# Relocalization under Substantial Appearance Changes using Hashing

Olga Vysotska Cyrill Stachniss

***Abstract*— Localization under appearance changes is essential for robots during long-term operation. This paper investigates the problem of place recognition in environments that undergo dramatic visual changes. Our approach builds upon previous work on graph-based image sequence matching and extends it by incorporating a hashing-based image retrieval strategy in case of localization failures or the kidnapped robot problem. We present a variant of hashing algorithm that allows for fast retrieval for high-dimensional CNN features. Our experiments suggest that our algorithm can reliably recover from localization errors by globally relocalizing the robot. At the same time, our hashing-based candidate selection is substantially faster than state-of-the-art locality sensitive hashing.**

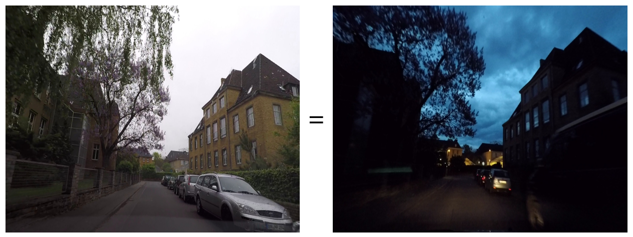

Fig. 1. Challenging image pairs for place recognition systems. Both images have been recorded at the same place but during different times resulting in strong appearance changes. The approach presented in this paper identifies such corresponding images via sequence information and can handle loops in the database sequences, recover from localization failures, as well as deal with the kidnapped robot problem.

## I. Introduction

The ability to localize itself is an essential capability for goal-directed robot navigation. A central ingredient of localization as well as mapping is the capability to identify that the robot is at a previously visited place, i.e., to make the data association between the current observation and a previously taken one. When operating in changing environments such as outdoor scenes, the localization system should be able to deal with substantial appearance changes. An example for such appearance changes is depicted in Fig. 1. Both images correspond to the observations taken at the same physical location but at different times during the day.

The task of localization through image matching for handling substantial changes in the appearance of a place has been tackled by several researchers [4], [5], [7], [14], [19], [21]. In line with previous work on this topic, we also rely on sequence information, i.e., exploit the fact the images are not obtained in a random order but according to the physical motion of the robot through the environment. We solve the problem by building a data association graph, where possible paths through this graph correspond to different image matching hypothesis. Efficient relocalization is achieved by a locality sensitive hashing strategy.

The main contribution of this work is a online approach for finding correspondences between the currently acquired image stream and a previously recorded image sequence even under strong appearance changes. Our work is an extension of our previous work as it uses the lazy search approach proposed in [21] but proposes several extensions. First, we provide a way for dealing with loops in the reference or database sequences and introduce new edges into the data association graph that is build up on the fly. Second, we provide an efficient way for relocalizing the robot in case it got lost. Both extensions naturally integrate with and extend [21] so that the same search approach and search heuristic can be re-used. This furthermore does not affect the online nature of the solution and the data association graph is still built incrementally.

We make the following three claims for our approach. It is able to (i) quickly relocalize the robot globally after getting lost and can handle the kidnapped robot problem, (ii) can be executed in an online fashion during navigation and requires only a small amount of image to image comparisons, and (iii) deal with loops in the reference images sequence while not relying on GPS information or similar means. These three claims are backed up through our experimental evaluation.

## II. Related Work

Localization is a relevant and frequently studied problem in robotics. A prominent approach to visual localization is FAB-MAP2 [5]. Dealing with substantial variations in the visual input, however, has been recognized as an obstacle for persistent autonomous navigation and one way to address it is to exploit sequence information for the image alignment [9], [13], [12], [14].

Over the past few years, different types of features have been investigated for place recognition. Some approaches use variants of HOG features such as [14] or Bag of Words models optimized to seasonal changes [16]. More recently, multiple researchers apply learned features as proposed by Sermanet *et al.* [17] and suggested for place recognition by Chen *et al.* [3]. These CNN features yield a high matching quality but are rather high-dimensional, i.e., comparisons are computationally expensive. This motivates the binarizations of such features and efficient comparisons using the Hamming distance [2]. Other alternatives are place-dependent features, which are optimized to the current location [11].

All authors are with the University of Bonn, Institute of Geodesy and Geoinformation, Bonn, Germany.

This work has partially been supported by the DFG under contract number FOR 1505 Mapping in Demand.

