SmartSketcher: Sketch-based Image Retrieval with Dynamic Semantic Re-ranking

Tiziano Portenier
University of Bern
portenier@inf.unibe.ch

Qiyang Hu
University of Bern
hu@inf.unibe.ch

Paolo Favaro
University of Bern
favaro@inf.unibe.ch

Matthias Zwicker
University of Maryland, College Park
zwicker@cs.umd.edu

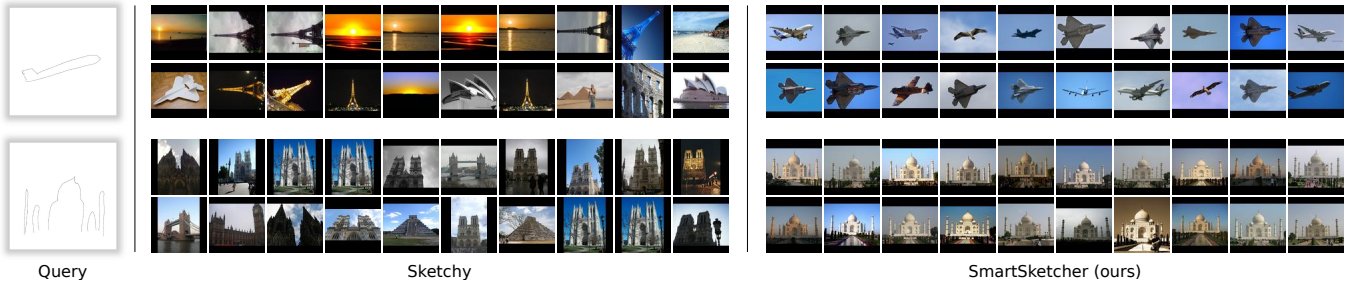

Figure 1: The key idea of our sketch-based image retrieval system is to dynamically re-rank query results by clustering them using convolutional neural network features learned for image classification. We show a comparison of a state of the art deep learning method for sketch-based image retrieval (left, the Sketchy features [Sangkloy et al. 2016]), to our approach (right). We present the top 20 results for two example queries on the Flickr15k dataset to demonstrate the improved quality of our query results.

ABSTRACT

We present a sketch-based image retrieval system, designed to answer arbitrary queries that may go beyond searching for predefined object or scene categories. While sketching is fast and intuitive to formulate visual queries, pure sketch-based image retrieval often returns many outliers because it lacks a semantic understanding of the query. Our key idea is to combine sketch-based queries with interactive, semantic re-ranking of query results. We leverage progress in deep learning and use a feature representation learned for image classification for re-ranking. This allows us to cluster semantically similar images, re-rank based on the clusters, and present more meaningful query results to the user. We report on two large-scale benchmarks and demonstrate that our re-ranking approach leads to significant improvements over the state of the art. Finally, a user study designed to evaluate a practical use case confirms the benefits of our approach.

CCS CONCEPTS

• **Information systems** → **Image search**;

KEYWORDS

sketch-based image retrieval, clustering

ACM Reference format:
Tiziano Portenier, Qiyang Hu, Paolo Favaro, and Matthias Zwicker. 2017. SmartSketcher: Sketch-based Image Retrieval with Dynamic Semantic Re-ranking. In *Proceedings of SBIM'17, Los Angeles, CA, USA, July 28-29, 2017,* 12 pages.
DOI: 10.1145/3092907.3092910

SBIM'17, Los Angeles, CA, USA
 978-1-4503-5079-2/17/07…$15.00
DOI: 10.1145/3092907.3092910

1 INTRODUCTION

We present an image retrieval system to find images based on simple visual scene descriptions that are given as sketches. We support sketches that can be more specific than just a scene or object category, and that are simple enough so they can be provided by an untrained user. Our approach is not limited to pre-determined sets of object or scene categories, does not rely on metadata, and does not require an already available example image. In addition, we designed our system to support a typical scenario where users refine their search iteratively to retrieve as many images as possible that match a desired concept in a limited amount of time. Finally, users would select the one retrieval result that they like best.

We tackle these challenges using a sketch-based approach. Sketching is attractive because it allows a user to quickly produce a visual representation of a desired scene, and sketches can be matched to images even in the absence of metadata or already available example images. Sketch-based image retrieval has been explored widely.

One approach is to recognize the object category that the user has drawn, and then retrieve images that contain the desired category. This approach is limited, however, to pre-existing categories that have been used to train the system. On the other hand, a handcrafted low-level representation of a sketch can be matched with images in the database directly. Unfortunately, this approach leads to inconsistent results in practice, often ranking images highly that do not semantically match the sketch. A more recent approach is to learn a representation by training a convolutional neural network using a large training set of sketch-image pairs, such as the Sketchy database [Sangkloy et al. 2016]. This approach works well on data that is sufficiently similar to the training set, but often fails on categories or more fine grained scene descriptions that were not seen during training, as we show in Figure 1 and in more detail in our results section.

A key idea of our approach is to address this issue by combining sketch-based image retrieval using handcrafted low level features with a re-ranking technique applied to the query results. For re-ranking we leverage recent progress in deep learning of image features, which leads to semantically meaningful clusters of sketch-based query results, and meaningful rankings as shown in Figure 1 on the right. It is interesting to observe that both approaches in Figure 1 leverage deep convolutional networks. While Sketchy [Sangkloy et al. 2016] is specifically trained on sketches, however, our approach performs re-ranking using a network trained on images only. Crucially, the Sketchy database is much smaller than the image database that trained the network we use. We believe that this is the main reason why our results are much more robust than the retrieval using Sketchy. Finally, our system supports an interactive, iterative query process, where the clustering of current query results is used to refine subsequent queries. This enables users to quickly find a large number of images that are relevant to their search.

We evaluate our re-ranking technique using two large-scale benchmarks for sketch-based image retrieval and show that it provides significant improvements over the state of the art. We further conducted an extensive user study, where we evaluate our interactive application in a more practical usage scenario. We show that our approach leads to significantly improved retrieval performance over two baseline systems, and it is generally preferred by the users in our study. In summary, we make the following contributions:

- An interactive, sketch-based image retrieval system that supports searching for images based on scene descriptions that are not restricted to a set of pre-existing object or scene categories.
- A powerful re-ranking technique based on deep learning features that significantly improves retrieval performance of the underlying descriptor.
- A user study that evaluates our system and shows how the combination of sketching and clustering leads to superior performance compared to two baseline systems restricted to sketching and clustering only.

2 RELATED WORK

Sketch-Based Image Retrieval. Our approach is most related to sketch-based image retrieval (SBIR), where the query is given as a sketch instead of a photographic image. Most SBIR techniques work by extracting low-level features from the query, and matching them with the features of the database images. Matching can be performed *locally*, by discretizing images into cells and matching features in each cell, or *globally*, using bag-of-words (BoW) techniques. In both cases, a main challenge is the design of appropriate features. For example, Hu et al. [2010] propose the Gradient Field HoG (histogram of oriented gradients [Dalal and Triggs 2005]) descriptor (GF-HoG) for sketch-based image retrieval, and they show improved accuracy over HoG. Eitz et al. [2011] propose a bag-of-features approach and benchmark different local descriptors. They observe that SIFT descriptors extracted by sampling keypoints on Canny edges (SHoG) outperform other methods (Tensor, HoG, shape context [Belongie et al. 2002], spark descriptor) on their benchmark. Saavedra et al. [2014] identify keyshapes (arcs, lines, circles, ellipses) instead of keypoints to extract local descriptors. Global descriptors are constructed using a bag-of-features approach. Improvement over SHoG [Eitz et al. 2011], however, is marginal. Saavedra [2014] also introduces Soft-Histogram of Edge Local Orientations (S-HELO), an extension of HELO [Saavedra and Bustos 2010] (cell-wise Histogram of Edge Local Orientations) that adds spatial information. Eitz et al. [2012] developed a representation based on Gabor filters (GALIF) for sketch-based 3D shape retrieval, which has been used successfully in SBIR methods [Li et al. 2015].

