DisLocation: Scalable descriptor distinctiveness for location recognition

Relja Arandjelović and Andrew Zisserman

Department of Engineering Science, University of Oxford

Abstract. The objective of this paper is to improve large scale visual object retrieval for visual place recognition. Geo-localization based on a visual query is made difficult by plenty of non-distinctive features which commonly occur in imagery of urban environments, such as generic modern windows, doors, cars, trees, etc. The focus of this work is to adapt standard Hamming Embedding retrieval system to account for varying descriptor distinctiveness. To this end, we propose a novel method for efficiently estimating distinctiveness of all database descriptors, based on estimating local descriptor density everywhere in the descriptor space. In contrast to all competing methods, the (unsupervised) training time for our method (DisLoc) is linear in the number database descriptors and takes only a 100 seconds on a single CPU core for a 1 million image database. Furthermore, the added memory requirements are negligible (1%).

The method is evaluated on standard publicly available large-scale place recognition benchmarks containing street-view imagery of Pittsburgh and San Francisco. DisLoc is shown to outperform all baselines, while setting the new state-of-the-art on both benchmarks. The method is compatible with spatial reranking, which further improves recognition results.

Finally, we also demonstrate that 7% of the least distinctive features can be removed, therefore reducing storage requirements and improving retrieval speed, without any loss in place recognition accuracy.

1 Introduction

We consider the problem of visual place recognition, where the goal is to build a system which can geographically localize a query image, and do so in near real-time. Such a system is useful for geotagging personal photos [1], mobile augmented reality [2], robot localization [3], or to aid automatic 3D reconstruction [4].

A common approach is to cast place recognition as a visual object retrieval problem: the query image is used to visually search a large database of geotagged images [5], and highly ranked images are returned to the user as location suggestions [6, 7, 8]. Visual retrieval is usually conducted by extracting local descriptors, such as SIFT [9], quantizing them into visual words [10], and representing images as bag-of-visual-words (BoW) histograms. The BoW histograms are sparse because large visual vocabularies are commonly used [10, 11]. Retrieval is then performed by computing distances between sparse BoW histograms, which can be done efficiently by using an inverted index [10]. Many works have further improved on this core system by using larger vocabularies [11, 12], soft-assignment [13, 14], more accurate descriptor matching [15], enforcing geometric consistency [10, 11, 15], or learning better descriptors [16, 17]. Further improvements of retrieval systems specifically targeted at location recognition have been made, namely removal of confusing features [6], training of per-location classifiers [18, 19], and better handling of repetitive structures commonly found on façades of modern buildings [8].

In this work we focus on improving localization performance by exploiting distinctiveness of local descriptors – distinctive features should carry more weight than non-distinctive features. Automatically determining which features are distinctive or not should be quite helpful in location recognition, especially in urban environments where many features look alike (e.g. descriptors extracted from corners of generic modern office windows are all very similar). A traditional method for weighting features based on their distinctiveness is the inverse document frequency (idf) weighting [10] which down-weights frequently occurring visual words. However, this method operates purely on the visual word level, while recent retrieval methods imply that a finer-level descriptor matching is required for better retrieval accuracy [15, 20, 21, 22, 23, 24, 25]. Therefore, we investigate descriptor distinctiveness on a sub-visual-word level for which we extend the Hamming Embedding (HE) approach [15, 20] (reviewed in section 2).

Our method, DisLoc (section 3), uses local density of descriptor space as a measure of descriptor distinctiveness, i.e. descriptors which are in a densely populated region of the descriptor space are deemed to be less distinctive (figure 1). This approach is in line with the idf weighting [10] where frequent visual words are deemed to be less distinctive, but, unlike idf, our method estimates descriptor density on a finer level than visual words. Similar motivation is used in the second nearest neighbour test [9] where descriptor matches are rejected if the nearest descriptor to the query descriptor is not significantly closer than the second nearest. This test tends to be overly aggressive as two database descriptors can naturally be very similar due to depicting the same object; in this case the second nearest neighbour test would reject perfectly good matches. In contrast, our method is much softer in nature because matches are weighted based on the local density of descriptor space, which is estimated robustly such that a few repeated descriptors do not affect density estimates much.

1.1 Related work

Distinctiveness has been investigated in a supervised setting where [6] remove confusing features based on their geographical distribution. In [26], only repeatable visual words for a particular scene are kept. Classifiers can be learnt to automatically estimate visual word importance for every location [18, 19]. However, all four methods suffer from two major problems: (1) much like idf weighting,

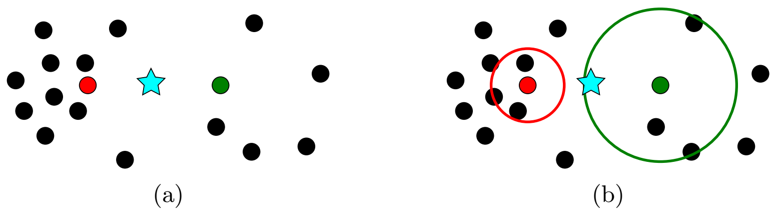


Fig. 1. Descriptor distinctiveness. Full circles represent database descriptors in the descriptor space. The red and green full circles are the closest database descriptors to the query (cyan star) and are equidistant from it. (a) The baseline Hamming Embedding method treats the green and red descriptors equally due to their equal distance from the query. (b) The red and green circles depict the local neighbourhood (radius is equal to the distance to the third nearest neighbour) for the red and green descriptors, respectively. DisLoc weights the green feature more because the relative distance to the query with respect to the neighbourhood radius is smaller for the green than for the red descriptor, implying that green is a better match.

