An evaluation of descriptors for large-scale image retrieval from sketched feature lines

Mathias Eitz*, Kristian Hildebrand*, Tamy Boubekeur+ and Marc Alexa*

* TU Berlin
+ Telecom ParisTech - CNRS *LTCI*

Abstract

We address the problem of fast, large scale sketch-based image retrieval, searching in a database of over one million images. We show that current retrieval methods do not scale well towards large databases in the context of interactively supervised search and propose two different approaches for which we objectively evaluate that they significantly outperform existing approaches. The proposed descriptors are constructed such that both the full color image and the sketch undergo exactly the same preprocessing steps. We first search for an image with similar structure, analyzing gradient orientations. Then, best matching images are clustered based on dominant color distributions, to offset the lack of color-based decision during the initial search. Overall, the query results demonstrate that the system offers intuitive access to large image databases using a user-friendly sketch-and-browse interface.

Keywords:
Sketch-based image retrieval, Image databases, Image descriptors, MPEG-7

1. Introduction

Digital cameras have lead to vast amounts of digital images, many accessible for free through the internet (e.g. Flickr). Finding an image in a database that is close to a mental model is an important and difficult task. Currently, most queries are either based on textual annotations, rough color sketches or other images, respectively parts of images [1, 2].

We feel that images cannot be succinctly communicated and searched based on words alone; humans would probably describe different parts of the

image and use different words depending on their cultural or professional background. On the other hand, searching an image based on a query that looks very similar to the intended result either requires an existing image, whose absence is usually the reason for a search, or great artistic skill if a shaded rendition of the image is necessary. It seems that it is much easier for humans to sketch the main feature lines of a shape or scene. This might be connected to how humans think of and memorize objects [3, 4, 5]. Note that the main feature lines of an image almost completely determine its shading [6]. This result has been exploited recently for creating a simple and intuitive to edit vector image representation [7].

1.1. Previous work

The task of comparing a rough sketch of feature lines to an image is natural yet difficult. First approaches to this problem go back to search based on pictorial description in 1979 [8]. Most approaches to image retrieval based on outline sketches up to now still use involved algorithms: Hirata et al. [9] search in a database of 205 colored oil-paintings by matching the edge image of the database images against the user sketch. Images are normalized in size and subdivided into 8×8 local blocks. For each local block, the best local correlation is computed by searching in a small window of local blocks. The global similarity is then computed as the sum of the local correlation values. Chan et al. [10] search in a database of 137 color images by comparing attributes of salient edge features, such as as length, curvature and spatial relationship. The edges are modeled as implicit polynomials, resulting in a retrieval algorithm that is tolerant towards local distortion in the input sketches. Rajendran and Chang [11] employ direction and curvature histograms for encoding the strongest edges in an image. The proposed method uses multiple scales to account for various levels of detail in user's sketches. The resulting system is used to search in a database of approximately 5,000 images of paintings and other objects. Lopresti et al. [12] recognized that a user sketch can be seen as a special form of handwriting and cleverly treat the search as a string matching problem in a database of 125 sketches. Jain et al. [13] combine color and shape information (using a linear combination of color histogram and edge histogram similarity measures) to retrieve trademark images out of a database of 400 images. Hurtut et al. [14] use curvature motion flows to analyze pictorial content in line-drawings in small databases.

Other works are based on matching a single curve to the sketch: Del Bimbo et al. [15] and Sclaroff [16] let the user sketch undergo bend and

stretch deformation to match the contours. Matusiak et al. [17] represent contours in curvature scale space [18] and define a distance measure for curves represented in curvature scale space. Ip et al. [19] present an affine invariant description for *single* contours.

As most current retrieval algorithms for *large* image collections, our system is based on a small descriptor (high dimensional feature vector) that captures essential properties of the images. Image similarity is then defined by a distance metric over the feature vector. Typical descriptors use global or localized histograms of intensity, color, directionality [20, 21, 22, 23] or coefficients of global image transformations [24, 25]. These descriptors fail to generate good results for sketched feature lines as input. A descriptor specifically designed for search based on edges [26] employs an angular radial partitioning (ARP) of the images and a histogram of the number of edge pixels falling into angular bins. The final feature vector is taken to be the magnitude of the Fourier transform of that histogram to achieve invariance to rotations. Initially proposed in the MPEG-7 standard [27, 28] as a descriptor for capturing location and orientation distribution of edges in texture images, the edge histogram descriptor (EHD) has been found to be as well applicable to sketch-based image retrieval [29, 30]. It has been extended to additionally capture semiglobal edge distributions by Won et al. [31].

