Sketch-Based Retrieval Using Content-Aware Hashing

Shuang Liang[1,⋆], Long Zhao[1], Yichen Wei[2], and Jinyuan Jia[1]

[1] Tongji University, Shanghai, China
{shuangliang,9012garyzhao,jyjia}@tongji.edu.cn
[2] Microsoft Research Asia, Beijing, China
yichenw@microsoft.com

Abstract. In this paper, we introduce a generic hashing-based approach. It aims to facilitate sketch-based retrieval on large datasets of visual shapes. Unlike previous methods where visual descriptors are extracted from overlapping grids, a content-aware selection scheme is proposed to generate candidate patches instead. Meanwhile, the saliency of each patch is efficiently estimated. Locality-sensitive hashing (LSH) is employed to integrate and capture both the content and saliency of patches, as well as the spatial information of visual shapes. Furthermore, hash codes are indexed so that a query can be processed in sub-linear time. Experiments on three standard datasets in terms of hand drawn shapes, images and 3D models demonstrate the superiority of our approach.

Keywords: sketch-based retrieval, LSH, content-aware windows.

1 Introduction

Using sketch as input to retrieve visual information, such as hand drawn shapes, images and 3D models, has attracted a lot of research interests in recent years. A sketch is regarded as a collection of hand drawn strokes representing the contour or skeleton of an object, while detailed appearance information are lost. As a result, sketch-based retrieval algorithms are usually quite different from traditional image retrieval systems. One research direction for sketch-based retrieval aims to produce sketch-like edge maps from images or 3D models. Such works focus on removing edge noises such as background clutters in an image [10,20,21] or selecting suitable viewpoints for 3D models [8]. In this paper, we try to answer another question: *How to extract and compare features carried by a sketch (or a sketch-like edge map) in a proper way?* Therefore, in the following we briefly discuss previous works from the two viewpoints of interest in this paper: the ways to extract sketch features and methods to measure feature similarities.

Recently, segmentation-based methods [9,11,12] are proved effective for sketch retrieval and recognition systems with line drawings. Stroke segmentation is usually involved to lower the computational complexity. Topology/geometry attributes are then calculated from the extracted segments. However, an accurate

⋆ Corresponding author.

W.T. Ooi et al. (Eds.): PCM 2014, LNCS 8879, pp. 133–142, 2014.

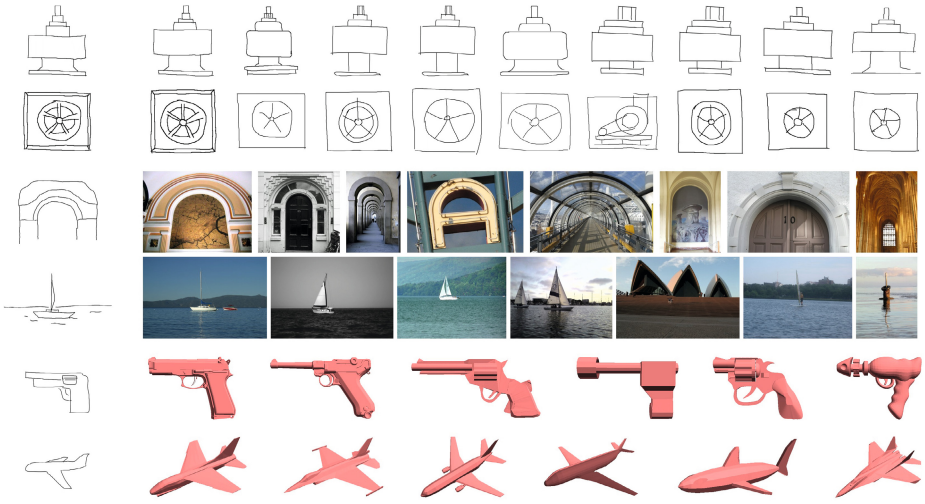

Fig. 1. Example results of our approach on the Magic Sketch Database [11] (top), TU Berlin Benchmark [7] (middle) and PSB Dataset [8] (bottom)

stroke segmentation is very hard to achieve. Especially, for natural images and 3D models where usually a lot of noises exist in their edge maps, there is almost no perfect segmentation methods. This limits segmentation-based methods unsuitable for generic sketch-based retrieval tasks.

