Online Loop-Closure Detection via Dynamic Sparse Representation

Moein Shakeri and Hong Zhang

Abstract Visual loop closure detection is an important problem in visual robot navigation. Successful solutions to visual loop closure detection are based on image matching between the current view and the map images. In order to obtain a solution that is scalable to large environments involving thousands or millions of images, the efficiency of a loop closure detection algorithm is critical. Recently people have proposed to apply l_1-minimization methods to visual loop closure detection in which the problem is cast as one of obtaining a sparse representation of the current view in terms of map images. The proposed solution, however, is insufficient with a time complexity worse than linear search. In this paper, we present a solution that overcomes the inefficiency by employing dynamic algorithms in l_1-minimization. Our solution exploits the sequential nature of the loop closure detection problem. As a result, our proposed algorithm is able to obtain a performance that is an order of magnitude more efficient than the existing l_1-minimization based solution. We evaluate our algorithm on publicly available visual SLAM datasets to establish its accuracy and efficiency.

1 Introduction

Autonomous mobile robots are beneficial to work in hazardous environments, or places out of range of human operators over long periods of time, such as exploration [4] and rescue [19]. In many environments robots have no prior

Moein Shakeri
Department of Computing Science, University of Alberta, Edmonton, Canada,
e-mail: shakeri@ualberta.ca

Hong Zhang
Department of Computing Science, University of Alberta, Edmonton, Canada
e-mail: hzhang@ualberta.ca

knowledge about their surroundings. Therefore, it is essential for a robot to be able to map an unknown environment itself in order to perform its tasks. Simultaneous Localization And Mapping (SLAM) has been a focus of robotics research and, among the many issues of concern, is the detection of loop closures, i.e., revisits to map locations.

In order to be able to handle a large environment, a loop closure detection algorithm must be efficient. The dominant approach in SLAM literature to meet this requirement is based on visual bag-of-words (BoW) that achieves efficiency through indexing. Visual BoW however requires offline construction of a visual vocabulary, which may not be representative of the environment that a robot will visit online, and the step of keypoint detection and vector quantization can be computationally costly.

An alternative to visual BoW for loop closure detection is compact whole image descriptors that avoid the step of keypoint detection and vector quantization [14]. In this case, loop closure detection is solved as a nearest neighbor search considering the descriptor of the current view as the search key. Recently, an interesting solution based on l_1-minimization has been proposed that solves this nearest neighbor search problem through sparse reconstruction. The proposed solution, although elegant, is less efficient than linear search to find the nearest neighbor, and offers little incentive for people to adopt.

In this paper, we improves the solution based on l_1-minimization by exploiting the sequential nature of the loop closure detection problem, i.e., successive robot views look similar so that l_1-minimization does not need to be solved from scratch. We make use of recent algorithms in dynamic algorithms for l_1-minimization to achieve a solution that is an order of magnitude more efficient than the static l_1-minimization, without sacrificing accuracy. Most importantly, our solution is more efficient than linear searsh and is therefore a competitive candidate in tackling the problem of visual loop closure detection with whole-image descriptors.

The remainder of this paper is organized as follows. In Section II, we discuss related works that address the loop closure detection problem. In Section III, we present a brief overview of sparse representation using l_1-minimization to solve the loop closure problem. Also dynamic update of the optimization problem to avoid solving l_1-minimization for each input image is described. In Section IV, we explain how the proposed dynamic sparse representation can be utilized in visual robot navigation and in Section V we present the experimental results on standard datasets. Finally, we summarize our approach and offer concluding remarks in Section VI.

2 Related Works

Loop closure detection is a fundamental problem in SLAM and is defined as the detection of the event when a robot returns to a previously visited place. This information is necessary, since it allows the robot to reduce and bound the errors and uncertainty in the estimated pose and environment map. Loop closure detection has been extensively studied and many solutions have been proposed over the years for robot navigation. In this work we focus only on image-based methods.