they are limited to operate on the visual word level; and (2) they are not scalable enough. The four methods are impractical on a large scale as they require querying with each image in the database: [6] does this to discover confusing features, [18] for selecting negatives with hard-negative mining, while [19, 26] need it for constructing the image graph [27]. Even though this processing is only performed offline, it is still impractical as the computational complexity is quadratic in the number of database features. For a database with millions of images (such as the San Francisco landmarks dataset [7] used in this work), which contains billions of local descriptors, it is unreasonable to use a method with quadratic computational cost. For example, using each image from the San Francisco dataset (1M images, section 4.1) to query the dataset using the baseline retrieval method (Hamming Embedding, section 4.2) with fast spatial verification [11] (required for all four methods), takes 2.2 days on a single core (estimated from a random sample of 10k images). The quadratic nature of the approaches means that for a 10M database one would need a cluster with 100 computers to work for 2.2 days. In contrast, we propose a method which is *linear* in the number of database features, which takes only 100 CPU seconds (section 3.1) to compute for the 1M image San Francisco dataset. Furthermore, unlike [6, 18], our method is completely unsupervised.

Measures of local descriptor density have been used to improve retrieval, but were commonly applied at the image descriptor level [28, 29, 30] (e.g. density of BoW histograms is investigated) rather than at the level of local patch descriptors (e.g. SIFT); applying these methods directly onto patch descriptors is impossible as it would require a prohibitive amount of extra RAM. Furthermore, all three methods are also quadratic in nature and therefore not scalable enough. Finally, [21] exploit descriptor-space density on the patch descriptor level, but their method suffers from two problems: (1) it is, again, quadratic in nature; and (2) requires storing an extra floating point value per descriptor. For the example of the San Francisco dataset from which we extract 0.8 billion features, assuming single-precision floats (i.e. 4 bytes), [21] would require an extra 3.3 GB of RAM, which is a 31% increase over the baseline and our method. In contrast, our DisLoc method is scalable as the necessary offline preprocessing is linear in the number of database features, and only a fixed (i.e. does not increase with database size) and negligible amount of extra memory is required (103 MB in total).

2 Hamming embedding for object retrieval

In this section we provide a short overview of the Hamming Embedding [15] method for large scale object retrieval, which has been shown to outperform BoW-based methods [15, 20, 23, 24]. We will use it as our baseline (section 4.2) and, in section 3, extend it to incorporate our descriptor distinctiveness weighting.

We follow the notation and framework of Tolias et al. [24] which encapsulates many popular retrieval methods, including bag-of-words (BoW) [10], Hamming Embedding (HE) [15], burstiness normalization [20] and VLAD [31]. An image is described by a set $\mathcal{X} = \{x_1, \dots, x_n\}$ of n local descriptors. The k-means vector quantizer q maps a descriptor x_i into a visual word ID $q(x_i)$, such that $q(x_i) \in \mathcal{C}$, where $\mathcal{C} = \{c_1, \dots, c_k\}$ is the visual vocabulary of size k. Finally, $\mathcal{X}_c$ is a subset of descriptors in $\mathcal{X}$ assigned to the visual word c, i.e. $\mathcal{X}_c = \{x \in \mathcal{X} : q(x) = c\}$. The similarity K between two image representations $\mathcal{X}$ and $\mathcal{Y}$ is defined as:

$$K(\mathcal{X}, \mathcal{Y}) = \gamma(\mathcal{X})\gamma(\mathcal{Y}) \sum_{c \in \mathcal{C}} w_c M(\mathcal{X}_c, \mathcal{Y}_c) \tag{1}$$

where $\gamma(.)$ is a normalization factor, w_c is a constant which depends on visual word c, and M is a similarity defined between two sets of descriptors assigned to the same visual word. For the case of BoW and HE, w_c is typically chosen to be the square of the inverse document frequency (idf). The normalization factor is usually defined such that the self-similarity of an image is $K(\mathcal{X}, \mathcal{X}) = 1$. For Hamming Embedding retrieval [15, 20], a B-dimensional binary signature is stored for every database descriptor, in order to provide more accurate descriptor matching; the signature is constructed in a LSH-like [32] manner (for more details see [15]). The similarity function M takes the following form for the special case of Hamming Embedding [15] with burstiness normalization [20] (assuming $\mathcal{X}$ and $\mathcal{Y}$ are representations of the query and database images, respectively):

$$M(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} |\mathcal{Y}_c(x)|^{-1/2} \sum_{y \in \mathcal{Y}_c} f(h(b_x, b_y)) \tag{2}$$

where b_x and b_y are binary signatures of local descriptors x and y, h is the Hamming distance, f is a weighting function which associates weights for all possible values of the Hamming distance, and $|\mathcal{Y}_c(x)|$ is the number of elements in the set $\mathcal{Y}_c(x)$ of database descriptors that match with x:

$$\mathcal{Y}_c(x) = \{y \in \mathcal{Y}_c : f(h(b_x, b_y)) \neq 0\} \tag{3}$$

Finally, the weighting function f is defined as the truncated non-normalized Gaussian [20]:

$$f(h) = \begin{cases} e^{-h^2/\sigma^2} & , \ h \leq 1.5\sigma \\ 0 & , \ otherwise \end{cases} \tag{4}$$

where the Gaussian bandwidth parameter σ is typically chosen to be one quarter of the number of bits B used for the binary signatures [20, 33] (e.g. a common setting is $B = 64$ and $\sigma = 16$).

Discussion. Here we explain, in less formal terms, the intuition behind mathematical definitions presented in this section (which were adapted from [24]). Equation (1) simply decomposes the image similarity across different visual words, which enables efficient computation of the similarity between the query and all database images by employing an inverted index. It also accounts for inverse document frequency weighting (w_c), and normalizes the scores in order to not bias the similarity towards images with a large number of descriptors (e.g. for the BoW case, equation (1) reduces to cosine similarity between tf-idf weighted BoW vectors). The binary signatures b_x and b_y help perform precise matching between the two descriptors by rejecting some false matches that a pure BoW system would accept, at a cost of increased storage (to store the signatures) and processing (to compute Hamming distances) requirements. This is done by thresholding the Hamming distance in equation (4), where descriptor matches whose Hamming distances are larger than 1.5σ are discarded, while others are given increasing weights for decreasing distances. For comparison, a BoW system would simply correspond to $f(h) = 1$ for all h. Finally, the visual burstiness effect is countered by the burstiness normalization [20] in equation (2).