Other works [2,7,8,14,21] originates from image retrieval. A sketch is divided into patches and visual descriptors are then extracted. Overlapping grids [2] or dense windows [14] are usually applied to describe the distribution of features over the whole shape. This is suitable for an image filled with details as colors and textures, but inappropriate for sparse edge maps, e.g., a sketch with a few strokes. In such cases, most of the content of a sketch is gathered into several salient patches while other patches are left almost empty. Due to this imbalance, it is really ineffective and even harmful when these empty patches are used for similarity computation, especially, when they are binarized before hashing.

Another issue in patch-based methods is that the similarity measurement for sketches is weak. As sketches are hand drawn artifacts with various line styles to represent objects other than colors and textures, they are different from images in two points: large intra-class differences (because of painters' subjective understanding) and small inter-class differences (due to the loss of visual information like colors and textures). Similar sketches may still have a lot of different patches. Therefore, the common concept (*images are similar when most patches are similar*) used in image retrieval and adopted by current patch-based methods is too strict and thus ineffective.

In this paper, we present a generic, content-aware sketch-based retrieval framework that overcomes the above issues in current patch-based methods by taking the sketch content distribution into consideration. Fig.1 shows some results of

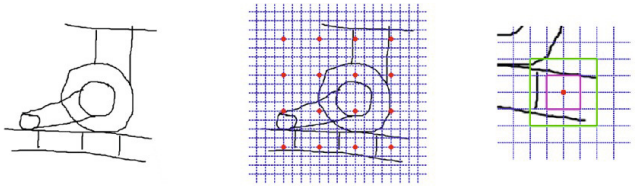

Fig. 2. Example of window selection. (a) input sketch; (b) an initial $n \times n$ grid and $m \times m$ uniformly sampled seeds; (c) the first $\delta w(x, y, 1)$ (magenta) and second $\delta w(x, y, 2)$ (green) rims of windows for seed (x, y). The ith $\delta w(x, y, i)$ is added to $w(x, y)$ iteratively.

our sketch-based retrieval on hand drawn shapes, images and 3D models. First, a window[1] selection scheme with regard to sketch content distribution is proposed to generate candidate patches. It ensures that features carried by a sketch are uniformly distributed in all patches. Second, we refine the common image retrieval concept in a more reasonable way: *sketches are similar when their most salient patches are similar*. Thus we introduce a salient window detection algorithm, which helps to compute shape similarity. We then follow hash-based retrieval approaches [2,16,23] and employ a locality-sensitive hashing (LSH) method, which combines the above cues together with structural information. To be applicable to large datasets, indexing is performed as well to enable sub-linear runtime performance. The rest of our paper is organized as follows. Sec. 2 presents the detailed algorithm, and our whole retrieval framework is introduced in Sec. 3. Finally, all experimental results are shown in Sec. 4.

2 Proposed Algorithm

Our proposed approach consists of three components: selecting candidate windows with regard to sketch content, detecting salient windows adaptively using key points, and combining these two features together with spatial cues into hash codes. Detailed algorithms of each step are described as follows.

2.1 Content-Aware Window Selection

Given a sketch, we begin by dividing it into a $n \times n$ grid. Then $m \times m$ seeds are uniformly sampled at the crossing points of the patches, as shown in Fig. 2(b). We define a set of windows $\delta w(x, y, i)$ as the ith-rim windows around the seed (x, y), as shown in Fig. 2(c). In order to generate a proper window, $\delta w(x, y, i)$ should be added to $w(x, y)$ iteratively until certain constraints are satisfied or $w(x, y)$ becomes invalid, e.g., it flows over the whole sketch or it is larger than a quarter of the sketch. We present two effective constraints as follows.

[1] We use "patches" and "windows" interchangeably in this paper.

