Efficient Panorama Database Indexing for Indoor Localization

Jean-Baptiste Boin
Stanford University
Stanford, CA, USA
jbboin@stanford.edu

Dmytro Bobkov
Technical University of Munich
Munich, Germany
dmytro.bobkov@tum.de

Eckehard Steinbach
Technical University of Munich
Munich, Germany
eckehard.steinbach@tum.de

Bernd Girod
Stanford University
Stanford, CA, USA
bgirod@stanford.edu

***Abstract*—We consider the task of indoor localization in large-scale environments using visual search on a database of geo-tagged panoramas. In this work we propose an efficient way to represent the database so as to maximize the search accuracy while minimizing the amount of computation required per query. The success of our method is due to a combination of (i) a hierarchical indexing method based on panorama image region information, and (ii) image descriptors aggregated from multiple views sampled finely over the panorama using generalized max pooling (GMP). Experiments on a large indoor dataset show that the complexity is reduced compared to common state-of-the-art retrieval methods such as FLANN (Fast Library for Approximate Nearest Neighbors): our scheme is more than twice as fast as an index based on FLANN while maintaining a similar retrieval performance.**

***Index Terms*—indoor localization, visual search, image retrieval, panoramic images, hierarchical search**

I. Introduction

Indoor positioning is important for applications such as autonomous navigation in robotics or augmented reality. Because other localization techniques such as GPS may fail indoors, there has been considerable interest in visual localization, which consists of localizing an image query in a given environment using only its visual information. The main approaches for this task either use 3D structure or 2D images.

Localization using 3D structure uses 3D geometry to perform 6-degrees-of-freedom (6DOF) pose estimation with high precision [1]–[5]. Local features extracted from the query are matched against a 3D point cloud built from a collection of images. Getting enough matches can be especially challenging for indoor scenes that contain large flat areas and repetitive structures, as was pointed out in [6]. Moreover, getting a good quality 3D reconstruction of the scene is not an easy task for some geometries [6] and scaling to large-scale 3D models is still a real challenge.

Localization using 2D images is illustrated for example in [7]–[9]. Using a database of localized images, the query pose is approximated by using the pose of the most similar image(s). By trading off localization accuracy for increased speed and robustness, these approaches present many advantages over 3D-based localization. Indeed, image retrieval can be made very efficient by using global descriptors that make images easy to match by visual similarity. These global descriptors can be built from sparse [10], [11] or dense [8] local descriptors, or be learned [7], [12], and are designed to be robust to a wide set of conditions (time of the day, lighting, etc.). The 2D-based solution is preferred for many applications that do not require a highly accurate pose and have to work with constrained computational resources, e.g., localization on a mobile phone. Even if 6DOF pose is desired, such a system is still very useful to limit the search space as it can easily be combined with a pose estimation step with the retrieved images.

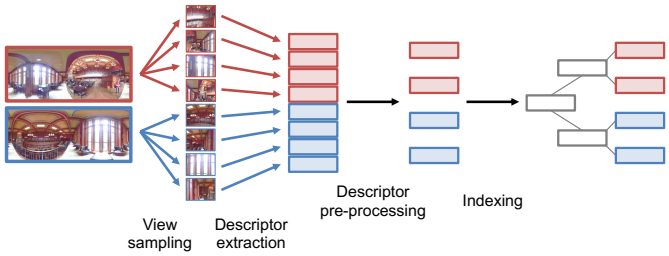

Fig. 1. Overview of the database-side processing of our localization system.

Our work is among these latter methods and focuses on databases of panoramic images. Some existing approaches like NetVLAD [7] design discriminative descriptors while other works, such as [8], improve performance by adding synthetic views to the database, echoing previous work by [13] on non-panorama images. The asymmetry between the omni-directional database images and the limited field-of-view queries is typically tackled by representing each panorama with the descriptors extracted from views sampled from that panorama, which may not be optimal. How to adequately represent a panorama gives rise to questions that have not been studied sufficiently and our work attempts to bridge this gap.