3 Scalable descriptor distinctiveness

This section proposes a method for determining descriptor distinctiveness and incorporating it into the standard retrieval framework presented in the previous section. Apart from improving retrieval performance, there are two main requirements: (1) the method must not have quadratic computational complexity in order to be scalable to databases containing millions of images, and (2) storage requirements should not increase drastically, i.e. no additional information should be kept on a per-descriptor basis but only a fixed amount (i.e. not dependant on the database size) of additional RAM can be justified. Both of these requirements distinguish our work from previous works, a review of which is given in section 1.1.

The key idea of our method is to estimate the local density of the descriptor space around each database descriptor, and weight descriptors depending on their distinctiveness which is inverse to the local density; we call it Local Distinctiveness (DisLoc). Figure 1 illustrates this point: given two database features (red and green) equally distant from the query (star), it is clear that the green one is more likely to be a correct match than the red one. This is because the red descriptor is surrounded by many other descriptors (i.e. descriptor density is large, distinctiveness is small) and is therefore not distinctive, and, for example, the second nearest neighbour test of [9] would reject the match. On the other hand, the green descriptor is quite distinctive as there is a small concentration of other descriptors around it, so it might be a correct match for the query. The Hamming Embedding retrieval system (reviewed in section 2), would add the same weight, $f(h)$ (equation (4)), to the images from which the red and green features came from, because h is the same for both of them. We therefore modify the weighting function to adjust the Gaussian bandwidth based on descriptor distinctiveness, i.e. we propose to make σ (equation (4)) a function of the database descriptor. Increasing σ allows for larger Hamming distances to be tolerated when deciding if two descriptors match, while reducing it increases the selectivity; figure 1b illustrates this. More formally, we modify the definition of M and f (equation (2) and (4), respectively) to incorporate σ being a function of the local descriptor, i.e. its visual word c and binary signature b (recall that $\mathcal{X}$ and $\mathcal{Y}$ are the representations of the query and database images, respectively):

$$M(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} |\mathcal{Y}_c(x)|^{-1/2} \sum_{y \in \mathcal{Y}_c} f(h(b_x, b_y), c, b_y) \tag{5}$$

$$f(h, c, b_y) = \begin{cases} e^{-h^2/\sigma(c,b_y)^2} & , \ h \leq 1.5\sigma(c, b_y) \\ 0 & , \ otherwise \end{cases} \tag{6}$$

3.1 Scalable estimation of $\sigma(c, b)$

The key remaining problem is how to robustly estimate $\sigma(c, b)$ for all database descriptors, and obey our two design goals: do not incur quadratic computational cost, nor store extra information on a per-descriptor basis (e.g. one could be tempted to store $\sigma(c, b)$ for every descriptor) thus requiring much more RAM which is usually the limiting factor for any large scale retrieval system.

We propose to precompute and store $\sigma(c, b)$ for all possible values of the visual word c and binary signature b. However, this is impractical – there are 2^B possible B-dimensional binary signatures, and for reasonably sized signatures (e.g. typically $B = 64$) there are too many combinations to compute and store. Therefore, we propose a small approximation – the B-dimensional binary signature b is divided into m blocks where each of the blocks is $l = \frac{B}{m}$ dimensional, so that b is a concatenation of $b^{(1)}, b^{(2)}, \ldots, b^{(m)}$. The blocks represent subspaces of the full descriptor space and we assume, akin to Product Quantization [34], that the subspaces are relatively independent of each other, i.e. if a descriptor is distinctive, it is also likely that it is distinctive in many of the m subspaces. Then, $\sigma(c, b)$ is approximated as the sum of σ's in the individual subspaces: $\sigma(c, b) = \sum_{i=1}^{m} \sigma_i(c, b^{(i)})$. Splitting the large B-dimensional binary vector into several parts makes our problem manageable due to dealing with smaller dimensional binary signatures – for each visual word c, instead of storing a table of $\sigma(c, b)$ values which has 2^B entries, we store m tables $\sigma_i(c, b^{(i)})$ with $2^l = 2^{B/m}$ values each. For a typical setting where $B = 64$, $m = 8$ and therefore

$l = 64/8 = 8$ bits, the number of stored and computed elements decreases from $2^{64} = 1.8 \times 10^{19}$ to $8 \times 2^8 = 2048$.

The problem now becomes: for all visual words c and all signature blocks (i.e. subspaces) s, compute and store $\sigma_s(c, b^{(s)})$ for all values of $b^{(s)}$. As discussed earlier, we propose to make $\sigma_s(c, b^{(s)})$ proportional to the binary signature distinctiveness, which is inversely proportional to the local descriptor density, and therefore proportional to the local neighbourhood size. The local neighbourhood for a given signature can be estimated as the minimal hypersphere which contains its p neighbours, the radius of this hypersphere is equal to the distance to the p-th nearest neighbour. This strategy is illustrated in figure 1b ($p = 3$), where the local neighbourhood is automatically estimated – the red descriptor's neighbourhood is smaller (i.e. descriptor density is larger, so it is less distinctive) than the green one's. Other measures of local neighbourhood size exist as well, such as a softer approach of [35], but we found the overall place recognition performance of our method to be robust to various neighbourhood size definitions.