Algorithm 1. Selecting candidate windows for the input sketch.

initialize $n \times n$ spatial grid for the input sketch
initialize $m \times m$ seeds uniformly sampled from the grid
initialize histogram $\boldsymbol{h}_i$ for each window w_i
initialize global histogram $\boldsymbol{\mathcal{H}} = \sum_{i=1}^{n^2} \boldsymbol{h}_i$
$W \leftarrow \{\}$, $H \leftarrow \{\}$
for $y = 1$ to m **do**
  for $x = 1$ to m **do**
    $w \leftarrow \{\}$, $\boldsymbol{h} \leftarrow \boldsymbol{0}$
    for $i = 1$ to $n/4$ **do**
      $w \leftarrow w + \delta w(x, y, i)$, $\boldsymbol{h} \leftarrow \boldsymbol{h} + \boldsymbol{\delta h}(x, y, i)$
      if $\mathcal{F}_{app}(\boldsymbol{h}) \geq k_{app} \times \mathcal{F}_{app}(\boldsymbol{\mathcal{H}})$ **then**
        if $\mathcal{F}_{var}(\boldsymbol{h}) \leq k_{var} \times \mathcal{F}_{var}(\boldsymbol{\mathcal{H}})$ **then**
          break
        end if
      end if
    end for
    $W \leftarrow W + w$, $H \leftarrow H + \boldsymbol{h}$
  end for
end for
return W and H

Since the histogram of oriented gradients (HoG) [5] is known to perform well in object detection and image retrieval problems, we use its unnormalized version to describe the feature inside windows for fast computation. Let $\boldsymbol{h} = \{b_1, \ldots, b_n\}$ denote the feature histogram of window w, $\boldsymbol{\delta h}(x, y, i)$ denote corresponding histograms of $\delta w(x, y, i)$ and $\boldsymbol{\mathcal{H}}$ denote the global histogram of the whole sketch. We define the *appearance constraint* C_{app} as

$$\mathcal{F}_{app}(\boldsymbol{h}) \geq k_{app} \times \mathcal{F}_{app}(\boldsymbol{\mathcal{H}}), \quad where \quad \mathcal{F}_{app}(\boldsymbol{h}) = \frac{1}{n} \sum_{i=1}^{n} b_i \tag{1}$$

$\mathcal{F}_{app}$ is the appearance objective function, which essentially computes the mean value of $\boldsymbol{h}$. When $\mathcal{F}_{app}$ is high, it shows information carried by the window is relatively large, while low $\mathcal{F}_{app}$ indicates that the window is almost empty. Apparently, C_{app} constrains each window to contain enough information as expcted. Our second constraint is called *variety constraint* C_{var}, which is formulated as

$$\mathcal{F}_{var}(\boldsymbol{h}) \leq k_{var} \times \mathcal{F}_{var}(\boldsymbol{\mathcal{H}}), \quad where \quad \mathcal{F}_{var}(\boldsymbol{h}) = \frac{1}{n} \sum_{i=1}^{n} (b_i - \mathcal{F}_{app}(\boldsymbol{h}))^2 \tag{2}$$

$\mathcal{F}_{var}$ is the variety objective function, which is the variance of $\boldsymbol{h}$ indeed. If $\mathcal{F}_{var}$ is low and $\mathcal{F}_{app}$ is high, it means that all bins in $\boldsymbol{h}$ have comparatively high values. As a result, windows satisfying C_{var} should have more diverse information, which is proved to be useful in sketch retrieval [14]. Note that parameters k_{app} and k_{var} control the global effects of $\boldsymbol{\mathcal{H}}$, and they are set to 0.8 and 1 experimentally. The whole algorithm is summarized in Alg. 1.

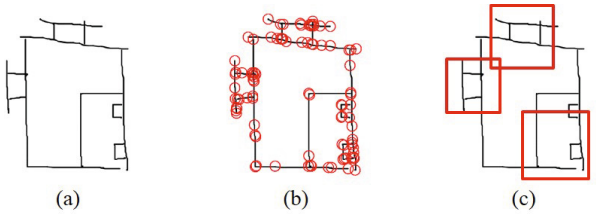

Fig. 3. Salient windows of a sketch. (a) input sketch; (b) salient points detected by Harris corner; (c) top 3 salient windows computed by Eq. 3 with overlapping area less than 20%.