One of the popular methods to address this problem is visual Bag-of-Words (BoW). The BoW approach has achieved considerable success in content-based image retrieval as well as in object recognition and image classification [6]. The solution uses an offline process in which features in training images are extracted and their descriptors are clustered. The cluster centres then serve as visual words and the collection of visual words form a visual dictionary or vocabulary [18]. Given a query image, its visual features are vector quantized through a nearest-neighbor (NN) algorithm to match with the visual words in the dictionary, and an image descriptor is built in terms of the histogram of the visual words appearing in the image. Candidate images that are similar to a query image can be retrieved efficiently using an inverted index. Because the visual dictionary is built offline, the online cost includes feature extraction, nearest neighbor search, and indexing of the query image. Although BoW has been shown to be an efficient method for producing loop closure candidates, it suffers two key weaknesses. First, an offline step is needed to build a visual vocabulary from training images, but the training images may not represent the future views of the robot appropriately. Secondly, the step of vector quantization, which converts visual features into visual words required by indexing, can be inefficient with a linear search and may cause perceptual aliasing [15], i.e., high similarity between different locations.

Nister *et al.* [16] proposed "vocabulary tree" as a way of speeding up nearest neighbor search in a large database and [3] used this method for loop closure detection in visual SLAM. Vocabulary tree was introduced as a hierarchical approach to Bag of Words although a tree structure does not guarantee the exact nearest neighbor. Cummins *et al.* [7] proposed FAB-MAP as a probabilistic appearance based approach using BoW and showed its performance on large scale environment. Although Galvez-Lopez *et al.* [12] advance the method by introducing a Binary BoW (BBoW) to speed up the method, it still needs an offline process to build a dictionary.

As a competing approach to visual BoW, compact whole-image descriptors such as Gist [17] have been recently employed in performing visual loop closure detection [14]. Rather than describing an image in terms of its keypoints, a whole-image descriptor may simply use a down-sampled version of an image, its gradient information, or its response to a filter bank, to describe the image. Whole image descriptors can avoid the computational complexity of feature detection and vector quantization in BoW, but introduces the need to perform nearest neighbor search in matching the descriptor of the current view and those of the map locations. In addition, the quality of detected loop closures can be affected as the result of simple representations. These issues have been alleviated with some success with the help of the Monte Carlo technique [14].

Most recently, an interesting solution has been presented to cast the loop closure detection problem as one of sparse reconstruction [13]. The solution uses l_1-minimization algorithms and is accurate in matching the current view with the map images. In their work, the current view of the robot is matched with a small number of the all observations from the map images through convex l_1-minimization which provides a sparse solution. By using a fast convex optimization technique, they showed their method to be fast enough for a map with 8,500 images. However, since their method needs to solve l_1-minimization problem from scratch for each newly captured image, increasing the number of images as the map size grows leads to a computational complexity that can be infeasible in large scale environments. In fact, their method has a time complexity that is worse than linear search, as we will show in the experimental result section, and this gives little incentive for one to choose this method for solving the loop closure detection problem.

To address the computational complexity issue of solving l_1-minimization from scratch, in this paper we introduce a highly efficient approach for loop closure detection by first solving a static l_1-minimization problem once and then updating the convex l_1-minimization solution dynamically to avoid solving a new optimization problem for each newly captured image. We exploit the fact that in visual SLAM the current robot view is similar to the recent previous views. We use this property of the loop closure detection problem to formulate our solution as the dynamic update of the solution to l_1-minimization in the previous step.

3 Sparse Solution for Loop-Closure Detection

Sparse Representation (SR) is a signal processing technique for reconstructing a signal by finding solutions to an underdetermined linear system and it is solved through convex optimization algorithms. SR has been extensively used for face recognition [20], denoising [10], etc. We use this framework to find the closest image in a robot map to a new observation for loop closure detection in SLAM. In this section, we first present a brief overview of sparse representation and l_1-minimization. Then we will describe the dynamic update of the convex minimization problem to approximate the solution without solving a complete new minimization.

