Improved Visual Localization via Graph Smoothing

Carlos Lassance, Yasir Latif, Ravi Garg, Vincent Gripon, Ian Reid

Improved Visual Localization via Graph Smoothing

Carlos Lassance[1,2], Yasir Latif[3], Ravi Garg[3], Vincent Gripon[1,2] and Ian Reid[3]

***Abstract*— Vision based localization is the problem of inferring the pose of the camera given a single image. One solution to this problem is to learn a deep neural network to infer the pose of a query image after learning on a dataset of images with known poses. Another more commonly used approach rely on image retrieval where the query image is compared against the database of images and its pose is inferred with the help of the retrieved images. The latter approach assumes that images taken from the same places consists of the same landmarks and, thus would have similar feature representations. These representation can be learned using full supervision to be robust to different variations in capture conditions like time of the day and weather. In this work, we introduce a framework to enhance the performance of these retrieval based localization methods by taking into account the additional information including GPS coordinates and temporal neighbourhood of the images provided by the acquisition process in addition to the descriptor similarity of pairs of images in the reference or query database which is used traditionally for localization. Our method constructs a graph based on this additional information and use it for robust retrieval by smoothing the feature representation of reference and/or query images. We show that the proposed method is able to significantly improve the localization accuracy on two large scale datasets over the baselines.**

I. Introduction

Vision-Based Localization (VBL) [1] is the problem of retrieving the location and orientation (pose) of the camera which will generate a given query image. VBL can be used to improve accuracy of vehicle tracking as well as for accurate visual maps creation via loop closure. The approaches for addressing VBL can be broadly divided into two categories [1]:

1) Direct methods: These methods directly retrieve pose from the visual query – usually by solving a regression problem. Use of deep learning techniques replacing this regressing have become prevalent forming the current state of the art where a set images with known poses are used to learn a mapping from raw pixel colors to image poses.
2) Indirect methods: In these methods, the pose information is inferred from the visual query using a reference database, where each image in the database has an associated pose. This can be seen as an image retrieval problem where the aim is to find images in the support set that might have been taken from the same location as that of the query image. Once a match or set of matches is found, the pose for the query image is computed as a function of the poses of the retrieved images.

Use of the deep learning for VBL approaches has recently received a lot of attention both in terms of learning direct image to pose mappings [2], [3] or to generate latent representation that are resilient to appearance changes [4]. A major drawback of direct approaches is their inability to generalize to previously unseen locations. Even small difference of query pose from the training set can cause gross localization errors and appending new query locations to the dataset for direct approach will require retraining the whole pose estimation network from scratch. On the contrary, indirect methods generalize well to new data without the need for this retraining.

However, the retrieval based indirect VBL is challenging because it is extremely difficult to learn a representation which is resilient to a huge amount of appearance variations. Moreover, if the pose of the query image is relatively different from the corresponding images in the database a correct match needs to be found by interpolation in the latent representations with additional information. Sequence to Sequence matching approaches like [5] have been proposed for making the retrieval more robust in these cases, but the success of these approaches simply rely on having a large number of images in the query set.

Proposing a very robust method for sequence/one to one image matching for localization due to external factors (e.g. different image acquisition conditions) is a hard problem. A principled solution will be capable of smartly interpolating the latent space for retrieval, given some additional information about the data. The large support dataset against which a query can be retrieved generally comes with rich information such as GPS location of the image and temporal order in which the images are captured.

In this work, we propose an indirect visual localization method that takes advantage of the additional information that might be available for each image in the database, including GPS coordinates, consecutiveness in the acquisition process and similar latent representations. This is particularly interesting for a robotics setting, where images are almost always acquired sequentially from a camera mounted on a vehicle. This sequential nature of the acquisition process suggests that images closer in time should also be close in the latent representation. Indeed, it is intuitive that temporally adjacent images have similar latent representa-

*This work was supported by FASIC, the Britanny region. Computations were performed using Nvidia GPUs, courtesy of Nvidia.