Given a set of panorama images captured densely in a set of buildings, we propose a way to build discriminative descriptors and index them for efficient retrieval. Similar in spirit to [14] that explored how to compactly represent a 3D object for retrieval, we show that performance is highly dependent on a good database representation. Our **contributions** are the following. (1) We speed up the search by introducing a hierarchical index based on the location and orientation of the views and show that it outperforms a comparable commonly used

978-1-7281-4673-7/19/$31.00 ©2019 IEEE

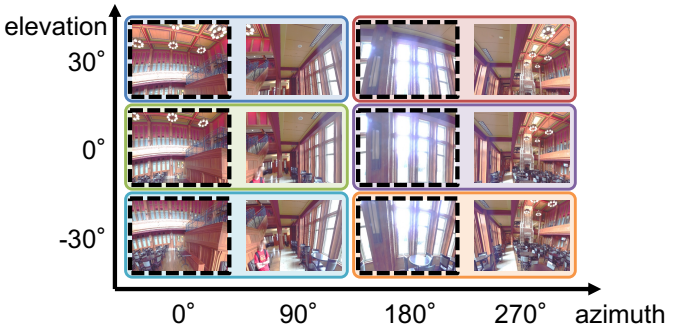

Fig. 2. View sampling for $N_H = 4$ and $N_V = 3$. The full database is represented as $(4, 3)$. The aggregated $(2, 3)$ database aggregates the views in each box, while the sub-sampled $(2, 3)$ database only keeps the views with a dashed outline while discarding the others. Both yield a database of 6 descriptors per panorama.

index based on descriptor statistics. (2) We perform systematic evaluation of view sampling and aggregation, showing that fine sampling of the panoramas followed by descriptor aggregation is preferred to coarse sampling, and that pooling descriptors using generalized max pooling (GMP) [15] is superior to mean pooling, thus confirming the use of GMP for aggregation in other works [16], [17]. The source code for this work is publicly available[1].

II. Methods

A. System design

Our system pipeline for the database-side processing is shown in Fig. 1. The retrieval method we use is based on high-dimensional global image descriptors. We extract limited field-of-view images from a panorama by sampling its viewing sphere with a regular pattern. The orientations of the views are obtained by regularly sampling N_V elevation angles in an interval $[-\alpha, +\alpha]$ and N_H azimuth angles in the interval $[0°, 360°)$. For each orientation, we then render a view using a fixed field-of-view and aspect ratio. This yields a total database comprising $N_P \cdot N_V \cdot N_H$ views, where N_P is the number of panoramas. The rationale behind sampling views instead of directly extracting descriptors from the panoramas is that we want to maximize the similarity between the descriptors in the database and the descriptors for potential queries. Since the queries are limited field-of-view images, it is logical to render limited field-of-view images from the panoramas, using typical camera parameters that we could expect to see in queries. For similar reasons, in practice we would typically use a high sampling rate to extract these views, meaning that neighboring views would have considerable overlap. This is also done to maximize the similarity between potential queries and the best matching database view.

We then compute one image descriptor for each view. Next we center and L2-normalize the set of descriptors. We observed that this step is important for high performance. The next stage is the descriptor pre-processing stage, the goal of which is to decrease the number of descriptors in our database. This stage will be described in the next sub-section. Finally, we store the pre-processed dataset into an index that allows for fast retrieval. All these steps are performed offline.

At query time, we extract the query descriptor, center it using the dataset mean, L2-normalize it and query our index. From this we get a ranked list of the database panoramas, which hopefully contains the spatially nearest panoramas at the top of the list.

B. Descriptor pre-processing

We consider two options: descriptor sub-sampling and descriptor aggregation. Both approaches are parameterized using two values, (n_h, n_v), corresponding to the number of descriptors we use in the horizontal and vertical direction respectively (we ensure that N_H and N_V are multiples of n_h and n_v respectively). For descriptor sub-sampling we only keep the view descriptors corresponding to an equally spaced subset of n_h azimuth angles and n_v elevation angles, and we discard the others. For descriptor aggregation we divide the azimuth angle and elevation angle intervals into n_h and n_v intervals and for each horizontal/vertical interval pair we pool the descriptors associated to these views into a single descriptor (Fig. 2). After this stage the database size is reduced to $N_P \cdot n_h \cdot n_v$ descriptors.

On the one hand, low values of n_h and n_v reduce both the size of the database and the redundancy among descriptors, which helps to accelerate search. On the other hand, too low values may decrease the similarity between a query and the most similar database descriptor, making the search task harder and decreasing the overall performance. By carefully choosing these values we can explore the trade-off between a fast or accurate search. Our experiments show that descriptor aggregation offers a much better trade-off than descriptor sub-sampling.