There is an increasing interest in the systems that can localize under appearance changes and different weather conditions. The experience-based navigation paradigm [4] stores multiple images or experiences for individual places and extends the place model whenever matching the current images to previous ones becomes challenging. Extension of experience-based navigation targets large-scale localization by exploiting a prioritized collection of relevant experiences so that the number of matches can be reduced [9]. SeqSLAM [13] aims at matching image sequences under strong seasonal changes and computes an image-by-image matching matrix that stores similarity scores between the images in a query and database sequence. It computes a straight-line path through the matching matrix and selects the path with the smallest sum of similarity scores across image pairs to determine the matching route. Milford *et al.* [12] present a comprehensive study about the SeqSLAM performance on low resolution images. Related to that, Naseer *et al.* [14] focus on offline sequence matching using a network flow approach and Vysotska *et al.* [21] extended this idea towards an online approach with lazy data association and build up a data association graph online on demand. Our paper extends [21] by allowing more flexible reference trajectories as well as efficient means for relocalization.

In visual place recognition, relocalization after getting lost can be achieved by comparing the query image to all images in the reference dataset as it was used by Neubert *et al.* [15]. To optimize the process of finding similar images in large datasets Gionis *et al.* [6] proposed using hashing algorithm, which was intensively used to solve text retrieval problems, to search for duplicates and even similar images in the large dataset, known as locality sensitive hashing (LSH). In LSH, slight variation in the image domain should only lead to slight variations in the hash. The disadvantage of LSH is that it relies on a quite large number hash tables ($> 100$ is suggested in practice) to obtain a high retrieval accuracy. To tackle this problem, Lv *et al.* [10] propose an efficient indexing strategy, which allowed to reduce the number of hash tables.

The popularity of the CNNs resulted in learned features, which are high-dimensional in comparison to those used by Lv *et al.* This slows down multi-probe LSH when matching full image sequences. An alternative approach to improve retrieval is spectral hashing [22], where a variant of spectral clustering is performed on the database before the operation in order to find better hash codes. Due to the high-dimensionality of CNN features, spectral clustering becomes computationally intractable and thus we rely on a variant of LSH proposed by Lv *et al.*

## III. Our Approach

### A. Lazy Data Association for Image Sequence Matching

The approach proposed in this paper is an extension of our previous approach for visual place recognition in changing environments [21]. The central idea of [21] is to perform visual place recognition by matching image sequences: Given a sequence of images, also called reference sequence, find for the stream of incoming images the visually most similar (sub-)sequence in the reference sequence. For making the required associations, the approach uses the ideas of lazy data association and solves the matching problem in an online fashion.

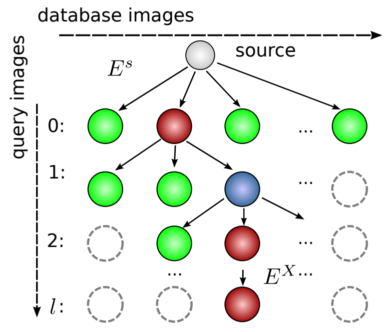

Fig. 2. Graph structure inherited from the [21]. Green circles denote expanded nodes (for which two feature vectors are compared); right circle - match; blue - non match, but support the path hypothesis.

We formulate the problem of matching image sequences as a graph search problem, in which every node represents the potential match between two images and edges encode possible transitions between nodes. The shortest path through this graph corresponds to the most likely data association between the image sequences. Fig. 2 depicts the structure of such a data association graph: green circle denotes the nodes for which the matching cost has been computed and blue (or red) nodes are those that belong to the found path. The blue nodes are so-called hidden nodes, the ones that support the path topology, but whose matching cost is higher than a specified parameter (non-matching cost), i.e. the image appear dissimilar. The red nodes are the ones for which corresponding matching cost is lower than the non-matching cost. The search ends (for each new image in the sequence) when the latest image gets associated to a reference image, either as hidden or real node.

This approach can accurately match image sequences but has also some limitations. First, it cannot relocalize well once lost. Second, it makes assumptions that query trajectory roughly follows the reference trajectory. In this paper, we overcome both limitations.

### B. Robust Image Matching Costs with CNN Features

To align image sequences, we need to match the individual images. Our approach represent each image by a single high-dimensional feature vector. In their extensive study, Chen *et al.* suggest that the $10^{\text{th}}$ layer of the convolutional neural network OverFeat [17] produces robust features for changing environments. The size of the output feature vector depends on the size of the input image. We opted for the smallest acceptable size of $450 \times 250$ pixels, which results in feature vector of approx. $200,000$ dimensions. Note, however, that our algorithm is not limited to this kind of features and we will plan to investigate alternative features such those from VGG-16 [18], Net-VLAD [1] and PoseNet [8] in our future work.