[1]IMT Atlantique and Lab-STICC, Brest, France
[2]Université de Montréal and Mila, Canada
[3]University of Adelaide, Australia

tion. Additional information such as GPS coordinates, if available, can aid in encoding global relationships between images in the database. We show that by considering such relationships between images, localization accuracy can be increased. Moreover, enhancements can be achieved using only minor adjustments to the inference process. Specifically, we exploit relationships via a graph smoothing operation [6] on top of pre-learned deep representations extracted from NetVLAD [4]. The graph smoothing operation is derived from the Graph Signal Processing (GSP) framework [6], and takes advantage of the graph based representation of the problem. In this graph, each vertex is associated one-to-one with an image (or equivalently its latent representation). Edges model relations between images and are derived from the additional source of information (e.g. temporal adjacency, GPS, similar latent representations).

Interestingly, the proposed method can be seen as a fine-tuning of the representations that does not require additional learning, allowing this operation to be possibly executed on a resource constrained system.

Main contributions: The main contributions of this work are two-fold:

1) We apply Graph Signal Processing techniques to the problem of indirect visual localization. To the best of our knowledge, we are the first to bring together the area of Graph Signal Processing and Visual Based Localization.
2) Through experiments on real-world datasets, we demonstrate the efficacy of the proposed method in improving the accuracy of the indirect VBL process on large scale datasets.

The rest of the paper is organized as follows: we present a brief overview of related techniques in Section II. In Section III, we formally introduce the proposed method and discuss its properties. In Section IV, we derive and discuss experiments. In Section V, we conclude the work and discuss future directions.

II. Related Work

Visual localization is a well studied problem in the vision community and a recent survey can be found in [1]. Traditional methods address the problem using point features using a Bag-of-Words (Bow) approach where each image is represented as histogram of visual word occurrences. Efficient indexing methods then allow retrieving images with similar features and a relate pose computation via the essential matrix. However, such methods can be adversely affected by changes in condition such as weather, time of the day and long term changes such as structure of the scene.

Deep learning in direct visual localization: with the recent revival of deep learning, work has focused on formulating VBL as an end-to-end learning problem where the pose of the image is regressed from the raw pixels via a deep neural network [3]. Other works such as [7] have explored scene coordinate regressed followed by RANSAC to compute the camera pose via 3D to 2D correspondences. This has shown great improvement over the end-to-end approach.

Deep learning in indirect visual localization: as mentioned in the introduction, various methods in the literature focus on deep learning for generating good embeddings for indirect visual localization, such as NetVLAD [4]. In this work, we build on top of these representations, though the proposed method could be adapted to any latent representation of the images. Its main advantage is that it is not required to perform any additional training. Recent work in robotics [8] has shown that using sequence information in Bayesian filtering approach, the accuracy of indirect methods can be vastly improved, even outperforming direct methods.

Graphs in visual localization: works [9], [2], [10] have used graphs to increase the performance of visual localization methods in various ways. One example is the re-ranking of candidates in indirect VBL, where one can use a graph to perform a ranking that takes into account more than one image at a time. This is achieved in [9] by using the closest pair of images and then performing linear combination of them. Other works such as [2] use techniques like Pose-Graph Optimization (PGO) [11] to take advantage of extra information available (in this case the relative poses of the "test"). Note that these approaches differ from ours as they are used only on the query data. As such, they could be combined with the proposed method, that also considers the reference database.

GSP: graph signal processing [6] is a mathematical framework that aims at extending harmonic analysis to irregular domains described using similarity graphs. As such, it is possible to define tools such as translations [12], convolutions [13], filtering [14] and wavelets [15] taking into account the complex structure of the inputs. GSP has successfully been applied to domains ranging from neuroimaging [16] to deep learning [13], [17], [18]. To our knowledge, the present work is the first usage of GSP in the context of indirect visual localization.