1.2. Scope

We believe that either deciding which single contour to extract or matching against a set of contours in *each* image is unlikely to scale to large databases. Instead, a method that is able to flexibly capture contours of an arbitrarily large number of objects in an image while representing the extracted data as a high-dimensional feature vector is desirable. This decouples the search from the representation and thus allows using *standard* nearest-neighbor search algorithms [32, 33]. In this work, we therefore only consider descriptors that can be represented as high-dimensional feature vectors. The focus of this work, however, is on evaluating retrieval quality, we are not trying to maximize retrieval speed. On our collection of 1.5 million images, a standard linear search algorithm yields reasonably interactive retrieval performance and we therefore generate all our results and perform all evaluations using *exact* linear search. To show that the proposed approach is indeed scalable to even larger collections, we present preliminary performance results using a state-of-the-art approximate nearest neighbor search algorithm in Section 4.

1.3. Contributions

In [34] we have presented a system for sketch-based image retrieval that yields interactive search results on a database of more than one million images. We extend this paper by providing additional technical details at all major steps as well as a new thorough evaluation of retrieval performance. The proposed image representations are shown to significantly outperform existing approaches.

Descriptors. We use both the ARP and EHD descriptor as a baseline for evaluation and compare their performance to that of the Tensor and HOG (histogram of oriented gradients) descriptor introduced in [34]. The Tensor and HOG descriptor efficiently capture distribution of location and orientation of gradients in the image but differ in the way this information is encoded. The first descriptor is based on structure tensors [35, 36], which encode the main gradient orientation in a certain image area. The second descriptor employs local histograms of oriented gradients [37, 38] and can be seen as an extension of the EHD descriptor. A main feature of both descriptors is that they elegantly address the asymmetry between the binary user sketch on the one hand and the full color image on the other hand. Both descriptors are constructed in such a way that both the full color image and the binary sketch undergo exactly the same preprocessing steps. This results in an elegant formulation and considerably eases implementation.

Evaluation. Evaluating an image retrieval system is simple if *correct* annotations for all images in the collection are given. Given a query image/sketch, each annotation would allow to make a binary decision whether the corresponding image is relevant to that query or not. Then, descriptors and systems can be compared by means of their precision/recall plots or advanced variants thereof. While such annotated dataset have been created, e.g. for relatively small shape databases [39], such datasets do not exist for sketch-based image retrieval. Creating such a dataset would be desirable but currently seems to be infeasible for large image collections such as the one used in this paper. Even if such a dataset existed, a fundamental problem remains: human performance (drawing a query sketch) and system performance both influence the result while it would be desirable to solely measure system performance. This is a problem in many experiments that require human input, and usually one tries to minimize the variation in human input as much as possible.

We propose to evaluate descriptors based on a set of reference images selected from the collection. For each reference image, we create a corresponding sketch and then measure on what rank a descriptor returns the reference image when querying with that sketch. Assuming that the shapes depicted in the sketch are close to those in the reference image, we expect a good descriptor to return the reference image in the top ranks, even under slight affine deformations of the sketch.

Since we are interested in isolating system performance, we need to make sure that sketched feature lines are close to where they are in an image, including only certain variation resulting from free-hand sketching. We therefore generate the query sketches by a) tracing the reference image, thus eliminating human influence as much as possible and as a compromise by b) drawing the sketches from memory while allowing the participants to have second looks at the reference image. On this basis we can objectively compare different descriptors, which is the main point of this paper.

2. System Overview

We have downloaded a set of 1.5 million pictures retrieved from *Flickr*, only images with a minimum resolution of 640x480 pixels have been retrieved. The maximum size of the downloaded images has been limited to 1024x768 pixels, downscaling larger images. All downloaded images have been stored in jpeg format in a simple folder structure on harddisk. The database memory footprint is 375 gigabyte resulting in an average jpeg filesize of 250 kilobyte.

The input of our image search engine is a set of binary outlines (see Figure 1, left) which are sketched by the user to define the desired *shape* of the searched content. The result of a query is a small collection of pictures with similar *shape* but spanning a potentially large range of hues. In order to provide the user with a mechanism for quickly finding the correctly colored image in the result set, we additionally cluster the search results according to a color histogram descriptor into a small number of clusters (typically in the order of five to ten). The user can then quickly find the cluster containing matches of desired color and choose from that cluster the image best matching the shape outlined in the sketch. The clustering is described in more detail in Section 4.