We aggregate the descriptors from multiple images into one with generalized max pooling (GMP) [15]. We consider the input descriptors $x_1, ..., x_n \in \mathbb{R}^d$ (where d is the dimension of the descriptors), stacked as columns of the matrix $X \in \mathbb{R}^{d \times n}$. We get the aggregated output with the formulation used in [17]:

$$\text{GMP}(X) = X(X^T X + \lambda I_n)^{-1} \mathbf{1}_n$$

where λ is a regularization parameter, commonly chosen as $\lambda = 1$. Unlike the original formulation [15] that requires inverting a $d \times d$ matrix, this only requires the inversion of a $n \times n$ matrix, which is beneficial because in general $n \ll d$. GMP was introduced so that the similarity between $\text{GMP}(X)$ and each x_i is close to a constant, which ensures that each input is well represented in the combined descriptor. We show in our experiments that GMP is usually a better choice than the more commonly used mean-pooling (MP).

C. Hierarchical indexing

We introduce two types of hierarchical indexes to store our descriptors. The first one (data-based hierarchy or DBH) is purely based on the statistics of the descriptors, while the second one (geometry-based hierarchy or GBH) is based on geometric characteristics of the views (orientation of the

[1]https://github.com/jbboin/panorama-indexing-localization

views, as well as position of the panoramas). We also compare the performance with linear indexes as our baseline.

1) Data-based hierarchy: This index is a k-means tree that is directly based on the FLANN (Fast Library for Approximate Nearest Neighbors) indexing approach [18], with some minor modifications. The tree is based on a branching factor B. The child nodes of the root are obtained by clustering the descriptors using k-means with B clusters. We then repeat the process recursively for each cluster until each cluster contains less than B descriptors. Each node of the tree corresponds to a cluster and is represented by a single descriptor that aggregates all descriptors in that cluster. The tree may not be balanced.

There are two modifications we add to the FLANN algorithm. Both modifications involve the way we compute the internal node descriptors from the set of input descriptors that are included in the corresponding cluster. First, we L2-normalize these internal node descriptors. This leads to a big difference in the performance of the index, especially because the input descriptors have unit norm. At retrieval time, when comparing a query descriptor to an internal node descriptor, our modified version of FLANN will thus compute the Euclidean distance between normalized descriptors. The second modification is that we aggregate the descriptors within a cluster by using GMP instead of MP.

2) Geometry-based hierarchy: The idea behind this tree structure is to hierarchically aggregate the descriptors according to their semantic content, which is correlated to the visibility information. This is done by aggregating descriptors with their neighbors in camera position space (not in descriptor space). This prevents from aggregating visually similar descriptors that could be in very different scene locations. As we get closer to the root, each level of the tree corresponds to an increasing amount of content aggregation.

In practice, we build the GBH tree by considering a sequence of aggregated databases: $(n_h^{(i)}, n_v^{(i)}), i = 0..L-1$, where we ensure that the values are such that $n_h^{(i)}$ (resp. $n_v^{(i)}$) divides $n_h^{(i+1)}$ (resp. $n_v^{(i+1)}$). The first level will contain all $N_P \cdot n_h^{(0)} \cdot n_v^{(0)}$ descriptors of the $(n_h^{(0)}, n_v^{(0)})$ database. Level $i+1$ is built from level i as follows. Because of the divisibility conditions, each descriptor from the $(n_h^{(i+1)}, n_v^{(i+1)})$ database can be associated to one and only one descriptor from the $(n_h^{(i)}, n_v^{(i)})$ database by looking at the subset of views it aggregates, and will become one of its children nodes in the tree. We note such a hierarchy as: $(n_h^{(0)}, n_v^{(0)}) > (n_h^{(1)}, n_v^{(1)}) > ... > (n_h^{(L-1)}, n_v^{(L-1)})$. An illustration is given in Fig. 3.

If we only perform intra-panorama aggregation as we just described, the first level of the hierarchy will grow linearly with N_P. This would lead to poor scalability. Hence we also perform inter-panorama aggregation over spatial units using extra semantic information from our dataset. An example of inter-panorama levels would aggregate the panoramas into rooms, then the rooms into buildings, etc. Fixing the hierarchy for this tree is akin to picking a branching factor in the DBH.