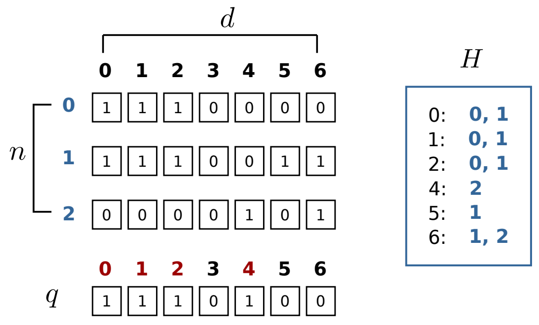

Fig. 3. Example of the proposed hashing algorithm. Here the dataset consists of 3 feature vectors of dimension 7 each. An entry of hash table $H[2]$ stores the IDs of the feature vectors 0 and 1, since for both of them, dimension 2 has the value of 1. For a query feature, the set of dimensions that take a value of 1 is $A = \{0, 1, 2, 4\}$. By collecting the values from $H$, the set of potential matching candidates is 0 with occurrence 3, 1 with occurrence 3 and 2 with occurrence 1. The resulting matching candidates for query $q$ are $\{0, 1\}$.

## C. *Efficient Relocalization Strategy*

No localization system is free of failures. Thus, it can happen that a robot gets lost, i.e, it cannot establish a correspondence between its current observations and the model anymore. In our case, this means that the (sub-)sequence of images, which the robot is currently acquiring, cannot be matched to the reference sequence anymore. A common reason for that in practice is the fact that the robot moves along a so far unseen trajectory, for example, when leaving the mapped area.

A good relocalization system should be able to detect whenever the robot reenters the previously mapped area in order to resume or restart the localization. To detect whether the robot is lost, we analyze the nodes of the best current matching hypothesis within the sliding window over time. If the percentage of the hidden nodes within this window exceeds 80%, i.e. only 20% of the images can be matched to the reference sequence, we consider the robot as lost. The size of the window depends on the framerate of the camera and potentially also on the speed of the robot[1].

A straightforward but computationally demanding way to find a reentry point is a brute force search through the whole reference database. Instead, we propose to use hashing for identifying potential reentry points. This results in comparing a query image only to the subset of the database images that are mapped to the same hash key. Hashing techniques are known to be robust and efficient to find image duplicates. In contrast to standard (cryptographic) hashing such as MD5 or SHA1, hashing for image retrieval is expected to assign similar features to the same or neighbouring buckets, i.e., to similar hash keys. This property is referred to as *locality sensitive*. Locality sensitive hashing (LSH) proposed in [6] was one of the first approaches to apply hashing for image retrieval problems.

We found that the use of an improved version of the LSH, called Multi-Probe LSH proposed by Lv *et al.* [10] is better suited for image matching tasks. Multi-Probe LSH builds on top of the LSH but specifies an intelligent strategy to probe specified buckets in multiple hash tables to get the higher probability of finding similar images. In our work, we use the Multi-Probe LSH in the following way: The moment the robot is considered *lost*, the algorithm starts to hash every incoming image $q_i$ and looks up for candidates $C(q_i)$, stored in the hash buckets, according to the probe strategy. More information about the probing strategy can be found in [10]. After the potential matching candidates are retrieved, we add the corresponding nodes to the graph

$$E^{\text{reentry}} = \{(x_{(i-1)j}, x_c)\}_{c \in C(q_i)} \tag{1}$$

where $i$ is the id of the current query image $q_i$, the term $x_{(i-1)j}$ corresponds to a node representing current best matching hypothesis, and $c$ refer to ids of the images in the reference dataset that were retrieved as matching candidates based on hashing.

Originally, Multi-Probe LSH was designed to match images that were taken under similar conditions and was used with relatively low dimensional features, e.g. 64 or 192 dimensions. In this work, however, we rely on high dimensional features (around 200K dimensions) as they show a better matching performance under changing conditions. This naturally leads to an increase in the querying time for computing the potential candidates from the database.

To tackle this issue, we propose an alternative hashing algorithm designed to explicitly take into account the high-dimensionality of the data and thus improve the querying time without compromising matching performance.