We simply make $\sigma_s(c, b^{(s)})$ equal to the local neighbourhood radius, i.e. the Hamming distance to the p-th nearest neighbour of signature $b^{(s)}$ in visual word c. The p parameter is set automatically as the average number of neighbours across all database descriptors (in the same visual word c and subspace s) which are closer than the default value of σ_{def}/m, where σ_{def} is defined as the value from section 2, i.e. $\sigma_{def} = B/4$ [20, 33]. In other words, if all descriptors are uniformly distributed in the descriptor space, the estimated $\sigma(c, b)$ would be identical for all b and equal to the default σ_{def} of the baseline Hamming Embedding method (section 2).

Implementation details: $\sigma_s(c, b^{(s)})$ computation. It is simple and fast to compute the distance to the p-th nearest neighbour for all possible values of $b^{(s)}$ (remember that $b^{(s)}$ is l-dimensional, and l is small, with $l = 8$ there are only 256 different values of $b^{(s)}$). For this purpose we define a lookup table $t_s(c, b^{(s)})$ which stores the number of descriptors quantized to visual word c which have the binary signature $b^{(s)}$ in subspace s. This table can be populated with a single pass through the inverted index (i.e. the computational complexity is by definition $O(n)$, where n is the number of descriptors in the database). To compute the distance to the p-th nearest neighbour for a particular value $b_i^{(s)}$, one can simply use a brute force approach: go through the list of binary signatures $b_j^{(s)}$ in the non-decreasing order of hamming distance $h(b_i^{(s)}, b_j^{(s)})$ and accumulate $t_s(c, b_j^{(s)})$ along the way. The traversal is terminated once the accumulated number reaches p, signifying that $h(b_i^{(s)}, b_j^{(s)})$ is the distance of the p-th nearest neighbour.

Implementation details: normalization. We make sure that on average $\sigma(c, b)$ is the same as the default σ_{def} so that no bias is introduced, such as consistently under/overestimating $\sigma(c, b)$ which by coincidence might work better for a particular benchmark. We therefore normalize $\sigma(c, b)$ by subtracting the mean over all $\mathcal{X}_c$ and adding σ_{def}. Therefore, the final estimate of $\sigma(c, b)$ is computed as $\sigma_{final}(c, b) = v_c + \sigma(c, b) = v_c + \sum_{i=1}^{m} \sigma_i(c, b^{(i)})$, where $v_c = \sigma_{def} -$

$mean_{x \in \mathcal{X}_c}(\sigma(c, b_x))$; it is clear that this ensures that $mean_{x \in \mathcal{X}_c}(\sigma_{final}(c, b_x)) = \sigma_{def}$. In order to be able to conduct the normalization at run-time, a single extra floating point number, v_c, needs to be stored for every visual word c.

Computational speed. As mentioned earlier, the table $t_s(c, b_i^{(s)})$ can be populated with a single pass over all database descriptors, which is $O(n)$, where n is their count. The brute force search for the p-th nearest neighbour distance is $O(2^l \times 2^l) = O(2^{2l})$ so this part of the algorithm is independent of the database size as l is a constant.

On a single core (i5 3.30 GHz), the entire computation takes only 100 seconds for the San Francisco dataset (section 4.1) which contains 1M images and 0.8 billion local features. Furthermore, even though there is no real need for speeding it up as 100 seconds for a one off preprocessing task is very efficient, the algorithm is easily parallelizable as the computations are performed completely independently for all visual words c and subspaces s.

Storage requirements. For every visual word c, at runtime one needs to have access to v_c (a single precision floating point number, 4 bytes) and m (typically equal to 8) lookup tables $\sigma_s(c, b^{(s)})$, which contain 2^l (typically equal to $2^8 = 256$) values. The $\sigma_s(c, b^{(s)})$ values can only take integer numbers from 0 to l as they are the only possible Hamming distances for l-length binary signatures. However, for our parameter settings we observe that all obtained values are between 1 and 4, therefore only 2 bits are needed to encode them. The total number of bits for storing all necessary information, for visual vocabulary size k, is therefore: $k \times (32 + 2 \times m \times 2^l)$, which for our parameter settings $k = 200k$, $m = 8$, $l = 8$ equals 103 MB. Note that no information is stored on a per-feature basis, i.e. for any size dataset, comprising of potentially millions of images, one only requires 103 MB of extra storage, which is negligible. On the other hand, the method of [21] requires storing a floating point number for every database feature, which for 0.8 billion features of the San Francisco dataset (section 4.1) would require extra 3.3 GB of RAM. This is a 31% increase in baseline's storage needs; in contrast, our method increases storage requirements by less than 1%.

3.2 Removal of unhelpful features

Up to this point we have shown how to compute distinctiveness of every descriptor in the database – larger $\sigma(c, b)$ corresponds to larger distinctiveness. We note that very non-distinctive features are not useful for retrieval or place recognition, as (1) they don't convey much information, and (2) it is unlikely that query features will match to them as the adapted Gaussian bandwidth $\sigma(c, b)$ is quite tight. Therefore, we propose to investigate removing non-distinctive descriptors, namely, to remove all descriptors from the database whose $\sigma(c, b)$ is below a certain threshold. Experimental results (section 4.3) indeed show that 7% of features can be removed safely without any degradation in place recognition performance, while 24% of features can be removed in exchange for a small reduction in recognition performance. The removal of features directly translates to reduced storage and RAM requirements, as well as place recognition speedup.

4 Experimental setup and recognition results

4.1 Datasets and evaluation procedure

Location recognition performance is evaluated on two standard large scale datasets containing street-view images of Pittsburgh [8] and San Francisco [7].