We note that the objective function $\mathcal{F}_{app}$ of C_{app} is *incrementally computable*, for it satisfies the equation: $\mathcal{F}_{app}(\boldsymbol{h} + \boldsymbol{\delta h}) = \mathcal{F}_{app}(\boldsymbol{h}) + \mathcal{F}_{app}(\boldsymbol{\delta h})$. As the window grows, it is unnecessary to compute $\mathcal{F}_{app}$ across the entire histogram but sufficient to only update the affected part. Therefore, to evaluate C_{app} in each iteration is considerably fast. Though F_{var} is not incrementally computable, it does not need to be evaluated until C_{app} is satisfied. So our algorithm still has competitive computational speed. Experiments show that using candidate windows generated by our algorithm can improve retrieval results achieved by many traditional visual descriptors such as SIFT descriptor [13], HoG [5] and GALIF [8]. Detailed results are shown in Sec. 4.1.

2.2 Salient Window Detection

For local feature detection, keypoint-based detectors, such as difference of Gaussian (DoG) [13], Hessian operator [1] and Harris-Laplace detector [15], are more suitable for sketch feature extraction than region-based methods [17,24]. This is because a sketch usually includes separate lines and points other than continuous areas. These keypoint-based methods are mainly designed for finding salient points, which correspond to the corners and end points in sketches. Harris corner detector is employed in our method, for it achieves better performance than other keypoint-based methods as shown in [14].

For each window $w_i \in W$, where W is a set of candidate windows output by Alg. 1, we compute k_i to measure the saliency of w_i in a sketch as

$$k_i = 1 + \frac{Number(S_i)}{\sqrt{Area(w_i)}} \tag{3}$$

where $Number(S_i)$ denotes the number of salient points in w_i detected by Harris corner detector and $Area(w_i)$ is the pixel-based area of w_i. It has the intuitive interpretation that *a window is salient when it is small while contains many salient points*. Note that we use the square root of the area to make it less sensitive to scales: the measure remains stable across different window resolutions. See Fig. 3 for visual examples of salient windows detected in a sketch.

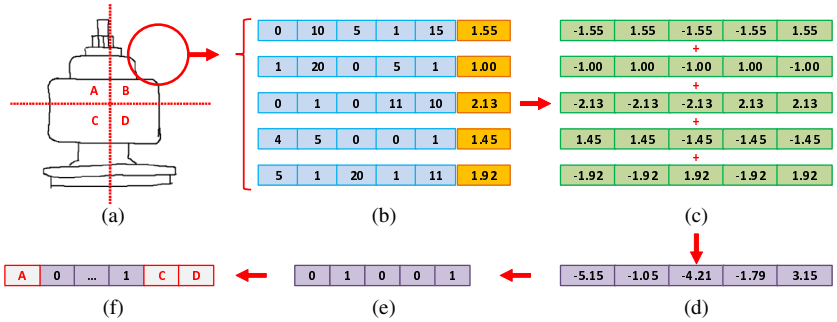

Fig. 4. Example of hashing. (a) input sketch partitioned into 4 parts spatially; (b) feature vectors $\boldsymbol{f_i}$ (blue) and k_i (orange) of all windows in part B; (c) weighted binary vectors by $\bar{\boldsymbol{f}}_{\boldsymbol{i}} \times k_i$; (d)-(f) vectors are summed, binarized and appended into the final hash code.

2.3 Hashing from Window Content and Saliency

Let $\boldsymbol{f_i}$ denote the feature vector extracted from w_i using certain visual descriptor (HoG in this paper). $\boldsymbol{f_i}$ is then binarized into $\bar{\boldsymbol{f}}_{\boldsymbol{i}}$ by setting the highest 40% values to 1 and the rest to -1. Note that thanks to the two constraints proposed in Sec. 2.1, features are evenly distributed and message loss can be reduced in this step. Then we follow the manner of Sim-hash [4] to produce locality-sensitive hashing for each sketch via $\bar{\boldsymbol{f}}_{\boldsymbol{i}}$ and k_i, where window saliency k_i is used as a weighting term for $\bar{\boldsymbol{f}}_{\boldsymbol{i}}$. Spatial information is shown to be helpful to sketch retrieval systems [6]. In order to capture spatial information of local features, we partition each sketch image into four parts. Other than hashing patch features in the entire sketch space, we do it in each part and finally append them together (a window is considered to belong to a certain part when its center is inside that part). Detailed steps are explained in Fig. 4.