As in every hashing algorithm, the first step is to construct the hash table $H$. We start with binarizing the feature vectors that represent the images from the reference dataset. We inherit the binarization strategy for CNN features proposed by Arroy *et al.* [2]. To receive the binary values, we first scale the feature vector so that the dimension with largest spread lies in the interval $[0, 255]$ and afterwards all dimensions higher than a middle value 128 are assigned to 1, others 0. Formally:

$$f^{\text{int}} = (f^{\text{cnn}} - \min(f^{\text{cnn}})) \frac{255}{\max(f^{\text{cnn}}) - \min(f^{\text{cnn}})}, \tag{2}$$

$$f^{\text{bin}} = \begin{cases} 1, \text{if} f^{\text{int}} >= 128, \\ 0, \text{if} f^{\text{int}} < 128 \end{cases} \tag{3}$$

Let us assume we need to hash a dataset that consists with $N$ features indexed with $n \in [0, N]$ and each of the feature has $D$ dimensions indexed with $d \in [0, D]$. Then, every entry in the hash table $H[d]$ stores the set of indices of all the features that have a value of 1 in dimension $d$:

$$H[d] = \{n \mid f_n^{\text{bin}}[d] = 1\} \tag{4}$$

In a query phase, the incoming image $q$ gets also binarized using the same procedure as in Eq. (3). Afterwards, we extract a set of indices $A(q)$ of dimensions take the value of 1 for the image $q$:

$$A(q) = \{d \mid f_q^{\text{bin}}[d] = 1\}, \tag{5}$$

[1]In our experiments (using car in an urban environment), the size was set to 10 s.

where $|A| = M < D$ and typically $M \ll D$. We collect all the feature indices from the hash table, that take a value of 1 for the dimension stored in $A$

$$H[A] = \hat{\cup}_{a \in A} H[a], \tag{6}$$

where $\hat{\cup}$ denotes the set union preserving duplicates. We intentionally keep the duplicates in the set to further select those feature candidates that have high number of occurrences in the set $H[A]$. This represents the fact that the query feature $q$ and candidate features from the database share a substantial set of feature dimensions taking a value of 1. Thus, they are likely to represent the same place. For an illustration of the hashing procedure, consider the toy-example in Fig. 3.

## D. *Loopy Reference Sequences*

While recording the reference image sequence, it can happen that the robot moves along the same route multiple times. It may even be unavoidable given the topology of the environment. In practice, this situation occurs frequently when considering a typical urban mapping run using a car. In Fig. 4 (left), we provide a sketch of a trajectory that shows the situation in which reference trajectory visits the same place in the environment twice from 'B' to 'C'. The cost matrix for a corresponding real world situation is depicted in Fig. 13 (left) in the experimental section. In these matrices, bright pixels correspond to a pair of images that appear similar given the feature vectors, whereas dark pixels suggest a low similarity. In this case after visiting the place 'C', there are two possibilities to proceed, either visiting 'C-D' or 'C-F'. As the query trajectory follows the 'C-F' route, we can see a brighter pattern on the right lower part of the cost matrix in the Fig. 13 (left), whereas if the query trajectory followed the 'C-D' direction, the pattern would appear in a left lower part of the matrix. The previous version of our approach [21], cannot handle this situation flexibly, because the query sequence is expected to roughly follow the reference one.

In this paper, we extend the approach so that the search algorithm can flexibly "jump" between similar places in reference sequence. The ability to "jump" is established by creating the edges in the graph between the nodes that correspond to the similar places in the environment. If we know that image $a$ corresponds to image $b$ within the reference sequence, then whenever the algorithm is requested to expand the graph from the node $x_{ia}$, it will also expand from the node $x_{jb}$

$$E^{\text{sim}} = \{(x_{ia}, x_{(i+1)k})\}_{k=b-K,\ldots,b+K} \tag{7}$$

where $K$ - is a fan-out parameter that compensates for different robot speeds or camera frame rates. The same thoughts hold if the graph gets expanded from the node $x_{ib}$.

To be able to establish these nodes, we need to identify which places, i.e. images, in the reference dataset represent the same place. Since the evaluations are performed only on the reference dataset, finding of the similar place can be done offline using standard place recognition algorithm such as FABMAP2 [5] since the images stem from the the same appearance within the reference trajectory.

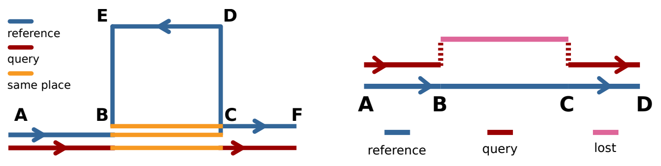

Fig. 4. Left: Sketch of similar places situation. Right: sketch of a query detour during which the robot is lost.

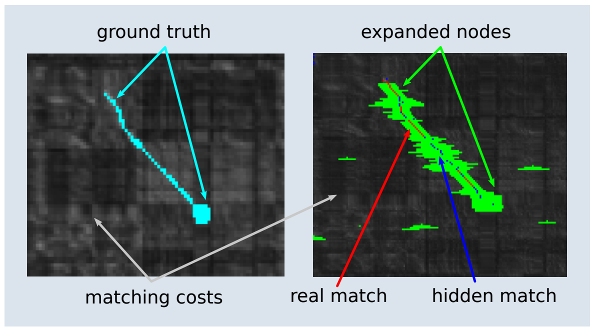

Fig. 5. Example of possible outputs in our experiments. The cost matrix stores the costs of matching individual images (not used in our algorithm). Expanded nodes - matching costs computed in our algorithm. Real matches - image pairs that represent the same place and hidden match - image pairs that support the path hypothesis, but have low matching cost. Ground truth matches that represent the same place in reality.