Our image ranking algorithm is based on descriptors which capture the main directions in each part of the image and are computed for all images in the database in an offline process. During the query, the user sketch provides

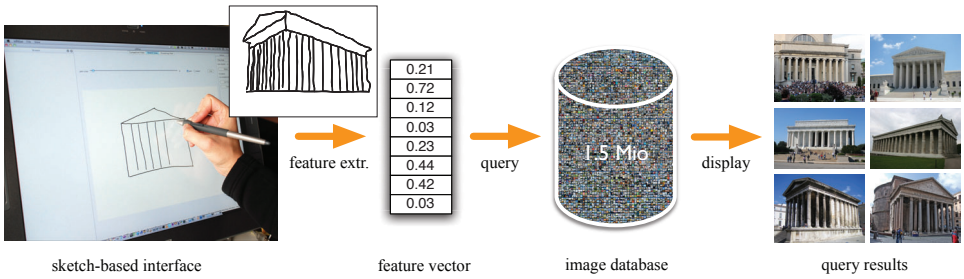

Figure 1: Overview over the proposed system pipeline: A sketch-based interface is used to generate hand-drawn binary sketches. A feature vector encoding essential properties is then extracted from the input sketch and used to query the image database for similar images which are presented to the user.

direction information for each spatial region. A descriptor is extracted from that sketch and compared to the descriptors in the database using a nearest neighbor search algorithm. Images corresponding to descriptors found as nearest neighbors to the query descriptor are returned and presented to the user.

We have implemented the proposed methods into an integrated sketch-based image retrieval system which can be used by any novice user to quickly query an image database (see Section 6 for timings) or even create new content using recent sketch-based photo synthesizer systems [40, 41]. The power of the system stems from exploiting the vast amount of existing images, which offsets obvious deficits in image descriptors and search. We analyze the properties of the proposed descriptors and evaluate their retrieval performance in Sections 5 and 6.

3. Sketch-based image descriptors

Almost all image descriptors are designed for matching entries in the database against a given (partial) image [20, 21, 22, 23, 24, 25, 42]. These descriptors can be used for user generated input only if this input resembles the image in color, intensity, or directionality. A vector-valued or scan-converted sketch of feature lines is not compatible with these descriptors, and we believe searching image databases based on this input can be considered harder than based on input already resembling the database entries.

In the following, we first quickly describe two existing descriptors for sketch-based image retrieval: angular radial partitioning proposed by Chalechale

et al. [26] and the edge histogram descriptor defined in the MPEG-7 standard [29]. We then describe the two descriptors introduced by Eitz et al. [34] and show in Section 5 that they overcome deficiencies of the existing approaches.

Partitioning strategy and features. All four descriptors subdivide the image using a regular partitioning strategy and extract image features from each resulting local region. Throughout this paper we call the local image regions arising from the subdivision process "cells". The local features of all cells (using a *fixed* spatial layout) are then taken to form a feature vector representing the image. The four descriptors analyzed in this paper differ in how they subdivide the image (regular grid in the case of the Tensor, HOG and EHD descriptor, radial-angular subdivision in the case of the ARP descriptor) and what information is extracted from each cell (gradient orientation in the case of the HOG and EHD descriptor, structure tensors in the case of the Tensor descriptor and Canny feature lines [43] in case of the ARP descriptor).

Partial matching. In order to obtain pictures which contain an object fitting the user sketch but also other objects in different locations, every empty cell (i.e. that has no intersection with the user sketch) is ignored in the descriptor-based distance computation and stored in a binary mask. This has three immediate consequences: first, the user can focus on specific picture content and does not have to sketch up an entire picture before querying the database; second, this increases the set of potentially acceptable results by avoiding restrictions on a picture's background; third, this reduces significantly the amount of distance computations during a query.

3.1. Angular Radial Partitioning

Chalechale et al. [26] propose a descriptor explicitly developed for sketch-based image retrieval that is robust against small offsets in location and scale and designed to be rotation invariant. The extraction of the descriptor from an input image requires the following steps: a) convert the image to its gray intensity representation; b) extract local edge features using the Canny edge filter; c) partition the edge map into $M \cdot N$ radial-angular partitions (illustrated in Figure 2); d) count the number of edge pixels falling into each partition e) apply the Fourier transform to each resulting radial histogram in order to achieve rotation invariance.