3 Retrieval Framework

Before feature extraction and hashing, images and 3D models are converted into edge maps. Given an image, we generate a coarse edge map E_c by Canny algorithm [3], which yet contains many erroneous detected edges from background clutter. In order to find the most salient area in the image, we use [24] to get its saliency map S. We also apply maximum filter $\mathcal{MF}$ on S to enlarge salient regions to avoid the degenerate case when object contours are missing due to segmentation errors. Then the final salient edge map E is computed as $E = E_c \times \mathcal{MF}(S)$, where E is binarized by threshold 0.5 empirically. Fig. 5 presents our final edge maps. Given a 3D model, we follow the pipeline in [8] to get its projection edge maps.

Both sketches and edge maps are cut out of their minimum square bounding boxes and resized into 160×160 resolution to be translation and scale invariant.

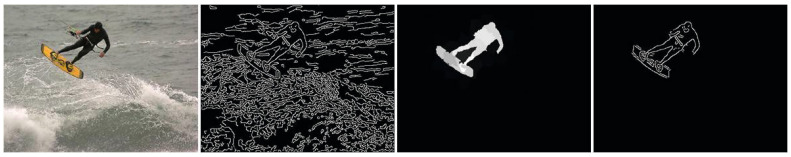

Fig. 5. Example of an image edge map. From left to right: input image, Canny lines [3], saliency map detected by [24] and our final edge map.

Algorithm 2. Pipeline of the retrieval framework.

Index:
1. produce edge maps E for all images/3D models in datasets
2. generate candidate windows w_i for E according to Alg. 1
3. compute HoG feature $\boldsymbol{f_i}$ and saliency k_i for each w_i according to Sec. 2.2
4. hash $\boldsymbol{f_i}$ and k_i into $\boldsymbol{h}$ according to Sec. 2.3
5. follow [16] to generate index I for all $\boldsymbol{h}$

Query:
1. follow the step 2-4 of **Index** to generate $\boldsymbol{h_s}$ for input sketch S
2. return similarity ranking R by $\boldsymbol{h_s}$ from I as query results

Following Alg. 1, a sketch is divided into a 80×80 grid, and 15×15 seeds are uniformly sampled. In each patch, we compute the unnormalized HoG with 8 orientations and produce a 8-bin histogram. After all candidate patches are generated, they are resized to 16×16. The HoG descriptor is applied to extract features in each patch, for its unnormalized version has already been calculated.

Given a query, we find the most similar edge maps via the hash code proposed in Sec. 2.3 based on their *Hamming distance*. For all hash codes are binary, their Hamming distances can be calculated by several bit *xor* and *shift* operations, which is competitively fast even without indexing. In order to gain better runtime performance, we further follow the indexing strategy proposed by [16], which can be combined with our approach directly. It enables our retrieval process to be performed in sub-linear time. The pipeline of our retrieval framework is summarized in Alg. 2.

4 Experimental Results

In this section, extensive experiments are conducted to compare the proposed algorithm with other state-of-the-arts for sketch-based retrieval tasks. All experiments are done on the following three standard datasets. 1) *Magic Sketch Database*: Liang et al. [11] establish this database with a total of 1100 sketches from 55 classes drawn by 10 people. We use it to evaluate variants of our approach. 2) *TU Berlin Benchmark*: This is a sketch-based image retrieval benchmark introduced by Eitz et al. [7] which consists of 31 subjects, each one including 1 sketch and 40 corresponding test images. We use this benchmark to evaluate our approach for sketch-based image retrieval tasks. 3) *PSB Dataset*:

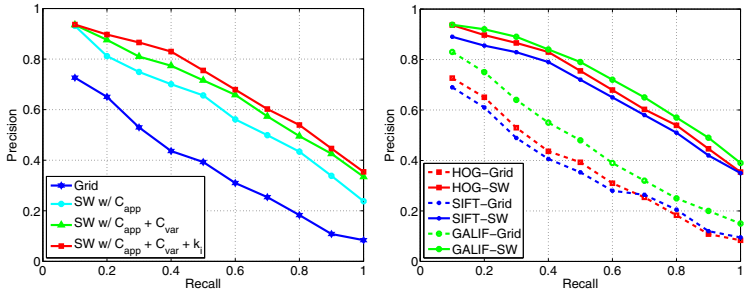

Fig. 6. Left: Evaluation of each component in our method. Right: Improvement of different visual descriptors caused by our method.

