LOAD-BALANCED LOCALITY-SENSITIVE HASHING: A NEW METHOD FOR EFFICIENT NEAR DUPLICATE IMAGE DETECTION

Yabo Fan, Junliang Xing, Weiming Hu

National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academic of Sciences
No. 95, Zhongguancun East Road, Beijing 100190, P. R. China
{yabo.fan, jlxing, wmhu}@nlpr.ia.ac.cn

ABSTRACT

Locality-Sensitive Hashing (LSH) is a mainstream method for the Near Duplicate Image Detection (NDID) problem. Previous LSH based methods, however, do not have a principled way to make the indexing structure generate the buckets of similar sizes, which will inevitably degrade the detection effectiveness and efficiency. In this work, we propose a Load-Balanced Locality-Sensitive Hashing (LBLSH) method with a new indexing structure to produce load-balanced buckets for the hashing process. As proved in the paper, the proposed LBLSH can guarantee load-balanced buckets in the hashing process and significantly reduce the query time and the storage space. Based on the proposed LBLSH method, we design an effective and feasible algorithm for the NDID problem. Extensive experiments on two benchmark datasets demonstrate the effectiveness and efficiency of our method.

Index Terms— Near Duplicate Image Detection, Nearest Neighbor Search, Locality-Sensitive Hashing, Load-Balanced Locality-Sensitive Hashing

1. INTRODUCTION

With the rapid development of multimedia technology, the amount of digital images has become overwhelmingly huge. It often occurs that a digital image has many near-duplicates on the Internet, which can be easily observed by using Google or Yahoo. This phenomenon dramatically leads to a huge waste of network resources, as well as problem like copyright infringement. Therefore, efficient Near Duplicate Image Detection (NDID) algorithm has become an urgent issue. Near duplicate images are defined as the many transformed versions of the original image like blurring, geometric manipulations [1], to name a few. All these potential transformations together makes NDID problem very challenging [2, 3].

In the last decade, various approaches have been proposed aiming at improving NDID accuracy and efficiency [4, 5, 6, 7, 8]. A major challenge for the NDID problem is the runtime performance, since the image database is usually very huge. There are several well-known algorithms for the case when feature dimension is low [9]. However, if the dimension exceeds 10 to 20, as demonstrated both in theory and practice, searching in k-d tree and related structures works no better than brute-force linear search [10]. Nevertheless, image features are usually multidimensional. Up to now, Locality-Sensitive Hashing (LSH) [11] has been the most popular indexing structure for multidimensional image features employed in NDID.

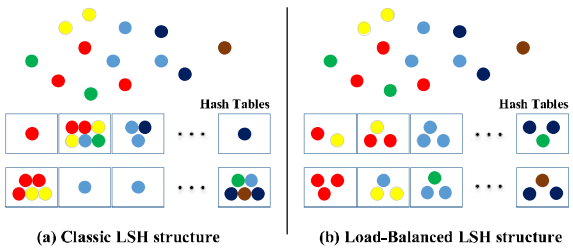


Fig. 1: Classic LSH versus Load-Balanced LSH. Classic LSH constructs unbalanced structure which naturally leads to inefficient searching. While our Load-Balanced LSH obtains balanced buckets and drastically accelerates the detection.

The LSH indexing structure actually acts as a classifier that projects similar items into the same bucket. The basic LSH was first applied in NDID by [4]. Based on various distances, different LSH families were proposed. For example, p-stable distribution LSH [12] for ℓ_p distance, min-Hash [6] for Jaccard coefficient distance, Kernelized LSH [13] for angle-based distance. TF-IDF weighting was combined with min-Hash method for NDID [6]. Weakly supervised LSH was proposed in [14] for NDID. LSH-based NDID algorithms are constructed as a two-stage model. The first stage uses LSH to largely reduce the candidates for a query while the second stage exploits the result of first stage by exhaustive searching. This model is often refereed to as coarse-to-fine model.