3.1 Loop closure detection via convex l_1-minimization

Image matching is essential for loop closure detection in visual SLAM. One recent successful image matching method, especially in large datasets, is the SR method [13]. Let n be the number of images in the robot map, and m be the dimension of the descriptor of each image in the map. Further, assume y to be the current view of the robot. The map images form a matrix $A \in R^{m \times n}$, and the linear representation of y can be rewritten in terms of all map images:

$$y = Ax_0 \quad , \quad y \in R^m \tag{1}$$

where x_0 represents the contributions of the map images to the reconstruction or representation of the current view. In SR the system is underdetermined with $m < n$. Therefore, recovering x_0 constitutes a non-trivial inverse problem. A classic solution to this problem is linear least squares, which finds the minimum l_2-norm solution to this system.

$$\hat{x}_2 = argmin\|x\|_2 \quad subject\ to \quad Ax = y \tag{2}$$

Equation (2) can be easily solved, but the solution $\hat{x}_2$ is dense (i.e., all its elements are non-zero in general) as is shown in [20] and is therefore not useful to retrieve y. Due to the fact that the query image can be represented using the map images at locations similar to the current robot location - if there is loop closure - the representation is naturally sparse, i.e., all but a small number of the elements of x are 0. The sparsest solution to $y = Ax$ is obtained by the following optimization problem:

$$\hat{x}_0 = argmin\|x\|_0 \quad subject\ to \quad Ax = y \tag{3}$$

The problem of finding the sparsest solution of an under-determined system of linear equations is NP-hard and difficult even to approximate [20, 1]. The theory of sparse representation [9] shows that if the solution $\hat{x}_0$ is sparse enough, the solution of the l_0-minimization is equal to the solution of the l_1-minimization, and x_0 can be retrieved by computing the minimum l_1-norm:

$$x^* = argmin\|x\|_1 \quad subject\ to \quad Ax = y \tag{4}$$

In real applications such as image matching in visual SLAM, a true loop closing image y can only be represented by map images approximately with slightly different illuminations, translations, and rotations. In such cases $\|Ax - y\|_2 \leq \epsilon$, where $\epsilon > 0$. So, to find a sparse solution x^*, one could use conventional l_1-regularized least squares regression as follows:

$$x^* = \underset{x}{argmin}\frac{1}{2}\|Ax - y\|_2^2 + \lambda\|x\|_1 \tag{5}$$

where l_1-regularization enforces sparsity on x^*; unfortunately, the complexity of solving (5) grows polynomially with m and n.

3.1.1 Solving l_1-minimization

In practice, for loop closure detection in visual SLAM, we have to solve (5) online and accurately. One of the fastest l_1-minimization methods is *homotopy* algorithm associated with the *basis pursuit denoising* (BPDN) [5] approach, which is applied by Latif *et. al.* [13] and described below for the completeness of presentation.

A solution x^* to (5) should follow the condition [11, 2]:

$$\|A^T(Ax^* - y)\|_\infty \leq \lambda \tag{6}$$

In the above equation, we distinguish between the nonzero components and the zero components of x^*. We denote $\bar{x}^*$ the reduced dimensional vector built upon the nonzero components of x^*. Similarly, A_Γ denotes the associated columns in A (Γ *is a set with the indexes of nonzero elements in* x^*). So, the optimality conditions for any given value of λ are as follows [11]:

$$A_\Gamma^T(Ax^* - y) = -\lambda z \tag{7}$$

$$\|A_{\Gamma^c}^T(Ax^* - y)\|_\infty < \lambda \tag{8}$$

where A_Γ is a $m \times |\Gamma|$ matrix from the columns of A indexed by Γ and vector z is signs of $\bar{x}^*$. A_{Γ^c} denotes all columns of A that are not in A_Γ. From the support Γ and z, the solution x^* can be calculated as follows [11, 2]:

$$x^* = \begin{cases} (A_\Gamma^T A_\Gamma)^{-1}(A_\Gamma^T y - \lambda z) & \textit{on } \Gamma \\ 0, & \textit{otherwise} \end{cases} \tag{9}$$

The algorithm proceeds by computing (7), (8), and (9) iteratively, until $A^T(Ax^* - y) < c$ (a small constant such as e^{-6}) and the final x^* represents the solution for (5).

