and a Bayes filter is used to enforce spatial and temporal consistency.

Network flows [14] has been used as an alternative to SeqSLAM's sequence searching. A directed acyclic graph (DAG) is created to store the similarity between pairs of images and formulate image matching is solved as a minimum cost flow problem while incorporating sequence information. Place recognition is equivalent to finding a sequence which has the minimum flow cost. However, the time complexity for this technique remains $O(n^2)$.

Hidden Markov Model (HMM) with Viterbi algorithm [15] has been used for finding the best matching sequence for a given sequence of images [16], inspired by Dynamic Time Warping (DTW) from speech recognition [17]. The state transition matrix is built using local velocity constraints of a robot. However, they need to calculate a complete similarity matrix for all the possible image pairs and hence the time complexity of this algorithm is $O(n^2)$.

To reduce the time complexity of SeqSLAM, a particle filter technique has been proposed by Liu and Zhang [18]. Rather than computing the matching scores for all candidate sequences in the map, a subset of these sequences are sampled and evaluated. The promising ones with high probabilities are kept for further evaluation, the unlikely ones are dropped, and new candidates are introduced in the next round of evaluation. Although a particle filter can alleviate to some extent the computational cost, convergence to the correct solution is probabilistic and the overall complexity of the algorithm is still $O(n^2)$.

III. Fast-SeqSLAM

The general setup of SeqSLAM, which we adopt, assumes two traversals of the same environment or route. We assume that the map is built in the first traversal from n images and the second traversal contains m images among which loop closures exist. The goal is to determine if any sequences in the second traversal overlap with any in the first. Both the length and the initial location of a possible matching sequence are unknown.

The key idea in our Fast-SeqSLAM algorithm is that we should avoid searching all the sequences in the map exhaustively for the best matching sequence to the current sequence; instead, we can search greedily among sequences defined by the most likely initial matching images and use motion continuity to extend and continue the search. This initial image set is small, and its size is constant (N) and does not grow with the size of the map.

Further, to identify promising initial locations of matching sequences, rather than linearly comparing with all map images, we use an approximate nearest neighbor algorithm to find the N best nearest neighbor images in the map with respect to the current robot view, and we create a sparse difference matrix with similarity scores only for these nearest neighbors. Then, we greedily search for the best matching sequence only from the K best matching locations among N locations. Our algorithm is approximate and its performance depends on the difference matrix. The approximation of the difference matrix depends on the value of N. On the other hand, choice of K depends on the reliability of the matching scores in the difference matrix and usually much smaller than N. Moreover, the searching is computationally more expensive than computing the difference matrix. Selection of K limits the searching.

We call each value in the difference matrix as *difference value*, and the summation of all the difference values of a trajectory sequence as *difference score*, using the notations in SeqSLAM [4]. Fig. 1 shows an example of a difference matrix where $n = m = 13$, and $N = 2$. The brighter a cell is, the larger the difference value. Sequence matching uses this sparse difference matrix, and proceeds as described below where the explanations refer to lines in Algorithm 1, the pseudocode of our sequence matching algorithm.

1) Initialize sparse difference matrix and score matrix with a high value (Lines 2-3).
2) Build a tree using the images of our map (Line 5).
3) Find N best nearest neighbor locations for the current view of the robot (Line 8).
4) Update the sparse difference matrix with difference values returned by ANN function for each of the N locations (Line 9).
5) For each probable loop closure locations, calculate the minimum difference score and update the robot's current location (Lines 14-22).
6) Finally, find the best matching location for the current robot view, and normalize the matching value (Lines 23-26).

In the following subsections, we describe three main components our algorithm in detail which are finding approximate nearest neighbors (Line 8), building a sparse difference matrix (Line 9) and sequence matching (Lines 13-21).