III. Methodology

In this section, we first describe the setting in which the current solution is applied and then present a formal overview of the GSP techniques as they are applied to the problem of visual localization.

A. Problem Setting

We consider the case of autonomous driving where a fleet of vehicles move around established roads in urban environments. This is a restricted setting than the more general case of localizing a freely moving tourist in a city using a mobile phone. Indeed the geometry of the road structure prevents significant view point variations. In our case the change in

viewpoint comes from traffic moving in different lanes along the same road. However, there might be significant viewpoint changes as vehicles can move during any season and any time of the day.

The camera mounted on the vehicle provide a stream of images, that is, we have information about the temporal adjacency of images. In addition, we are also provided additional information in the form of GPS location for each image.

For image representation, we assume a mapping function that maps each image to a fixed dimensional latent space, with some resilience to viewpoint and appearance changes. For the rest of the section, we use images and latent representation interchangeably to mean a lower dimensional embedding of the original image into a resilient subspace. A key asset of the latent space is that it *linearizes* representations. As such, by taking the linear combination of latent representations of actual images, we usually obtain a latent representation of a natural looking (artificial) image.

B. Graph Signal Processing and Graph Signals Smoothing

In this work we consider graphs defined as tuples $G = \langle V, \mathbf{A} \rangle$, where V is the finite set of vertices and $\mathbf{A}$ is the weighted adjacency matrix: $\mathbf{A}[\mu\nu]$ is the weight of the edge between vertices μ and ν, or 0 if no such edge exists. Vertices are associated one-to-one with images, and an edge defines the similarity between two vertices.

In order to avoid irregular artifacts, we consider a normalized adjacency matrix $\mathbf{A} = \mathbf{D}^{-1}\mathbf{W}$ where $\mathbf{W}$ is the direct measure of similarity between two vertices and $\mathbf{D}$ is the degree matrix associated with $\mathbf{W}$:

$$\mathbf{D}[\mu\nu] = \begin{cases} \sum_{k \in V} \mathbf{W}[\mu k] & \text{if } \mu = \nu \\ 0 & \text{otherwise} \end{cases}.$$

Note that this normalization is only well-defined if the graph has no isolated vertex, what we consider to be true in the following.

Given a graph $G = \langle V, \mathbf{A} \rangle$, consider a matrix $\mathbf{s} \in \mathbb{R}^{V \times d}$, where $d \in \mathbb{N}$. We refer to $\mathbf{s}$ as a signal in the remaining of this work, and typically we consider $\mathbf{s}$ to be composed of the concatenation of latent representations of images corresponding to vertices in V. As such, a row of $\mathbf{s}$ corresponds to an image in the dataset, whereas a column correspond to a dimension of the feature vectors representing the images. We define the *graph smoothing* $h_G(s)$ of $\mathbf{s}$ as:

$$h_G(\mathbf{s}) = \mathbf{A}^m \mathbf{s}. \quad (1)$$

Graph smoothing simply consists of multiplying the normalized matrix $\mathbf{A}$ of the graph with the signal. This operation can be repeated multiple times (represented by the parameter m). Note that smoothing can be achieved in other ways, for example using low-pass filters [6] on the graph, which can be computationally expensive for large graphs. In this work, we focus on this particular smoothing method (1) for its simplicity and performance.

Let us explain briefly why this operation has the effect of smoothing the representations in $\mathbf{s}$. First note that because $\mathbf{A}$ is symmetric and real-valued, it admits $|V|$ eigenvalues (where $|\cdot|$ denotes the cardinal). The way $\mathbf{A}$ has been normalized, all these eigenvalues are between -1 and 1. Other interesting properties include that the eigenspace associated with the eigenvalue 1 is composed of constant vectors and -1 is not an eigenvalue if the graph is not bipartite.

