Sketch based Image Retrieval using Learned KeyShapes (LKS)

Jose M. Saavedra
jose.saavedra@orand.cl
Juan Manuel Barrios
juan.barrios@orand.cl
Orand S.A.
Estado 360, Of. 702-A,
Santiago, Chile

Abstract

Sketch based image retrieval is a particular case of the image retrieval problem, in which a query is not a regular example image. Instead, the query is a hand-drawn sketch representing what the user is looking for. This kind of problem has a lot of applications, in particular when an example image is not available. For instance, in searching for design pieces in digital catalogs. The natural ambiguity of sketches as well as the poor skills of drawing make the problem very challenging, which is reflected in the low performance achieved by current methods. In this work, we present a novel method for describing sketches based on detecting mid-level patterns called *learned keyshapes*. Our experiments were performed in two datasets, one with 1326 images and the other with approximately 15k images. Our results show an increase of effectiveness around 17% on the smaller dataset and 98% on the larger one, which represent new state-of-the-art performance in the sketch based image retrieval domain. We also show that our method allows us to achieve good performance even when we use around 20% of the sketch content.

1 Introduction

In the last years, Internet has undergone a considerable growth of multimedia content which includes photos, videos, music, etc. The proliferation of mobile devices and the expanded use of social networks have played an important role in this phenomenon. Considering the huge amount of images on the web, users are requiring new effective an efficient technology to interact with that kind of content. A content based image retrieval (CBIR) system is an important application in this domain, which allows users to search large catalogs using a query example as input. A vast number of research has been devoted to content based image retrieval [2],[22], leading to very effective results on large datasets. For instance, *Google Images* is a good representative of this kind of system.

Unfortunately, the usability of a CBIR system is limited by requiring an example image as query. How could we deal with scenarios where we need to find some image, but we do not have an example query to do it?. For instance, suppose that a someone is redesigning his living room, he would like to buy a lamp with a certain shape; he could have the idea of what he wants to get, but he does not have an image resembling what he needs. A classical content based image retrieval system is not possible in this scenario.

© 2015. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

Pages 164.1-164.11
DOI: https://dx.doi.org/10.5244/C.29.164

A natural alternative for querying in an image retrieval system is by simply drawing what the user has in mind. Indeed, drawing was the primitive means of communication between humans. One of the goals of an image retrieval scenario is to provide users a simple modality for querying. Thereby, in this context, a drawing means a simple hand-drawn sketch composed only of strokes that users can do easily, lacking color or texture. In this regard, although color may provide relevant information for the retrieval task [27], the use of this attribute is beyond the scope of this work. Examples of hand-drawn sketches are shown in Figure 1. This kind of querying modality leads to the *sketch based image retrieval problem* (SBIR), that is also supported by the widespread touchscreen technology which allows users to make sketches easily. Even, the task of drawing may result very amused for users. Nonetheless, dealing with sketches in the image retrieval domain is a challenging problem because of two main reasons: First, images that we want to retrieve are not sketches, requiring an intermediate step to pass from images to sketches. Second, query sketches show certain level of ambiguity by nature that may make a method get confused easily. Indeed, a sketch exhibits different kinds of variations based, particularly, on non-rigid transformations. In addition, sometimes a sketch produced by an user will look very rough because of poor drawing skills or limited time for drawing.

Figure 1: Examples of hand-drawn sketches. The last two are from the Eitz's dataset [14].

There are many problems that may take advantage of the SBIR applications. A potential application is, for instance, to promote the cognitive development of children who have not yet acquired writing skills. In this way, a SBIR system will allow children to draw simple shapes and the SBIR system will return a set of images resembling what the child drew. This may allow children to associate abstract shapes with real objects, improving their ability to understand the real environment. Another interesting application is for searching in product catalogs when we have not any image resembling what we want to get.

In the last few year, we have seen different approaches addressing the SBIR problem [13, 15, 25], showing still low performance in public datasets. The current approaches are based on low level features like gradient orientation histograms that in some cases are aggregated using a *bag of feature* framework [13, 15]. However, these approaches are far from the way in which our brain works to interpret patterns. Indeed, diverse studies in neurosciences revel that our brain constructs complex representations from the signal received on the retina [10, 23]. In addition, the work of Hubel and Wiesel [10] about *visual perception* also revels the existence of complex cells specialized in detecting oriented patterns similar to those that we could find in sketches. These complex cells can be related to finding higher-level patterns in sketches.