When applied to the NDID problem, the classic LSH-based approaches can achieve very good accuracy. Their efficiencies, however, are not satisfied. The main reason is due to the peculiarity of the NDID problem: the number of categories is very huge and some "hot spot" images have too many duplicates while some have very few. Under these situations, as shown in Fig. 1(a), the classic LSH usually maps too many items into some buckets while others contain only a few, which we refer to as unbalanced indexing structure. Obviously, the candidate number returned by the LSH structure

dominates the detection efficiency. The larger bucket is easier to hit, and results in much more time for carrying out exhaustive searching. To surmount these problems, we propose new Load-Balanced LSH method for the NDID problem. Our proposed approach suits the peculiarity in the NDID problem by adopting local redistribution rules and probing appropriate number of neighbor buckets for detection to produce load-balanced LSH structure, as illustrated in Fig. 1(b).

The main contribution of this work can be summarized in three-fold: 1) we propose Load-Balanced LSH, an efficient indexing structure which contains load-balanced buckets aiming at improving efficiency for the NDID problem; 2) we derive an appropriate bucket size threshold for the LBLSH, with guaranteed searching performance; 3) we present an effective and feasible algorithm associated with LBLSH for the NDID problem, including initialization, basic hashing, local redistribution and neighbor-probe searching.

2. LSH BASIS

Given a query point q, if there exists a point p such that $\text{dist}(p,q) \leq R$, then the indexing structure needs to report points within distance cR from q for some constant $c > 1$. This problem is referred as (R, c)-near neighbor (NN) problem. Locality-Sensitive Hashing (LSH) is introduced to solve the (R, c)-NN problem by mapping similar items, represented by feature vectors, into the same buckets with higher probability than dissimilar items. LSH is based on the definition of LSH family, a family of hash function with the property that similar items have higher probability of colliding than dissimilar items. Formally, an LSH family is defined as follows:

Definition 1 *(Locality-Sensitive Hashing)*: A family of H is called (R, c, p_1, p_2)-sensitive if for any items p and q,

- If $\text{dist}(p,q) \leq R$, then $\text{Prob}_{h\sim H}[h(p) = h(q)] \geq p_1$,
- If $\text{dist}(p,q) \geq cR$, then $\text{Prob}_{h\sim H}[h(p) = h(q)] \leq p_2$.

Here $c > 1$, and $p_1 > p_2$. The parameter $\rho = \frac{\log(1/p_1)}{\log(1/p_2)}$ governs the searching performance, *i.e.*, the smaller ρ, the better searching efficiency. Given the query q, the items lying in the bucket $h(q)$ are considered as candidates for q. Constructing an LSH structure needs to determine two parameters: L, the number of hash tables, and K, the length of hash code.

3. LOAD-BALANCED LSH

The key idea of our Load-Balanced LSH is to hash the image feature vectors into buckets which are guaranteed to be balanced. The threshold value for bucket size is derived from theoretical analysis about LSH. In [15], Lv *et al.* show that if the item q near to an item p is not hashed into the same bucket as p, two buckets containing p and q are close to each other with a high probability. Our proposed LBLSH exploits this property to distribute extra items into neighbor buckets and probes appropriate number of neighbor buckets accordingly. As shown in Fig. 2, our approach consists of four steps.

We first initialize LSH function family based on classic LSH family, and then we carry out basic hashing on all items regardless of the bucket size threshold. When the size of a bucket exceeds the threshold, local redistribution is performed on this bucket. When all buckets conform the constraint condition, we finish the LBLSH construction and probe appropriate number of neighbor buckets for detection.

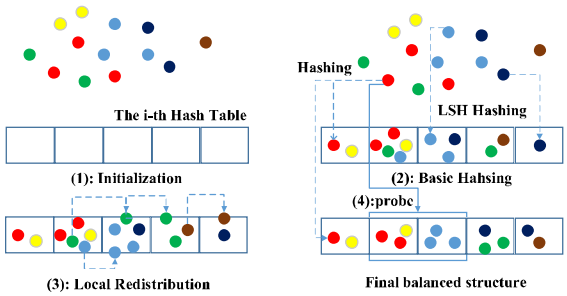


Fig. 2: Overview of the proposed Load-Balanced LSH.

3.1. Initialization

Our Load-Balanced LSH function is based on classic LSH, such as E2LSH [12] and Hamming LSH [10]. For example, the function for Euclidean distance is formulated as:

$$h_{w,b}(x) = \lfloor \frac{w^T x + b}{r} \rfloor. \quad (1)$$

Here, w is a d-dimension vector with entries chosen independently from p-stable distribution, and b is a real number chosen uniformly from range [0, r]. This hash function: $h_{w,b} : \Re^d \rightarrow Z$ maps a d-dimension vector onto a set of integers. Specifically, to solve the searching problem under the Euclidean distance, the 2-stable distribution, *i.e.*, the Gaussian distribution, is chosen to generate random projection w.