3.2 Dynamic Update for Homotopy

The static homotopy solution described in the previous section has a complexity that is polynomial in n and m, and can therefore be too slow for a large scale map. However, in loop closure detection, we expect the current image captured by a robot to be similar to the image that robot captured in the previous time instance. So, we can update the l_1-minimizer for the last image, described in Section 3.1.1, to obtain the solution to the current image without solving the optimization problem from scratch. Asif and Romberg [2] explained the problem of estimating a time varying sparse signal from a series of linear measurement vectors to update the standard BPDN homotopy dynamically. They assumed that the signal changes only slightly between measurements, so that the reconstructions will be closely related, an assumption that holds true in visual SLAM for finding the best match between the current image and the map images. This dynamic method enables us to arrive at a solution that is highly efficient and capable of handling large-scale robot environments. In the rest of this section, we apply the dynamic algorithm [2] to loop closure detection in visual SLAM.

Assume that we have solved the BPDN problem (5) for a given value of λ. Now, for a new image, we express it as a m-dimensional feature vector $\breve{y}$, and the problem we have to solve for the new image approximately is:

$$\underset{x}{argmin}\ \frac{1}{2}\|Ax-\breve{y}\|_2^2+\lambda\|x\|_1 \tag{10}$$

with the same value of λ in (5). In classical approaches (10) is solved for each image without benefiting from the just-completed solution. Our goal is using the information from the solution of (5) to quickly compute the solution for (10). Thus, we use the homotopy formulation in [2]:

$$\underset{x}{argmin}\ \frac{1-\epsilon}{2}\|Ax-y\|_2^2+\frac{\epsilon}{2}\|Ax-\breve{y}\|_2^2+\lambda\|x\|_1 \tag{11}$$

where ϵ is the homotopy parameter. By increasing ϵ from 0 to 1, (11) moves from the solution of (5) to the solution of (10). By adapting the optimally conditions of (7) and (8) for (11), we have:

$$A_\Gamma^T(Ax^*-(1-\epsilon)y-\epsilon\breve{y})=-\lambda z \tag{12}$$

$$\|A_{\Gamma^c}^T(Ax^*-(1-\epsilon)y-\epsilon\breve{y})\|_\infty<\lambda \tag{13}$$

where Γ is the support of solution x^* and z is its sign sequence on Γ. From (12) the solution x^* for (11) follows a piecewise linear path as ϵ varies. The critical point in this path occurs when an element is either added or removed from the solution x^*. Parameter ϵ increases incrementally from 0 to 1 and [2] proved that the direction of the solution x^* moves by:

$$\partial x=\begin{cases}(A_\Gamma^T A_\Gamma)^{-1}A_\Gamma^T(\breve{y}-y), & on\ \Gamma\\ 0, & otherwise\end{cases} \tag{14}$$

With the moving direction given by (14), we are able to find the step-size θ [2], which leads us to the next critical value of ϵ. Afterwards, the solutions at that point are as follows:

$$\epsilon\longleftarrow\epsilon+\theta,\qquad x^*\longleftarrow x^*+\theta\partial x \tag{15}$$

This procedure is repeated from (12) to (15) until $\epsilon=1$ and the final x^* represents the solution for (11), which means the best matched images could be found by this dynamic updating method without solving (11) independently.

4 Implementation Details and Discussion

We have formulated loop closure detection problem with the dynamic update of BPDN homotopy algorithm in Section 3.2. Here, we explain how this novel formulation can be utilized in visual loop closure detection.

4.1 Initialization

To initialize our solution to loop closure detection and construct our A matrix, we use the first n images or keyframes captured by the robot where n is a small number (e.g., 10). In addition, as is customary, we exclude the last l images seen by the robot from consideration in matching the current view with the map images in order to avoid triggering false loop closure detection due to the similarity between successive images. l is another small number, e.g., 20.

After constructing A, for the next query image we use standard homotopy to obtain the initial solution x^* just once, and this solution is updated via dynamic method for all the subsequent images while the robot moves and captures additional keyframes.