## IV. Experimental Evaluation

Our experiments are designed to show the capabilities of our method and to support our key claims, which are: Our approach is able to (i) quickly re-localize after a period while navigating without additional pose information, (ii) handle the kidnapped robot problem, (iii) be executed in an online fashion, and (iv) deal with loops in the reference images sequence. We furthermore provide comparisons to an existing methods such as [14], [13], [21]. We perform the evaluations on own datasets as well as on publicly available ones. To support the claims made in the beginning of the paper, we have selected the datasets that explicitly represent a particular challenge for localization. Some of them stem from the *Freiburg datasets* used in [20], [21] and others from the VPRICE Challenge dataset. Additionally, we collected a more challenging dataset in terms of trajectory shapes. We collected the data in Bonn with a car and a dashboard camera. The query as well as reference trajectory contains several revisits of the same places. In this paper, we use the datasets collected in the morning with slight rain and overcast as well as in the evening and very late evening on different days. Example images can be seen in Fig. 1.

Note, that throughout all the experiments the cost matrix was only computed for visualization purposes. The matching algorithm only computes the matching costs for the image pairs visualized in green. To enhance the visualization of the cost matrix for larger datasets, we also overlayed the ground truth results, see Fig. 5 for further notations.

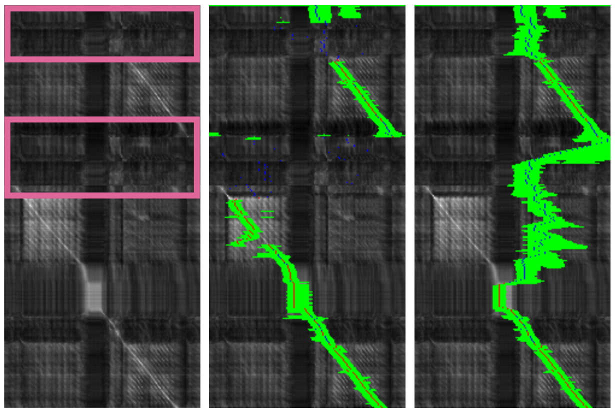

Fig. 6. Example of a cost matrix for matching two trajectories, where the query trajectory deviates from the reference one twice. Once at the beginning and then in the middle. The places are marked with pink rectangles. Left: matching matrix. Middle: result of proposed algorithm. Right: Result using previous approach. Note that the full cost matrix for matching is only shown for visualization and does not need to be computed by our approach.

### A. Matching Performance and Localization Recovery

The first set of experiments is designed to show that our approach is able to quickly relocalize after the system has identified that it cannot find matching images for a certain amount of time. This typically has the reason that the robot is navigating outside the mapped area or that the robot has been "kidnapped and teleported" to a different location the map. In our setup, kidnapping is equivalent to the situation that robot has deviated from a previously taken path and now moves on a different subsequence of the reference data.

The first experiment is designed to show how our approach can deal with situations, in which the robot is navigating outside the mapped area (reference sequence). This means that there are no corresponding images with respect to the reference sequence. An example for that can be seen in Fig. 6 (left) marked by the large rectangles. Fig. 6 (middle) illustrates that our approach localizes the robot in such a situation (as can be seen from the red (matched) and blue (unmatched) pixels. In contrast, the standard lazy DA approach [21] finds the matches only partially as is searches in the wrong area of the graph, see Fig. 6 (right).

We further tested our system on the publicly available VPRICE Challenge dataset to illustrate the handling of the kidnapped robot problem. We depict here the part that contains images where a person is moving with a hand-held camera during the day in the reference sequence and repeats the same path during the day and at night within the query sequence. Since the image sequences between the day and night runs are appended to each other, this corresponds to the kidnapped robot situation, i.e. the robot was teleported from the current location (end of the sequence) to another place (beginning of the sequence). As Fig. 7 (middle) suggests also in this case we can dramatically improve the matching quality with respect to our previous approach Fig. 7 (right). Furthermore, we evaluated our approach on a more challenging dataset that has multiple revisits of the same places in query as well as reference sequence, see Fig. 8. The left