Image features are extracted by considering pixels $I(\phi,\theta)$ in the binary edge map resulting either from the Canny edge extraction step or the binary hand-drawn sketch. Figure 2 shows the edge map of a typical image from our database overlaid with the radial-angular partitioning scheme. The algorithm uses the surrounding circle of I to create $M \cdot N$ radial partitions, where M is the number of radial partitions and N is the number of angular partitions. The angle θ between adjacent angular partitions is $\theta = 2\pi/N$ and the radius of successive concentric circles is $\phi = R/M$ where R is the radius of the surrounding circle of the image (see Figure 2 for an illustration). The image features $f(k,i)$ in each cell are now defined as follows:

$$\mathrm{f}(k,i) = \sum_{\phi=\frac{kR}{M}}^{\frac{(k+1)R}{M}} \sum_{\theta=\frac{i2\pi}{N}}^{\frac{(i+1)2\pi}{N}} I(\phi,\theta) \tag{1}$$

for $k = 0, 1, 2, ..., M-1$ and $i = 0, 1, 2, ..., N-1$. In other words, the ARP descriptor simply counts the number of Canny pixels falling into each cell.

When I is rotated by an angle of θ, the extracted features for each cell are shifted into a neighboring angular bin. To achieve rotation invariance for $f_\theta(k,i)$, the final descriptor exploits this property and stores the magnitude $|F(k,u)|$ of the 1D-Fourier transform of $f(k,i)$ for each i. When shifting the signal (rotating the image) by an angle of θ, the resulting Fourier coefficients change, but the magnitude $|F(k,u)|$ of the transform stays the same due to the shift invariance property of the Fourier transform.

The distance between two ARP feature vectors is defined as their l_2 distance.

3.2. Edge histogram descriptor

The edge histogram descriptor (EHD) has been proposed in the MPEG-7 standard [29] for texture image characterization and has been adapted for SBIR by Manjunat et al. and Chalechale [29, 30].

The EHD represents the distribution of 5 types of edges in local image patches. As illustrated in Figure 3, the image is subdivided into $k \times k$ non-overlapping cells. The edge distribution in each cell is characterized by a 5-bin histogram, distinguishing 5 different types of edges: vertical, horizontal, 45-degree, 135-degree and non-directional edges (illustrated in Figure 3). A histogram is computed for each of the cells individually, resulting in a feature vector of size $5k^2$.

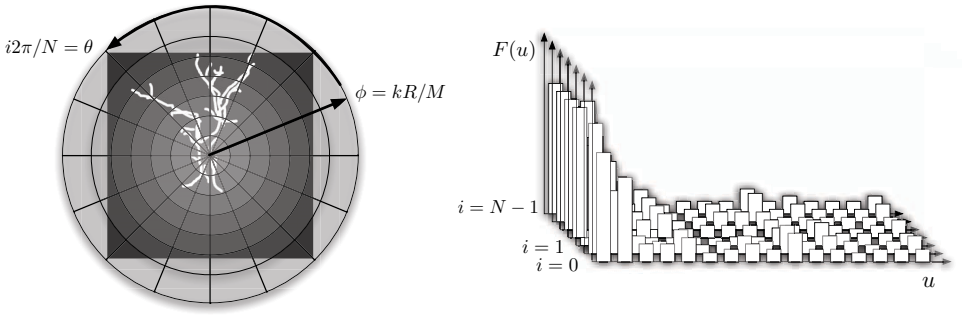

Figure 2: Left: angular radial partitioning of the Canny edge map into M radial and N angular sectors, where $k = 0, 1, 2...M-1$ and $i = 0, 1, 2...N-1$. Right: rotation invariant features for each radial partition i extracted using the 1D Fourier transform.