By exploiting additional information, the GBH is expected to have several advantages over the DBH. First, visually distinct parts of a scene that would appear in neighboring database views may be captured by the same query, so it should help to aggregate such views together, independently of their content. The GBH is also more balanced which entails that the panorama locations are more evenly distributed in the tree. Finally, the GBH is easy to edit. Adding or removing a building or a room consists solely of appending or removing a child sub-tree to the root node. This feature is essential for applications where GPS or other techniques are first used to narrow the search down to a few candidate buildings out of many.

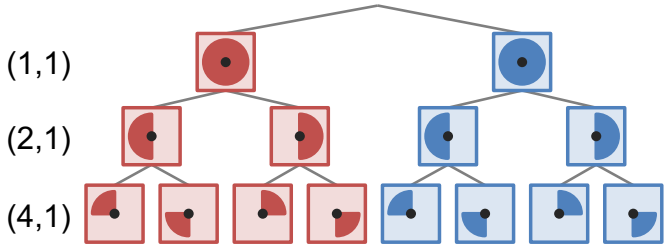

Fig. 3. GBH tree structure for the hierarchy $(1,1) > (2,1) > (4,1)$ with $N_P = 2$. For illustration purposes we only consider horizontal aggregation. Each square represents an aggregated descriptor and its associated circular sector represents which view descriptors are aggregated. Different colors correspond to different panoramas.

D. Index search

Once the index is built, we want to search it with a query descriptor in order to rank all the panoramas by likelihood of a match. The first part of the process is identical to the nearest-neighbor search used in FLANN: starting from the root, we compute the distance between the query and the descriptors representing each child node. We continue the tree traversal with the node that gets the lowest distance, while other nodes are put in a priority queue. That queue keeps track of the unexplored branches by increasing distance to the query, and is used when reaching the end of a branch to get the next one to explore. This is repeated until a fixed number of leaf nodes k is visited. The cost of the search is dominated by the distance computations between the query and each visited node descriptor. This traversal strategy ensures that each node is only explored once so at most (when k is the size of the database) the number of distance computations is equal to the number of nodes in the tree (subtract the root). Pushing the exploration to this extreme would obviously make the cost higher than a linear search given that there is some extra overhead due to the internal nodes. But in practice we usually reach a performance on par with linear search even with values of k that are considerably lower, which makes the whole search much faster. Adjusting k allows us to explore the trade-off between search cost and search performance.

After retrieving k database descriptors, we re-rank them by increasing distance. This is sufficient for FLANN because the goal is only to find the nearest neighbors to the query. In our case two extra steps are needed. First, we are retrieving panoramas, not database descriptors, so a conversion from a list of descriptors to a list of panoramas is required. Since the pre-processing stage never aggregates image descriptors across

distinct panoramas, each descriptor at the output of the pre-processing stage is only associated to one panorama, so there is a natural mapping. We remove duplicates by only keeping the first occurrence of each panorama in our list. Second, we want to rank all panoramas but if k is not large enough, we may not have visited them all and we may have missed relevant panoramas. We can complete the list with no extra distance computation cost by using the priority queue from the traversal and pulling nodes one by one. Each node can be associated to several panoramas that we can easily order by decreasing number of occurrences in the associated set of database descriptors. We then add the non-retrieved panoramas to our ranked list in order.

III. Experiments

A. Datasets and evaluation

Datasets: The WUSTL (Washington University in St. Louis) Indoor RGBD dataset [19] contains 129 geo-localized panoramas captured in a large building with a panorama scanner. It was extended with the InLoc dataset [20], which includes 329 realistic queries captured in the same building with a smartphone at a different time and months after the capture of the panoramas. InLoc also provides 6DOF manually verified poses for all queries.

To study the effects of retrieval on a large dataset, we extend the dataset with additional distractor panoramas. The Matterport3D dataset [21] contains panoramas captured with the Matterport Pro Camera in 90 different buildings and that are accurately localized in a reference coordinate system. Room segmentation was also performed, so that each panorama is associated with a room label, which is useful for the GBH index. Because of practical memory limitations, we only include 5 of those buildings, which corresponds to an extra 632 panoramas divided in 114 rooms.

