Large Scale Sketch Based Image Retrieval Using Patch Hashing[⋆]

Konstantinos Bozas and Ebroul Izquierdo

School of Electronic Engineering and Computer Science
Queen Mary University of London, Mile End Campus, UK, E1 4NS
{k.bozas,ebroul.izquierdo}@eecs.qmul.ac.uk

Abstract. This paper introduces a hashing based framework that facilitates sketch based image retrieval in large image databases. Instead of exporting a single visual descriptor for every image, an overlapping spatial grid is utilised to generate a pool of patches. We rank similarities between a hand drawn sketch and the natural images in a database through a voting process where near duplicate in terms of shape and structure patches arbitrate for the result. Patch similarity is efficiently estimated with a hashing algorithm. A reverse index structure built on the hashing keys ensures the scalability of our scheme and at the same time allows for real time reranking on query updates. Experiments in a publicly available benchmark dataset demonstrate the superiority of our approach.

1 Introduction

The exponential growth of publicly available digital media during the last two decades highlighted the need for efficient and user friendly techniques to index and retrieve images and videos from large scale multimedia databases. Despite the considerable progress of content-based image retrieval (CBIR) [1], where the goal is to return images similar to a user provided image query, most of the multimedia searches are text-based (e.g. Google Images, YouTube). The latter requires user intervention to tag all the available data and has two main drawbacks: (i) it is time consuming and (ii) most importantly user subjective. It is a common belief that images cannot be succinctly communicated based on words; humans would probably use different words to describe a scene based on their cultural background and experience. It follows from the above that in specific scenarios searching images by text query will return frustrating and dubious results to the user. Sketch based image retrieval (SBIR) emerged as a more expressive and interactive way to perform image search; here the query is formed as a hand-drawn sketch of the imaginary picture. Obviously, a shaded rendition of the query requires great artistic skill [2] and most users will refrain from exercising considerable effort and time to draw it. A more intuitive way is

[⋆] This work is partially supported by EU project CUBRIK under grant agreement FP7 287704.

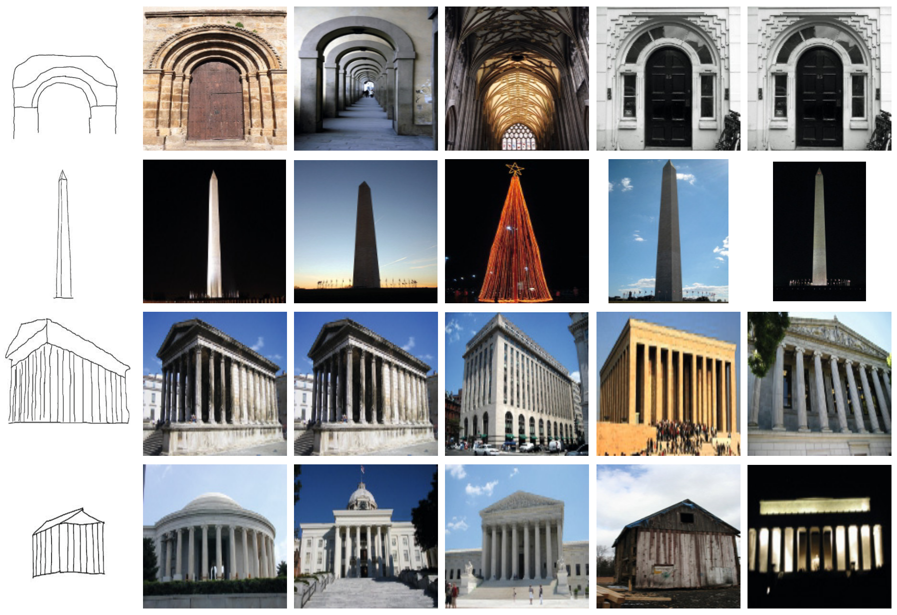

Fig. 1. Top 5 results returned from our system in some queries of [5]

to sketch the main feature lines of a shape or scene, a design choice we follow in this paper, and is supported by recent studies where it has been demonstrated that lines are drawn along contours [3] and line drawings can encode certain shapes almost as well as shaded images [4]. Binary sketches can be easily drawn using the mouse in a personal computer or the touch screen of a modern touch screen device.