Therefore, taking some ideas of the human visual perception we present a novel method for sketch based image retrieval. Our method, is based on detecting the occurrence of mid-level patterns on a sketch. To this end, we figure out, by means of an unsupervised learning process, a set of patterns that we called *learned keyshapes*. We then build a histogram that counts the occurrences of the patterns in the sketch. The histogram is built using soft-voting, spatial division [17] and squared root normalization [2]. We show that our method improves the effectiveness in two available datasets. Furthermore, our

proposal doubles the precision achieved by current methods [15, 25].

This document is organized in five sections. Section 2 presents the related work, Section 3 describes the proposal in detail, Section 4 discusses the experiments and parameter analysis, and finally, Section 5 shows the conclusions.

2 Related Work

Early works on sketch based image retrieval were based on global representations, using for instance the distribution of edge pixels [7] or using elastic contours [11]. Others approaches transform the query sketch into a colorful image (*image montage*) [8, 17] to proceed with a classical CBIR strategy. The montage process involves an extra cost that may be impractical in real environments. Following the advances in computer vision, many researcher have also addressed the SBIR problem applying local features which are then aggregated by the bag of features (BoF) framework. In this vein, Eitz *et al.* [13] proposed two techniques based on *SIFT* descriptors [19] and *Shape Context* descriptors [4].

In contrast of regular images, like photos, a sketch undergoes a sparseness condition, where a great part of the image belong to the background. To deal with this problem Hu and Collomosse [15, 16] proposed to represent a sketch by a gradient field (GF) image. The GF images are then used to compute HOG descriptors in different scales. After that, a BoF model is applied to form a frequency histogram. To obtain the GF image, the Hu's method requires solving a sparse linear equation system where the number of variables in the equation is the order of the size of the input image. In addition, Hu and Collomosse proposed an large dataset containing 14660 images and 329 query sketches. To our knowledge this dataset represent the largest public dataset in the context of SBIR. Yan Cao et al. [6] also presented a method based on the Chamfer Distance [5]. Even though the authors present a technique to deal with large database based on the inverted index structure, the proposed method does not present any kind of invariance.

The majority of SBIR methods are based on histograms of orientations either to compute a global representation or a local one. In this context, HOG [9] seems to be the favorite descriptor in the community. For instance, Arandjelović and Zisserman [1] use a multi-scale HOG for representing boundaries in the context of smooth object retrieval. However, because of the sparseness of sketches, HOG descriptors may be also sparse which may impact negatively in the final effectiveness. To address this problem, Saavedra and Bustos proposed the HELO (histogram of edge local orientations) descriptor [24], where the orientation histogram is formed by local orientations that are estimated by grouping pixels in cells and determining just one representative orientation for each cell. The representative gradient is computed using the *square gradient* approach. In a recent work [25], Saavedra presented SHELO, a improved version of HELO, where a soft computation for computing representative gradient as well as for computing spatial division are applied. It has been demonstrated that SHELO shows an outstanding performance over HOG and HELO in the context of sketch based image retrieval. However, the performance of SHELO seems to be comparable to the performance of GF-HOG when it is evaluated in the Flickr15k dataset.

A critical problem with the sketch based image retrieval is about the disparity between a test image and a query sketch. The first one is a regular image, containing a lot of details, but the second one is a more abstract representation containing only strokes. To address this disparity, an edge detector is commonly applied on the test images. The Canny detector is commonly used for this task. However, Canny method produces high response in front of high variability in the image, leading to a noisy representation. Nonetheless, the problem of getting reliable edges from images has been studied by many researchers [3, 20], who have proposed effective by costly methods. In this vein, Lim *et al.* [18] have recently presented an efficient mid-level based method for detecting contour in images that we can use for converting images into a sketch-like representation.