Pittsburgh [8]. The dataset contains 254k perspective images generated from 10.6k Google Street View panoramas of Pittsburgh downloaded from the internet. There are 24k query images generated from 1k panoramas taken from an independent dataset, Google Pittsburgh Research Dataset, which has been created at a different time. We follow the evaluation protocol of [8] where ground truth is generated automatically by using the provided GPS coordinates; a database image is deemed as a positive if it is within 25 meters from the query image. It should be noted that a perfect location recognition score is unachievable as some queries (1.2%) do not have any positives (as all database images are further away than 25 meters), some positives are not within 25 meters due to GPS inaccuracies ([8] reports GPS accuracy to be between 7 and 15 meters), and the construction of the ground truth does not take into account occlusions which can occur due to large camera displacements.

San Francisco landmarks [7]. The dataset contains 1.06 million perspective images generated from 150k panoramas of San Francisco, while the query set contains 803 images taken at different times using mobile phones. Ground truth is provided in terms of building IDs which appear in the query and database images; as in [7, 8], positives for a particular query are all database images which contain a query building.

There are two versions of the ground truth provided by the database authors – the original April 2011 version used by [7, 8], and an updated April 2014 version which contains fixes but has not been used in a paper yet due to its recency. Unless otherwise stated, we report results on the latest ground truth version (April 2014), while when comparing to previous methods [7, 8] we use the same version as them (the first version from April 2011) in order to be fair.

Finally, it should be noted that there are still some problems with the ground truth – 6.5% queries do not contain any positives making the maximal obtainable location recognition score to be 93.5%. Furthermore, we have encountered a few more ground truth errors, such as cases where the side of the building imaged in a query is not visible in any of the database images of the same building.

Evaluation measure. Localization performance is evaluated in the same way for the two datasets, as defined by their respective authors [7, 8], as recall at N retrievals. Namely, a query is deemed to be correctly localized if at least one positive image is retrieved within the top N positions. We use two types of graphs to visualize the performance: (1) *recall@N*: recall as a function of the number of top N retrievals; and (2) *rank-gain@N*: relative decrease in the number of required top retrievals such that the recall is the same as the baseline method at N retrievals. An example of a *rank-gain@N* curve is figure 2d, where our method, DisLoc, achieves 33.3% at $N = 30$, which means that 33.3% less retrievals were needed for DisLoc to achieve the same recall as the baseline achieves at $N = 30$ (i.e. DisLoc only needs to return 20 images).

4.2 Baseline

We have implemented a baseline retrieval system based on the Hamming Embedding [15] with burstiness normalization [20], details of which are discussed in section 2. We extract upright RootSIFT [36] descriptors from Hessian-Affine interest points [37], and quantize them into 200k visual words. To alleviate quantization errors, multiple assignment [13] to five nearest visual words is performed, but in order not to increase memory requirements this is done on query features only, as in [14]. A 64-bit Hamming Embedding (HE) [15] signature is stored together with each feature in order to improve feature matching precision. The visual vocabulary and Hamming Embedding parameters are all trained on a random subsample of features from the respective datasets.

As shown in figure 4, our baseline (HE) already sets the state of the art on both datasets, and by a large margin. For example, on the Pittsburgh benchmark at $N = 10$ the baseline gets 77.3% while the best previous result (Adaptive weights [8]) achieves 61.5%. The superior performance of the baseline can be explained by the fact that Hamming Embedding with burstiness normalization has been shown to outperform bag-of-words methods due to increased feature matching precision [15, 20, 23, 33].

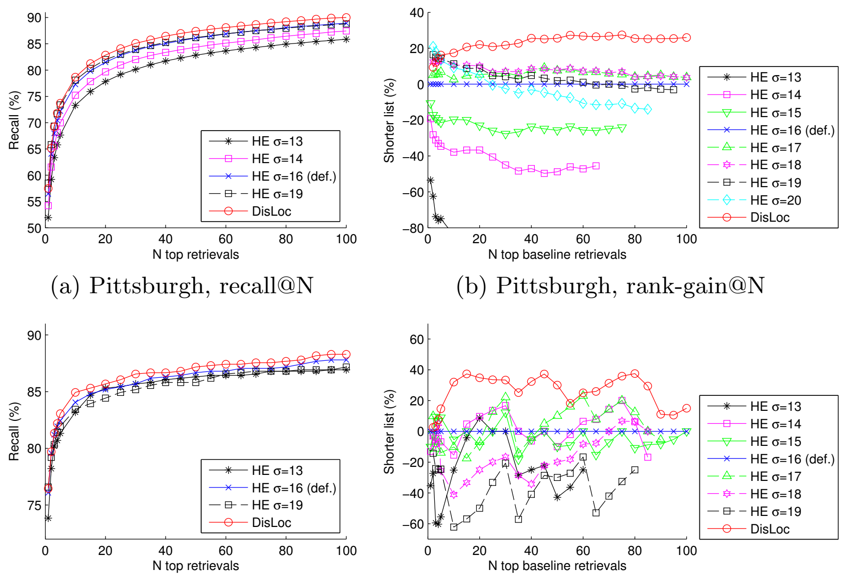
4.3 Results

Unless otherwise stated, none of the following experiments performs spatial reranking [11] as our aim is to improve the performance of the core retrieval system. Spatial reranking or any other postprocessing technique [29, 33, 38, 39] can be applied on top of our method, and we show results for spatial reranking later in section 4.4.

Note that the Pittsburgh benchmark contains many more query images than San Francisco (24k compared to 803), which explains why the Pittsburgh performance graphs are smoother and differences between methods are easier to see (large number of query images implies performance differences are statistically significant).

Figure 2 shows the performance of our DisLoc method compared to the Hamming Embedding baseline. DisLoc clearly outperforms the baseline on both benchmarks and at all sizes of retrieved lists. For example, for the San Francisco dataset DisLoc achieves a rank-gain of 37.5% at $N = 80$, namely DisLoc only needs to retrieve 50 images in order to achieve the same recall (87.2%) as the baseline obtains with 80 retrievals. This directly corresponds to a more user-friendly system as a much shorter list of suggestions has to be shown to a user in order to achieve the same success rate. Rank-gain is consistently larger than 20% for all N on both benchmarks.