Qiang et al. [2015] introduce a framework based on Product Quantization with sparse coding to construct an optimized codebook, using a state-of-the-art local descriptor (GALIF). The key idea is to reduce information loss by using a large codebook size, and they show performance improvements on standard benchmarks over previous BoW techniques. Finally, Li et al. [2014] tackle the problem of fine-grained SBIR within object categories. A deformable part-based model is learned to encode poses. Graph matching is performed for retrieval. They outperform both bag-of-feature and spatial pyramid approaches. Other techniques have explored the use of additional information that may be extracted from sketches, or provided by the user, such as saliency [Takahiko and Ryutarou 2014], regions of interest [Liang et al. 2014], symmetry [Cao et al. 2013; Parui and Mittal 2014], or color [Sun et al. 2013].

Achieving interactivity is challenging with huge image databases. Yang et al. [2011] address this by proposing a raw contour-based matching algorithm with an index structure. The idea is based on Oriented Chamfer Matching (OCM) but with an efficient index structure for scalability. Zhou et al. [2012] achieve large scale image retrieval by identifying a main region and a region of interest (RoI) for each image. The main region is a weighted centroid of image features and enables scale and translation invariance, and the RoI is used to identify important objects in complicated scenes.

Our approach is mostly orthogonal to the above techniques since our main goal is not to develop a novel representation for SBIR. Instead, we introduce a system that combines an SBIR technique with on-the-fly clustering to provide users with effective, interactive image retrieval capabilities. Our technique shares similarities with Sun et al.'s framework [2012]. They use an existing representation [Yang et al. 2011] for SBIR, and leverage the query results to estimate the most likely category using a graphical model, which

may also include metadata such as text labels. In contrast, our approach is completely unsupervised. Most importantly, our approach supports users to answer queries that go beyond finding images that match certain pre-existing labels or categories. Instead, we leverage powerful clustering based on convolutional neural network (CNN) features that groups query results in a semantically meaningful, unsupervised manner. Qian et al. [2016] propose a SBIR system that also leverages re-ranking. Their approach is based on visual feature verification in SIFT feature space. In contrast, our method is much simpler, uses deep learning features instead of hand-crafted features, and performs better on average.

Convolutional Neural Network Features. Progress in deep learning using CNNs has led to astonishing advances in many computer vision tasks recently. As a consequence, features extracted from networks trained on object recognition or image classification have largely replaced previous handcrafted features. A survey of this fast evolving field, however, is outside the scope of this paper. To provide an example, Razavian et al. [2014] demonstrate that features extracted from a deep CNN trained on image classification work very well as a generic image representation to tackle various recognition tasks, namely scene recognition, fine grained object recognition, attribute detection, and image retrieval.

Deep CNNs have also been used for sketch recognition. Seddati et al. [2015] train a network on the TU Berlin dataset [Eitz et al. 2012], and Sarvadevabhatla [2015] takes features directly from networks trained on image classification. Both report superior performance compared to previous work based on hand-crafted features for sketch classification. Directly using CNN features of a query sketch for SBIR, however, does not lead to successful results in our experience. Cross-domain retrieval can be achieved by training Siamese networks to learn features for two domains simultaneously. Wang et al. [2015] use this approach to learn features for both sketches and line renderings of 3D meshes for sketch-based 3D shape retrieval. Similar approaches were also used to train features for SBIR [Sangkloy et al. 2016; Yu et al. 2016]. We show that the state of the art approach by Sankloy et al. [2016] still benefits from our re-ranking technique.

3 SKETCH-BASED IMAGE RETRIEVAL WITH DYNAMIC SEMANTIC RE-RANKING

Here we present our interactive SBIR system, called SmartSketcher. The input to this system is an image collection of up to several hundred thousand images, typically gathered from Internet images using a keyword search, thus representing a specific semantic idea such as "airplane", "dog", or "boat", or any other image dataset. Using such datasets, SmartSketcher enables the user to retrieve specific images that he has in mind, for example, *"an airliner on the ground seen from the side, with a tower in the background"*. The user expresses this conception by drawing a rough sketch and the system presents matching images to the user in real-time. The system supports an interactive, iterative workflow, where subsequent queries are refined based on previous query results. It is designed to retrieve as many matching images as possible with few interactions in a short time. The final output is a list of images that matches the user's conception.

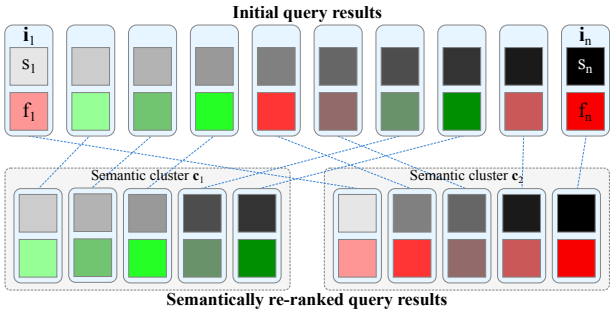

Figure 2: Schematic visualization of our re-ranking technique. Each image i_i in the query result has (1) an associated similarity score s_i based on its distance to the query sketch in a low-level representation (grayscale), and (2) a semantic representation f_i (green and red) derived from deep learning on a large image database. The top row shows the retrieval results using the similarity scores, before re-ranking. The bottom row shows the semantically re-ranked retrieval results and the corresponding semantic clusters c_1 and c_2.

The key component of SmartSketcher is a re-ranking technique that improves the performance of an arbitrary retrieval system and enables the user to resolve semantic ambiguities, described in Section 3.1. We present the overall system with the underlying geometric SBIR method and support for iterative queries in Section 3.2.

3.1 Dynamic Semantic Re-ranking

Typical SBIR methods query nearest neighbors by computing similarity scores in some low-level feature domain. Most users do not have the skills to provide precise and detailed input sketches, however, and ordering query results according to low-level similarity scores typically includes many undesired images ranked close to the top. As a consequence, significant user effort is required to browse through query results and select images that match the query. Hence a key idea in our system is to reorganize the query results in a semantically meaningful manner, which we achieve by re-ranking them using a representation derived from deep learning on a large image database. Figure 2 shows a schematic visualization of the proposed re-ranking technique.

Razavian et al. [2014] demonstrate that features extracted from a deep CNN trained on image classification work very well as a generic image representation to tackle various recognition tasks. Inspired by this, we use hidden layer activations of a deep CNN trained on image classification with the intuition that because the CNN is trained on image classification, the representation learned by the network will capture semantic information, which enables a semantic re-ranking of the query results. Given the results from an underlying SBIR system, we first compute their semantic features, which are independent of the query sketch, to group them into k semantic clusters, or modes, using k-means clustering. If the results from the initial SBIR contain some correct images, these will form a dedicated semantic mode. The goal is to re-rank the results such that the the cluster containing the correct images will be ranked on top. To achieve this, we rank each cluster based on

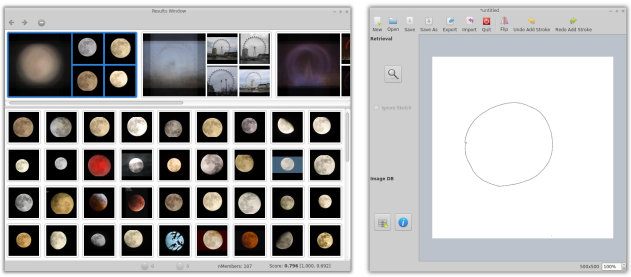

Figure 3: The user interface of SmartSketcher consists of two windows. The left window shows the grouped retrieval results and the right window is used to express the query. The user interacts with the canvas using the Wacom pen. Once the user collected a matching image, he can optionally ignore the sketch for further retrieval. The results window is divided into two parts: the top pane shows the clusters and the bottom pane shows the cluster members. The user can mark a cluster in the top pane as "undesired".

the average similarity score of its members. The intuition is that if the result from the initial SBIR contains enough signal, i.e. the correct images are scored higher than the noise images on average, the proposed approach will rank the desired cluster on top. Within individual clusters, we finally order the images according to the initial similarity scores. It is clear that this approach only works if there are enough highly ranked correct images in the initial results, and we show in our experiments that this approach works extremely well on three different SBIR systems. We also experimented with re-ranking based on per-cluster average rank instead of average similarity score, and we find that similarity score outperforms rank on average.