3.2. Basic Hashing

After the initialization step, the Load-Balanced LSH carries out the Basic Hashing on the input items regardless of the bucket size threshold as the classic LSH approach does, namely: item x goes into $h_i(x)$ bucket in the i-th hash table.

3.3. Local Redistribution

When the size of a bucket exceeds the threshold Δ_{LB}, we need to execute the Local Redistribution. The threshold Δ_{LB} which confines the bucket size is important for our Load-Balanced LSH. According to the definition of LSH family, the parameter $\rho = \frac{\log(1/p_1)}{\log(1/p_2)}$ governs the searching performance. And given a family of $(1, c, p_1, p_2)$-sensitive hash function for n items in $\Re^d$, where each function can be evaluated in time τ, one can construct a data structure for (R, c)-NN problem with $O(dn+n^{1+\rho})$ space [16]. For any vector $x \in \Re^d$, the value $\| w^T x \|^2 / \| x \|^2$ is distributed with probability $tP_{\chi^2}(xt)$, where $P_{\chi^2}(x) = \frac{x^{t/2-1}e^{-x/2}}{\Gamma(t/2)2^{t/2}}$ is the chi-squared distribution with t degrees of freedom. As proved in [16], we can establish boundary for p_1 and p_2:

$$p_1 \geq \frac{1}{2\sqrt{t}} \cdot \frac{1}{(1+1/4\sqrt{t}+t/2)^{t/2}}, \; p_2 \leq \frac{2}{(1+c^2/4\sqrt{t})^{t/2}} \quad (2)$$

Thus, ρ can be bounded as:

$$\rho = \frac{\log(1/p_1)}{\log(1/p_2)} \leq \frac{\log(1+1/4\sqrt{t}+t/2)+2\log 2\sqrt{t}/t}{\log(1+c^2/4\sqrt{t})-2\log 2/t} \quad (3)$$

$$\leq \frac{1/4\sqrt{t}+t/2}{c^2/4\sqrt{t}-(c^2/4\sqrt{t})^2/2} \cdot (1+O(\frac{\log t}{t\cdot\log(1+1/4\sqrt{t})})) \quad (4)$$

$$\leq 1/c^2 \cdot (1+2/\sqrt{t}) \cdot (1+O(c^2/\sqrt{t})) \cdot (1+O(\log t/\sqrt{t})) \quad (5)$$

$$\leq 1/c^2 \cdot (1+O(\log t/\sqrt{t})). \quad (6)$$

Note that in step (4) and (5), Taylor series expansion is adopted to perform the derivations.

Since it is very common that the amount n of images in the database is very large in real NDID problem, with this information we can perform the following approximations:

$$t \propto n, \text{so} \lim_{n\to\infty} \frac{\log t}{\sqrt{t}} \to 0, \quad (7)$$

$$\rho \leq \frac{1}{c^2} \cdot (1+O(\frac{\log t}{\sqrt{t}})) \to \frac{1}{c^2}. \quad (8)$$

Therefore, space efficient LSH algorithm can allocate $(dn+n^{1+1/c^2})$ space. In the Load-Balanced LSH, we define this threshold as:

$$\Delta_{LB} = \lceil \frac{(dn+n^{1+1/c^2})}{L \cdot B} \rceil \quad (9)$$

for n d-dimension items stored in L hash tables, each of which will maintain at most B buckets (c with the value 2 is sufficient in most experiments).

After the Basic Hashing operation, we need to find the virtual center VC for every bucket, whose d-dimension coordinates are the average values of the current items in the bucket. Then, we will check every bucket according to its position in the hash table. If the size $n_i(t)$ of a bucket exceeds its threshold Δ_{LB}, we need to compute the distance between its each item and its VC, and sort the items by their distances in descending order. Then $(n_i(t) - \Delta_{LB})$ items are chosen with the farthest distance, and these chosen items will be sent to neighbor buckets, as shown in Algorithm 1.