Finding similarities between a binary drawing and a database of colored pictures taken under arbitrary conditions consists a challenging problem. Images and binary sketches do not share many common modalities. Images contain rich information in domains such as color and texture, while sketches can be described only by their shape and spatial configuration, therefore traditional CBIR methods relying on texture and color cannot be inherited in SBIR. Furthermore, the vast amount of photos uploaded to social media websites signify the crucial role of *scalability*. We need to be able to search large collections of images in reasonable query times as well as update databases and index files with minimum computational cost.

This paper focuses both on scalability and retrieval quality. We tackle these challenges by retrieving *near duplicate in terms of shape* images to a binary sketch query using a hashing technique. Our approach assumes that users are looking for images spatially consistent with their query, for instance if they draw a sunset scene and the sun has been placed at the top right of the canvas that indicates that images displaying the sun in (approximately) the same spot will be preferred, therefore our system is designed to retrieve near duplicate images to the hand drawn sketches up to small translations. In the core of our retrieval scheme lies min-hash, a set similarity estimation technique originally applied to identify duplicate web pages [6] and later modified for near duplicate image search [7]. Our method differs from [7] where an image is described by a single set of visual words, alternatively we extract local image patches represented with a binary version of the HoG [8] descriptor which allows the utilization of the min-hash algorithm. This provides more detailed image description as well as flexibility during query time since we can omit searching for patches that have not been filled during drawing. An index structure facilitates quick queries and online result updating. We compare the retrieval accuracy of our method in a publicly available benchmark [5] and demonstrate state-of-the-art results.

2 Related Work

The idea of retrieving similar photos given a hand-drawn query has first received attention during the nineties. Hirata and Kato [9] performed visual search by estimating block correlation between digitized oil paintings and a rough sketch illustration of a database painting. Lopresti and Tomkins [10] proposed a stroke matching approach to retrieve hand-drawings. They take advantage of the inherently sequential nature of ink creation in the temporal domain, so they segment a digital pen's input into strokes and export a set of basic strokes types from every drawing. This approach was successfully applied to handwritten text recognition where the stroke order is well defined but struggles with pictorial queries due to the arbitrary order of the strokes. Liang, Sun, and Li [11] decompose sketches into basic geometric primitives by using pen speed and curvature and form a topological graph to quantify similarity. Edge map segmentation, crucial for this technique, fails when applied to natural images. Del Bimbo and Pala [12] presented a template matching technique, where the elastic deformation energy spent is employed to derive a measure of similarity between the sketch and the images in the database and to rank images to be displayed. The method was applied to a pool of 100 images and in spite of its success in that particular database it cannot be employed in large scale due to the computational cost of the elastic deformation calculation. Matusiak *et al.* [13] parametrized the hand-drawn sketch curves in the curvature scale space and match them against a database of eight hundred closed curved shapes. This approach fails when images contain non closed curve shapes, therefore fails to describe the majority of user photos since the presence of a perfect non-occluded shape rarely occurs due to background clutter and other sources of noise.

More recent approaches attempt to extract a visual footprint for each image that satisfies two properties, first similar images are mapped to similar feature vectors and secondly the size of footprint must be kept as small as possible without losing its discrimination power so as to accommodate large scale indexing. Towards this direction, an image is usually partitioned in blocks and a small descriptor is computed for each block, the final footprint is acquired by concatenating all the feature vectors to a longer one. Chalechale, Naghdy, and Mertins

[14] performed angular partitioning in the spatial domain of images and used the magnitude of the Fourier coefficients to obtain a shape description which is rotation invariant but suffers from noisy edge artifacts. The first approach that reported the use of a large scale database for evaluation presented from Eitz *et al.* [15], the authors compute the main gradient orientations for each spatial block achieving robustness to background clutter. Following its success in text retrieval, the well-known bag-of-words (BoW) model has been applied to image search [16], each image is represented with a set of visual words obtained by clustering a large pool of feature vectors. Hu *et al.* [17,18] employed a gradient vector field in an attempt to smooth the divergence between images and hand-drawn sketches, and used the BoW model to facilitate the retrieval, however their experiments were conducted in a small database of 320 images. Cao *et al.* [19] achieved efficient indexing of 2 million images and approximated the Orientated Chamfer Distance to measure database similarities with a binary sketch query. Our work is inspired from ShadowDraw [20], where a min-hash based retrieval mechanism is utilized to acquire image patches similar to the user hand-drawn input, yet our approach employs different patch description process and relying on the patch votes infers a ranking on the database images.