In addition to the improved ranking, the proposed method allows us to present different semantic modes in the data to the user. Our system displays average images and representative images for each cluster, which enables the user to quickly find the desired mode. To maintain interactivity, we use a fast k-means implementation on the GPU [Catanzaro 2013] and only consider the top n results for clustering. The resulting clusters from the k-means algorithm are very sensitive to the initialization step. To achieve a more robust re-ranking, we run k-means 50 times with different random initializations. We run a fixed number of 20 iterations for each initialization to maintain interactivity. We then select the initialization with the smallest error and run k-means with this initialization until convergence.

3.2 The SmartSketcher Image Retrieval System

In our SmartSketcher image retrieval system we combine SBIR and dynamic clustering in a unified interactive framework. To make better use of the power of the CNN features, we leverage them not only to re-rank and present query results but also to refine the similarity score computation for subsequent queries. Figure 3 shows the user interface of SmartSketcher.

3.2.1 Low-level Representation. Since our system is designed to retrieve very specific images, our low-level representation to match sketches and images consists of a single global feature vector that encodes both spatial and rotational information. For this purpose we leverage a representation based on histogram of oriented gradient (HoG) features. We use a dense SIFT implementation [Vedaldi and Fulkerson 2008] to extract local features on a regular grid of dense keypoints and we concatenate these local descriptors to form one global feature vector. As shown in Section 4 we find that HoG yields very high precision for small recall values, which is important in our application, and a quantitative comparison to state of the art deep learning features [Sangkloy et al. 2016] supports our choice. We emphasize, however, that other approaches could be used instead, and the key of our technique is to semantically re-rank the results from an arbitrary SBIR system, which enables users to answer complex queries.

3.2.2 Real-Time Retrieval. For image retrieval, the user draws a query sketch that is matched against all the images in the database. Let g^I denote the global HoG descriptor extracted from image I and g^Q the descriptor extracted from the query sketch Q. We denote l_i^I the local descriptor extracted from image I at keypoint i, i.e. $g^I = (l_0^I, \dots, l_{n-1}^I)$, where n is the number of keypoints on the grid. The partial similarity between Q and I is computed as the mean of the dot products between local features at the corresponding keypoints,

$$sim(Q, I) = \frac{\sum_{i=0}^{n-1} l_i^I \cdot l_i^Q}{\sum_{i=0}^{n-1} nz(l_i^Q)}, \tag{1}$$

where

$$nz(v) = \begin{cases} 1, & \text{if } v \neq 0 \\ 0, & \text{otherwise} \end{cases}. \tag{2}$$

To avoid penalization of sparsity of the query sketch, we normalize by the number of non-empty local features in the query. Note that the dot product in the nominator in Equation 1 is not normalized. Local features in Q are typically sparse and a normalization of l_i^I would penalize images with denser features. Since our system is interactive and we assume the user to draw sparse strokes, such a normalization would not be meaningful.

Evaluating Equation 1 produces a ranking of all images in the dataset. We compute this ranking twice, once for the original query sketch and once for a normalized version. We normalize the query by scaling the sketch such that its bounding box covers 0.7 times the canvas size and translate it to be centered. The final similarity score for each image I is

$$s(Q, I) = \max(sim(Q, I), sim(\bar{Q}, I)), \tag{3}$$

where $\bar{Q}$ is the normalized query. With this approach, the user can precisely define the locations of objects in the image, but does not have to care about the exact scale and position in case he wants to retrieve images with just a single object.

3.2.3 Iterative Queries. To support iterative queries, SmartSketcher includes two extensions enabled by the incorporation of the CNN features. First, we enable the user to mark entire modes in the query results as "undesired", that is, to express that a particular mode does not represent the concept in mind. During subsequent retrievals, we penalize images that are close to the marked modes

in CNN feature space. If the number of marked modes exceeds a certain threshold, we greedily merge a newly marked mode with its nearest neighbor in the list of marked modes by replacing the appropriate mode with the average to maintain scalability. Second, SmartSketcher maintains a list of positive images, that is, the images that the user already selected as matches for the desired concept. The user continuously adds images to this list during interactive search, and our system considers them as targets in subsequent retrievals. A naive implementation would be to favor images that are close to at least one positive image. However, this approach scales linearly to the number of positive images and retrieval becomes slow once the user collects more images. Also, the approach often does not work well since the image content that is salient to the CNN does not necessarily correspond to the semantic concept the user has in mind. To overcome these two issues, we cluster the list of positive images to come up with a model of the concept the user has in mind in CNN feature space. Subsequent retrievals favor images that are close to at least one cluster centroid. This approach scales well and yields superior results.

We modify Equation 3 to include these two extensions and obtain the final similarity score for image I in database D as

$$\frac{s(Q,I)}{\max_{J\in D} s(Q,J)} + \alpha \max_{a\in A}(1-d(a,I)) + \beta \sum_{b\in B} d(b,I), \tag{4}$$

where A denotes the set of centroids of the positive images, B is the set of undesired modes marked by the user, and d is the normalized L_2 distance in CNN feature space, that is,

$$d(x,y) = \frac{\|x-y\|_2}{\max_{z\in D}\|x-z\|_2}. \tag{5}$$

In our experiments, we set $\alpha = 1$ and $\beta = \max(|B|, 2)^{-1}$. The user interface of SmartSketcher also features an option to ignore the sketch drawn by the user during retrieval, once the user collected some positive images. If the user enables this option, the first term in Equation 4 is omitted for subsequent retrieval.

3.2.4 Implementation Details. In a preprocessing step, we compute our HoG descriptor for each image in the database off-line. During on-line retrieval, the same descriptor is extracted from the query sketch and matched against the descriptors of all images in the database. To achieve real-time performance, we perform matching on the GPU. Before computing the descriptors, we scale the images in the database to 500 pixel resolution, that is, the larger size is scaled to 500 pixels. We extract Canny edge maps [Canny 1986] from the resized images, and pad the edge maps with zeros to fit a resolution of 500x500 pixels. For each edge map, we then extract a global HoG feature vector. A local feature has 4x4 spatial bins and 10 orientation bins for gradient directions. Similar to Eitz et al. [2011], we ignore information about edge direction and only distinguish 5 orientations from 0 to π. We use a relatively large local feature size of 128 pixels, that is roughly 25% of the image size. Local features are extracted on a grid with a stride of 32 pixels which results in a 11520-dimensional global feature vector for each image.

For the semantic re-ranking we use CaffeNet, a deep CNN trained on the ILSVRC 2012 image classification challenge from the Caffe deep learning framework [Jia et al. 2014] as feature extraction pipeline. In a preprocessing step, we compute feature vectors for all database images by propagating them through the trained CNN and extracting hidden layer activations from the "pool5" layer, which results in a 9216-dimensional vector for each image. We first scale the images such that the smaller size is 256 pixels, and crop them to a square size to fit the input layer of the CNN. Similar to Szabo et al. [2015], we apply L_2 normalization on the feature vectors. We store the feature vector components as single bytes by scaling and quantizing to 256 bins to reduce memory requirements. We did not notice an appreciable difference due to this loss of precision for our application.