To the best of our knowledge, the proposed SBIR methods are based on low level features that could then be aggregated using techniques as the *bag of features*. Different from these traditional approaches, we present a novel method that exploits mid-level patterns on sketches. These patterns are

presented as *learned keyshapes*. The idea of using *keyshapes* for sketches was previously studied by Saavedra and Bustos [26]. In that work, they proposed to use six fixed *keyshapes* that rarely represent the variety of shapes that we could find in sketch images. Instead, we figure out *keyshapes* by clustering on a huge amount of sketch patches. In this way, we present state-of-the-art results using our *keyshape*-based approach (LKS) showing an effectiveness increase of around 98% in the Flickr15k dataset and around 17% in the Saavedra's dataset.

3 Learned KeyShape (LKS) approach

Our proposal consists of two stages (2). The first one is an offline process that is devoted to figure out a set of *keyshapes*. The second one, is dedicated to generate the LKS descriptors based on the detected set of keyshapes. In the following, we describe in detail each of the steps involved in this proposal.

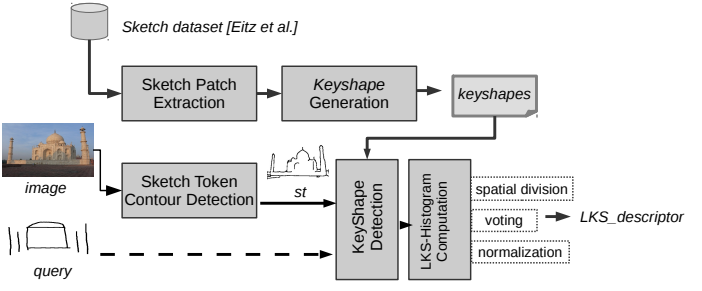


Figure 2: A scheme of our Learned KeyShapes based proposal (LKS).

3.1 KeyShape Generation

To address this task, we got inspired by the work of Lim *et al*. [18], where reliable contour maps are obtained by detecting a set of edge structures called *sketch tokens*, that are learned from a training collection of contour images. For learning sketch tokens, Lim *et al*. used the *Berkeley Segmentation Dataset*(BSD500) [3, 21]. In our case, as our goal is to detect *keyshapes* on sketches, we require a different training dataset, one representing hand-drawn sketches instead of contour maps. Thereby, to accomplish this task, we use the sketch dataset proposed by Eitz *et al*. [14] in the context of sketch classification, which contains 20000 sketches organized in 200 different classes. Using this dataset we extract one million of 31×31 sketch patches, each one centered in a stroke point.

Following the work of Lim *et al*. we represent each patch by DAISY descriptors [28] setting the parameters as RQ=2 (radius quantization) , TQ=8 (angular quantization), and HQ=8 (histogram quantization). The descriptor is computed with respect to the center of the patch which yields a 136-dimensional descriptor ($(2 \times 8 + 1) \times 8 = 136$). DAISY was chosen because of its effectiveness and efficiency for computing local descriptors on a dense set of points. In our case, DAISY descriptors will be computed for each stroke point on a sketch image during the search process. The DAISY descriptors computed for the training patches are then clustered by K-means. We run experiment with different values of K (number of clusters). In Figure 3, we show a sample of learned *keyshapes* using $K = 150$.

It is also important to mention that the dataset for discovering *keyshapes* is independent on the datasets used for evaluation. This allows us not only to avoid any possible bias in the learning process but also to learn more general *keyshapes*.

Finally, each cluster C_i is represented by its mean μ_i, and the distance r_i of the farthest point in C_i to μ_i. In this sense, the set of K clusters are represented by the following:

$$Clusters_K = \{(\mu_1, r_1), \cdots, (\mu_K, r_K)\} \tag{1}$$

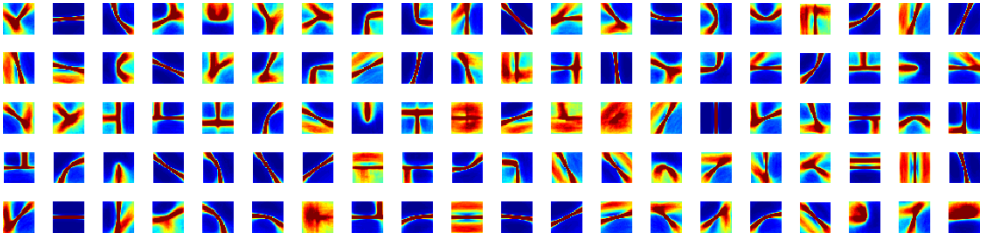
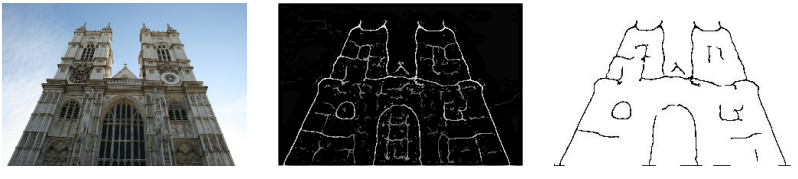


Figure 3: A sample of learned keyshapes when the number of clusters $K = 150$.

3.2 LKS Descriptor Computation

3.2.1 Sketch Token Contour Detection

As we mention in previous sections, a critical task in a SBIR system is to represent an image as a sketch. Instead of using low level methods as Canny, we prefer to use the sketch token based approach proposed by Lim *et al*. [18]. The sketch tokens are a kind of mid-level patterns. Therefore, given an image we will get an skecth token map, where for each image pixel a score of being a contour point is assigned. In our proposal we chose a threshold to decide whether a score value is enough to be considered as a contour point or not. Empirically we set this threshold as 60% of the maximum response in the underlying sketch token image. An example of a color image with its corresponding sketch token and contour image is depicted in Figure 4.



(a) (b) (c)

Figure 4: (a) Original image, (b) sketch token image (c) contour image.

3.2.2 KeyShape Detection

We extract sketch patches from the input query as well as from the sketch-like representation of a test image (the contour map). Each patch is centered in each stroke point. Therefore, we have as much sketch patches as the number of stroke points in the image. For each patch, we search for the P nearest *keyshapes* using DAISY descriptors.

We define S_i as the i-th sketch patch in an image. Let p_j^i be the index of the j-th nearest keyshape of S_i and δ_j^i the corresponding distance between S_i and p_j^i. Thereby the set KS_P^i of P nearest keyshapes of S_i is defined by the following:

$$KS_P^i = \{(p_1^i, \delta_1^i), \cdots (p_P^i, \delta_P^i)\} \tag{2}$$

3.2.3 LKS-Histogram Computation

We build a K-size histogram, where K is the number of keyshapes. In this process, three steps are involved: (1) Voting, (2) Spatial Division, (3) Normalization.

1. **Voting**: Each sketch patch S_i (represented as a DAISY descriptor) votes for P bins according to KS_P^i. The vote is computed in a soft manner by means of a gaussian kernel function $G(\cdot,\cdot)$ [29],

defined as follows:

$$G(S_i, p^i_j) = \frac{1}{\sigma_{p^i_j}\sqrt{2\pi}} \exp\left(-0.5\frac{\delta^{i\,2}_j}{\sigma_{p^i_j}{}^2}\right) \tag{3}$$

where $G(\cdot,\cdot)$ is computed with respect to a sketch patch (S_i) and its corresponding j-th nearest keyshape (p^i_j). Considering a gaussian distribution of the distances for each cluster, we set the standard deviation of the kernel depending on the maximum distance in the cluster. Thereby, we set $\sigma_{p^i_j} = \alpha * r_{p^i_j}$, with $\alpha = 1$, which means that around 68% of the points in a cluster have a distance no greater than the maximum distance on the cluster estimated during training.

Then S_i votes for the bin p^i_j of the LKS histogram h_{LKS} as follows:

$$vote^i_j = \frac{G(S_i, p^i_j)}{\sum_{k=1}^{P} G(S_i, p^i_k)} \tag{4}$$

$$h_{LKS}(p^i_j) += vote^i_j \tag{5}$$

where $h_{LKS}(\cdot)$ is the LKS histogram for the underlying sketch or contour.