We also investigate if the baseline's performance can be improved by tweaking the σ parameter; figure 2 also shows the results of these experiments. The default



(a) Pittsburgh, recall@N (b) Pittsburgh, rank-gain@N

(c) San Francisco, recall@N (d) San Francisco, rank-gain@N

Fig. 2. Localization performance evaluation. DisLoc always outperforms the baseline by a large margin. It also outperforms various settings of the baseline's σ parameter.

parameter value $\sigma = 16$ [20, 33] indeed performs the best with the performance being relatively stable in the range between 14 and 18. DisLoc outperforms all baselines regardless of the tweaked σ, further proving its superiority.

Figure 3 shows some qualitative examples of place recognition, where DisLoc outperforms the baseline due to successful estimation of distinctive vs non-distinctive features.

Comparison with state of the art. As noted in section 4.2, our Hamming Embedding baseline already advances the new state of the art on both benchmarks (figure 4). Since DisLoc consistently outperforms this baseline, it sets the new state of the art for both datasets. The best competitor is the Adaptive weights method [8] which discovers repetitive structures in an image and uses them to perform a more natural soft assignment of local descriptors. The paper [8] also tests several baselines (included in figure 4) such as Fisher Vectors (FV), tf-idf, etc. DisLoc consistently beats all existing methods, for example, on the Pittsburgh dataset at $N = 10$ DisLoc achieves 78.7% while the best competitor, Adaptive weights [8], gets 61.5%. Furthermore, the recall at $N = 50$ when the performance of all methods starts to saturate is also much larger – 87.4% compared to 73%. DisLoc with only top 3 retrievals achieves a better recall than Adaptive weights at 25 and 50 retrievals for the Pittsburgh and San Francisco benchmarks, respectively.

Removal of unhelpful features. Figure 5 shows the effects of removing non-distinctive features, i.e. all features whose σ is estimated to be below a threshold

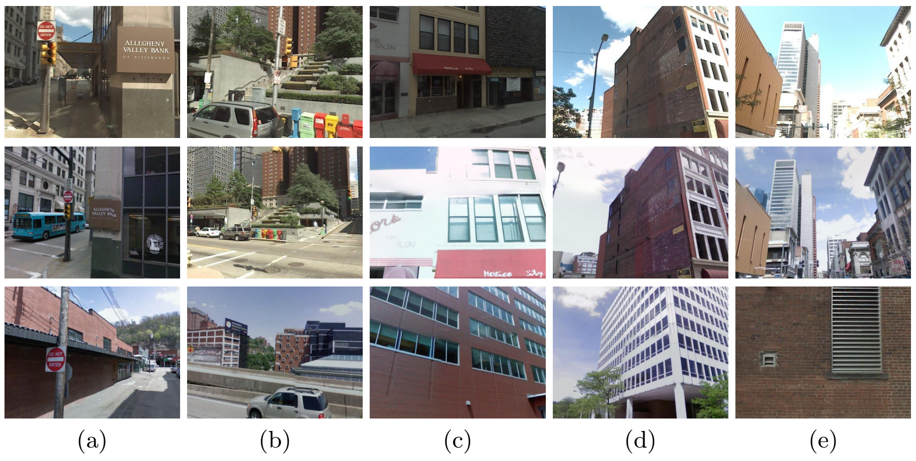
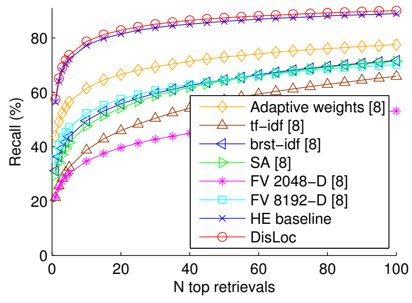
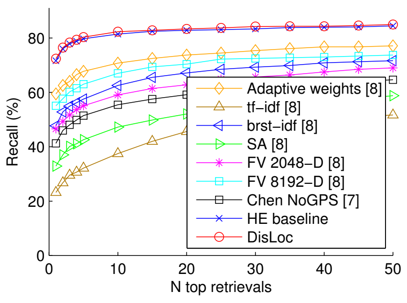


Fig. 3. Qualitative examples from the Pittsburgh benchmark. Each column shows one example, the query image is shown in the top row, and the first results returned by DisLoc and the baseline are shown in the middle and bottom rows, respectively. The baseline is often confused by non-distinctive features coming from traffic signs (a), cars (b), windows (c-d), and various repetitive structures (e). DisLoc often successfully overcomes these problems.



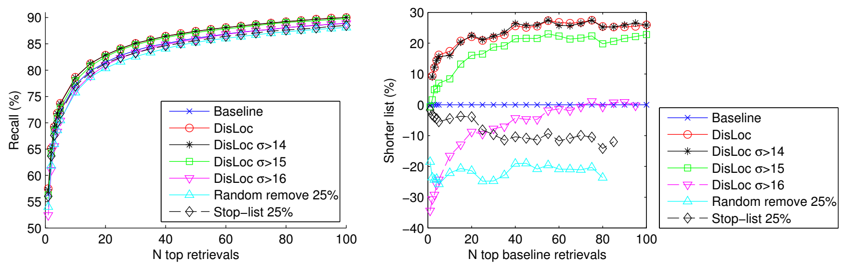
(a) Pittsburgh, recall@N



(b) San Francisco, recall@N

Fig. 4. Comparison with state of the art. The two graphs are taken from [8] and amended with our Hamming Embedding baseline and the DisLoc method. For the San Francisco dataset and this figure only, we use the April 2011 version of the ground truth for fair comparison with [7, 8], as explained in section 4.1.

are removed. It can be seen that removing 7% of features doesn't change the localization performance, while quite good performance is maintained after removing 24% of the features. Removing 50% of features makes the system work worse than the baseline for $N < 55$. Therefore, without compromising localization quality one can save 7% of storage/RAM while simultaneously increasing localization speed (as the posting lists get shorter due to a smaller number of features). With a small decrease in localization performance, a 24% saving in storage is obtainable. We have observed similar trends on the San Francisco benchmark as well.