Eitz et al. [8] perform a large-scale experiment to collect 1,914 sketches for all 3D model categories in the Princeton Shape Benchmark (PSB) [19]. Our approach is evaluated on it to show the performance in sketch-based 3D model retrieval tasks.

4.1 Evaluation of Our Approach

Our method spends about 1.87 seconds to retrieve a sketch on the Magic Sketch Database, tested on a desktop computer with Intel 3.39GHz Quad-core CPU and 16GB memories and implemented by MATLAB without parallelized.

To evaluate the effectiveness of selecting windows with constraints C_{app} (Eq. 1) and C_{var} (Eq. 2), we implement a baseline using overlap grid (Grid) according to [2]. Then results via selected windows (SW) with C_{app} only and both of them are compared. To evaluate the performance of window saliency k_i (Eq. 3), we equally set window saliency by 1 in previous results for comparison. Results in Fig. 6(left) shows that our two constraints are complementary, and removing any of these three components would cause performance drop.

To further understand our method, we evaluate results using traditional visual descriptors such as SIFT [13] and GALIF [8] in addition to HoG [5]. Fig. 6(right) shows the performance of their grid-based version and the other one with our constrainted selected windows (SW). All of them are improved by our approach. We note that although GALIF achieves the best performance due to its multi-scale sampling strategy other than histogram-based method, it requires more time to be calculated. Therefore, HoG is employed in our approach.

4.2 Comparison with State-of-the-Art

On the TU Berlin Benchmark, we compare our algorithm with other methods using Bag-of-Words (BW) [7], key shapes (KS) [18] and Min-hash (MH) [2]. We use the benchmark scores introduced by [7] to evaluate the performance. On the Magic Sketch Database, methods via biased SVM (BSVM) [11], spatial relations

Table 1. Results on the TU Berlin Benchmark [7]

Method	BW[7]	KS[18]	MH[2]	Ours
Score	0.277	0.289	0.336	**0.352**

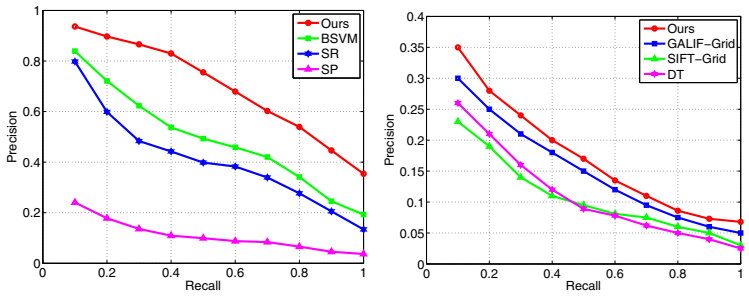

Fig. 7. PR curves of different algorithms on the Magic Sketch Database [11] (left) and PSB Dataset [8] (right)

(SR) [12] and spatial proximity (SP) [9] are compared with ours. On the PSB Dataset, we compare ours with methods via diffusion tensor (DT) [22], grid-based SIFT (SIFT-Grid) and GALIF (GALIF-Grid) [8]. All results on these two datasets are evaluated by standard Precision-Recall (PR) Curves. Both results in Table 1 and Fig. 7 show that our approach outperforms other state-of-the-arts. Some visual retrieval results of our approach are shown in Fig. 1.

5 Conclusion

In this paper, we present a novel sketch-based retrieval framework using content aware hashing. Feature window patches are selected with appearance and variety constraints to account for content distribution. A saliency value of each window is then calculated. For feature extraction, hashing is performed to combine content, saliency and spatial information of visual features and enables efficient retrieval in sub-linear time. Experimental results on sketch, image, and 3D model datasets demonstrate our proposed algorithm performs favorably against other alternatives for sketch-based retrieval systems. Future work includes exploring more effective visual descriptors for sketches.