So, considering the graph is not bipartite, multiplying the signal by $\mathbf{A}$ has the effect of diminishing the influence of all components of the signal that are not aligned with a constant vector, while maintaining the latter. As a result, the difference between representations of neighboring vertices in the graph is reduced. This operation has the effect of smoothing the signal, in the sense that the i-th column of the smoothed signal is such that the difference in values between two (strongly) connected vertices is going to be smaller than that before smoothing. In the extreme case of smoothing multiple times (i.e. large m), this would eventually have the effect of averaging all representations in connected components of the graph.

In brief, graph smoothing has the effect of smoothing the signal values, taking into account strongly connected vertices in the graph. As a result, outliers are smoothed using similar images in the graph. In this work, we consider the vertices to be either the reference database or the query database. In both cases, the goal is to use graph smoothing to reduce the noise in the latent representations. This is illustrated in Figure III-B, where we consider a unidimensional signal represented using blue (for positive values) and red (for negative values) bars. Before smoothing (on the left), neighboring vertices can have large variations in their signal values. After smoothing (on the right), these variations are lowered. Note that the parameter m in Equation (1) controls the intensiveness of smoothing: when m is small (i.e. almost 0), $\mathbf{A}^m$ becomes close to the identity matrix and the smoothing has almost no effect. When m is large (i.e. $m \gg 1$), $\mathbf{A}^m$ becomes an averaging matrix on each of its connected components.

C. Graph definition

In order to make the graph smoothing improve the accuracy of VBL, we need to make sure that the edges of the graph are well chosen to reflect the similarity between two images represented as vertices, as our main goal is to exploit extra information available at the database. In this work, we consider three different sources:

- Metric distance (`dist`): the distance measured by the GPS coordinates between vertices μ and ν;
- Sequence (`seq`): the distance in time acquisition between two images (acquired as frames in videos);

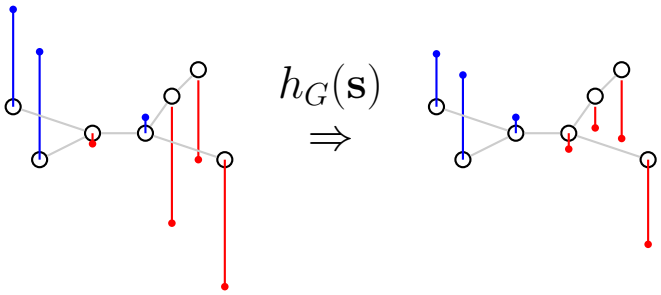

Fig. 1. Illustrative example of the graph smoothing operation. The signal is represented by the blue(positive) and red(negative) bars.

- Latent similarity (`latent_sim`): the cosine similarity between latent representations.

The matrix $\mathbf{W}$ can therefore be derived from the three sources as:

$$\mathbf{W} = \mathbf{W}_{\texttt{dist}} + \mathbf{W}_{\texttt{seq}} + \mathbf{W}_{\texttt{latent\_sim}}.$$

1) Metric distance: In order to transform the metric distance into a similarity, we use an exponential kernel. This is parametrized by a scalar α that controls the sharpness of the exponential and a threshold parameter $max_{distance}$ that cuts edges between distant vertices:

$$\mathbf{W}_{\texttt{dist}}[\mu\nu] = \begin{cases} e^{\alpha dist_{\mu,\nu}} & \text{if } dist_{\mu,\nu} < max_{distance} \\ 0 & \text{otherwise} \end{cases}.$$

Note that the choice of an exponential kernel may seem arbitrary, but is often used in the area of Graph Signal Processing [6].