3 Min-Hash Algorithm

In this section, a brief description of the min-hash algorithm is presented. Min-hash is a Local Sensitive Hashing [21] based technique that estimates similarity between sets. Assume a set of tokens S of size $|S|$. The set $D_i \subseteq S$ can be represented by a sequence of size $|S|$ where the presence of a token $s \in S$ is indicated by 1 and the absence by 0. The set overlap similarity (or Jaccard similarity) between two sets D_1 and D_2 is defined as the ratio of their intersection and union and is a number between 0 and 1; it is 0 when the two sets are disjoint, 1 when they are equal, and between 0 and 1 otherwise.

$$sim(D_1, D_2) = \frac{|D_1 \cap D_2|}{|D_1 \cup D_2|} \in [0,1] \,. \tag{1}$$

Eq. 1 values equally every member of the set. In [7] an extension proposed to assign to each set member a weight according to its importance. Min-hash approximates the set similarity by creating a collection of hash functions $h_i : S \to \mathbf{R}$ mapping each element of S to a real number. Each hash function h_i infers an ordering on the members of D_i. Min-hash is defined as the smallest element of D_i under ordering induced by h_i and has the property that the probability of two sets having the same min-hash value equals to their Jaccard similarity.

$$P\{h_i(D_1) = h_i(D_2)\} = \frac{|D_1 \cap D_2|}{|D_1 \cup D_2|} = sim(D_1, D_2) \,. \tag{2}$$

An unbiased similarity estimation is obtained by computing for each set D_i multiple independent min-hash functions h_i and counting the occurrences of identical min-hash values for the two sets. To reduce the possibility of false positive retrievals hash functions are organized in s-tuples (often called sketches of min-hash values). Two sets are characterized similar if they share at least one sketch, this follows from the fact that similar sets share many min-hash values while dissimilar share only few. The sketches can be computed fairly fast (linear in the size of D_i) and given two sketches the resemblance of the corresponding sets can be computed in linear time in the size of the sketches. Moreover, each set is represented by k s-tuples to increase the recall of the retrieval. The probability of collision under this scheme is:

$$P\{h(D_1) = h(D_2)\} = 1 - (1 - sim(D_1, D_2)^s)^k \, . \tag{3}$$

Min-hash has been successfully applied to text [6] and image [7] domains to detect near duplicate instances of a given set. Section 4 describes how we adopt min-hash to sketch based image retrieval.

4 Near Duplicate Patch Retrieval with Hashing

4.1 Method Overview

We retrieve similar images to a sketch query based on a local patch technique. Every image in the database is divided into overlapping patches and for each patch a sequence of min-hash values is extracted. A reverse index is built on the unique min-hash values pointing to the patches containing these values. A sketch query undergoes the same process and for each sketch patch we look into the index to retrieve similar patches. Every index hit contributes a vote to the corresponding image and the final ranking is generated by summing the votes for each image.

Min-hash is employed to estimate the similarity between two patches. Chum *et al.* [7] proposed to represent an image by an unordered set of visual words acquired by clustering on the feature space. Under this scheme, an image is represented by a single sequence of 0 and 1. Our approach, similarly to [20] adopts a sequence description for every local patch, but instead of utilizing a visual codebook to derive the sequences, we use the non-zero indexes of the binarized patch descriptor. A visual illustration of our method is presented in Fig. 2.

4.2 Preprocessing

Bridging the modality gap between photos and hand-drawn sketches consists the first impediment a SBIR system has to overcome. Ordinarily, this is achieved by extracting edge lines from the images. The well-known Canny algorithm [22] requires in many cases manually tuning of the detection thresholds to return a desired edge map without many erroneous detected edges originating from background clutter. On the other hand, the state-of-the-art edge detector Berkeley Bg operator [23] excels in quality at the expense of computational cost. We found that if we keep the image dimensions reasonably small we can benefit from the