Algorithm 1: Algorithm for Finding Matches

Input:
Q = Images in the robot views
M = Images in the map
Output:
matches = A vector containing matching indices and matching values

1 **Procedure** `findMatches(Q, M)`
2 **global** D = initialize(len(M), len(Q))
3 **global** S = infinity(size(D)) ▷ matrix for score values
4 matches = []
5 T = Tree(M)
6 **foreach** *Image I_i in Q* **do**
7 g_i = descriptor(I_i)
8 N-matches, distances = ANN(T, g_i, precision)
9 D = updateDifferenceMatrix(D, i, N-matches, distances)
10 K-locations = N-matches(1:K)
11 probable-loop-closures = K-locations ∪ locations
12 locations = []
13 **foreach** *location j in probable_loop_closures* **do**
14 min_score = $\min_{v_{min},\ldots v_{max}} score(j,T,V)$
15 $S_{j,T}$ = min_score
16 estimated_velocity = $\arg\min_{v_{min},\ldots v_{max}} score(j,T,V)$
17 updated-location = j+estimated_velocity
18 **if** *min_score* $<$ *max_score* **then**
19 locations.add(updated-location)
20 **end**
21 **end**
22 value = $\min_{j=1,2\cdots m}(\hat{S}_j^T)$
23 value = normalize(value)
24 index = $\arg\min_{j=1,2\cdots m}(\hat{S}_j^T)$
25 matches(T) = (index,value)
26 **end**
27 **return** matches
28 **Procedure** `score(j, T, V)`
29 $S = \sum_{t=T-d_s}^{T} D_k^t$ where, $k = j + V(t-T)$
30 **return** S

A. Approximate Nearest Neighbor

For finding the approximate nearest neighbor (ANN) for an image, we use FLANN by Muja and Lowe [7]. We control the accuracy of the result by setting its precision parameter. FLANN selects the ANN parameters automatically for a given dataset and a target accuracy. It uses either randomized kd-trees [19] or hierarchical k-means tree [20] algorithm. One of the factors that has a great impact on the nearest neighbor matching is data dimensionality. Muja and Lowe demonstrated speedup by a factor of 10^3 for data with dimensions from 10^1 to 10^3 and a target accuracy of 80%. In our study, the dimensions of the image descriptors are under 1024.

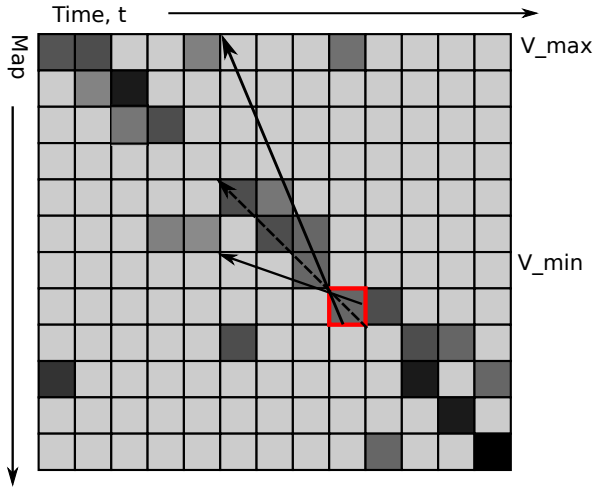


Fig. 1: The figure shows a sparse difference matrix where $N = 2$. The brighter a cell is, the larger the difference value. Trajectory line with maximum velocity and trajectory line with minimum velocity are shown. We calculate 2 nearest neighbors for each node (only two cells in each column are darker than others). The dotted line is the trajectory which has minimum difference score for the red bordered cell.

In one implementation of our algorithm, we first down-sample each image to 32×32 and then extract features using the HOG [21] descriptor, which has 334 dimensions. Alternatively, we can also use raw images as their descriptors as in SeqSLAM so that a descriptor has 1024 dimensions. HOG has a smaller dimension size and therefore less ANN search time, than the raw image descriptor.

B. Sparse Difference Matrix

We initialize the difference matrix to large values (bright cells in Fig. 1). We use FLANN to create a tree structure to store the map image descriptors to facilitate efficient search. For a map or traversal of n images, each image can find its approximate nearest neighbors in the map in $O(\log(n))$ time, vs. $O(n)$ in the case of linear search. The distance values returned by the ANN algorithm are used to replace the initial values whenever they are computed (where N=2 for the example in Fig. 1). Consequently, our difference matrix is sparse containing distances to (approximate) N nearest neighbors along each column (row).