2) Sequence: To exploit the information of time acquisition of frames, we use the function $seq(k, \mu, \nu)$ which returns 1 if the frame distance between μ and ν is exactly k and 0 otherwise. We then build a matrix $\mathbf{W}_{\texttt{seq}}$ parametrized by scalars β_k and k_{max}:

$$\mathbf{W}_{\texttt{seq}}[\mu\nu] = \sum_{k=1}^{k_{max}} \beta_k seq(k, \mu, \nu).$$

3) Latent similarity: Finally, we define a matrix $\mathbf{W}_{\texttt{latent\_sim}}$ for the latent representations cosine similarity. This is parametrized by a scalar γ that controls the importance of the latent similarity. We only compute this similarity if either the distance similarity or the sequence similarity is nonzero:

$$\mathbf{W}_{\texttt{latent\_sim}}[\mu\nu] = \begin{cases} \gamma sim(\mu, \nu) & \text{if } W_{dist}[\mu\nu] > 0 \\ & \text{or } W_{seq}[\mu\nu] > 0, \\ 0 & \text{otherwise} \end{cases}.$$

where sim is the latent similarity function. In this work we use the cosine similarity, but any similarity function could be used.

D. NetVLAD

For image representation in a latent space, we use features from a pretrained NetVLAD [4], trained on the Pittsburgh dataset [19], [20]. The model is available online at [21]. NetVLAD is specifically trained to cater for viewpoint and appearance changes. It maps an image to 32768 dimensional deeply learnt representation, which we then compressed to 4096 dimensions using PCA (trained on the support database) and then finally whitened [22]. We follow the same image preprocessing from the training of the model, where images are first resized so that the smaller part has a size of 256 pixels, then we perform a center-crop of 224, and finally perform standardization.

IV. Experiments

A. Dataset generation

In order to verify the effectiveness our method in the setting of autonomous driving, we need a dataset that is collected from roads and is large enough to demonstrate appearance changes and limited viewpoint changes due to road structure. We collect images from Mapillary API[1], which contains publicy sourced data over time for major roads. To show the generalization ability of the proposed work, we collect road imagery from two Australian cities. The first covers the Central Business District (CBD) area of Adelaide, Australia and spans an area of roughly 10km^2. Since the data is publicly sourced, there is a lot of viewpoint, illumination and dynamic changes (cars, pedestrian, etc). The second set is collected around the Greater Sydney region and covers an area of around 200km^2. We note that the data collected for the Greater Sydney region contains some sequences that were generated using different equipment (panoramic cameras) or different positioning (camera pointed to a vehicule window instead of the windshield) from the ones used during the training of the NetVLAD network, which combined with the total area of the support database creates a much more challenging problem. In addition to imagery, the collected data provides sequence information and GPS. The GPS tracks for the collected data are shown in Figures 2 and 3.

In the rest of the experiment section, we use the terminology of indirect visual localization, that is, support database refers to the reference database, validation and test queries refer to query inputs.

To split the Adelaide dataset in support/validation/test we randomly choose 4 sequences for validation and 5 sequences for testing. For the Sydney database, we choose 5 sequences that could be retrieved with reasonable performance using our pre-trained NetVLAD (named easy query) and 5 sequences at random (hard query). Using GPS as ground truth, we remove all examples from the query sets that are further than 25m from the support dataset (i.e. there are no examples