For 500k images, GPU memory footprint is less than 6 GB for the HoG descriptors and less than 5 GB for the CNN features. Retrieval with subsequent re-ranking takes less than one second on an Nvidia TITAN X GPU. One could easily reduce the memory footprint and decrease retrieval time by using an inverted index or approximate nearest neighbor methods.

Table 1: MAP for different SBIR methods on the Flickr15k benchmark. Our re-ranking boosts the performance of the previous state of the art (Sketchy) by about 20%.

method	re-ranking parameters	MAP
HoG	-	0.1506
HoG re-ranked	$k = 5, n = 500$	0.1923
HoG re-ranked	$k = 30, n = 3000$	0.2532
Sketchy	-	0.2515
Sketchy re-ranked	$k = 5, n = 500$	0.2699
Sketchy re-ranked	$k = 30, n = 3000$	0.2993
GF-HoG BoW	-	0.1222

4 RESULTS

To evaluate our re-ranking technique, we report results on two large-scale SBIR benchmarks (Section 4.1). In addition, we conducted a user study (Section 4.2) to evalute SmartSketcher, our interactive application introduced in Section 3.

4.1 Large-scale SBIR Benchmarks

We employ the Flickr15k benchmark [Hu and Collomosse 2013] for SBIR to compare our re-ranking approach with HoG features, and the Sketchy features [Sangkloy et al. 2016], a state of the art SBIR technique based on deep learning. Flickr15k consists of approximately 15k images and 330 query sketches, and each image-query pair has a label as either relevant or irrelevant. We use Mean Average Precision (MAP) as a metric for comparison. To obtain the MAP score, we compute the average precision (the average fraction of retrieved images that are relevant, over all retrieval thresholds) for each of the 330 queries, and then take the mean over all queries. We follow the approach by Hu et al. [Hu and Collomosse 2013] and use *trec_eval*, an implementation from the TRECVID benchmark for the computation of MAP.

Table 1 shows MAP values for our global descriptor proposed in Section 3 (HoG) and the re-ranked version with two different parameter sets, which clearly shows the benefits of re-ranking. We see that the re-ranking parameters (number of clusters k, number of clustered images n) have a clear influence on the MAP score, and

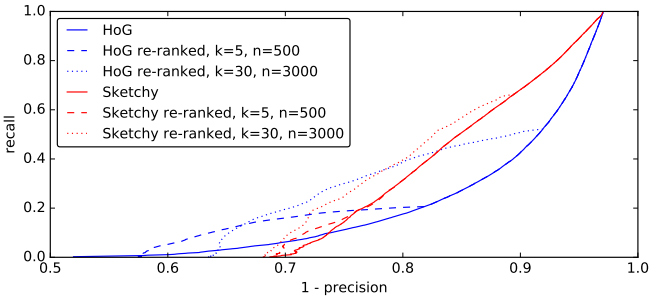

Figure 4: Precision-recall curves for HoG and Sketchy on Flickr15k, averaged over all queries. Re-ranked HOG provides better precision at lower recall values (the top 394 retrieval results in average, see Figure 5) compared to Sketchy.

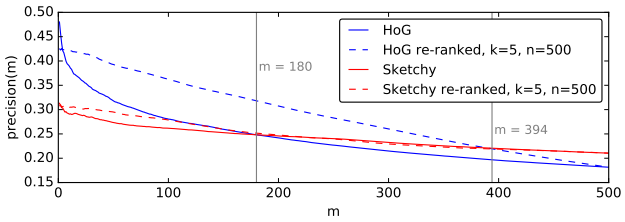

Figure 5: Precision in the first m retrieval results for HoG and Sketchy on Flickr15k, averaged over all queries. Re-ranked HOG outperforms Sketchy up to almost 400 retrieved images, which is attractive for interactive applications.

we justify our setting for SmartSketcher ($k = 5, n = 500$) below. Interestingly, re-ranking the HoG results even outperforms the Sketchy features [Sangkloy et al. 2016]. For comparison purposes we also applied our re-ranking to the Sketchy results, and observe that our approach also boosts the MAP performance of Sketchy by about 20%. This may be surprising, since the Sketchy features were learned by pretraining the Sketchy network on ImageNet for image classification, similar to the features we use for clustering. It indicates that fine tuning the cross-domain mapping on a smaller amount of data (compared to ImageNet) causes the Sketchy network to ignore useful information acquired during pretraining and to degrade the distinctiveness of the Sketchy features. Lastly, we include the MAP of GF-HoG reported by Hu et al. [2013] as an additional reference.

The results in Table 1 reveal that Sketchy clearly outperforms our HoG descriptor in terms of MAP. We emphasize, however, that MAP is not necessarily appropriate to measure performance of SBIR systems because it weights precision equally over all recall values (fraction of relevant images that are retrieved). For a practical application it is important to obtain high precision for lower recall values (the first few hundred relevant results), since users will typically not browse through thousands of images. Precision beyond recalling several hundred relevant results is less important.

We illustrate this in Figure 4, where we plot precision-recall curves for our HoG features and the Sketchy features. We see that our HoG features yield much higher precision at low recall values. In practice this means that the first m results contain more relevant images when using HoG. This is shown quantitatively in Figure 5 and qualitatively in Figure 6. We refer to the supplemental material for more qualitative comparisons. On average over all Flickr15k queries, HoG provides higher precision (more relevant results) than Sketchy in the first $m = 180$ results, as shown in Figure 5. Moreover, after re-ranking HoG outperforms Sketchy in the first $m = 394$ results. This can be explained as follows: large datasets are likely to contain many images with similar edges as in the query sketch, and HoG does a very good job at ranking such images highly. The drawback of our HoG features is their lack of invariance and semantic information. Hence, there are usually many more relevant images in the database than can be identified with HoG. In contrast, the Sketchy features are trained to be invariant to geometric variations and represent semantic information. Hence, Sketchy works well for object categories that were seen during training. Sketchy also manages to rank results highly where object location, pose, or scene composition do not match with the query sketch, and retrieve many more relevant results than HoG, which leads to higher precision at larger recall values compared to HoG. However, Sketchy does not generalize well to unseen categories.

As a consequence, our HoG descriptor combined with a powerful re-ranking technique to overcome the lack of semantics is a better choice for fine-grained image retrieval using a large dataset. In SmartSketcher we set $n = 500$ and $k = 5$, which yields the best precision for small recalls as shown in Figure 4. This also allows us to perform clustering in less than one second and provide an interactive user experience.

We next compare our approach to the recent re-ranking technique by Qian et al. [2016] using the large-scale benchmark that they introduced. The dataset consists of 362 query sketches, a sketch-describable image collection of 68,647 labeled images, and a distractor set of 227,915 images. We evaluate our re-ranking approach on this dataset and compare to the SBIR system of Qian et al. [2016]. They use an initial SBIR system based on the Edgel Index [Yang et al. 2011], which serves as input to their re-ranking and query expansion framework. To compare our re-ranking approach to their framework, we use the same baseline SBIR system based on the Edgel Index. Following Qian et al. [2016], we compute precision in the first m retrieval results, averaged over all 362 queries,

$$p(m) = \frac{1}{Z} \sum_{t=1}^{Z} \frac{1}{m} \sum_{i=1}^{m} R_t(i), \tag{6}$$

where $Z = 362$ and $R_t(i)$ is the binary relevance for query sketch t and retrieval result i. The comparison in Figure 7 shows that our re-ranking improves the performance of the underlying Edgel Index retrieval by a large margin. In addition, our mean average precision (i.e. the area under the average precision curve) improves over the best performing method proposed by Qian et al. [2016], although our approach does not significantly improve the top-1 average precision. We do not currently perform query expansion, but could likely further improve our results with such a strategy. We believe our advantage over Qian et al. stems from the semantic CNN features that are superior to the classical SIFT features in