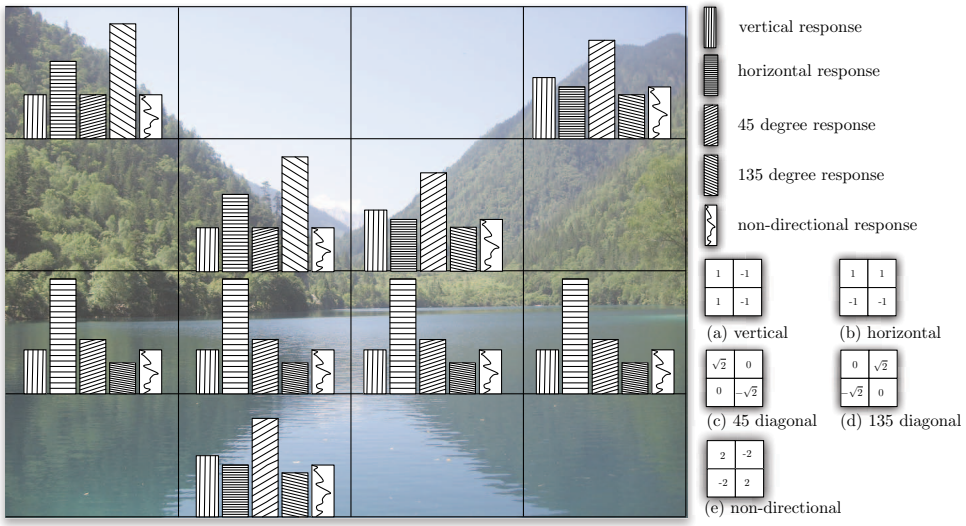

Figure 3: Left: the EHD distinguishes 5 types of edges for each cell and stores the values in a 1D array. Bottom right: five edge filters used for the EHD.

For extracting directional edge features for each subimage we follow Yamada et al. [28]. The count for the determined direction in the appropriate histogram bin of the image patch is increased by one if the measured edge response is greater than a certain threshold t. For a binary sketch, we use $t = 0$ and for images $t = 11$ [30]. Yamada et al. also propose to improve the resulting descriptor using semiglobal histograms that are generated from the local histograms over all rows, columns and four-by-four patches of the subdivision grid [28]. We also evaluate this extension and refer to it as the EHD semiglobal descriptor throughout the rest of the paper.

The distance between two EHD feature vectors is defined as their l_1 distance; a weighted l_1 distance is used for the semiglobal version of the descriptor [28].

3.3. Histogram of oriented gradients descriptor

Clearly, the main type of information in a sketch is the direction of the stroke (i.e. the tangents, resp. normals) relative to its position. This information relates best to the direction of gradients. Note that it is important to ignore the sign of the gradient, as the feature line only contains the information that gradients in the image are expected to be orthogonal to the line, but not which of the two regions is supposed to have higher intensity. In the following we review two descriptors that collect information about the gradients in each image in the database, specifically designed to be independent of the sign of gradients [34].

Let I denote an image with dimensions $m \times n$. We write $\mathbf{g}_{uv} = \nabla I_{uv}$ for the gradient. For both approaches we consider a regular decomposition of the image into cells C_{ij}. We say $(u, v) \in C_{ij}$ if the pixel with coordinates u and v is contained in the cell with index (i, j). We compute gradients using Gaussian derivatives with $\sigma = 1$. The additional smoothing is essential for computing reliable orientation information, especially in the case of a rough binary user sketch. This is different from the implementation in [34] and slightly improves retrieval performance.

The main point of the descriptors described below is to determine the orientation of large gradients in each cell in the image hoping that they correlate with the normal directions of the user sketch. Note that the normals of the user sketch not only lack information on the sign but also have no "magnitude". This means we have to normalize the gradients of the descriptors, which results in regions with large and small gradients being treated equivalently. We discard very small consistent gradients reflecting smooth

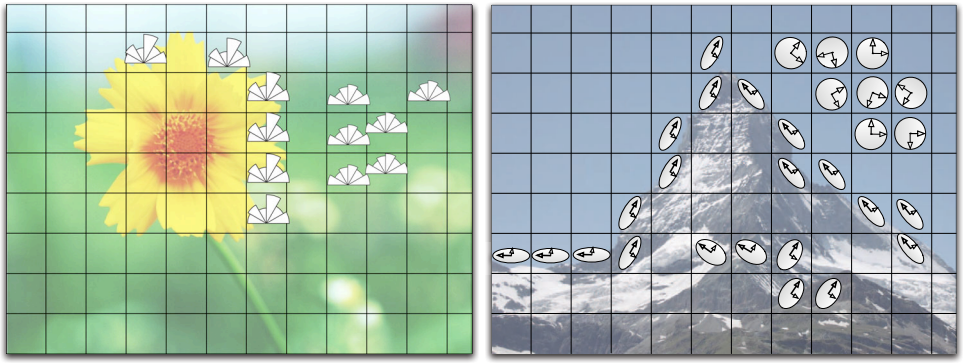

Figure 4: Left: for each image cell the histogram of oriented gradients descriptor stores the sum of squared gradient magnitudes falling into one of six discrete orientation bins. Right: the tensor descriptor subdivides the image into rectangular tiles. For each tile a structure tensor is computed, depicted by the ellipses.

intensity or color transitions, noise or jpeg compression artifacts. In practice we set $\mathbf{g}^\mathsf{T}\mathbf{g} < \varepsilon^2$ to zero. We use $\varepsilon = \sqrt{2}/20$, which corresponds to 5% of the maximum gradient magnitude. We compute gradients on a grayscale image produced from the intensity channel of the input color image. When computing a descriptor from a binary image (user sketch), gradients are directly computed from the binary representation.