[1] `https://www.mapillary.com/developer/api-documentation/`

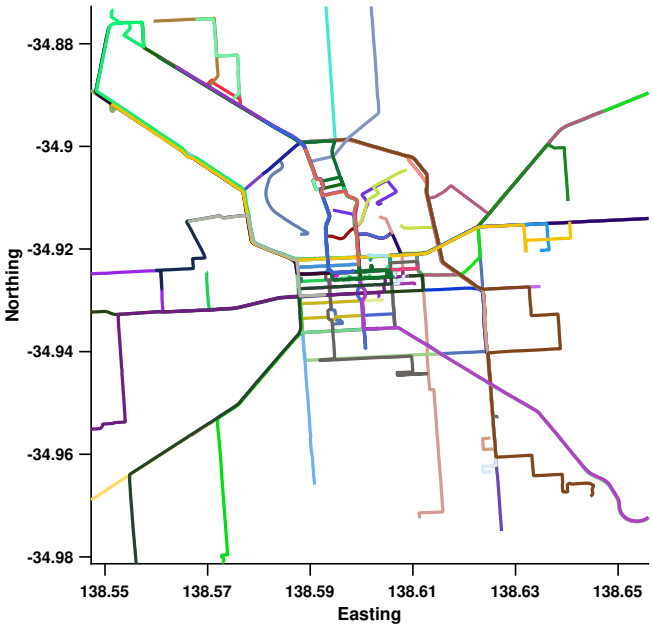

Fig. 2. GPS Tracks of image sequence collected around Adelaide CBD from Mapillary.

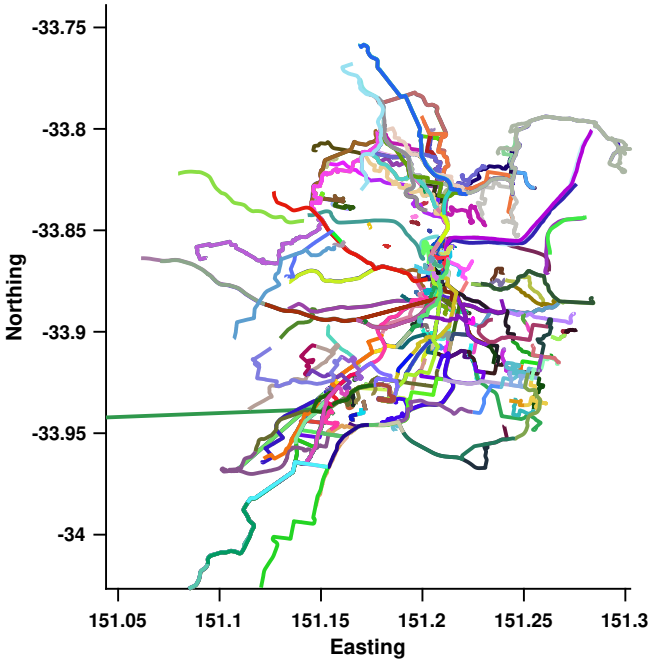

Fig. 3. GPS Tracks of image sequence collected around Sydney from Mapillary.

TABLE I

SUMMARY OF THE DATASETS USED IN THIS WORK

City	Adelaide	
	# Sequences	# Images
Support Database	44	24263
Validation Query	4	2141
Test Query	5	1481
	Sydney	
	# Sequences	# Images
Support Database	284	117860
Easy Query	5	1915
Hard Query	5	2285

in the support set in a 25m radius from them)[2]. The statistics for each dataset are summarized in table I.

B. Parameter definition

For all the results in the subsequent sections we use the same parameters, which were obtained using a grid search and keeping the best score on the Adelaide validation query. We use the Adelaide test query to ensure that the parameters are not overfitted to the validation query. Also note that we use the same parameters for all cities to further validate the fact that we do not need additional training/parameter search for each new city. The parameters are $\alpha = 0.25, \beta_1 = 0.75, \beta_2 = 0.0625, \beta_3 = 0.0625, k_{max} = 3, \gamma = 0.33, m = 2$.

C. Results

We test the graph smoothing effect in three different cases. First the extra data is available only for the support, second it is available only for the query and finally it is available in both cases. In each case we report two metrics, the median localization error over all the queries and the percentage of localizations that have less than 25m error.

First we perform the tests on the Adelaide dataset and present the results in table II. The graph smoothing operation was able to increase performance, even when applied only on the query database, and as expected, adding the graph smoothing during both query and support gave the best results. Recall that the parameters were defined based on the validation query, under the case where the extra data is available only for the support database.