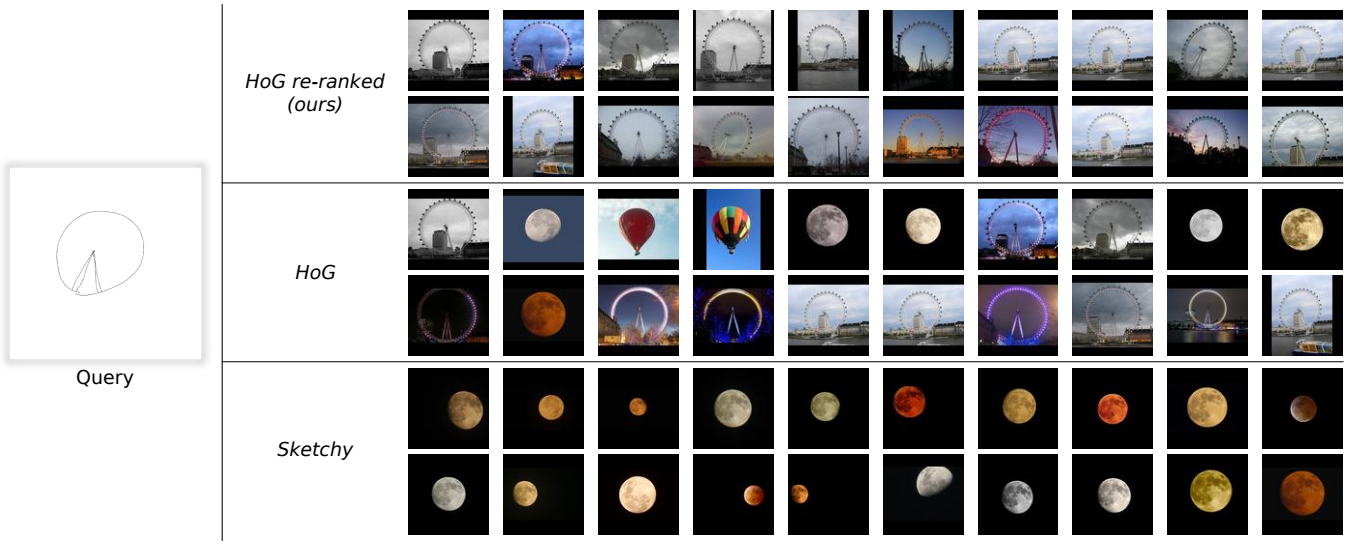

Figure 6: Top 20 results for two example queries on the Flickr15k dataset, with (HoG re-ranked) and without re-ranking (HoG) of query results. The state of the art Sketchy features lead to lower precision for the top few hundred query results (see also Figure 5). We refer to the supplemental material for more qualitative results.

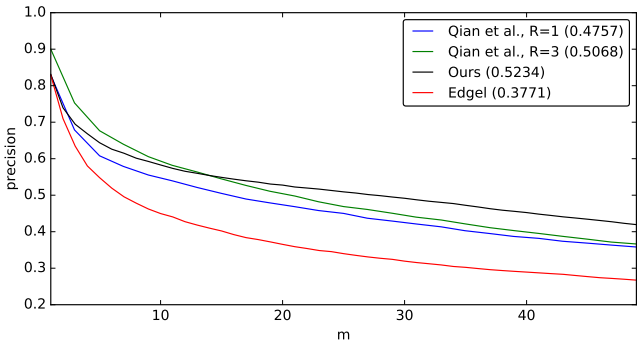

Figure 7: Precision in the first $m = 50$ retrieval results in the benchmark by Qian et al. [2016], averaged over all queries. Our mean average precision values (shown in the legend) improves over their technique.

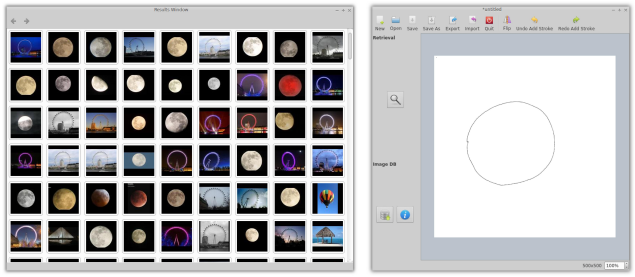

Figure 8: The user interface of Sketcher consists of two windows. The left window shows the retrieval results and the right window is used to express the query. The user interacts with the canvas using the Wacom pen. The user can load the Canny edge map of an image from the results list into the drawing canvas.

their technique. We refer to the supplemental material for a qualitative comparison, where we include a visual example of how our technique can outperform Qian et al.'s.

4.2 User Study

We conducted a formal user study to validate the effectiveness of SmartSketcher. For this purpose, we designed specific image retrieval tasks to be solved by a set of users. We implemented SmartSketcher in C++ as an interactive image retrieval application. We run the system on a PC running Linux with an Intel(R) Core(TM) i7-5960X CPU @ 3.00GHz, 64 GB RAM, and an NVIDIA GeForce GTX TITAN X GPU with 12 GB VRAM. We use a Wacom Cintiq 21UX 2 display as drawing device.

We experimented with different image datasets downloaded from Flickr using keyword searches: *Airplane* (499,970 images), *Bicycle* (154,919 images), *Car* (199,858 images), and *High-Rise* (500,000 images). In our user study we used the *Airplane* and *Bicycle* dataset and we trained the users on the *Car* dataset.

4.2.1 Baseline Systems. To measure performance of SmartSketcher, we developed two reference systems for comparison. The first baseline system is restricted to sketching only and the second baseline system works entirely based on clustering.

Sketcher. For the first baseline system called Sketcher we use the pure geometric sketch-based retrieval (Section 3.2) from SmartSketcher without the additional tools enabled by the CNN feature representation, that is, the dynamic re-ranking of query results, the ability to mark modes as undesired, and the clustering of positive

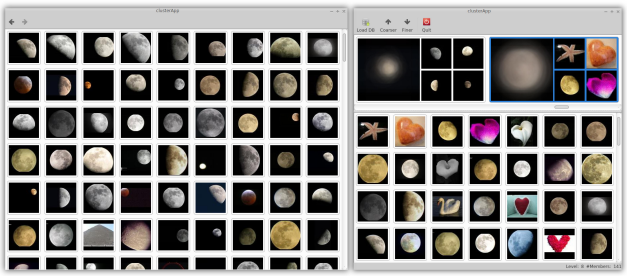

Figure 9: The user interface of ClassBrowser consists of two windows. The left window shows the retrieval results and the right window is used to navigate through the clusters. The right window is divided into two parts: the top pane shows all clusters in the current hierarchy level and the bottom pane shows the members of the selected cluster. The user can select an image in the bottom pane to perform retrieval. In addition, the user can also use images from the results window as query.

images with the option to ignore the sketch entirely (Section 3.2). In addition, Sketcher enables the user to load the Canny edge map of an arbitrary image from the results list into the drawing canvas. The user can modify the edge map by erasing or adding strokes and use it to find further similar images. Note that we do not provide this feature in SmartSketcher to keep the user interface simple, even though it would be straightforward to include it. Figure 8 shows the user interface of Sketcher.

ClassBrowser. To investigate how useful sketching is for image retrieval, we developed a second baseline system called ClassBrowser that works entirely based on clustering and without sketching. For this purpose, we use the CNN features from CaffeNet to build a cluster hierarchy of the data in a pre-process. We construct a binary tree for each dataset by clustering the data in a top-down fashion using k-means with $k = 2$. We repeatedly split the nodes in the tree until there are less than 100 images left in a node. For image retrieval, the user browses the tree starting at the root. The user interface has a "coarser" and a "finer" button to browse through the levels of the tree. The ClassBrowser interface always shows all the nodes for a given hierarchy level, thus the user can easily switch between branches in the tree at any time. We visualize the tree nodes using an average image and four representatives, similar to the visualization in SmartSketcher. Once the user spots an image that represents the desired concept, he can use the image as query and the system will present results based on an L_2 nearest neighbor search in CNN feature space. Figure 9 shows the user interface of ClassBrowser.