The WUSTL dataset has the following issue: it contains two non-independent image sets, DUC1 and DUC2, that correspond to the first and second floor of the building, and that are not accurately aligned to one another. Each query is registered either in one or the other coordinate system (DUC1 contains 49 panoramas and 198 queries; DUC2 contains 80 panoramas and 131 queries). We bypass this problem by evaluating these query sets independently: the DUC1 (resp. DUC2) queries are evaluated on a dataset that combines the DUC1 (resp. DUC2) panoramas and the distractor panoramas. We then average our reported metrics across all queries.

Details on system design: The image descriptor we use is the 2048-dimensional descriptor from [12]. It is based on a convolutional neural network and achieves state-of-the-art results on image retrieval benchmarks. Because our approach is agnostic to the type of global descriptor, we use the pre-trained descriptor network as it is without fine-tuning it. Clearly, fine-tuning the network on a specific dataset would improve performance, but we are interested in a practical scenario where the network is often used as is in order not to limit the generalization property.

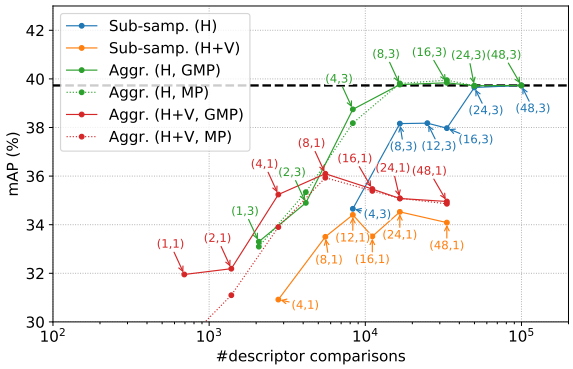

Fig. 4. Search performance (mAP) vs. number of descriptor comparisons for different aggregation and sampling modes using linear search. Condition H (resp. H+V) correspond to sub-sampling or aggregation on the horizontal direction only (resp. on both horizontal and vertical directions). The parameters (n_h, n_v) are shown for each point. We also compare aggregation with different pooling methods (MP, GMP). The black dashed line indicates the value of the baseline mAP.

For panorama sampling, we use $N_H = 48$ and $N_V = 3$, which gives 144 views per panorama. The views are rendered at a resolution of 640×480 px and with a focal length of 350 px. The horizontal (resp. vertical) field-of-view is $85°$ (resp. $69°$), which are typical values found in smartphone cameras.

Ground truth and evaluation: As is common for retrieval systems, we use mean average precision (mAP) as a performance metric. If the recognized location is simply the vantage point of the most similar panorama, the most natural criterion would be top-1 accuracy. But, as mentioned previously, our retrieval could be followed by a more complex pose estimation step that would process the most promising candidates only. Thus, even if the nearest panorama is not retrieved at the very top of the list, it is still useful. The mAP metric is a good proxy to evaluate the quality of the retrieved ranked list.

In order to perform the evaluation with mAP we first need to define the ground truth for query-panorama pair matches. We consider that matching panoramas are within a certain distance threshold t of a query. This criterion may give false positives by matching panoramas and queries located in different rooms. We avoid this by performing room segmentation using a system similar to [22]: the point cloud provided by the dataset is segmented into rooms based on the free space criterion utilizing potential field. We use the room correspondence as a second criterion for a match. In the end, the ground truth matches are defined as the panoramas that are in the same room as the query and within a distance $t = 10$m from it.

For retrieval speed quantification, we observe that the retrieval time is dominated by the distance computations, hence we can use the average number of distance computations (also the average number of descriptors from our index that we access per query) as a metric.

B. Descriptor pre-processing

We compare descriptor pre-processing conditions using exhaustive search (Fig. 4) for different values of (n_h, n_v). The mAP value reached without pre-processing ($(48, 3)$ database)

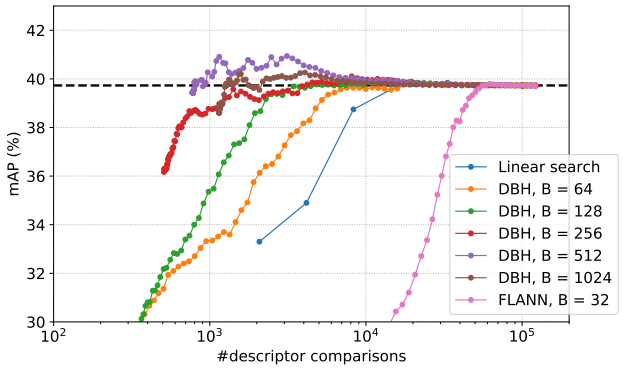

Fig. 5. Search performance of the DBH index for various values of the branching factor B as well as the performance of the FLANN index. The reference plot for linear search corresponds to the trade-off obtained with horizontal aggregation. The black dashed line indicates the value of the baseline mAP.