Acknowledgement. This research work was supported by the National Science Foundation of China (No.61272276, No. 61305091), the National Twelfth Five-Year Plan Major Science and Technology Project of China (No.2012BAC11B01-04-03), Special Research Fund of Higher Colleges Doctorate (No. 20130072110035), the Fundamental Research Funds for the Central Universities (No.2100219038), and Shanghai Pujiang Program (No.13PJ1408200).

References

1. Bay, H., Tuytelaars, T., Van Gool, L.: Surf: Speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part I. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
2. Bozas, K., Izquierdo, E.: Large Scale Sketch Based Image Retrieval Using Patch Hashing. In: Bebis, G., et al. (eds.) ISVC 2012, Part I. LNCS, vol. 7431, pp. 210–219. Springer, Heidelberg (2012)
3. Canny, J.: A Computational Approach to Edge Detection. IEEE Transactions on PAMI Pattern Analysis and Machine Intelligence 8(6), 679–698 (1986)
4. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: STOC (2002)
5. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)
6. Eitz, M., Hays, J., Alexa, M.: How Do Humans Sketch Objects? In: SIGGRAPH (2012)
7. 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(11), 1624–1636 (2011)
8. Eitz, M., Richter, R., Boubekeur, T., Hildebrand, K., Alexa, M.: Sketch-Based Shape Retrieval. In: SIGGRAPH (2012)
9. Fonseca, M., Ferreira, A., Jorge, J.: Sketch-based Retrieval of Vector Drawings. In: Sketch-based Interfaces and Modeling, pp. 181–201. Springer, London (2011)
10. Furuya, T., Ohbuchi, R.: Visual saliency weighting and cross-domain manifold ranking for sketch-based image retrieval. In: Gurrin, C., Hopfgartner, F., Hurst, W., Johansen, H., Lee, H., O'Connor, N. (eds.) MMM 2014, Part I. LNCS, vol. 8325, pp. 37–49. Springer, Heidelberg (2014)
11. Liang, S., Sun, Z.: Sketch retrieval and relevance feedback with biased SVM classification. Pattern Recognition Letters 29(12), 1733–1741 (2008)
12. Liang, S., Sun, Z., Li, B.: Sketch Retrieval Based on Spatial Relations. In: CGIV (2005)
13. Lowe, D.: Distinctive image features from scale-invariant keypoints. IJCV 42(3), 145–175 (2001)
14. Ma, C., Yang, X., Zhang, C., Ruan, X., Yang, M.H.: Sketch Retrieval via Dense Stroke Features. In: BMVC (2013)
15. Mikolajczyk, K., Schmid, C.: Scale and Affine Invariant Interest Point Detectors. IJCV 60(1), 63–86 (2004)
16. Norouzi, M., Punjani, A., Fleet, D.J.: Fast Search in Hamming Space with Multi-Index Hashing. In: CVPR (2012)
17. Perazzi, F., Kr, P., Krahenbuhl, P., Pritch, Y., Hornung, A.: Saliency Filters: Contrast Based Filtering for Salient Region Detection. In: CVPR (2012)
18. Saavedra, J.M., Bustos, B.: Sketch-based image retrieval using keyshapes. In: Multimedia Tools and Applications, pp. 1–30. Springer, US (2013)
19. Shilane, P., Min, P., Kazhdan, M., Funkhouser, T.: The Princeton Shape Benchmark. In: Shape Modeling International (2004)
20. Sun, X., Wang, C., Xu, C., Zhang, L.: Indexing Billions of Images for Sketch-based Retrieval. In: ACM Multimedia (2013)
21. Wang, C., Cao, Y., Zhang, L.: MindFinder: A Sketch-based Image Search Engine based on Edgel Index. In: CVPR (2011)
22. Yoon, S.M., Scherer, M., Schreck, T., Kuijper, A.: Sketch-based 3D model retrieval using diffusion tensor fields of suggestive contours. In: ACM Multimedia (2010)
23. Zhang, L., Zhang, Y., Tang, J., Lu, K., Tian, Q.: Binary Code Ranking with Weighted Hamming Distance. In: CVPR (2013)
24. Zhu, W., Liang, S., Wei, Y., Sun, J.: Saliency Optimization from Robust Background Detection. In: CVPR (2014)