4.2.2 Participants and Tasks. We invited twenty people to participate in our study: nine women and eleven men, aged 22-37, eight with some background in computer vision or computer graphics, and nine without any computer science background at all. Nine participants specified their drawing skills to be bad, ten moderate and one good.

A task consists of a textual description of the content of an image as shown in Table 2. In addition, we also display an example image that matches the requested description. We then asked the users to find as many images as possible that match the requested description. The participants had three minutes available to solve a single task with one system. As soon as a user started a task, its description would disappear and users had to rely on their mental image of the task while solving it. We show the test set of five tasks to be completed by the participants in Table 2, three on the *Airplane* dataset and two on the *Bicycle* dataset.

Table 2: Tasks for user study. Water plane image courtesy of Flickr user Tiberiu Ana, airport scene image courtesy of Flickr user m-takagi, space shuttle image courtesy of Flickr user zoxcleb, and tandem bicycle image courtesy of Flickr user bert_m_b.

Task 1 *Water plane, right side view: Find images of a water plane, seen from the right hand side.*	
Task 2 *Airport scene: Find images of an airliner on the ground, with a tower-like building in the background. The building should be located in the center of the image.*	
Task 3 *Airplane with space shuttle: Find images of an airplane transporting a space shuttle piggyback.*	
Task 4 *Child on bicycle with helmet, frontal view: Find images of a child with helmet on a bicycle, seen from frontal view.*	
Task 5 *Tandem bicycle: Find images of a tandem bicycle. That is, a bicycle built for two persons.*	

4.2.3 Procedure. Each participant completed all five test tasks with all three systems. To avoid ordering, first-order carryover, and learning effects, both the system orders and task orders were randomized and balanced.

For each system, each participant was first given an introduction to the user interface of the system, followed by a four minutes tutorial video that demonstrates solving an example task using the particular retrieval system. Next, the participants were given time to practice with the system using example tasks that we prepared on the *Car* dataset. They were given time until they felt familiar with the system. Such a training phase typically took between five and ten minutes, depending on the particular system. Finally, the participants solved all five tasks in the test set using the particular retrieval system. All systems allow the user to repeatedly query the database, and harvest desired images from each query result. After a short break, the procedure started anew with the next system.

After the experiment, we asked the participants to complete a short questionnaire. A typical session took about three hours: 90

Table 3: Testing the null hypothesis that the median number of retrieved images with system x is lower or equal than with system y. We report p-values of right-tailed two-sample Wilcoxon signed rank tests. Red values indicate rejection of the null hypothesis at the 5% significance level. The alternate hypothesis states that system x retrieves more images than system y.

Task	x = SS, y = S	x = SS, y = CB	x = S, y = CB	x = CB, y = S
1	0.001	0.002	0.406	0.612
2	0.001	0.000	0.000	1.000
3	0.000	0.000	0.020	0.982
4	0.002	0.000	0.079	0.926
5	0.007	0.000	0.004	0.997

minutes for introductions, training, and breaks, and 90 minutes for the actual measurement. During the experiment we recorded all user interactions, user drawings, and the lists of collected images.

4.2.4 Experimental Results. We include all results from our user study in the supplemental material, and we provide a statistical analysis of the results here. Figure 10 shows the number of images found by the participants with all three systems for each task. Although the variance between different users is high, participants find more matching images with SmartSketcher on average. In Table 3 we test whether the increase in retrieved images with the SmartSketcher system compared to the others is statistically significant. The table shows the p-values of a paired, right-tailed two-sample Wilcoxon signed rank test on the number of retrieved images for pairs of retrieval systems. We do not use a paired t-test but rely on the Wilcoxon signed rank test instead to avoid the assumption that the random variable (number of retrieved images) follows a normal distribution. The test examines the null hypothesis that participants do not find more images with system x in comparison to system y with statistical significance, or more precisely, that the data in $x - y$ comes from a distribution with median not greater than zero. The resulting p-values refute the null hypothesis and show that our system (SS) outperforms both baselines (S and CB) on all tasks with statistical significance at the 5% significance level. In addition, the comparison of the two baseline systems shows that users find more images with Sketcher for task 2, 3, and 5, whereas the difference is not statistically significant for task 1 and 4.

Table 4: Testing the null hypothesis that system x is ranked worse or equal than system y. We report p-values of right-tailed two-sample Wilcoxon signed rank tests on the normalized system ranks for pairs of retrieval systems. Red values indicate rejection of the null hypothesis at the 5% significance level. The alternate hypothesis states that system x is ranked better (higher) than system y.

	x = SS, y = S,	x = SS, y = CB,	x = S, y = CB,
p-value	0.000	0.000	0.001

Next, we evaluate the ranking of the systems across all tasks. For this, we normalize the number of images found by a user over all systems. Let $n^X_{i,j}$ denote the number of retrieved images for participant i with system X on task j, where $X \in \{\text{S}, \text{CB}, \text{SS}\}$. The normalized number of images found by participant i with system X on task j is defined as

$$\bar{n}^X_{i,j} = \frac{n^X_{i,j}}{\max\limits_{Y \in \{\text{S},\text{CB},\text{SS}\}} n^Y_{i,j}}. \tag{7}$$

To summarize the overall performance of the three retrieval systems, we define a rank r^X_i for system X and participant i by summing the normalized number of images for system X over tasks j,

$$r^X_i = \sum_{j=1}^{5} \bar{n}^X_{i,j}, \tag{8}$$

such that $r^X_i \in [0, 5]$ and a value of 5 indicates that system X performed best in all 5 tasks for participant i. Figure 11 shows the system ranks for each participant. Wilcoxon signed rank tests for pairs of retrieval systems show that the system rank for SmartSketcher is significantly higher than for the two baseline systems (see Table 4). Moreover, Sketcher is ranked higher than ClassBrowser with statistical significance.

Table 5: Testing the null hypothesis that the median time without finding a single image in system x is higher or equal than in system y. We report p-values of left-tailed two-sample Wilcoxon signed rank test. Red values indicate rejection of the null hypothesis at the 5% significance level. The alternate hypothesis states that the median time in system x is lower than system y.

Task	x = S, y = CB	x = SS, y = CB	x = S, y = SS	x = SS, y = S
1	0.000	0.003	0.412	0.603
2	0.000	0.014	0.023	0.979
3	0.000	0.000	0.588	0.428
4	0.000	0.000	0.693	0.320
5	0.000	0.000	0.245	0.767

During the user study, we measured how much time the participants spent to find a first matching image. Table 5 shows for pairs of retrieval systems whether the times spent without finding a single image is significantly lower for the first than for the second system. We report the p-values of a paired, left-tailed two-sample Wilcoxon signed rank tests, which examine the null hypothesis that the data in $x - y$ comes from a distribution with median greater or equal to zero. That is, it tests whether participants spend less time to find the first image with system x in comparison to system y with statistical significance. The resulting p-values show that users spend significantly less time to find a first matching image using the systems with sketching for all five tasks at the 5% significance level. The comparison of Sketcher and SmartSketcher shows that the differences are not statistically significant, with the exception of task 2, where the participants were significantly faster using Sketcher. We attribute this to the fact that the user has to browse through the clusters in SmartSketcher, which can be time consuming if there are only few matching images in the top n results, since