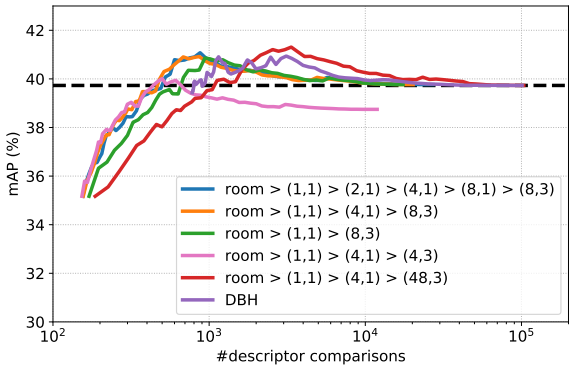

Fig. 6. Search performance of the GBH index for various hierarchies, as well as the best values obtained for the DBH index for reference. The black dashed line indicates the value of the baseline mAP.

is, as expected, among the best: 39.73%. This value will be called the baseline mAP. Our objective is to keep the mAP as close as possible to this value while minimizing the number of descriptor comparisons. Aggregation offers a better trade-off because the database can be considerably reduced without a significant mAP drop: the aggregated database $(8, 3)$ gives an mAP almost equal to the baseline mAP with a $6\times$ database size reduction. Meanwhile horizontal sub-sampling by any factor larger than 2 causes a noticeable mAP drop. Finally, it is interesting to point out that the best results are obtained when we only perform aggregation in the horizontal direction, although it comes at an extra cost in the number of descriptors.

Another observation is that both pooling methods perform similarly when few descriptors are aggregated (right part of the plot). However, GMP outperforms MP when many dissimilar descriptors are aggregated, for example for $(1, 1)$ when all descriptors from a panorama are aggregated. This insight is critical for our hierarchical indexes, because the quality of the search is highly dependent on the first levels of the hierarchy, where large amounts of aggregation occur.

We saw that for exhaustive linear search, the best performance trade-off is obtained when pre-processing the database with aggregation, using GMP as the aggregating method.

C. Evaluation of indexing methods

In this section we will now study the influence of the choice of indexing method on the performance trade-off.

1) DBH: Here we first evaluate the DBH index without any pre-processing, i.e. on the full $(48, 3)$ database, for various values of branching factor B. This is compared with the original FLANN implementation, also without any pre-processing. FLANN depends on the parameter B as well but in order not to overload the plot, we only report the results obtained with the best performing value of B. Finally, we also show for reference the best trade-off obtained in the previous section with linear search (i.e., aggregation on the horizontal direction only with GMP). The latter is the only one where pre-processing is performed. Results are shown in Fig. 5.

As we can see, the modifications to FLANN are critical for our task: the unmodified FLANN tree is even outperformed by linear search, but after modifications (DBH index) we obtain a performance trade-off that is much better. This justifies the use of a hierarchy. The best results are obtained for DBH with a branching factor of 512. Another interesting observation is that the DBH curves rise fast: a performance close to the baseline mAP is obtained with a small number of leaf nodes. The mAP is even higher than the baseline mAP in some cases, although, as expected, the mAP drops back to the baseline value when the number of leaf nodes explored increases.

Evaluation of the DBH index with pre-processing is not shown because we found that it did not help for that index.

2) GBH: For GBH evaluation, we consider a hierarchy with one inter-panorama level: room level aggregation. The distractor dataset already provides room segmentation. For the WUSTL dataset, we do not use the complex segmentation algorithm mentioned previously (we only use it to compute the ground truth matches). This is because it produces much larger rooms than the ones in the distractor dataset, and it requires 3D reconstruction of the building, which in general is not available. Instead, in order to keep the dataset consistent (rooms of similar size), we aggregate the panoramas by clustering their position with k-means, which is simpler. The number of rooms is chosen so that the average distance between a panorama and the associated centroid is the same as for the distractor dataset. This results in 25 rooms for DUC1 and 37 for DUC2. Using our insights from part III-B, at each level aggregation is performed with GMP instead of MP.