(a) Pittsburgh, recall@N (b) Pittsburgh, rank-gain@N

Fig. 5. Removal of unhelpful features. Keeping only features for which DisLoc assigns σ larger than 14, 15 and 16 reduces storage/RAM requirements by 7%, 24% and 50%. Random removal or stop-word removal of 25% of features combined with the baseline method works much worse than the DisLoc competitors.

We compare the DisLoc-based feature removal method with two additional baselines which don't use automatic distinctiveness estimation: (1) random removal: discards 25% of the features randomly (i.e. no selection criterion is used); and (2) stop-list [10]: removes 25% of the features by discarding the most frequent visual words. Both strategies perform poorly (figure 5) compared to the DisLoc alternative – with the same number of removed features (25%) DisLoc outperforms the two baselines with a large margin. Even with 50% of the features removed, the DisLoc method significantly outperforms random removal for $N > 4$, as well as stop-list for $N > 25$.

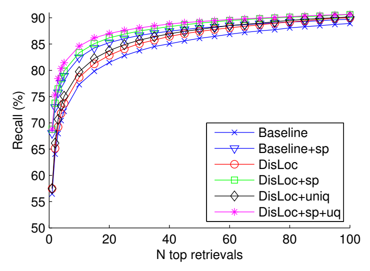
4.4 Pushing the localization performance further

In this section we evaluate using postprocessing methods to further increase place recognition performance.

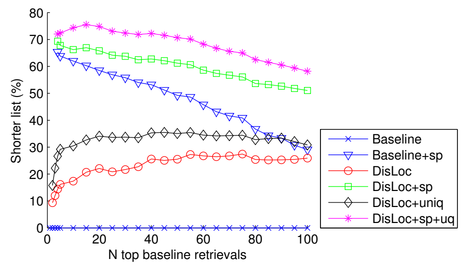
Spatial reranking. We use the standard fast spatial reranking method of [11] where the top 200 images are checked for spatial consistency with the query, using an affine transformation with verticality constraint. As expected, the method increases precision (figure 6) reflected in the increased recall at small N. For the San Francisco benchmark, DisLoc without spatial reranking beats the baseline with spatial reranking. Our method, DisLoc, continues to outperform the baseline method after spatial reranking on both benchmarks.

Unique landmark suggestions. In real-world location recognition, retrieval results should be processed to improve user experience. Namely, it would be frustrating for a user if a place recognition system provides the same wrong answer multiple times. Simple diversification of results alleviates this problem and prevents the user from being buried with false retrievals.

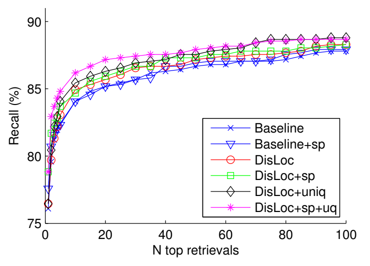
For the San Francisco dataset where building IDs are known for every database image, one can simply avoid returning the same building ID more than once, i.e. only the first instance of a building ID is kept. For the Pittsburgh dataset there is no building ID meta data available, but GPS coordinates of all database images are known. We therefore tessellate Pittsburgh into 25-by-25 meter squares



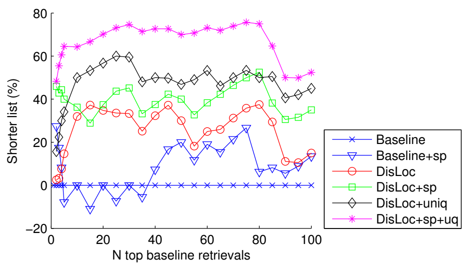
(a) Pittsburgh, recall@N



(b) Pittsburgh, rank-gain@N



(c) San Francisco, recall@N



(d) San Francisco, rank-gain@N

Fig. 6. Postprocessing performance evaluation. The "+sp" suffix signifies that spatial reranking is performed (on top 200 images), the "+uniq" suffix signifies that unique landmark suggestions are returned to the user. DisLoc continues to outperform the baseline after spatial reranking. For the San Francisco benchmark, DisLoc without spatial reranking outperforms the baseline with spatial reranking. Returning unique landmarks further improves the place recognition performance.

and only return the top ranked images from each square. For datasets which do not contain any meta information, a vision-based diversification approach can be used, such as [19, 40, 41].

As expected, the proposed diversification approach further improves location recognition performance (figure 6).

5 Conclusions

DisLoc has been shown to consistently outperform the baseline Hamming Embedding system on standard place recognition benchmarks: Pittsburgh and San Francisco landmarks, containing street-view images of those cities. Furthermore, DisLoc sets the state-of-the-art for both benchmarks by a large margin. Standard post-processing methods such as spatial reranking and result diversification have been shown to be compatible with DisLoc, and to further improve its performance. Furthermore, non-distinctive local descriptors can be discarded from the inverted index, therefore lowering memory requirements by 7%-24% and speeding up the system.

Acknowledgement. We thank A. Torri and D. Chen for sharing their datasets, and are grateful for financial support from ERC grant VisRec no. 228180 and a Royal Society Wolfson Research Merit Award.