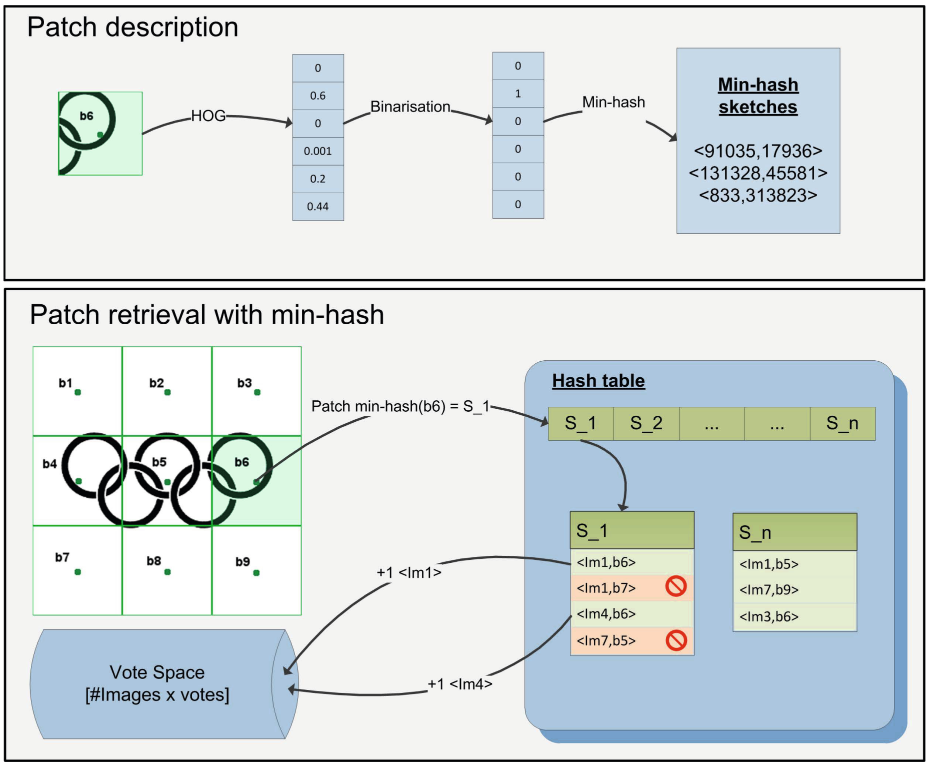

Fig. 2. Patch retrieval framework overview. (Top) Feature extraction for a single patch. The HoG descriptor is computed and then a binarization process is applied to the feature vector. A list of min-hash tuples calculated from the non-zero descriptor indexes is the final patch representation. (Bottom) The patch retrieval mechanism. Every image patch is described individually, for each computed min-hash tuple a look-up in the hash table is performed. We only allow votes originating from neighboring patches.

superior performance of BG detector without burdening much with the computational cost, hence every image is resized to a 200 × 200 square and edge detection is carried out with the BG detector in less than one second. Weak detected edges are further reduced by thresholding the returned edge map, edgels with edge intensity less than 10 are discarded. Query preprocessing is similar to photos, the edge detection step is substituted with a morphological thinning operation as a means to reduce thick drawn lines to single pixel width followed by downscaling to the same dimensions as photos.

4.3 Feature Extraction

An *overlapping spatial grid* is applied to finely describe the generated edge map and feature vectors are extracted for every patch of the grid. We set the size of spatial grid to 17 × 17 patches with each patch occupying 40 × 40 pixels.

The stride between neighbor patches is set to 10 pixels which equals to 75% overlap between them. We employed the Eitz *et al.* database [5] to tune the above parameters. Due to space limitations their impact analysis is omitted. The patch extraction process is visualized in the top part of Fig. 2. Two blocks are considered similar if they share similar shapes, *i.e.* their edges have similar orientation histogram and spatial arrangement. We quantify this similarity with the HoG descriptor [8] known to perform well in general object detection problems. The HoG algorithm further divides a patch in overlapping blocks and calculates an orientation histogram for each block. We configure the HoG to blocks of 2×2 cells, computing an 8-bin histogram of unsigned edge orientations for each cell(orientations are between $0°$ and $180°$). The resulting patch descriptor is a 512-dimensional vector. Min-hash requires each entry to be described by a set of visual words, this can be formed as a binary vector where a word presence/absence is indicated by 1/0. The HoG feature vector can be modified to abide to this scheme, indeed, without losing much information we can binarize the descriptor by setting the b% highest values to 1 and the rest to 0. The binarization process boosts the retrieval performance as it highlights the strongest patch orientations corresponding to bold continuous contours while eliminating weak responses from noisy edges. The parameter b has been empirically set to 20%. Finally, for each binarized descriptor we calculate k s-tuples of min-hash values which will be used to efficiently retrieve similar patches.