Fig. 6 compares hierarchies with different branching factors for the intra-panorama levels (first 3 curves) and different last levels (curves 2, 4 and 5), as well as the best DBH index as reference. The depth of the GBH tree can directly be obtained by counting the levels in the hierarchy. Two observations can be made from this comparison. First, comparing the performance with different last levels shows that the $(8, 3)$ aggregated database as the last level of the hierarchy gives the best trade-off. This confirms what we observed with the linear index: the $(8, 3)$ database is descriptive enough to keep the

TABLE I

LOWEST AVERAGE NUMBER OF DESCRIPTOR COMPARISONS PER QUERY REQUIRED TO GET WITHIN 0.5% OR 1% OF THE BASELINE MAP. THE VALUE IN RED CORRESPONDS TO THE SPEEDUP RATIO: THE FACTOR BY WHICH THE NUMBER OF DESCRIPTORS IS REDUCED COMPARED TO THE LINEAR SEARCH WITH SUB-SAMPLING BASELINE.

	Target mAP	
Method	38.73% (-1%)	39.23% (-0.5%)
GBH	**302** (165×)	**374** (133×)
DBH	778 (64×)	778 (64×)
Aggreg., linear search	8320 (6×)	16640 (3×)
Sub-samp., linear search	49921	49921

retrieval performance high, while still giving impressive speed gains. Using the $(48, 3)$ database as the last stage slows down the process because more redundant descriptors are explored, while using $(4, 3)$ gives a good performance early on but the mAP converges to a lower value. Second, the hierarchy is not too sensitive to the branching factor used for the intra-panorama stages. A value around 4 or 6, like in the hierarchy room $>$ $(1, 1)$ $>$ $(4, 1)$ $>$ $(8, 3)$ exhibits good performance.

Our results are summarized in Table I. Switching from linear search to a hierarchical index brings the most impressive gains, but we note that our GBH index outperforms a purely data-based approach: we get within 0.5% of the baseline mAP with only half the number of comparisons.

D. Other retrieval methods

Instead of reducing the number of descriptors compared per query, other techniques such as product quantization (PQ) [23] may lead to lower search complexity by approximating the distance computations. Here we justify that such an approach may not give optimal performance for this task.

PQ introduces two techniques: ADC and IVFADC. For ADC, the number of operations is proportional to Nm, where N is the number of descriptors in the database and m the number of quantizers. As a rough estimation, we consider that for GBH the cost of each descriptor comparison is d operations, the dimension of the descriptors, so our method reaches 0.5% of the baseline mAP with only 7.67e5 operations. This is to be compared with the values in Table II, obtained with the FAISS implementation of PQ [24]. The PQ approximation degrades our search performance too much to be acceptable. In order to get within 0.5% of the baseline mAP we need to use 1024 quantizers, which is almost as high as d, and this considerably decreases any possible speed benefit. IVFADC improves on ADC by only searching part of the index but it still relies on the same approximation as ADC so even though complexity will decrease, it will still be necessary to keep a high number of quantizers, which defeats the purpose of PQ. Although more work would be needed to explore how PQ could be modified for this task, we believe that at the moment our hierarchical search is among the most suitable techniques, but future work could explore how the ideas from PQ could benefit our index.

TABLE II

NUMBER OF OPERATIONS (#OP.) AND MAP OBTAINED WITH PQ FOR DIFFERENT NUMBERS OF QUANTIZERS m.

m	32	64	128	256	512	1024	2048
#op.	3.19e6	6.39e6	1.28e7	2.56e7	5.11e7	1.02e8	2.04e8
mAP	29.65	33.56	34.95	37.69	38.58	39.41	39.70

IV. CONCLUSIONS

In this work, we proposed an efficient representation of a database of indoor panoramas which can be used for image-based positioning in indoor environments. We represent each panorama by finely sampling views, extracting global descriptors and aggregating them into a smaller set of descriptors. These descriptors are then indexed by our novel structure (GBH) which uses additional information like view orientation and panorama location instead of descriptor statistics as is the case for other common methods. Internal nodes are represented using aggregation with GMP, which is shown to be superior to MP when dissimilar descriptors are aggregated. Experimental results on a large panorama dataset showed that hierarchical search with this index drastically accelerates the process while keeping the mAP unchanged: we could reach a performance similar to a brute-force method where descriptors from all sampled views are accessed (only a 0.5% mAP drop) by only comparing 374 descriptors per query on average, which corresponds to a 133-fold decrease in complexity. This is also better than another common hierarchical index approach based on FLANN (2-fold decrease in complexity).