TABLE II

RESULTS UNDER DIFFERENT GRAPH SMOOTHING CONDITIONS FOR THE MAPILIARY ADELAIDE DATASET. GS MEANS GRAPH SMOOTHING

Measure	None	GS Support	GS Query	GS S+Q
Validation				
acc $<$ 25m	66.84%	74.64%	70.06%	**79.03%**
median distance	8.76m	**7.29m**	13.02m	9.17m
Test				
acc $<$ 25m	44.63%	50.03%	46.39%	**51.32%**
median distance	110.66m	24.08m	41.84m	**22.81m**

Second we validate that the operation can be used on other cities and that we do not need to perform an additional

[2]The dataset and code for reproducing the results will be made public on acceptance of the work.

grid search for the new data. The results are presented in Table III. As expected the graph smoothing operation allowed us to get better performance in both median distance and accuracy, while using the parameters optimized for the Adelaide dataset. This is inline with our goal that is to have an operation that we do not have to retrain or re-validate parameters for a new dataset. We note that the performance of the hard query set is not inline with a good retrieval system (several kilometers from the correct point), but it is included to show that our method allows us to increase the performance both when the NetVLAD features are already very good for the task and when they are very bad.

TABLE III

RESULTS UNDER DIFFERENT GRAPH SMOOTHING CONDITIONS FOR THE MAPILIARY SYDNEY DATASET. GS MEANS GRAPH SMOOTHING

Measure	None	GS Support	GS Query	GS S+Q
Easy				
acc $<$ 25m	49.45%	56.16%	55.93%	**64.21%**
median distance	28.25m	13.48m	18.41m	**12.13m**
Hard				
acc $<$ 25m	13.87%	17.33%	16.67%	**24.07%**
median distance	4000km	3373m	3149m	**2151m**

D. Ablation studies

To verify that each part of the graph is important, we perform ablation studies using the Adelaide test query. The results are presented in Table IV. The table shows that different sources of information are important, with each one adding to increase in performance. Metric distance and sequence being the most important features and latent similarity being more of a complementary feature (this is expected, as it is being thresholded by the other two features). This is encouraging since in the absence of any other external information (GPS, etc), one can rely on the sequential nature of data collection to get a boost in localization performance. This information is readily available in a robotics setting.

TABLE IV

ABLATION STUDY ON THE MAPILIARY ADELAIDE TEST QUERY.

$\mathbf{W}_{\texttt{dist}}$	$\mathbf{W}_{\texttt{seq}}$	$\mathbf{W}_{\texttt{latent\_sim}}$	median distance	acc $<$ 25m
			110.66m	44.63%
X			46.10m	47.26%
	X		39.11m	47.53%
X		X	42.92m	47.60%
X	X		24.75m	50.03%
	X	X	37.39m	47.47%
X	X	X	**24.08m**	**50.30%**

In the next experiment, we demonstrate the effect of successive smoothing. This is achieved by applying smoothing operation m times. Theoretically, this should help increase the performance until it hits a ceiling and then it should start to slowly decrease (as it enforces connected examples of the database to be too similar to each other). The results are presented in Fig. 4. As can be seen, there is a clear pattern of increased performance until $m = 2$ after which the performance starts to degrade. It should be noted that even for $m = 10$ the graph smoothing operation still performs better than the baseline ($m = 0$).

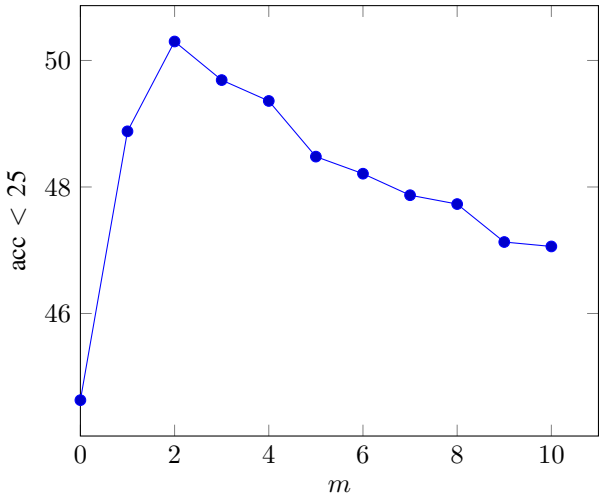

Fig. 4. Effect of the parameter m on the retrieval accuracy under 25m for the Adelaide test query.