2. **Spatial division**: To take into account the spatial distribution of strokes, we divide the image into $B \times B$ blocks yielding a final histogram with $B \times B \times K$ bins. The histogram for each block is computed using bi-linear interpolation. Finally, the LKS histogram is the concatenation of all the block histograms.

3. **Normalization**: Each LKS histogram computed on a block is then normalized to unit length. For retrieving images, the histogram of a query sketch will be compared with the histogram of a test image. This comparison is carried out by a distance function, commonly Minkowski's functions are used. However, this comparison may be biased by peaks in the histograms. To reduce the negative impact of this phenomenon, Arandjelović and Zisserman [2] suggest to use a *Hellinger* distance function. This process is similar to normalize the histogram taking the square root of each element and then applied a classical L_2 distance. Thereby, we normalize the complete LKS histogram by the square root rule.

4 Experimental Evaluation

4.1 Dataset

For evaluation purposes we used two datasets: The first one proposed by Saavedra and Bustos [24], which is a small dataset consisting of 1326 test images and 53 sketch queries. The second one is the *Flickr15k* dataset [15], which is a larger dataset containing 14660 images labeled in 33 different classes. The *Flickr15k* dataset also provides 329 sketches drew by different users, each sketch also belong to one out of 33 classes. For measuring the performance of the SBIR methods, we use the precision-recall graphic as well as the mean average precision metric (mAP).

4.2 Result Analysis

In Table 1, we present the mAP for different approaches including HOG[9], GF-HOG [15] and SHELO [25]. We can note that our proposal achieves an effectiveness gain of around 17% in the Saavedra's dataset and around 98% in the Flickr15k dataset. In particular, our method achieves a mAP of 0.245 in Flickr15k, while the others are around 0.12. This result shows the goodness of using mid-level features.

In addition, we present in Figure 5 the recall-precision graphic comparing our proposal LKS with SHELO in the two evaluation datasets.

We achieve the best performance for similar settings of LKS in the two datasets. In the Saavedra's dataset we found the best performance (mAP=0.3251) when $B = 4, P = 10$, and $K = 150$, and in the

	HOG	GF-HOG[15]	SHELO[25]	**LKS**	gain
Saavedra's	0.2355	*unreported*	0.2766	**0.3251**	17.5%
Flickr15K	0.0771	0.1222	0.1236	**0.2450**	98.2%

Table 1: Mean Average Precision comparing our proposals LKS with state-of-the-art methods.

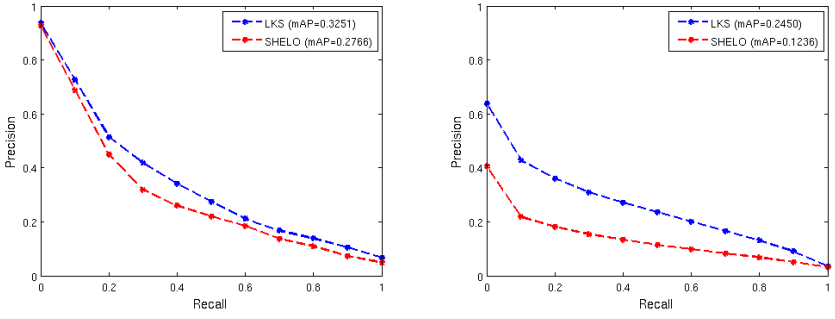


Figure 5: Precision-Recall graphic showing the performance of LKS (blue curve) and SHELO (red curve) on the Saavedra's dataset (on the left) and Flickr15k dataset (on the right).

Flickr15k the best parameters were $B = 3, P = 20$, and $K = 300$ obtaining a $mAP = 0.2497$. However, we also achieve a very competitive $mAP = 0.2450$ in Flickr15k when we use $B = 3, P = 10, K = 150$, an we prefer this setting since the final descriptor size is twice lower. A sample of our results obtained for different parameters are shown in Table 2 and Table 3.

4.2.1 Parameter Analysis

We first analyze the performance of our method varying the parameter P, the number of nearest keyshapes used in the voting process, and B, the number of blocks used for spatial division. We run experiments where P takes values in $\{5, 10, 15, 20\}$ and B in $\{1, 2, 3, 4, 5, 6, 7\}$. In Figure 6, we show the mAP achieved in the two evaluation datasets when $K = 150$. The best performance is obtained when $B = 3$ or 4 and $P = 10$. Note that a low value of P is required in order to reduce the complexity of the method.