C. Sequence Matching

We calculate minimum difference score only from the locations in the map where the best K matches for the current view of the robot are found while we have a total N matching locations (Algorithm 1 Line 10). N determines how approximate our difference matrix is. Then we update the robot's current location using robot's current velocity. We update robot's location so that the loop-closure node between

two sequences are temporally continuous. In following, we describe how we calculate difference score and greedily update robot's location.

1) Difference Score: Each column of our difference matrix **D** is a difference vector. Difference vector $\hat{\mathbf{D}}^{(T)}$, where T is the current time, contains N valid image difference values for an image I_{2j} with I_{1T} where $j = 1,2 \ldots m$. I_{2j} is an image in the map and I_{1T} is the current robot view. We search for a sequence of images in the map along the rows of **D** which best match with the sequence, $I_{1,T-d_s}$, $I_{1,T-d_s+1}$, $\cdots$ $I_{1,T}$, across the columns of the robot views where d_s is the length of sequence. To recognize the current sequence of robot views in the map, a search is performed through the space **M** of image difference vectors.

$$\mathbf{M} = \left[\hat{\mathbf{D}}^{\mathbf{T-d_s}}, \hat{\mathbf{D}}^{\mathbf{T-d_s+1}} \ldots \hat{\mathbf{D}}^{\mathbf{T}}\right] \tag{1}$$

In the example in Fig. 1, different trajectory lines that originate from the red bordered cell are drawn for different velocities. We calculate difference score S of a trajectory line by summing up the difference values the line passes through. Using Equations (2) and (3), we calculate difference score for a particular velocity V. In Equations (2) and (3), D^t_j is the difference value between the robot view t and map image j, V is the trajectory velocity and s is the image number in the map for which we are calculating the sequence score.

$$S = \sum_{t=T-d_s}^{T} D^t_j \tag{2}$$

$$j = s + V(t - T) \tag{3}$$

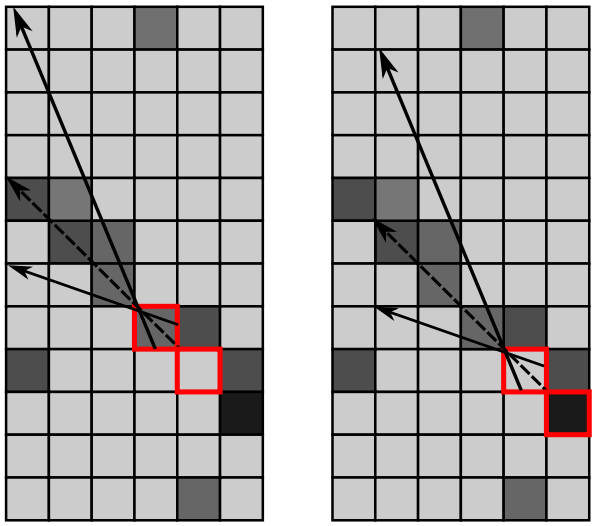
In Algorithm 1, we use the procedure $score(j,T,V)$ in Lines 32-34 to encapsulate the calculation by the above equations.

2) Greedy Motion Model Estimation: To find continuous or temporally consistent loop-closure nodes efficiently, we use a greedy motion model estimation technique. For a current sequence of robot views, we find the best sequence of images in the map. For each probable loop closure node, we calculate a difference score for different trajectories where each trajectory line corresponds to a velocity or motion model. If a node is actually a loop closure node, the robot should move along the trajectory defined by the velocity from previous nodes in the sequence. In other words, the best fit trajectory is the one that has the minimum difference score. We can update the robot's current location using the motion model or velocity that corresponds to the best-fit trajectory.

We only update the robot's current location when the best-fit trajectory line has at least one nearest neighbor; the trajectory is otherwise abandoned. In Fig. 2a, the dotted trajectory line has the min-difference-score, and we use that velocity to update robot's current location. Fig. 2b shows the current updated location at the red bordered cell.

IV. Experimental Results

In this section, we demonstrate the effectiveness of our algorithm.



(a) (b)

Fig. 2: In Fig. 2a, the dotted black trajectory has the minimum difference-score for the red bordered cell. Red bordered cells are current and next locations of the robot. We follow the min-difference score trajectory line to update robot's current location (next red bordered cell). In Fig. 2b, we show the updated location.