4.2 Selection of the Top Candidate from Solution x^*

The solution x^*, obtained by either homotopy or its dynamic update, is naturally sparse and represents candidate images from the robot map to best match with the current view. To find a unique image and potentially close a loop, we select the greatest contribution α_i from the solution $x^* = [\alpha_1, ..., \alpha_n]$. The index i, corresponds to the column of the matched image in the map. To improve the chance of true positive detection and reduce false alarm, we use the heuristic that if 2-norm between the matched image and the current image is less than a predefined threshold τ, a loop closure is detected ($\|A_{:,i} - \breve{y}\|_2 < \tau$). τ can be chosen empirically and, as will be shown in our experiments, the precision of the detected loop closures can reach 100%.

We should add that the proposed method accommodates the growth of the robot map, when a novel image is detected, by adding a column to the end of the matrix A, i.e. $A_k = [A_{k-1},\ f_k]$ so that the map grows incrementally similar to [13]. f_k denotes the descriptor of k^{th} image being added to the map. The main steps of our method are summarized in Algorithm 1.

Algorithm 1 Closing Loops via the Proposed Method

```
 1: Initialization:
 2:     Preparing Matrix $A$ (see Section 4.1) , $\lambda = 0.5$,
 3: For the first query image $i$
 4:     Obtain $x^*$ with Solving (5) via standard homotopy
 5:     Closing loops (see Section 4.2)
 6:     Update matrix $A$ (see Section 4.2)
 7: For all query images $i$ to $n$
 8:     Update $x^*$ via dynamic method (see Section 3.2)
 9:     Closing loops (see Section 4.2)
10:     Update matrix $A$ (see Section 4.2)
```

4.3 Discussion

In terms of computational cost, although homotopy is one of the popular and fastest solvers for SR, the computational complexity of homotopy is still $O(dm^2+dmn)$ to recover a d-sparse signal in d steps. Obviously, this complexity grows polynomially with m and it is expensive for large-scale datasets or maps. In contrast, in the proposed method, for each query image, the main computational cost comes from solving a $|\Gamma| \times |\Gamma|$ system of equations to compute the direction in (14). $|\Gamma|$ is equal to the number of nonzero elements of the sparse solution x^* which means its size is too small. Therefore, the computational cost of the proposed method is significantly lower than the static homotopy method.

5 Experimental Results

In this section, we perform a set of experiments to demonstrate and validate the capability of the dynamic updating of l_1-minimization method to perform loop closure detection in visual SLAM. In particular, we evaluate the computational cost and the accuracy of the proposed method and compare it with the standard l_1-minimization in [13] and a nearest neighbor (NN) method. Since the proposed method is appropriate to detect loop closure in large scale datasets, we compare our method with FAB-MAP as well. We use three datasets: New College, City Centre, and a Google Street View dataset, with the following details.

- New College: This dataset consists of 2146 images along a 2.2km trajectory. Each image has originally a resolution of 640×480 and is down sampled to 320×240. The dataset provides stereo images from the left and right of the robot, and we use both images from each location as query image with a combined resolution of 640×240 so that $n = 1073$.

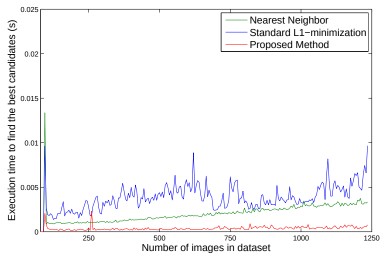

Fig. 1: Execution time comparison (in seconds) on "City Centre" dataset

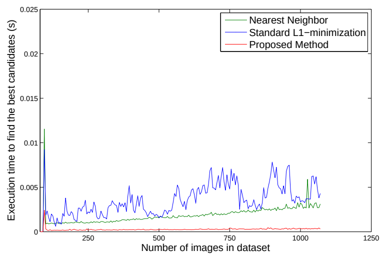

Fig. 2: Execution time comparison (in seconds) on "New College" dataset

- City Centre: This dataset consists of 2474 images and similar to New College dataset provides stereo images. Each image has a resolution of 640×480 and is down sampled to 320×240. Again, we use both images from each location as query image with a combined resolution of 640×240 so that $n = 1237$.
- Google Street View: This dataset consists of about 50000 images captured in downtown Pittsburg by Google. This dataset has omni-directional images by four cameras at each location, and we reduce the resolution of the obtained panoramic view to 640×240. The number of locations in this dataset is around $n = 12,500$.