We also evaluate the impact of the number of keyshapes K. In Figure 7, we show the performance of LKS for the two datasets. For both datasets, the best performance is achieved when $K = 150$. In this experiments K takes values in $\{50, 100, 150, 200, 300\}$ when $B = 4, P = 10$ in the case of Saavedra's dataset and $B = 3, P = 10$ in Flickr15k. For reducing the complexity of the method a low value of K is preferable.

LKS detects keyshapes for all stroke points of an input image, so the complexity of the descriptor

	K=50	K=100	K=150	K=200	K=300
B=3, P=10	0.2802	0.2901	0.2911	0.2908	0.2949
B=4, P=10	0.2925	0.3186	**0.3251**	0.3248	0.3180
B=3, P=15	0.2773	0.2888	0.2900	0.2911	0.2916
B=4, P=15	0.2828	0.3076	0.3210	0.3241	0.3250
B=3, P=20	0.2783	0.2863	0.2886	0.2878	0.2893
B=4, P=20	0.2789	0.3019	0.3144	0.3191	0.3220

Table 2: mAP for a sample of different parameters in the Saavedra's dataset.

	K=50	K=100	K=150	K=200	K=300
B=3, P=10	0.2246	0.2407	**0.2450**	0.2431	0.2381
B=4, P=10	0.2208	0.2370	0.2351	0.2272	0.2104
B=3, P=15	0.2124	0.2348	0.2437	0.2452	0.2472
B=4, P=15	0.2056	0.2330	0.2390	0.2375	0.2300
B=3, P=15	0.2058	0.2276	0.2387	0.2446	0.2497
B=4, P=15	0.1964	0.2255	0.2361	0.2402	0.2387

Table 3: mAP for a sample of different parameters in the Flick15k dataset.

is directly affected for the complexity of the sketch that means that more stroke points on the image more point to classify. In this regard we evaluate the performance of our proposal when only a fraction of the total stroke points are used. We found that in both evaluation dataset, our proposal does not undergo significant loss of precision when we used until 20% of the stroke points (See Figure 8).

Finally, in Figure 9, we present examples of retrieval results using our proposal on the Flickr15k dataset. For each row, we present a query sketch together with the five first responses.

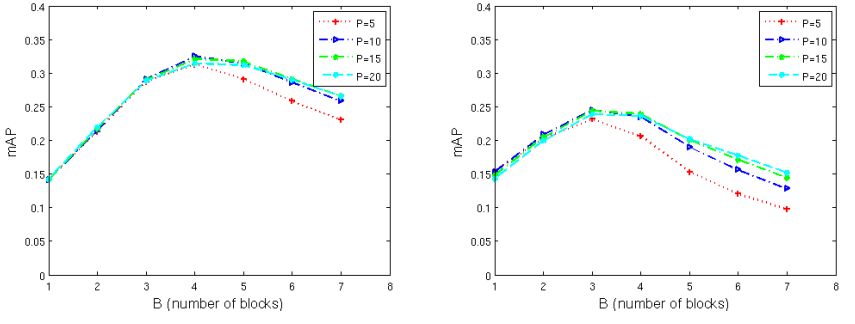


Figure 6: Performance of LKS varying the number of blocks for spatial division and the number of P for voting when $K = 150$. On the left, we see the performance on the Saavedra's dataset. On the right, we show the performance on the Flick15k dataset

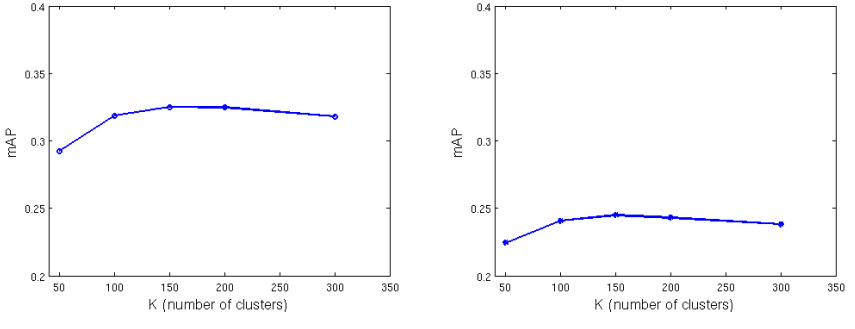


Figure 7: Performance of LKS varying the number of clusters on the Saavedra's dataset (on the left), and Flickr15k (on the right).