A. Datasets

In this section, we run our algorithm for loop-closure detection on three different datasets used commonly in previous studies of visual loop closure detection in SLAM, each of which consisted of at least two traversals of the same route but at different times and, as a result, all datasets involve severe appearance changes. The datasets are as follows:

- Nordland Dataset, train ride in northern Norway,
- UA dataset, from University of Alberta, Edmonton, Canada, and
- Garden Points Walking, from Queensland University of Technology, Brisbane, Australia.

The Nordland dataset [22] contains four traversals of the same route in winter, spring, summer and fall. It is a 728 km long train ride in northern Norway and has a total 37500 images. The UA dataset was collected by a Husky robot with a normal RGB camera and the Garden Points Walking dataset is collected by Arren Glover and has two traversals, one during day and the other at night. All these datasets have ground truth information to allow easy performance evaluation. We compared our algorithm with OpenSeqSLAM [22], an implementation of SeqSLAM, in terms of metrics for accuracy and efficiency as described in detail below.

TABLE I: Parameters

Parameter	Description and Values
R_x, R_y	Reduced image size for all the datasets, we used $[R_x, R_y]$ = [32,32] in all the datasets
Cell size	For HOG, we used Cell size = [8,8]
d_s	Trajectory length in numbers, we used d_s = 20 in Nordland dataset, d_s = 30 in UA and d_s = 20 in Garden points
N	Number of nearest neighbor matches we calculate for current view, we used N = 100 in Nordland dataset, N = 10 in UA and N = 10 in Garden walking dataset
K	Number of best matching locations among N locations where we search for the best matching sequence, K = 5 in Nordland dataset, K = 2 in UA and K = 2 in Garden points
V_{min}	Minimum trajectory speed, V_{min} = $.4V_{ave}$
V_{max}	Maximum trajectory speed V_{max} = $1.5V_{ave}$

B. Precision-Recall

In order to measure the accuracy of the different algorithms for loop closure detection, we generate precision and recall values while varying the threshold on similarity between sequences to determine if a loop closure has occurred. We consider a match as positive if it is within a certain offset distance from the ground truth location.

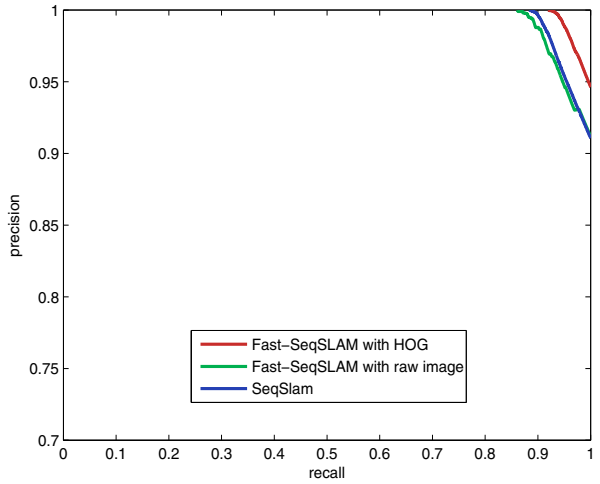


Fig. 3: Nordland dataset.

We compared our algorithm with SeqSLAM and we plotted the precision-recall curve. As mentioned, we used HOG in our algorithm in order to reduce the dimensionality of the image descriptor and allow efficient and accurate ANN search. However, even if we use raw image as a descriptor as in SeqSLAM, our Fast SeqSLAM is still able to achieve good performance. Therefore, in our experiments, the following algorithms were included in the comparison, and the results are summarized in Figs. 3-5.

- Fast-SeqSLAM with HOG (red).
- SeqSLAM (blue).
- Fast-SeqSLAM with raw image as descriptor (green).

Fig. 3 shows the comparative results for Nordland dataset. All algorithms have almost the same performance. For the Garden Points dataset, in Fig. 4, our algorithm with HOG descriptor performs better than SeqSLAM. This result is somewhat surprising as our algorithm is a greedy version of SeqSLAM. We observed that this result is due to the fact that the images in the dataset are captured closely with each other and, as a result, the contrast enhancement step on the difference matrix in SeqSLAM would degrade its performance, as we verified experimentally. Contrast enhancement on the difference matrix did not help SeqSLAM in this case since each image matches with nearby images because of their high similarity. Using the dataset collected on the University of Alberta Campus at two different times of a day with a Husky robot, all three algorithms have a similar performance as shown in Fig. 5, with SeqSLAM performing better for low recall values and Fast SeqSLAM better for high recall values. We have given the parameter values we used in our experiments in Table I.