To describe an image, we use HOG [8] with $m = 576$ dimensions and a constant weighting parameter $\lambda = 0.5$. In this section, we focus on the capability of our dynamic model in comparison with the static homotopy and NN method. In all methods we use the same algorithm for closing loops.Also, to be consistent with [13] we just pick the highest α as a top candidate (the first method in Section 4.2).

5.1 Execution Time

In the first set of experiments, we compare the computational cost of the proposed method with the standard l_1-minimization and the NN method, when the size of the dataset is increased incrementally. We run the experiments in Matlab 2011b on a desktop computer with Core-i7 CPU of 3.40GHz and 16 GB RAM.

Fig. 1 and Fig. 2 show the execution time for finding the best candidates on "City Centre" and "New College" datasets respectively, when the images are added to the map incrementally. These figures illustrate the proposed method is faster and more stable (smaller standard deviation) than the standard L1-minimizer, by a factor of four on average. Also, our dynamic model is around two times faster than the nearest neighbor method on both datasets.

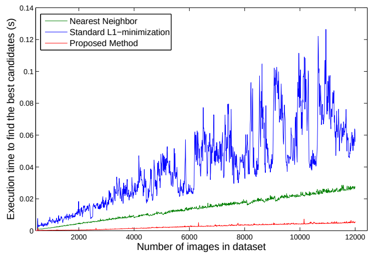

Fig. 3: Execution time comparison (in seconds) on "Google Street View" dataset with using of 576 dimensional HOG descriptor for images

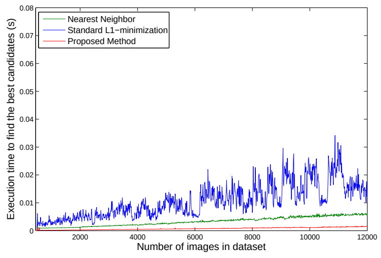

Fig. 4: Execution time comparison (in seconds) on "Google Street View" dataset with using of 81 dimensional HOG descriptor for images

To show the capability of the proposed method on larger datasets, we compare the computational time of the dynamic method with the standard l_1-minimizer and NN method to find the best match on "Google Street View" dataset in Fig. 3. This experiment confirms that the proposed method is much faster than both the standard homotopy and NN methods when the map is large. Also, Fig. 3 demonstrates the dynamic update method has much smaller standard deviation than the standard homotopy method to obtain the solution as the map expands with additional images. Furthermore, comparison of Fig. 3 and Fig. 4 shows the scalability of our model in comparison with two other methods in terms of feature vector dimension. By increasing m as the dimension of feature vector, computation time of the two other competing methods increases at a faster rate than our model. The qualitative result on the Google Street View dataset is also shown in Fig. 5 where the blue lines represent the robot map and the red dots represent detected loop closures by the proposed method. The ground truth of loop closures (in green dots) can be found in Fig. 6.

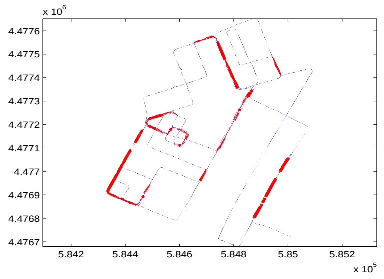

Fig. 5: qualitative result of the proposed method on "Google Street View" dataset to find loop closures