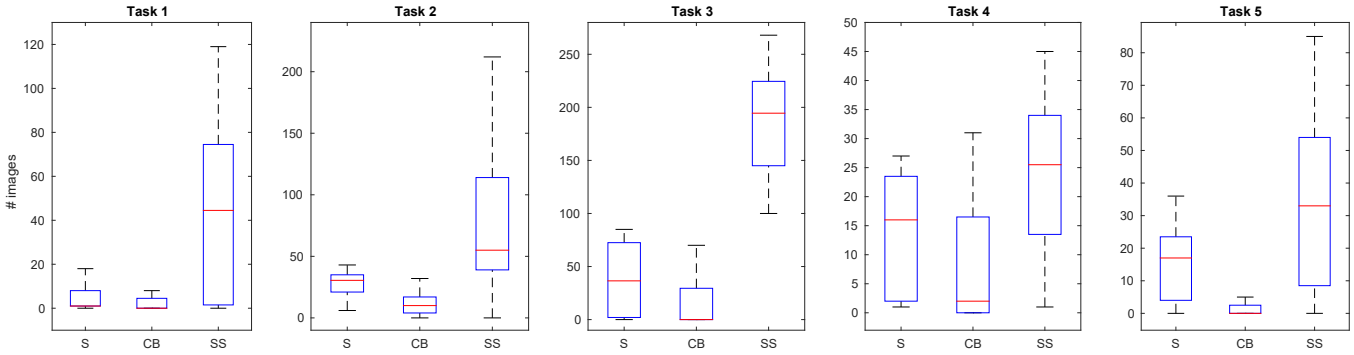

Figure 10: Number of images retrieved with each system (S: Sketcher, CB: ClassBrowser, SS: SmartSketcher) for all five tasks. The red lines show the median number of images, the black lines the mininum and maximum, and the blue boxes the variance.

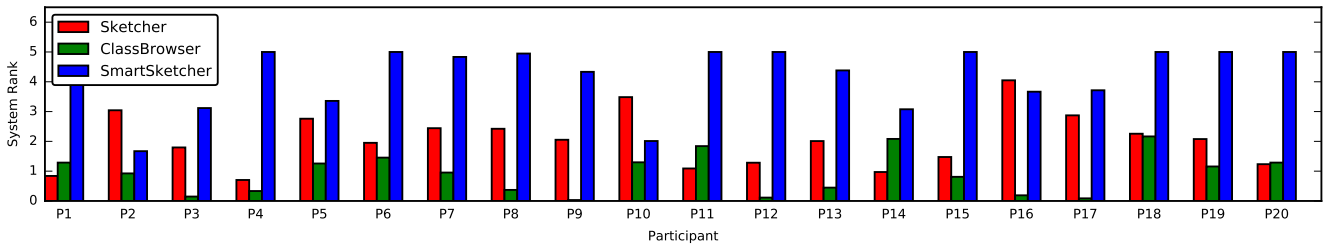

Figure 11: System ranks for all twenty participants P1 to P20.

the matching images end up distributed in different clusters and the system does not detect the desired mode in these cases.

In Figure 12 we plot the average number of retrieved images over time for all tasks. SmartSketcher is very effective in finding more matching images, once a few positive samples are given, and it consistently outperforms both baseline systems. An interesting case is task 2: for this task, SmartSketcher does not detect the requested mode at first. In the first place SmartSketcher proposes images of airplanes on the ground without tower in the background. Only after the user marks the proposed mode as "undesired", the system proposes correct images and retrieval becomes effective. We attribute this to the fact that the CNN features learned to ignore the background of such images during training on object classification. However, the background information is nevertheless encoded in CNN feature space and the user can quickly guide the system towards the correct mode in an "unsharp masking" manner. This example shows that interactivity is key for a robust retrieval system.

In a post questionnaire the participants were asked to specify which system they liked most and least and to explain their answers. Figure 13 shows that users prefer sketching for image retrieval over ClassBrowser in general. In addition, by far most participants preferred SmartSketcher over Sketcher: Seven participants specified that the additional option to mark clusters as "undesired" is very effective. Six participants found the fact that SmartSketcher uses already selected images to refine the query to be effective. Eight participants indicated that they found more images in general, using SmartSketcher. Three participants liked the option to ignore the query sketch for subsequent retrievals. Two participants (P16

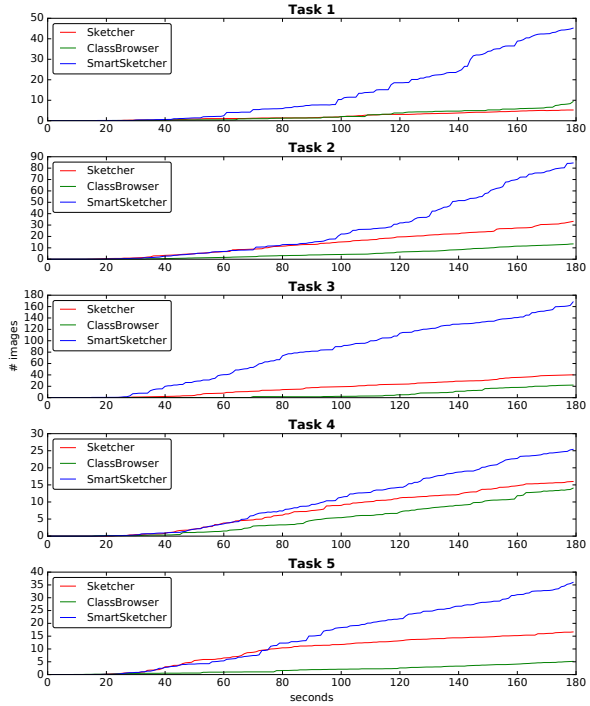

Figure 12: Mean number of images over time over all users.

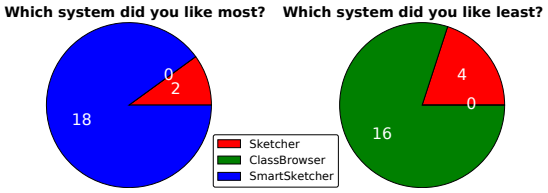

Figure 13: User preference and dislike.

and P17) preferred Sketcher over SmartSketcher for the following reasons: P16 missed the feature to load Canny edges of an arbitrary image into the drawing canvas, which is only available in Sketcher. He used this option to find a first image by modifying the edge map of an almost matching image. As shown in Figure 11 this participant indeed found more matching images using Sketcher on average. P17 found SmartSketcher too complicated but mentioned that he would prefer it over Sketcher if there was no time pressure. Interestingly, the participant found more images using SmartSketcher on average (see Figure 11). Sixteen users did not like ClassBrowser at all. Fifteen of them specified that it is hard to navigate through the classes and that the grouping is not always meaningful. Five participants mentioned they found images only by chance. Four participants found it arduous to use the system, since one has to look through so many images. Four participants disliked Sketcher, all of them mentioned that they have very poor drawing skills and since drawing is the only means to interact with the system they found it hard to use.

5 CONCLUSIONS

We presented an interactive image retrieval system that combines sketching and semantic clustering to re-rank query results and iteratively refine queries. An evaluation on two large-scale benchmarks proves the effectiveness of the proposed technique on three different SBIR systems. In an extensive user study we show that our system is significantly more effective than two baseline systems using sketching or clustering only. Sketching allows users to find initial matches to their queries more quickly than browsing a hierarchically clustered image database, and clustering query results helps to find more matching images faster than in a system purely based on sketching. The success of our clustering approach stems from the use of semantic features that we extract from a CNN trained on image classification, which leads to robust, semantically meaningful groupings of query results. This is especially useful in cross-domain settings such as SBIR, where enough training data is available only in one domain. We justify the use of a low-level feature representation for initial sketch-to-image matching by showing that it leads to higher precision in the first several hundred query results compared to a state of the art, deep learning based approach. For future work, this raises the question how effective cross-domain representations can be learned without suffering from potential imbalances in the available training data in different domains.