4.4 Index Construction

As discussed in Sec. 4.3, for every patch we compute k sketches of min-hash values. Every value requires 64 bits of memory, hence if we use twenty 2-tuples an image will be described with ~90.3KB of memory, a non-negligible amount especially when the database size grows. Without compression 1 million images require ~90GB of memory, a quantity beyond the memory capacity of a modern computer. The need of an index mechanism is apparent, we take advantage of the sketch collisions that will occur between similar images in a large database and we construct a hash table from the unique min-hash sketches. Every hash table entry can accommodate multiple pairs of type $<image\_id, block\_id>$. Under this scheme, memory is saved by disregarding duplicate sketch values enabling faster search times. Additionally the current patch position in the grid is stored for each entry (*block_id* field), information that can be capitalized during the query process when spatial constraints can be enforced as a means to reject non adjacent patches. Patches containing a small portion of edgels are not taken into account during the index construction.

4.5 Patch Retrieval and Voting

Images similar to a sketch query are returned based on a voting process (Fig. 2 bottom). The pipeline of the query system is as follows: given a binary drawing, features are extracted according to the process illustrated in Sec. 4.3, obviously the edge detection step is omitted. For every patch with adequate number of edgels, k s-tuples are calculated and for each tuple a look-up in the hash table is performed. Retrieved patches located at grid positions above a predefined distance threshold from the current patch center are discarded, the rest vote for their corresponding image. Over-segmentation introduced by the spatial grid leads to multiple votes emanating from one image to the same min-hash tuple. To reduce the bias of this effect an image can cast only one vote for every min-hash tuple extracted from a query patch. The final ranking is generated by counting the votes for each image and sorting them in descending order. The suggested patch based retrieval scheme enhances flexibility since look-ups are taking place only for patches that have been drawn by the user, efficiently reducing query time and facilitating real time result updating when a new stroke is drawn. The query routine can be easily parallelized to enhance scalability even further. Patch queries can be executed independently in different machines and return a vote count for every image. An integration process will then merge all the votes to generate the final ranking.

5 Results

We evaluate our system in the Eitz *et al.* dataset[5] especially designed to measure retrieval performance in a large scale environment. The database consists of 31 user drawn sketch queries outlining objects and sceneries (Fig. 1 depicts some of them). Each sketch query is associated with 40 photos assigned with a value between 1 (similar) and 7 (dissimilar). These 1240 photos are mixed with 100,000 distractor images. A SBIR algorithm must generate a ranking of the database images for each query and retrieve the order of the 40 query related photos, a correlation value between the ground truth and the returned ranking measures the retrieval accuracy. The final benchmark score is the average correlation value across the 31 queries. Eitz *et al.* reported a state-of-the-art score of 0.277 with their tensor descriptor [15] in conjunction with the BoW model. Our approach outperforms [15], we report average correlation value of 0.336. Fig. 1 demonstrates the top 5 returned photos for some of the queries, the reader can verify the spatial coherence between the query and the acquired photos achieved with our patch voting scheme.

Table 1. Results on dataset [5]

Method	Benchmark Score
Eitz *et al.* [5]	0.277
Our method	0.336

We performed the experiments in a desktop machine with 4 cores and 4GB of memory using MATLAB. Twenty 2-tuple min hash values were extracted for each patch. The database outputs ~29.2 million patches, 5.51% of them were empty or with not much information and discarded. The hash table was constructed on the remaining patches outcoming 49,535 unique keys and ~552 million entries. Each entry requires 3 bytes of storage space, therefore the hash table equips ~1.5GB of memory. Hash keys with too many entries were removed from the index achieving two goals, reduction of storage requirements and attenuation of bias caused by very common hash keys (similar to stop words in text retrieval). Building the index takes a considerable amount of time, but needs to be done only once and offline. The code was not parallelized, still a query is executed in less than $2s$. These timings are achieved through the index mechanism which performs a search for a hash key in logarithmic time ($O(log\, n)$) in the size of index.

6 Conclusions

We presented a patch based retrieval technique which can scale to millions of images with the utilization of hashing. Shape and structure information is extracted from a patch with the HoG descriptor and a binarization process further enhances strong continuous contours while facilitating the application of min-hash algorithm. An inverted index created on the unique hash key values allows for sublinear search times and parallelization. Experiments on a large scale database demonstrated the efficiency and scalability of our technique by achieving state-of-the-art results.

Future work includes the optimization of the algorithm in a parallel architecture and a patch description technique that will reduce false positive collisions. Towards this direction a feature descriptor designed to identify and penalize patches with noisy edges could be useful.