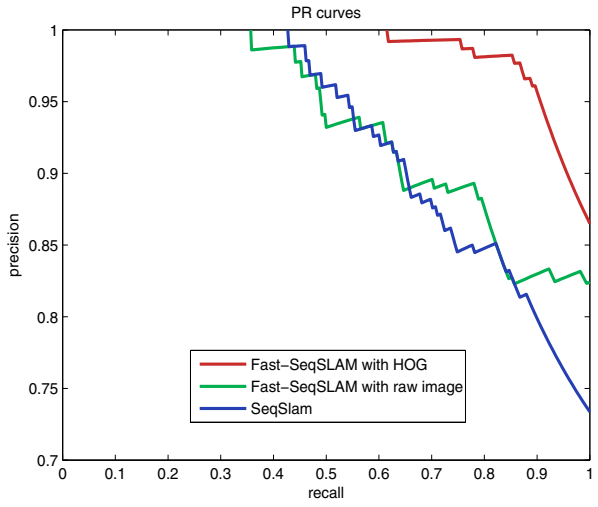


Fig. 4: Garden Points dataset.

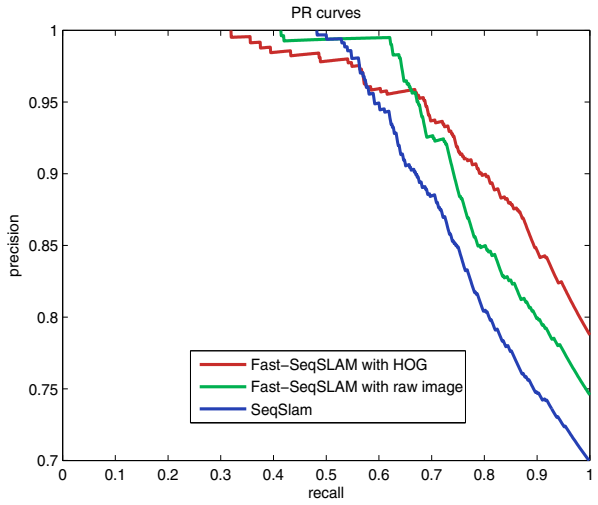


Fig. 5: UA dataset.

C. Computational Complexity and Execution Time

In our algorithm, we use a tree structure to store the image descriptors of a map, and tree construction takes $O(n \log n)$ time. Calculation of the nearest neighbors for the robot's current view has a time complexity of $O(\log n)$. If we use a single nearest neighbor for each image, we will have m total values in the matrix where m is the number of columns in the difference matrix. We search sequences greedily and we continue searching until the matching sequence has at least one nearest neighbor. In the worst case, we may calculate a difference score for d_s extra cells in the difference matrix. So we have a time complexity of $O(md_s)$ for finding all the loop closure nodes between the two traversals. Since d_s, the trajectory length, is a constant, we can write $O(m)$ instead of $O(md_s)$. The overall time complexity becomes $O(n \log n + m)$ for finding all loop closure nodes, for a map with n images and a traversal of m images. If we assume $n \approx m$ in general, then the overall complexity of Fast SeqSLAM is $O(n \log n)$.

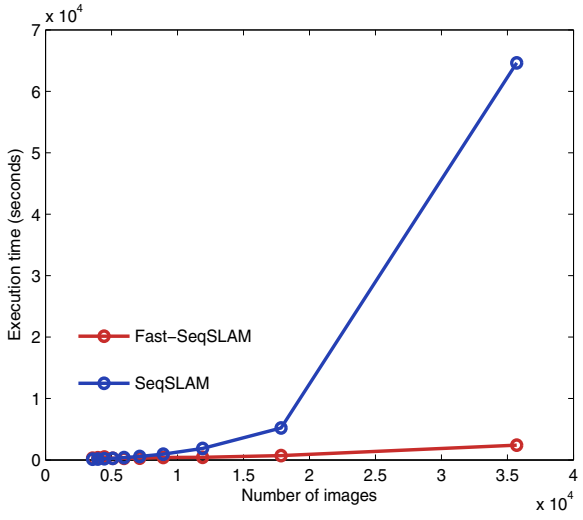


Fig. 6: Nordland dataset, execution time vs number of images.