REFERENCES

S. Belongie, J. Malik, and J. Puzicha. 2002. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, 4 (Apr 2002), 509–522. DOI: https://doi.org/10.1109/34.993558

J Canny. 1986. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 8, 6 (June 1986), 679–698. DOI: https://doi.org/10.1109/TPAMI.1986.4767851

Xiaochun Cao, Hua Zhang, Si Liu, Xiaojie Guo, and Liang Lin. 2013. SYM-FISH: A Symmetry-Aware Flip Invariant Sketch Histogram Shape Descriptor. In *The IEEE International Conference on Computer Vision (ICCV)*.

Bryan Catanzaro. 2013. kmeans. https://github.com/bryancatanzaro/kmeans. (2013).

N. Dalal and B. Triggs. 2005. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 1. 886–893 vol. 1. DOI: https://doi.org/10.1109/CVPR.2005.177

Mathias Eitz, James Hays, and Marc Alexa. 2012. How Do Humans Sketch Objects? *ACM Trans. Graph. (Proc. SIGGRAPH)* 31, 4 (2012), 44:1–44:10.

Mathias Eitz, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. 2011. Sketch-Based Image Retrieval: Benchmark and Bag-of-Features Descriptors. *Visualization and Computer Graphics* 17, 11 (2011), 1624–1636. DOI: https://doi.org/10.1109/TVCG.2010.266

Mathias Eitz, Ronald Richter, Tamy Boubekeur, Kristian Hildebrand, and Marc Alexa. 2012. Sketch-based Shape Retrieval. *ACM Trans. Graph.* 31, 4, Article 31 (July 2012), 10 pages. DOI: https://doi.org/10.1145/2185520.2185527

Mathias Eitz, Ronald Richter, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. 2011. Photosketcher: interactive sketch-based image synthesis. *IEEE Computer Graphics and Applications* (2011).

Rui Hu, Mark Barnard, and John Collomosse. 2010. Gradient field descriptor for sketch based retrieval and localization. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*. IEEE, 1025–1028. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5649331

Rui Hu and John Collomosse. 2013. A Performance Evaluation of Gradient Field HOG Descriptor for Sketch Based Image Retrieval. *Comput. Vis. Image Underst.* 117, 7 (July 2013), 790–806. DOI: https://doi.org/10.1016/j.cviu.2013.02.005

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093* (2014).

Qiang Li, Yahong Han, and Jianwu Dang. 2015. Sketch4Image: a novel framework for sketch-based image retrieval based on product quantization with coding residuals. *Multimedia Tools and Applications* (2015). DOI: https://doi.org/10.1007/s11042-015-2645-y

Yi Li, Tim Hospedales, Yi-Zhe Song, and Shaogang Gong. 2014. Intra-category sketch-based image retrieval by matching deformable part models. In *Proceedings of the British Machine Vision Conference*. BMVA Press. DOI: https://doi.org/10.5244/C.28.115

Shuang Liang, Long Zhao, Yichen Wei, and Jinyuan Jia. 2014. Sketch-based retrieval using content-aware hashing. In *Advances in Multimedia Information ProcessingâĂŞPCM 2014*. Springer, 133–142. http://link.springer.com/chapter/10.1007/978-3-319-13168-9_14

Sarthak Parui and Anurag Mittal. 2014. Similarity-invariant sketch-based image retrieval in large databases. In *Computer VisionâĂŞECCV 2014*. Springer, 398–414. http://link.springer.com/chapter/10.1007/978-3-319-10599-4_26

Xueming Qian, Xianglong Tan, Yuting Zhang, Richang Hong, and Meng Wang. 2016. Enhancing sketch-based image retrieval by re-ranking and relevance feedback. *IEEE Transactions on Image Processing* 25, 1 (2016), 195–208.

Jose M. Saavedra. 2014. Sketch based image retrieval using a soft computation of the histogram of edge local orientations (S-HELO). In *Image Processing (ICIP), 2014 IEEE International Conference on*. IEEE, 2998–3002. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7025606

Jose M. Saavedra and Benjamin Bustos. 2010. *Pattern Recognition: 32nd DAGM Symposium, Darmstadt, Germany, September 22-24, 2010. Proceedings*. Springer Berlin Heidelberg, Berlin, Heidelberg, Chapter An Improved Histogram of Edge Local Orientations for Sketch-Based Image Retrieval, 432–441. DOI: https://doi.org/10.1007/978-3-642-15986-2_44

Jose M. Saavedra and Benjamin Bustos. 2014. Sketch-based image retrieval using keyshapes. *Multimedia Tools and Applications* 73, 3 (2014), 2033–2062. http://link.springer.com/article/10.1007/s11042-013-1689-0

Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. 2016. The Sketchy Database: Learning to Retrieve Badly Drawn Bunnies. *ACM Transactions on Graphics (proceedings of SIGGRAPH)* (2016).

Ravi Kiran Sarvadevabhatla and R. Venkatesh Babu. 2015. Freehand Sketch Recognition Using Deep Features. *CoRR* abs/1502.00254 (2015). http://arxiv.org/abs/1502.00254

O. Seddati, S. Dupont, and S. Mahmoudi. 2015. DeepSketch: Deep convolutional neural networks for sketch recognition and similarity search. In *2015 13th International Workshop on Content-Based Multimedia Indexing (CBMI)*. 1–6. DOI: https://doi.org/10.1109/CBMI.2015.7153606

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. CNN Features Off-the-Shelf: An Astounding Baseline for Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Xinghai Sun, Changhu Wang, Avneesh Sud, Chao Xu, and Lei Zhang. 2013. MagicBrush: Image Search by Color Sketch. In *Proceedings of the 21st ACM International Conference on Multimedia (MM '13)*. ACM, New York, NY, USA, 475–476. DOI:

https://doi.org/10.1145/2502081.2502276

Zhenbang Sun, Changhu Wang, Liqing Zhang, and Lei Zhang. 2012. Query-adaptive Shape Topic Mining for Hand-drawn Sketch Recognition. In *Proceedings of the 20th ACM International Conference on Multimedia (MM '12)*. ACM, New York, NY, USA, 519–528. DOI: https://doi.org/10.1145/2393347.2393421

Attila Szabo, Andrea Vedaldi, and Paolo Favaro. 2015. Building the View Graph of a Category by Exploiting Image Realism. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*.

Furuya Takahiko and Ohbuchi Ryutarou. 2014. Visual Saliency Weighting and Cross-Domain Manifold Ranking for Sketch-based Image Retrieval. *Multi-Media Modeling* (2014). http://www.kki.yamanashi.ac.jp/~ohbuchi/online_pubs/MMM_2014_Furuya/MMM2014_Furuya_Web.pdf

A. Vedaldi and B. Fulkerson. 2008. VLFeat: An Open and Portable Library of Computer Vision Algorithms. (2008).

Fang Wang, Le Kang, and Yi Li. 2015. Sketch-Based 3D Shape Retrieval Using Convolutional Neural Networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cao Yang, Wang Changhu, Zhang Liqing, and Zhang Lei. 2011. Edgel Index for Large-Scale Sketch-based Image Search. *CVPR* (2011). http://research.microsoft.com/pubs/149199/0630.pdf

Qian Yu, Feng Liu, Yi-Zhe Song, Tao Xiang, Timothy M. Hospedales, and Chen Change Loy. 2016. Sketch Me That Shoe. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Rong Zhou, Liuli Chen, and Liqing Zhang. 2012. Sketch-based Image Retrieval on a Large Scale Database. In *Proceedings of the 20th ACM International Conference on Multimedia (MM '12)*. ACM, New York, NY, USA, 973–976. DOI: https://doi.org/10.1145/2393347.2396360