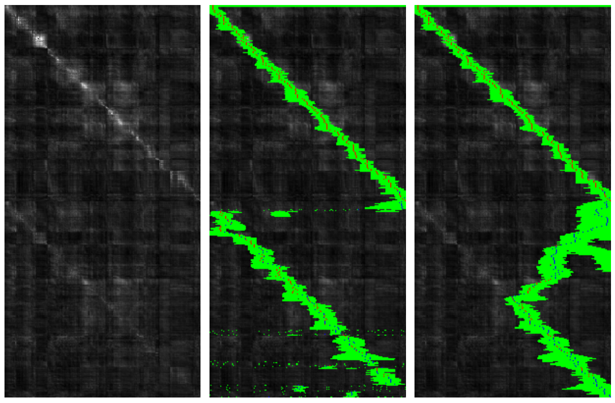

Fig. 7. Example of the trajectory matching from the VPRICE datasets. Here the query trajectory follows the reference trajectory twice, once during the day time (upper matrix part) and once during the night time (lower matrix part). Left: cost matrix. Middle: result of proposed algorithm. Right: Result using previous approach.

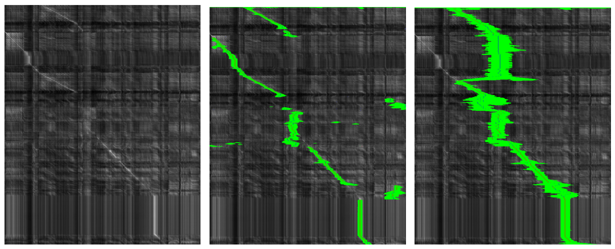

Fig. 8. Matching example from the Freiburg dataset. The trajectory contains partial revisits of the reference sequence as well as detours in the query sequence. Left: cost matrix. Middle: result of proposed algorithm. Right: Result using previous approach.

image of Fig. 9 depicts the precision-recall curve and as can be seen the quality of the result is also better than in our previous approach [21] (here labeled as "RAL'16") exactly due to the ability to detect loops. Fig. 9 (right) visualizes the results for a dataset with a query loop and relocalization part, as in Fig. 6 and it clearly outperforms the RAL'16 approach.

The next experiment is performed using more challenging shapes of trajectories, recorded in downtown Bonn. In Fig. 10 the query sequence was collected in the morning with a slight rain, whereas the reference in the late evening around a week later. The trajectories only partially overlapped, which results in broken bright patterns in the cost matrix. As can be seen the proposed approach (middle) is able to find the underlying pattern, i.e. find the matches, whereas our previous approach can only perform reliably within the continuous pattern. Fig. 11 shows the results for another pair of trajectories. The query sequence was collected in the early evening, whereas the reference in the late evening. As can be seen the proposed approach finds the underlying pattern as well as ignores the areas, where the query trajectory deviates from the reference one, thus no matching images and minimal expansion are expected. Due to inability of our previous method to handle the loops in the trajectories, without (GPS) priors, it performs poorly on this dataset.

To evaluate the performance of our approach in a more quantitative way, we computed precision recall as well as

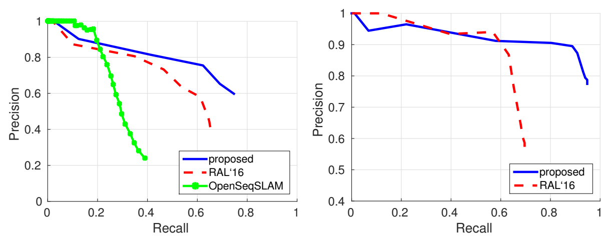

Fig. 9. Precision recall plots for the dataset with multiple loops in query in references sequences (left) and the dataset with a query connected through the similar places in the reference sequence (right).

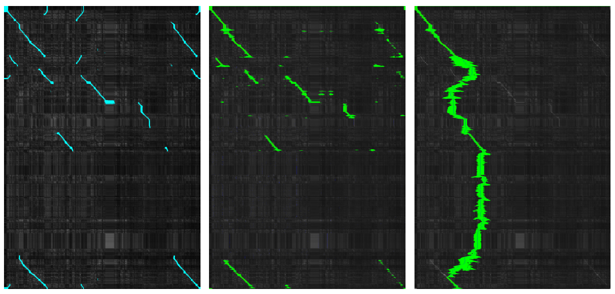

Fig. 10. Matching example of trajectories from Bonn in which both trajectories contain loops. The query also deviates for reference trajectory for a significant amount of time. Left: cost matrix with ground truth overlayed. Middle: proposed approach; Right: Result using previous approach.

F1-score statistics for the given datasets, see in the Tab. I. As it can be seen, by introducing the additional constraints and an efficient relocalization strategy, we are able to increase the number of found image matches (recall) with almost the same precision rate as in our previous paper. This naturally leads to an increase in accuracy in terms of F1-score.