In order to guarantee the stability of the hash buckets and the accuracy of detection, the virtual center VC for every bucket is pre-computed after Basic Hashing, and not updated during the iteration. If the iteration goes to the last bucket in a hash table and its size exceeds Δ_{LB}, the chosen farthest items will be sent to the first bucket, and the iteration will start over again from the first bucket. When all buckets conform the Δ_{LB} constraint, we finish the LBLSH construction.

3.4. Neighbor-probe Searching

To cooperate with the Local Redistribution operation which sends extra items to neighbor buckets, Load-Balanced LSH probes more than one bucket for k approximate nearest neighbors search. Our neighbor-probe searching method is inspired by the Multi-probe LSH [15], but it is simple and intuitive in comparison with Multi-probe LSH. Specifically, for a query

Algorithm 1: Local Redistribution for the i-th hash table

```
for j = 1 : n do
    compute h_i(v_j) for item v_j based on the LSH function;
    insert v_j into the bucket_{h_i(v_j)};
end for
for every bucket t in the i-th table do
    n_i(t) = size(bucket_{h_i(t)});
    for x = 1; x ≤ d; x++ do
        VC_{h_i(t)} = (Σ_x v_{h_i(t)}) / n_i(t);
    end for
    if n_i(t) > Δ_LB then
        dist(h_i(t)) = Σ_x (VC_{h_i(t)} − v_{h_i(t)})^2;
        chose (n_i(t) − Δ_LB) items with farthest distance;
        send these chosen items to next neighbor bucket;
    end if
end for
```

q, we need to first probe the $h_i(p)$-th bucket and then check its next np neighbor buckets in the i-th hash table. Here,

$$np = \lfloor \frac{\Delta_{LB}}{\Delta_{LB} - \text{mean}(\text{size}(bucket_{h_i}))} \rfloor. \quad (10)$$

The number np of neighbor buckets is related to the mean value of buckets size in the i-th hash table.

4. EXPERIMENTAL EVALUATION

4.1. Experimental Setup

We adopt the public benchmark *UKbench* [17] and *INRIA Copydays* dataset [18] for evaluation. *UKbench* consists of 2550 different scenes. Each one has four images taken from different viewpoints, thus *UKbench* contains 10200 images. *INRIA Copydays* dataset consists of 157 images, and for each image, we generate 20 near-duplicate versions by using JPEG compression, scaling and noise blurring. Querying the datasets with one image should return its k near-duplicates. The performance is measured by N-S score for *UKbench* (maximum is 4), and mean average precision (mAP) for both.

The proposed LBLSH can directly adopt the global features. In the experiments, two global representations are used to verify the effectiveness of proposed method, namely the 320-D GIST [19], and the 400-D SIFT [20] based Bag-of-Feature (BoF). Besides, for *UKbench*, we adopt early fusion of three normalized features, namely color, LBP [21] and RootSIFT [22] based VLAD [23] feature and then the PCA is used to decrease the dimension to 3000 which keeps about 98% of the energy.

Accuracy and speed are the main evaluation criteria for NDID. For accuracy, the mean Average Precision (mAP) is used as the evaluation metric. In fact, mAP refers to the area under the precision-recall curve. The results of exhaustive searching is used as baseline. The mAP is evaluated as:

$$\text{mAP(ratio)} = \frac{\text{top } k \text{ results of indexing method}}{\text{top } k \text{ results of exhaustive searching}}$$

For detection speed, we discuss the acceleration factor that

can be estimated by N/N_r, where N is the total number of images in the dataset and N_r is the number of image candidates returned by the indexing structure.

4.2. Parameter Analysis for LBLSH

The key idea of our Load-Balanced LSH is to hash the image feature vectors into buckets which contain appropriate number of items. Thus the parameter Δ_{LB} needs to be carefully considered during the degign of LBLSH. The threshold value can be derived from theoretical analysis about LSH, as shown in section 3.3. In the following, we will discuss the effect of the parameter Δ_{LB} on detection accuracy and speed.