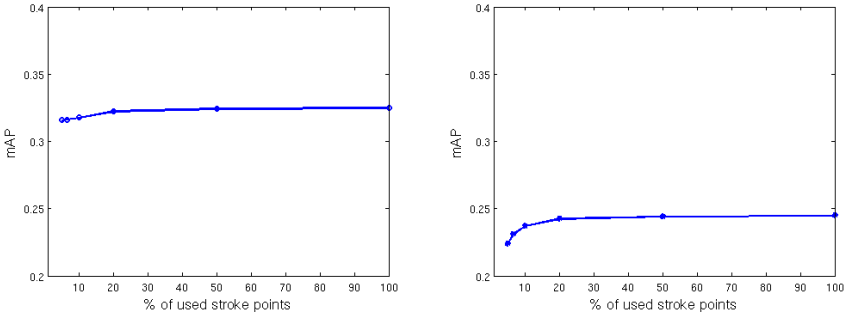


Figure 8: Impact of reducing the number of patches for computing the LKS histogram. On the left, the performance on the Saavedra's dataset. On the right, the performance on Flickr15k.

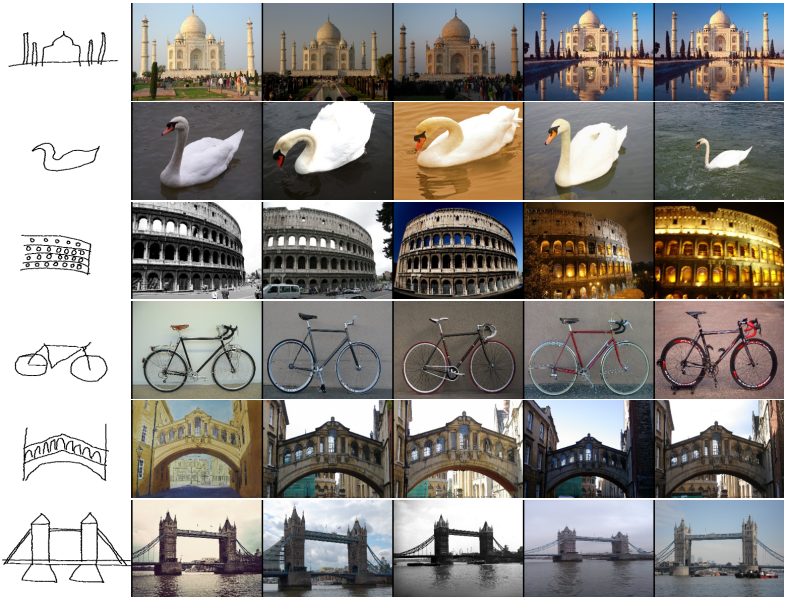


Figure 9: Examples of results using LKS. Each row shows a query sketch together with the five first responses.

5 Conclusions

We have presented a novel method for sketch based image retrieval, showing new state-of-the-art performance in this context. We demonstrate that our approach based on mid-level patterns is very promising for representing sketches. In fact, we demonstrate an effectiveness gain of about 98% in a large dataset with respect to state-of-the-art methods. We also show the benefits of using mid-level contour detection for obtaining sketch-like representations of regular images.

We also have demonstrated that our method achieves good results when a fraction of the total of stroke points are used. In particular, LKS does not undergo significant variation when the 20% of points are used.

6 Acknowledgment

We are grateful for financial support from CONICYT, Chile through the projects PAI-781204025 and PAI-781204026 as well as CORFO-INNOVA, Chile through the project 15ITE2-38948.

References

[1] Relja Arandjelovic and Andrew Zisserman. Smooth object retrieval using a bag of boundaries. In *Proc. of the 2011 Int. Conference on Computer Vision*, ICCV '11, pages 375–382, 2011.