Fig. 6 demonstrates the comparison of execution time of our algorithm with SeqSLAM on one dataset using different number of images of the dataset. The red curve is for our Fast-SeqSLAM algorithm and the blue curve is for SeqSLAM. The y-axis shows the execution time in seconds. For a small map, for example 5000 images, execution time does not differ much. However, for a large map, for example 37000 images, our algorithm runs in 40 minutes whereas SeqSLAM takes 18 hours, i.e., Fast-SeqSLAM improves SeqSLAM by a factor of 27 in execution time in this case. Recall that each dataset contains two traversals of the same route, and the above times are what is required to find all corresponding sequences or loop closures between the two traversals.

V. CONCLUSIONS

In this paper, we have proposed an efficient version of SeqSLAM, which we refer to as Fast SeqSLAM. Our proposed algorithm is able to retain the key advantages of SeqSLAM in terms of place recognition accuracy, and at the same time, much more efficient than SeqSLAM, speeding it up as much as by a factor of 27, on a map with 37K keyframes. Our algorithm is greedy, and its key ideas that allow us to achieve this efficient are (a) we avoid computing a complete difference matrix between the map images and those observed by the robot, and (b) possible matching sequences in the map are searched strategically and selectively rather than exhaustively. In fact, it is possible to develop an online version of SeqSLAM based on these two key ideas. We are leaving this as our future work.

The implementation of Fast SeqSLAM is available at: https://github.com/siam1251/Fast-SeqSLAM. It is written in Matlab.

REFERENCES

[1] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3607–3613.

[2] N. Sünderhauf and P. Protzel, "Switchable constraints for robust pose graph slam," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 1879–1884.

[3] M. Cummins and P. Newman, "Appearance-only slam at large scale with fab-map 2.0," *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1100–1123, 2011.

[4] M. J. Milford and G. F. Wyeth, "Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1643–1649.

[5] M. Milford, C. Shen, S. Lowry, N. Suenderhauf, S. Shirazi, G. Lin, F. Liu, E. Pepperell, C. Lerma, B. Upcroft, *et al.*, "Sequence searching with deep-learnt depth for condition-and viewpoint-invariant route-based place recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 18–25.

[6] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5162–5170.

[7] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration." *VISAPP (1)*, vol. 2, no. 331-340, p. 2, 2009.

[8] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 1470–1477.

[9] D. Filliat, "A visual bag of words method for interactive qualitative localization and mapping," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 3921–3926.

[10] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer, "Fast and incremental method for loop-closure detection using bags of visual words," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1027–1037, 2008.

[11] M. Cummins and P. Newman, "Fab-map: Probabilistic localization and mapping in the space of appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[12] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[13] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[14] T. Naseer, L. Spinello, W. Burgard, and C. Stachniss, "Robust visual robot localization across seasons using network flows." in *AAAI*, 2014, pp. 2564–2570.

[15] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[16] P. Hansen and B. Browning, "Visual place recognition using hmm sequence matching," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 4549–4555.

[17] L. Rabiner and B.-H. Juang, "Fundamentals of speech recognition," 1993.

[18] Y. Liu and H. Zhang, "Towards improving the efficiency of sequence-based slam," in *2013 IEEE International Conference on Mechatronics and Automation*. IEEE, 2013, pp. 1261–1266.

[19] C. Silpa-Anan and R. Hartley, "Optimised kd-trees for fast image descriptor matching," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[20] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 2161–2168.

[21] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 886–893.

[22] N. Sünderhauf, P. Neubert, and P. Protzel, "Are we there yet? challenging seqslam on a 3000 km journey across all four seasons," in *Proc. of Workshop on Long-Term Autonomy, IEEE International Conference on Robotics and Automation (ICRA)*, 2013, p. 2013.