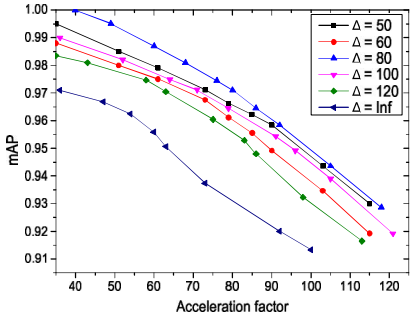
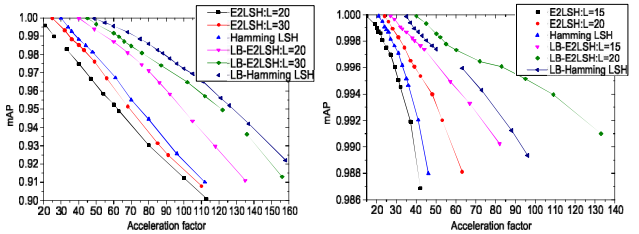


Fig. 3: Results on *UKbench* of different Δ_{LB} for LB-E2LSH using GIST. K is sampled to obtain different performance when $L = 20$.

For 320-D GIST of *UKbench*, we assume that 10200 images are stored in 20 hash tables each containing at most 2000 buckets. Thus, based on section 3.3, we can estimate Δ_{LB} as $(320 \cdot 10200 + 10200^{1+0.25})/(20 \cdot 2000) \approx 80$. We present results with values of Δ_{LB} set to 50, 60, 80, 100, 120 and Inf (infinitc, namcly classic LSH), as shown in Fig. 3. From the results, we clearly observe that the proposed calculation method can achieve the best detection accuracy and speed.

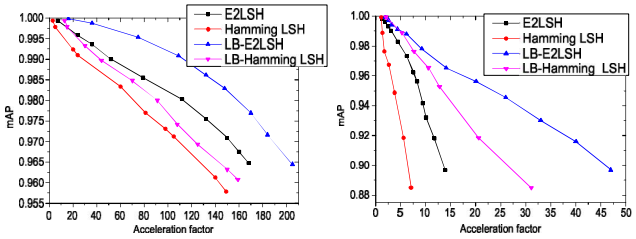
4.3. Performance Comparison with Classic LSH

In order to show the effectiveness of our proposed LBLSH for NDID, we compare our method (LB-E2LSH and LB-Hamming LSH) with classic LSH-based approaches, namely E2LSH and Hamming LSH. Fig. 4 and Fig. 5 show the comparison results (in terms of mAP versus acceleration factor) using 320-D GIST and 400-D BoF. The length K of hash code is varied to obtain different performance. From the comparisons, we observe that, at the same detection accuracy, LBLSH (LB-E2LSH and LB-Hamming LSH) performs much faster than the classic LSH. Such results reflect the effectiveness of the proposed LBLSH to some degree.



(a) 320-D GIST (b) 400-D SIFT based BoF

Fig. 4: Comparisons between our LBLSH (LB-E2LSH and LB-Hamming LSH, $\Delta_{LB} = 80, 100$ for GIST and BoF) and classic LSH using two different features on *UKbench*. For E2LSH, we present results with different hash table numbers L.



(a) 320-D GIST (b) 400-D SIFT based BoF

Fig. 5: Comparisons between our LBLSH (LB-E2LSH and LB-Hamming LSH, $L = 20, 30$ and $\Delta_{LB} = 27, 34$ for GIST and BoF) and classic LSH using two different features on *INRIA Copydays*.

Finally, we evaluate our method compared with the classic E2LSH using early fusion of color, LBP and RootSIFT based VLAD (L=30, $\Delta_{LB} = 550$) on *UKbench*. Table 1 presents the comparison results, including N-S score, image candidates for exhaustive searching and acceleration factor. We are not trying to compare with image retrieval. Our method is designed to detect near duplicate images efficiently. Despite this, the N-S scores of our LB-E2LSH exceed the score of 3.17 for min-Hash in [6] and 3.42 for BoF in [24] on *UKbench*.

Table 1: Comparison between E2LSH and our LB-E2LSH using early fusion of color, LBP and RootSIFT on the *UKbench* dataset.