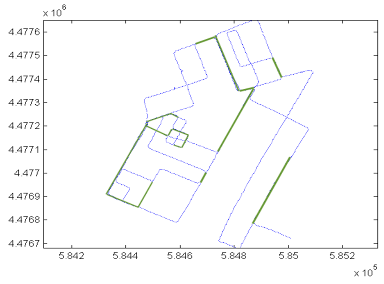

Fig. 6: Graphical ground truth for "Google Street View" dataset from Pittsburg

Table 1: Execution time statistics for one iteration of the loop detection algorithm of the proposed method, the standard homotopy, and NN method on different datasets

	Map size	Nearest Neighbor		Standard homotopy		Proposed Method	
		mean (ms)	std (ms)	mean (ms)	std (ms)	mean (ms)	std (ms)
City Centre	1.2K	1.91	0.20	3.59	2.28	0.40	0.26
New College	1.1K	1.74	0.23	3.32	2.52	0.39	0.22
Google Street	12.5K	26.80	1.53	77.34	33.40	4.97	0.82

Table 1 shows the quantitative results of Figs. 1, 2, and 3 in terms of average execution time and standard deviation on the three datasets. For small maps like City Centre or New College datasets, the proposed method is 10 times faster than the standard homotopy on average. The execution time on the Google Street View dataset in Table 1 shows the capability of our method on large maps in comparison with the standard homotopy and even NN method. The average computational time of the standard homotopy increases more than 20 times from around 3.5 ms to 77 ms, when the map grows 10 times from around 1200 images to 12000 images; however, the computational time for the proposed method only increases linearly, and the standard deviation increases approximately 3 times when the map grows 10 times. In absolute terms, with crude extrapolation, our proposed algorithm could potentially perform loop closure detection in a map with a million images in under one second.

5.2 Loop Closure Detection Accuracy

In this part, we compare the accuracy of the proposed dynamic method against the standard homotopy method for the loop closure detection. Fig. 7 and Fig. 8 show the precision recall curves of the proposed method and the standard homotopy method on "City Centre" and "New College" datasets respectively. Like before, no specific verification step was used and the decision for closing loops is only based on simple thresholding of the top α_i as described in the Section 4.2. According to these figures, the accuracy of the proposed method using a dynamic algorithm is essentially the same as the standard l_1-minimization method. Therefore, using the proposed method, loop closure detection can be solved much faster without losing accuracy. We also compare the proposed method with FAB-MAP as a baseline method for large-scale dataset in Table 2. Although FAB-MAP is not the state-of-the-art in terms of loop detection accuracy, it has been evaluated on the same datasets as used in this work, and is therefore directly comparable. At 100% precision, the proposed method achieves 68% and 57% recall on "City Centre" and "New College" datasets with $\tau = 0.98$ and $\tau = 0.45$ respectively, which is

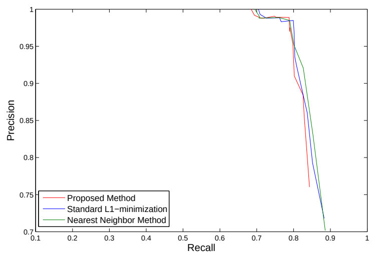

Fig. 7: Precision and recall curves of the proposed method, the standard homotopy, and NN method for loop closure detection on "City Centre" dataset

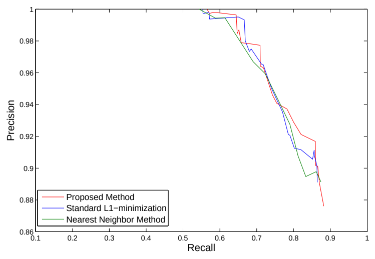

Fig. 8: Precision and recall curves of the proposed method, the standard homotopy, and NN method for loop closure detection on "New College" dataset

higher than the recall for FAB-MAP method reported in [7]. τ is empirically chosen to allow comparison with other methods at 100% precision.

6 Conclusion

We have presented in this paper a novel technique to detect loop closure that is highly efficient in time and competitive in detection accuracy. The proposed method formulates the loop closure detection as a sparse representation problem. Since in visual SLAM the current view of the robot is similar to the most recent previous image, we are able to update the obtained solution from one iteration of static l_1-minimization for loop closure detection using the subsequent robot view without solving the minimization problem from scratch. Using our dynamic update method, loop closure detection can be solved much faster than the static method without losing accuracy. The proposed method is therefore more scalable and able to handle larger robot maps. The reliability and efficiency of the proposed method have been validated on three different publicly available datasets. In the future, we plan to implement the proposed algorithm on real robots in an online SLAM system.