By using only existing panorama locations that may not be uniform across the space, the index introduced in this work presents a disconnect between the intra-panorama and inter-panorama stages. The former have some regularity (same number of views per aggregated descriptor or per panorama) that the latter lack (aggregation at the "room" level is not guaranteed to involve a constant number of panoramas) and it could be interesting to see whether this affects the search performance. An approach like [8] could bridge this disconnect, by rendering synthetic panoramas regularly sampled on a grid that could then be aggregated in our index to make it more balanced. This could be the subject of future work.

REFERENCES

[1] T. Sattler, B. Leibe, and L. Kobbelt, "Efficient & effective prioritized matching for large-scale image-based localization," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2017.

[2] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua, "Worldwide pose estimation using 3D point clouds," in *ECCV*, 2012.

[3] Y. Li, N. Snavely, and D. P. Huttenlocher, "Location recognition using prioritized feature matching," in *ECCV*, 2010.

[4] L. Svärm, O. Enqvist, F. Kahl, and M. Oskarsson, "City-scale localization for cameras with known vertical direction," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2017.

[5] T. Sattler, B. Leibe, and L. Kobbelt, "Fast image-based localization using direct 2D-to-3D matching," in *ICCV*, 2011.

[6] F. Walch, C. Hazirbas, L. Leal-Taixe, T. Sattler, S. Hilsenbeck, and D. Cremers, "Image-based localization using lstms for structured feature correlation," in *ICCV*, 2017.

[7] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *CVPR*, 2016.

[8] A. Torii, R. Arandjelovic, J. Sivic, M. Okutomi, and T. Pajdla, "24/7 place recognition by view synthesis," in *CVPR*, 2015.

[9] T. Sattler, A. Torii, J. Sivic, M. Pollefeys, H. Taira, M. Okutomi, and T. Pajdla, "Are large-scale 3D models really necessary for accurate visual localization?," in *CVPR*, 2017.

[10] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *CVPR*, 2010.

[11] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2012.

[12] A. Gordo, J. Almazan, J. Revaud, and D. Larlus, "End-to-end learning of deep visual representations for image retrieval," *International Journal of Computer Vision*, 2017.

[13] R. Huitl, G. Schroth, S. Hilsenbeck, F. Schweiger, and E. Steinbach, "Virtual reference view generation for CBIR-based visual pose estimation," in *ACM Multimedia*, 2012.

[14] J.-B. Boin, A. Araujo, L. Ballan, and B. Girod, "Effective Fisher vector aggregation for 3D object retrieval," in *ICASSP*, 2017.

[15] N. Murray and F. Perronnin, "Generalized max pooling," in *CVPR*, 2014.

[16] R. Sicre and H. Jégou, "Memory vectors for particular object retrieval with multiple queries," in *ICMR*, 2015.

[17] A. Iscen, G. Tolias, Y. Avrithis, T. Furon, and O. Chum, "Panorama to panorama matching for location recognition," in *ICMR*, 2017.

[18] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration.," *VISAPP*, 2009.

[19] E. Wijmans and Y. Furukawa, "Exploiting 2D floorplan for building-scale panorama RGBD alignment," in *CVPR*, 2017.

[20] H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, and A. Torii, "InLoc: Indoor visual localization with dense matching and view synthesis," in *CVPR*, 2018.

[21] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3D: Learning from RGB-D data in indoor environments," *3DV*, 2017.

[22] D. Bobkov, M. Kiechle, S. Hilsenbeck, and E. Steinbach, "Room segmentation in 3D point clouds using anisotropic potential fields," in *ICME*, 2017.

[23] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2011.

[24] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *arXiv preprint arXiv:1702.08734*, 2017.