## B. *Hashing comparison*

The second experiments is designed to show that the performance of the proposed relocalization strategy and the Multi-probe locality sensitive hashing is comparable. We also confirm that the proposed hashing algorithm runs faster for the data with very high dimensional features. For the Multi-probe hashing we use the OpenCv implementation. We select the following parameters for all of the experiments: number of tree = 1, key size = 10 and probe level = 2. From our experience selecting a higher number of trees or key size does not improve the performance of the algorithm, but, dramatically increases the computation time. Fig. 12 depicts only small deviations of the precision recall curves between the locality sensitive hashing (LSH) and the proposed dimension hashing (DH), which indicates that both hashing algorithm perform equally good on multiple datasets. On the other hand, the run time of the individual hashing strategies differ dramatically. For querying a set of candidates the LSH on average takes $120\,ms$, whereas DH retrieves the candidates in on average in $600\,\mu s$, which makes the candidates extraction time around 200 times faster. This speedup has a substantial impact on the overall timing.

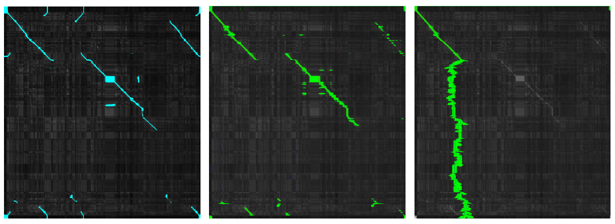

Fig. 11. Additional matching example from Bonn. Left: cost matrix with ground truth overlayed. Middle: proposed approach; Right: Result using previous approach.

TABLE I

QUALITATIVE COMPARISONS BETWEEN THE IMAGE SEQUENCE MATCHING APPROACH FROM [21] (RAL'16) AND PROPOSED METHOD. TRAJECTORIES OF VARIOUS SHAPES.

| | RAL'16 | | | Proposed | | |
|---|---|---|---|---|---|---|
| | exp | pr; re | F1 | exp | pr; re | F1 |
| 1 | 7,3% | 0.98; 0.35 | 0.51 | 6% | 0.95; 0.92 | 0.93 |
| 2 | 12,6% | 0.89; 0.63 | 0.74 | 11% | 0.89; 0.91 | 0.86 |
| 3 | 10,3% | 0.55; 0.64 | 0.59 | 4.2% | 0.7; 0.71 | 0.70 |
| 4 | 2.9% | 0.99; 0.31 | 0.48 | 1.5% | 0.95; 0.76 | 0.84 |
| 5 | 2.7% | 0.72; 0.35 | 0.46 | 1.8% | 0.8; 0.81 | 0.80 |

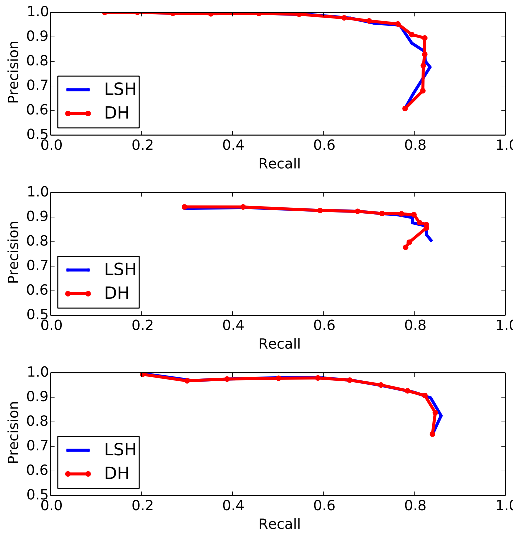

Fig. 12. Precision recall plots for different pairs of trajectories from Bonn dataset. LSH - results for locality sensitive hashing, DH - proposed hashing algorithm (dimension hashing).

Given the OverFeat feature vectors, we obtain the following timings. Processing a single image while being localized takes $2-3\,ms$. In contrast, processing a single image while being lost takes $30-80\,ms$ using our DH hashing and $150-200\,ms$ using LSH. Thus, our approach reduces the runtime for the relocalization by a factor of 2.5-5 in our experiments.