Table 2: Comparison of the recall between the proposed method and FAB-MAP as baseline at precision 100%

	City Centre	New College
Proposed Method	68%	57%
FAB MAP	37%	48%

References

1. Amaldi E, Kann V(1998) On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems, In: Theoretical Comp. Sci., vol. 209, 237:260.
2. Asif M. S, Romberg J (2010) Dynamic updating for l1-minimization, In: IEEE J. of Selected Topics in Signal Processing, vol. 4, no. 2, 421:434.
3. Callmer J, Granstrom K, Nieto J, and Romas F (2008) Tree of words for visual loop closure detection in urban SLAM, In: Proc. of the Australian Conf. on Robotics and Automation.
4. Cannell C. J, Stilwell D. J (2005) A comparison of two approaches for adaptive sampling of environmental processes using autonomous underwater vehicles. In: MTS/IEEE OCEANS, 1514:1521.
5. Chen S. S, Donoho D. L, Saunders M. A (1999) Atomic decomposition by basis pursuit, In: SIAM J. Sci Comput., vol. 20, no. 1, 33:61.
6. Csurka G, Dance C. R, Fan L, Williamowski J, Bray C (2004) Visual categorization with bags of keypoints. In: ECCV International Workshop on Statistical Learning in Computer Vision, 1:22.
7. Cummins M, Newman P (2008) FAB-MAP: Probabilistic localization and mapping in the space of appearance, In: The Int. J. of Robotic Research. vol. 27, No. 6, 647:665.
8. Dalal N (2005) Histograms of oriented gradients for human detection, In: IEEE Conf. on Computer Vision and Pattern Recognition Workshops, 886:893.
9. Donoho D, (2006) For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution, In: Comm. Pure and Applied Math, vol. 59, no. 6, 797:829.
10. Elad M, Figueiredo M, Ma Y (2010) On the role of sparse and redundant representations in image processing, In: Proc. of the IEEE. vol. 98, no. 6, 972:982.
11. Fuchs J (2004) On sparse representations in arbitrary redundant bases, In: IEEE Trans. on Information Theory, vol. 50, no. 6, 1341:1344.
12. Galvez-Lopez D, Tardos J. D (2012) Bag of binary words for fast place recognition in image sequences, In: IEEE Trans. on Robotics, vol. 28, no. 5, 1188:1197.
13. Latif Y, Huang G, Leonard J, Neira J(2014) An online sparsity-cognizant loop-closure algorithm for visual navigation, In:Proc. of Robotics: Science and Systems.
14. Liu Y, Zhang H (2012) Visual loop closure detection with a compact image descriptor, In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 1051:1056.
15. Newman P, Cole D, Ho K(2006) Outdoor SLAM using visual appearance and laser ranging, In: proc. of Int. Conf. on Robotics and Automation, 1180:1187.
16. Nister D, Stewenius H (2006) Scalable recognition with a vocabulary tree, In: Proc. of IEEE Conf. on Computer Vision and Pattern Recognition, vol. 2, 2161:2168.
17. Oliva A, Torralba A(2001) Modelling the shape of the scene:A holistic representation of the spatial envelope, In: Int. J. of Comp. Vision, vol.42, no.3, 145:175.
18. Sivic J, Zisserman A (2003) Video google: A text retrieval approach to object matching in videos, In: 9th IEEE ICCV, 1470:1477.
19. Sugiyama H, Tsujioka T, Murata M (2005) Collaborative movement of rescue robots for reliable and effective networking in disaster area, In: Int. Conf. on Collaborative Computing: Networking, Applications and Worksharing.
20. Wright J, Yang A, Ganesh A, Sastry S, Ma Y (2009) Robust face recognition via sparse representation, In: IEEE Trans. on PAMI, vol. 31, no. 2, 210:227.