V. CONCLUSION

This work showed that using techniques from Graph Signal Processing, the performance of indirect visual based localization can be improved by incorporating additional available information. This additional information acts on the latent representation by making it smoother on a graph designed using all available information, leading to a boost in localization. One of the encouraging observation of the work is that this additional information can take the form of a simple temporal relationship between surrounding images acquired in a sequence, and still lead to a significant increase in performance.

In future work, we would like to use the graph during the localization inference, to add temporal consistency to the position inference and also to train the smoothing operation in an end-to-end fashion.

REFERENCES

[1] N. Piasco, D. Sidibé, C. Demonceaux, and V. Gouet-Brunet, "A survey on visual-based localization: On the benefit of heterogeneous data," *Pattern Recognition*, vol. 74, pp. 90 – 109, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320317303448

[2] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz, "Geometry-aware learning of maps for camera localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2616–2625.

[3] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2938–2946.

[4] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5297–5307.

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

[6] D. Shuman, S. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 3, no. 30, pp. 83–98, 2013.

[7] E. Brachmann and C. Rother, "Learning less is more-6d camera localization via 3d surface regression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4654–4662.

[8] A.-D. Doan, Y. Latif, T.-J. Chin, Y. Liu, T.-T. Do, and I. Reid, "Scalable place recognition under appearance change for autonomous driving," in *International Conference on Computer Vision (ICCV)*, 2019.

[9] A. Torii, J. Sivic, and T. Pajdla, "Visual localization by linear combination of image descriptors," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE, 2011, pp. 102–109.

[10] S. Cao and N. Snavely, "Graph-based discriminative learning for location recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[11] L. Carlone, G. C. Calafiore, C. Tommolillo, and F. Dellaert, "Planar pose graph optimization: Duality, optimal solutions, and verification," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 545–565, 2016.

[12] N. Grelier, B. Pasdeloup, J. Vialatte, and V. Gripon, "Neighborhood-preserving translations on graphs," in *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Dec 2016, pp. 410–414.

[13] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," 2015.

[14] S. Segarra, A. G. Marques, and A. Ribeiro, "Optimal graph-filter design and applications to distributed linear network operators," *IEEE Transactions on Signal Processing*, vol. 65, no. 15, pp. 4117–4131, Aug 2017.

[15] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129 – 150, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1063520310000552

[16] M. Ménoret, N. Farrugia, B. Pasdeloup, and V. Gripon, "Evaluating graph signal processing for neuroimaging through classification and dimensionality reduction," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Nov 2017, pp. 618–622.

[17] M. Bontonou, C. Lassance, G. B. Hacene, V. Gripon, J. Tang, and J. Tang, "Introducing graph smoothness loss for training deep learning architectures," in *2019 IEEE Data Science Workshop (DSW)*, June 2019, pp. 160–164.

[18] R. Anirudh, J. J. Thiagarajan, R. Sridhar, and T. Bremer, "Influential sample selection: A graph signal processing approach," *arXiv preprint arXiv:1711.05407*, 2017.

[19] A. Torii, J. Sivic, M. Okutomi, and T. Pajdla, "Visual place recognition with repetitive structures," 2015.

[20] A. Torii, J. Sivic, T. Pajdla, and M. Okutomi, "Visual place recognition with repetitive structures," in *CVPR*, 2013.

[21] N. van Noord, "pytorch-netvlad," https://github.com/Nanne/pytorch-NetVlad, 2019.

[22] H. Jégou and O. Chum, "Negative evidences and co-occurences in image retrieval: The benefit of pca and whitening," in *Computer Vision – ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 774–787.