## C. *Loops in Reference Sequences*

The last experiment is designed to show that taking into account the similarity of the places in the reference sequence leads to better localization results. As it can be seen, in Fig. 13 (middle), our proposed approach finds the underlying pattern as oppose to the approach that does not take into account the notion about the place similarity Fig. 13 (right). Again this is an expected results and was the reason for

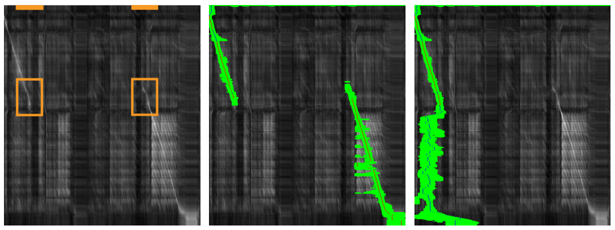
Fig. 13. Example of trajectories, where the reference sequence traverses the same place in the environment twice (marked with orange squares) and deviates in two difference direction upon exiting similar area. The query trajectory then also passes the "marked" area and follows one of the direction in the reference sequence. Middle: result of proposed algorithm. Right: Result using previous approach.

implementing this approach. Experiments with other datasets show similar results, but are omitted here in sake of brevity.

## V. Conclusion

In this paper, we presented a new approach for quickly finding correspondences between a currently observed image stream and a previously recorded image sequence under strong appearance changes. Related to [21], we build a data association graph incrementally and search for a data association sequence using an effective search heuristic. The work proposed here overcomes two key limitations of our previous method. First, we provide an efficient way for re-localizing the robot in situations, in which it got lost, after the robot has left the previously mapped areas and is reentering the known part of the environment or to solve the kidnapped robot problem. Second, our new approach can deal with loops in the reference sequences effectively without additional pose priors, like GPS. We implemented and evaluated our approach on different publicly available datasets. Our evaluations and comparisons show that we can handle the above mentioned situations, which could not been solved with the approach in [21]. We furthermore show through the experiments that the our approach runs online, provides an effective image matching, and supports all claims made in this paper.

## Acknowledgment

The authors would like to thank Nived Chebrolu for providing hardware support during the experiments and Andres Milioto for hacking the GoPro GPS interface for simplifying the evaluations.

## References

[1] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[2] R. Arroyo, P.F. Alcantarilla, L.M. Bergasa, and E. Romera. Fusion and binarization of cnn features for robust topological localization across seasons. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 4656–4663, 2016.

[3] Z. Chen, O. Lam, A. Jacobson, and M.Milford. Convolutional neural network-based place recognition. In *In Proc. of the Australasian Conf. on Robotics and Automation.*, 2014.

[4] W. Churchill and P. Newman. Experience-based navigation for long-term localisation. *Int. Journal of Robotics Research*, 32(14):1645–1661, sep 2013.

[5] M. Cummins and P. Newman. Highly scalable appearance-only SLAM - FAB-MAP 2.0. In *Proc. of Robotics: Science and Systems*, 2009.

[6] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

[7] A.J. Glover, W.P. Maddern, M. Milford, and G.F. Wyeth. FAB-MAP + RatSLAM: Appearance-based slam for multiple times of day. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3507–3512, 2010.

[8] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, 2015.

[9] C. Linegar, W. Churchill, and P. Newman. Work smart, not hard: Recalling relevant experiences for vast-scale but time-constrained localisation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

[10] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *VLDB*, pages 950–961, 2007.

[11] C. McManus, B. Upcroft, and P. Newman. Learning place-dependant features for long-term vision-based localisation. *Autonomous Robots*, 39(3):363–387, 2015.

[12] M. Milford. Vision-based place recognition: how low can you go? *Int. Journal of Robotics Research*, 32(7):766–789, 2013.

[13] M. Milford and G.F. Wyeth. Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.

[14] T. Naseer, L. Spinello, W. Burgard, and C. Stachniss. Robust visual robot localization across seasons using network flows. In *Proc. of the AAAI Conf. on Artificial Intelligence*, 2014.

[15] P. Neubert, S. Schubert, and P. Protzel. Exploiting intra database similarities for selection of place recognition candidates in changing environments. In *Proc. of the CVPR Workshop on Visual Place Recognition in Changing Environments*, 2015.

[16] P. Neubert, N. Sunderhauf, and P. Protzel. Appearance change prediction for long-term navigation across seasons. In *Proc. of the Europ. Conf. on Mobile Robots (ECMR)*, 2013.

[17] P. Sermanet, D. Eigen, Z. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *Intl. Conf. on Learning Representations (ICLR)*, 2014.

[18] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[19] E. Stumm, C. Mei, S. Lacroix, and M. Chli. Location graphs for visual place recognition. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

[20] O. Vysotska, T. Naseer, L. Spinello, W. Burgard, and C. Stachniss. Efficient and effective matching of image sequences under substantial appearance changes exploiting gps priors. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

[21] O. Vysotska and C. Stachniss. Lazy data association for image sequences matching under substantial appearance changes. *IEEE Robotics and Automation Letters (RA-L)*, 1(1):1–8, 2016.

[22] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009.