Evaluation criteria		N-S score	Image cand.	Acce. factor
classic E2LSH	K=24	3.462	939.0	10.9
	K=25	3.507	953.0	10.7
	K=26	3.493	943.0	10.8
	K=27	3.484	931.5	11.0
	K=28	3.442	883.5	11.5
proposed LB-E2LSH	K=24	3.490	620.8	16.4
	K=25	3.508	636.2	16.0
	K=26	3.506	634.8	16.1
	K=27	3.495	625.6	16.3
	K=28	3.494	621.7	16.4

5. CONCLUSIONS

We present a Load-Balanced LSH method for the efficient and effective NDID problem. LBLSH guarantees to map images into buckets with similar sizes and probe appropriate number of neighbor buckets for detection, thus drastically accelerating the detection and also obtaining very good accuracy. The proposed method is efficient and flexible compared with the classic LSH. Experimental results on benchmark datasets demonstrate its effectiveness and efficiency.

6. ACKNOWLEDGEMENT

This work is partly supported by the 973 basic research program of China (Grant No. 2014CB349303), the Natural Science Foundation of China (Grant No. 61303178 and 61472421), the National 863 High-Tech R&D Program of China (Grant No. 2012AA012504), and the Project Supported by Guangdong Natural Science Foundation (Grant No. S2012020011081).

7. REFERENCES

[1] A. Joly, O. Buisson, and C. Frelicot, "Content-based copy retrieval using distortion-based probabilistic similarity search," *IEEE Transactions on Multimedia*, vol. 9, no. 2, pp. 293–306, 2007.

[2] Z. Wu, Q. Xu, S. Jiang, Q. Huang, P. Cui, and L. Li, "Adding affine invariant geometric constraint for partial-duplicate image retrieval," in *IAPR International Conference on Pattern Recognition*, 2010.

[3] Y. Lin, C. Xu, L. Yang, Z. Lin, and H. Zha, "L1-norm global geometric consistency for partial-duplicate image retrieval," in *IEEE International Conference on Image Processing*, 2014.

[4] Y. Ke, R. Sukthankar, and L. Huston, "Efficient near-duplicate detection and sub-image retrieval," in *ACM International Conference on Multimedia*, 2004.

[5] W. Zhao, C. Ngo, H. Tan, and X. Wu, "Near-duplicate keyframe identification with interest point matching and pattern learning," *IEEE Transactions on Multimedia*, vol. 9, no. 5, pp. 1037–1048, 2004.

[6] O. Chum, J. Philbin, and A. Zisserman, "Near duplicate image detection: min-hash and tf-idf weighting," in *British Machine Vision Conference*, 2008.

[7] Z. Xu, H. Ling, F. Zou, Z. Lu, and P. Li, "Robust image copy detection using multi-resolution histogram," in *ACM International Conference on Multimedia Information Retrieval*, 2010.

[8] Y. Lei, G. Qiu, L. Zheng, and J. Huang, "Fast near-duplicate image detection using uniform randomized trees," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 10, no. 4, pp. Article No. 35, 2014.

[9] H. Samet, *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[10] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *International Conference on Very Large Data Bases*, 1999.

[11] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *ACM Symposium on Theory of computing*, 1998.

[12] M. Datar, N. Immorlica, P. Indyk, and V.S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *ACM Symposium on Computational Geometry*, 2004.

[13] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *IEEE International Conference on Computer Vision*, 2009.

[14] Y. Cao, H . Zhang, and J. Guo, "Weakly supervised locality sensitive hashing for duplicate image retrieval," in *IEEE International Conference on Image Processing*, 2011.

[15] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-probe lsh: Efficient indexing for high-dimensional similarity search," in *International Conference on Very Large Data Bases*, 2007.

[16] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *IEEE Symposium on Foundations of Computer Science*, 2006.

[17] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[18] M. Douze, H. Jégou, H. Sandhawalia, L. Amsaleg, and C. Schmid, "Evaluation of gist descriptors for web-scale image search," in *ACM International Conference on Image and Video Retrieval*, 2009.

[19] A. Oliva and A.B. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.

[20] D.G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[21] T. Ojala, M. Pietikäinen, and T. Mäenpää, "A generalized local binary pattern operator for multiresolution gray scale and rotation invariant texture classification," in *IEEE International Conference on Advances in Pattern Recognition*, 2001.

[22] R. Arandjelović and A. Zisserman, "Three things everyone should know to improve object retrieval," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[23] H. Jégou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1704–1716, 2012.

[24] H. Jégou, M. Douze, and C. Schmid, "Improving bag-of-feature for large scale image search," *International Journal of Computer Vision*, vol. 87, no. 3, pp. 316–336, 2010.