References

[1] Quack, T., Leibe, B., Van Gool, L.: World-scale mining of objects and events from community photo collections. In: Proc. CIVR. (2008)

[2] Chen, D.M., Tsai, S.S., Vedantham, R., Grzeszczuk, R., Girod, B.: Streaming mobile augmented reality on mobile phones. In: International Symposium on Mixed and Augmented Reality, ISMAR. (2009)

[3] Cummins, M., Newman, P.: FAB-MAP: Probabilistic localization and mapping in the space of appearance. The International Journal of Robotics Research (2008)

[4] Agarwal, S., Snavely, N., Simon, I., Seitz, S.M., Szeliski, R.: Building Rome in a day. In: Proc. ICCV. (2009)

[5] Schindler, G., Brown, M., Szeliski, R.: City-scale location recognition. In: Proc. CVPR. (2007)

[6] Knopp, J., Sivic, J., Pajdla, T.: Avoiding confusing features in place recognition. In: Proc. ECCV. (2010)

[7] Chen, D.M., Baatz, G., Koeser, K., Tsai, S.S., Vedantham, R., Pylvanainen, T., Roimela, K., Chen, X., Bach, J., Pollefeys, M., Girod, B., Grzeszczuk, R.: City-scale landmark identification on mobile devices. In: Proc. CVPR. (2011)

[8] Torii, A., Sivic, J., Pajdla, T., Okutomi, M.: Visual place recognition with repetitive structures. In: Proc. CVPR. (2013)

[9] Lowe, D.: Distinctive image features from scale-invariant keypoints. IJCV **60** (2004) 91–110

[10] Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: Proc. ICCV. Volume 2. (2003) 1470–1477

[11] Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: Proc. CVPR. (2007)

[12] Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: Proc. CVPR. (2006) 2161–2168

[13] Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Lost in quantization: Improving particular object retrieval in large scale image databases. In: Proc. CVPR. (2008)

[14] Jégou, H., Douze, M., Schmid, C.: Improving bag-of-features for large scale image search. IJCV **87** (2010) 316–336

[15] Jégou, H., Douze, M., Schmid, C.: Hamming embedding and weak geometric consistency for large scale image search. In: Proc. ECCV. (2008) 304–317

[16] Philbin, J., Isard, M., Sivic, J., Zisserman, A.: Descriptor learning for efficient retrieval. In: Proc. ECCV. (2010)

[17] Simonyan, K., Vedaldi, A., Zisserman, A.: Learning local feature descriptors using convex optimisation. IEEE PAMI (2014)

[18] Gronat, P., Obozinski, G., Sivic, J., Pajdla, T.: Learning and calibrating per-location classifiers for visual place recognition. In: Proc. CVPR. (2013)

[19] Cao, S., Snavely, N.: Graph-based discriminative learning for location recognition. In: Proc. CVPR. (2013)

[20] Jégou, H., Douze, M., Schmid, C.: On the burstiness of visual elements. In: Proc. CVPR. (2009)
[21] Jégou, H., Douze, M., Schmid, C.: Exploiting descriptor distances for precise image search. Technical report, INRIA (2011)
[22] Aly, M., Munich, M., Perona, P.: Compactkdt: Compact signatures for accurate large scale object recognition. In: IEEE Workshop on Applications of Computer Vision. (2012)
[23] Sattler, T., Weyand, T., Leibe, B., Kobbelt, L.: Image retrieval for image-based localization revisited. In: Proc. BMVC. (2012)
[24] Tolias, G., Avrithis, Y., Jégou, H.: To aggregate or not to aggregate: selective match kernels for image search. In: Proc. ICCV. (2013)
[25] Qin, D., Wengert, C., Gool, L.V.: Query adaptive similarity for large scale object retrieval. In: Proc. CVPR. (2013)
[26] Turcot, T., Lowe, D.G.: Better matching with fewer features: The selection of useful features in large database recognition problems. In: ICCV Workshop on Emergent Issues in Large Amounts of Visual Data (WS-LAVD). (2009)
[27] Philbin, J., Zisserman, A.: Object mining using a matching graph on very large image collections. In: Proc. ICVGIP. (2008)
[28] Jégou, H., Harzallah, H., Schmid, C.: A contextual dissimilarity measure for accurate and efficient image search. In: Proc. CVPR. (2007)
[29] Qin, D., Gammeter, S., Bossard, L., Quack, T., Van Gool, L.: Hello neighbor: accurate object retrieval with k-reciprocal nearest neighbors. In: Proc. CVPR. (2011)
[30] Delvinioti, A., Jégou, H., Amsaleg, L., Houle, M.E.: Image retrieval with reciprocal and shared nearest neighbors. In: VISAPP – International Conference on Computer Vision Theory and Applications. (2014)
[31] Jégou, H., Douze, M., Schmid, C., Pérez, P.: Aggregating local descriptors into a compact image representation. In: Proc. CVPR. (2010)
[32] Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Comm. ACM (2008)
[33] Tolias, G., Jégou, H.: Visual query expansion with or without geometry: refining local descriptors by feature aggregation. Pattern Recognition (2014)
[34] Jégou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE PAMI (2011)
[35] Van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. Journal of Machine Learning Research (2008)
[36] Arandjelović, R., Zisserman, A.: Three things everyone should know to improve object retrieval. In: Proc. CVPR. (2012)
[37] Mikolajczyk, K., Schmid, C.: Scale & affine invariant interest point detectors. IJCV **1** (2004) 63–86
[38] Chum, O., Philbin, J., Sivic, J., Isard, M., Zisserman, A.: Total recall: Automatic query expansion with a generative feature model for object retrieval. In: Proc. ICCV. (2007)
[39] Chum, O., Mikulik, A., Perďoch, M., Matas, J.: Total recall II: Query expansion revisited. In: Proc. CVPR. (2011)

[40] Kennedy, L., Naaman, M.: Generating diverse and representative image search results for landmarks. In: Proc. World Wide Web. (2008)
[41] van Leuken, R.H., Garcia, L., Olivares, X., van Zwol, R.: Visual diversification of image search results. In: Proc. World Wide Web. (2009)