References

1. Datta, R., Joshi, D., Li, J., Wang, J.Z.: Image retrieval: Ideas, influences, and trends of the new age. ACM Comput. Surv. 40, 1–60 (2008)
2. Eitz, M., Kristian Hildebrand, T.B., Alexa, M.: A descriptor for large scale image retrieval based on sketched feature lines. In: Eurographics Symposium on Sketch-Based Interfaces and Modeling, pp. 29–38 (2009)
3. Cole, F., Golovinskiy, A., Limpaecher, A., Barros, H.S., Finkelstein, A., Funkhouser, T., Rusinkiewicz, S.: Where do people draw lines? Communications of the ACM 55, 107–115 (2012)
4. Cole, F., Sanik, K., DeCarlo, D., Finkelstein, A., Funkhouser, T., Rusinkiewicz, S., Singh, M.: How well do line drawings depict shape? ACM Transactions on Graphics (Proc. SIGGRAPH) 28 (2009)
5. Eitz, M., Hildebrand, K., Boubekeur, T., Alexa, M.: Sketch-based image retrieval: Benchmark and bag-of-features descriptors. IEEE Transactions on Visualization and Computer Graphics 17, 1624–1636 (2011)
6. Broder, A.Z., Charikar, M., Frieze, A.M., Mitzenmacher, M.: Min-wise independent permutations. Journal of Computer and System Sciences 60, 327–336 (1998)
7. Chum, O., Philbin, J., Zisserman, A.: Near duplicate image detection: min-hash and tf-idf weighting. In: Proceedings of the British Machine Vision Conference (2008)

8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, vol. 1, pp. 886–893 (2005)
9. Hirata, K., Kato, T.: Query by Visual Example. In: Pirotte, A., Delobel, C., Gottlob, G. (eds.) EDBT 1992. LNCS, vol. 580, pp. 56–71. Springer, Heidelberg (1992)
10. Lopresti, D., Tomkins, A.: Computing in the ink domain. In: Yuichiro Anzai, K.O., Mori, H. (eds.) Symbiosis of Human and Artifact - Future Computing and Design for Human-Computer Interaction. Proceedings of the Sixth International Conference on Human-Computer Interaction (HCI International 1995). Advances in Human Factors/Ergonomics, vol. 20, pp. 543–548. Elsevier (1995)
11. Liang, S., Sun, Z., Li, B.: Sketch retrieval based on spatial relations. In: Proc. Int. Computer Graphics, Imaging and Vision: New Trends Conf., pp. 24–29 (2005)
12. Del Bimbo, A., Pala, P.: Visual image retrieval by elastic matching of user sketches. IEEE Transactions on Pattern Analysis and Machine Intelligence 19, 121–132 (1997)
13. Matusiak, S., Daoudi, M., Blu, T., Avaro, O.: Sketch-Based Images Database Retrieval. In: Jajodia, S., Özsu, M.T., Dogac, A. (eds.) MIS 1998. LNCS, vol. 1508, pp. 185–191. Springer, Heidelberg (1998)
14. Chalechale, A., Naghdy, G., Mertins, A.: Sketch-based image matching using angular partitioning. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 35, 28–41 (2005)
15. Eitz, M., Hildebrand, K., Boubekeur, T., Alexa, M.: A descriptor for large scale image retrieval based on sketched feature lines. In: Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling, SBIM 2009, pp. 29–36. ACM, New York (2009)
16. Sivic, J., Zisserman, A.: Video Google. In: Proceedings of the International Conference on Computer Vision, vol. 2, pp. 1470–1477 (2003)
17. Hu, R., Barnard, M., Collomosse, J.: Gradient field descriptor for sketch based retrieval and localization. In: Proc. 17th IEEE Int. Conf. Image Processing, ICIP, pp. 1025–1028 (2010)
18. Hu, R., Wang, T., Collomosse, J.: A bag-of-regions approach to sketch-based image retrieval. In: 2011 18th IEEE International Conference on Image Processing, ICIP, pp. 3661–3664 (2011)
19. Cao, Y., Wang, C., Zhang, L., Zhang, L.: Edgel index for large-scale sketch-based image search. In: 2011 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, pp. 761–768 (2011)
20. Lee, Y.J., Zitnick, C.L., Cohen, M.F.: Shadowdraw: real-time user guidance for freehand drawing. ACM Trans. Graph. 30, 27 (2011)
21. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, STOC 1998, pp. 604–613. ACM, New York (1998)
22. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. 8, 679–698 (1986)
23. Martin, D., Fowlkes, C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE Transactions on Pattern Analysis and Machine Intelligence 26, 530–549 (2004)