[2] Relja Arandjelovic and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 2911–2918, 2012.

[3] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, May 2011.

[4] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(4):509–522, April 2002.

[5] Gunilla Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–865, November 1988.

[6] Yang Cao, Changhu Wang, Liqing Zhang, and Lei Zhang. Edgel index for large-scale sketch-based image search. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–768. IEEE Computer Society, 2011.

[7] Abdolah Chalechale, Golshah Naghdy, and Alfred Mertins. Sketch-based image matching using angular partitioning. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 35(1):28–41, 2005.

[8] Tao Chen, Ming-Ming Cheng, Ping Tan, Ariel Shamir, and Shi-Min Hu. Sketch2photo: internet image montage. *ACM Transactions on Graphics*, 28(5):124:1–124:10, December 2009. ISSN 0730-0301.

[9] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, pages 886–893. IEEE Computer Society, 2005.

[10] Hubel David and Wiesel Torsten. *Brain and visual perception: the story of a 25-year collaboration*. Oxford University Press US., USA, 2005. ISBN 978-0-19-517618-6.

[11] Alberto Del Bimbo and Pietro Pala. Visual image retrieval by elastic matching of user sketches. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):121–132, 1997.

[12] Mathias Eitz, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. Photosketch: a sketch based image query and compositing system. In *SIGGRAPH 2009: Talks*, SIGGRAPH '09, pages 60:1–60:1, 2009.

[13] Mathias Eitz, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. Sketch-based image retrieval: Benchmark and bag-of-features descriptors. *IEEE Transactions on Visualization and Computer Graphics*, 17(11):1624–1636, 2011.

[14] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Trans. Graph.*, 31(4):44:1–44:10, July 2012.

[15] Rui Hu and John Collomosse. A performance evaluation of gradient field hog descriptor for sketch based image retrieval. *Computer Vision and Image Understanding*, 117(7):790–806, July 2013.

[16] Rui Hu, M. Barnard, and J. Collomosse. Gradient field descriptor for sketch based retrieval and localization. In *17th IEEE International Conference on Image Processing (ICIP), 2010*, pages 1025 –1028, 2010.

[17] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178, 2006.

[18] J.J. Lim, C.L. Zitnick, and P. Dollar. Sketch tokens: A learned mid-level representation for contour and object detection. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3158–3165, June 2013.

[19] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[20] Michael Randolph Maire. *Contour Detection and Image Segmentation*. PhD thesis, EECS Department, University of California, Berkeley, Sep 2009.

[21] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *8th International Conference on Computer Vision*, volume 2, pages 416–423, July 2001.

[22] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. Ieee, 2007.

[23] K. Ramakrishnan, H. S. Scholte, I. I. A. Groen, A. W. M. Smeulders, and S. Ghebreab. Visual dictionaries as intermediate features in the human brain. *Frontiers in Computational Neuroscience*, 2014.

[24] Jose Saavedra and Benjamin Bustos. An improved histogram of edge local orientations for sketch-based image retrieval. In *Pattern Recognition*, volume 6376 of *Lecture Notes in Computer Science*, pages 432–441. 2010.

[25] Jose M. Saavedra. Sketch based image retrieval using a soft computation of the histogram of edge local orientations (s-helo). In *International Conference on Image Processing, ICIP'2014 (To appear)*, 2014.

[26] Jose M. Saavedra and Benjamin Bustos. Sketch-based image retrieval using keyshapes. *Multimedia Tools and Applications*, 2013.

[27] Xinghai Sun, Changhu Wang, Avneesh Sud, Chao Xu, and Lei Zhang. Magicbrush: Image search by color sketch. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, pages 475–476. ACM, 2013.

[28] Engin Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(5), May 2010.

[29] JanC. van Gemert, Jan-Mark Geusebroek, CorJ. Veenman, and ArnoldW.M. Smeulders. Kernel codebooks for scene categorization. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision âĂŞ ECCV 2008*, volume 5304 of *Lecture Notes in Computer Science*, pages 696–709. Springer Berlin Heidelberg, 2008.