The HOG descriptor can be seen as an improved variant of the edge histogram descriptor (EHD) proposed in the MPEG-7 standard [27, 28]. Histograms of oriented gradients have also been employed for human recognition in images [38], as an alternative to shape contexts [44] or SIFT [45]. The descriptor fits our requirement in that it only considers the gradients of the image and can easily be used without considering the sign of the gradient.

For each cell we compute gradient orientations and insert them into the corresponding histogram bin. We weigh each entry by its squared length based on the assumption that relatively stronger gradients are more likely to be sketched by the user. Let h_{ij} be the histogram of cell C_{ij} with d bins, then we define the weight in the k-th bin as

$$h_{ij}(k) = \sum_{(u,v)\in C_{ij}, o(\mathbf{g}_{ij})\in[k/d,k+1/d[} \mathbf{g}_{uv}^\mathsf{T}\mathbf{g}_{uv} \tag{2}$$

with

$$o(\mathbf{x}) = \arccos\left(\operatorname{sgn}\left(\mathbf{e}^\mathsf{T}\mathbf{x}\right)\mathbf{e}^\mathsf{T}\mathbf{x}/\|\mathbf{x}\|\right) \tag{3}$$

where $\mathbf{e}$ is an arbitrary unit direction vector and sgn $\left(\mathbf{e}^{\mathsf{T}}\mathbf{x}\right)$ accounts for the desired equivalence $\mathbf{x} \equiv -\mathbf{x}$.

For the computation of distances between histograms we first compute normalized histograms H_{ij} to account for the possibly different number of gradients in two corresponding cells:

$$H_{ij} = \frac{1}{\sum_k h_{ij}(k)} h_{ij} \tag{4}$$

Now let H_{ij} and $\tilde{H}_{ij}$ denote two normalized histograms. Let d_{ij} denote the l_1 distance between H_{ij} and $\tilde{H}_{ij}$:

$$d_{ij} = \sum_k |H_{ij}(k) - \tilde{H}_{ij}(k)| \tag{5}$$

Finally, we define the distance between two edge histogram descriptors H and $\tilde{H}$ as:

$$\mathrm{dist}(H, \tilde{H}) = \sum_i \sum_j d_{ij} \tag{6}$$

The resulting image description is visualized using pie charts in Figure 4.

3.4. Tensor descriptor

Contrary to the histogram approach where orientations are *discretized* into bins we are interested in finding a single vector in each cell that is *as parallel as possible* to the local image gradients. This vector would be a representative for the image "structure" in that cell. We pose this as a maximization problem and see that the system matrix corresponds to the so-called structure tensor. We only consider discrete scalar images containing luminances here, while the approach can be easily extended for multi-band images [46].

Let $\mathbf{x}$ be a unit vector, which we want to define such that it represents the main direction in cell C_{ij}. As $\mathbf{x}^{\mathsf{T}}\mathbf{g}_{uv}$ attains a maximum if $\mathbf{x} \parallel \mathbf{g}_{uv}$ we pose the definition of $\mathbf{x}$ as the following optimization

$$\mathbf{x} = \underset{\|\mathbf{x}\|=1}{\arg\max} \sum_{(u,v)\in C_{ij}} \left(\mathbf{x}^{\mathsf{T}}\mathbf{g}_{uv}\right)^2. \tag{7}$$

Note that

$$\sum_{(u,v)\in C_{ij}} \left(\mathbf{x}^\mathsf{T}\mathbf{g}_{uv}\right)^2 = \sum_{(u,v)\in C_{ij}} \mathbf{x}^\mathsf{T}\mathbf{g}_{uv}\mathbf{g}_{uv}^\mathsf{T}\mathbf{x} = \mathbf{x}^\mathsf{T}\left(\sum_{(u,v)\in C_{ij}} \mathbf{g}_{uv}\mathbf{g}_{uv}^\mathsf{T}\right)\mathbf{x} = \mathbf{x}^\mathsf{T}G_{ij}\mathbf{x} \tag{8}$$

which means we are maximizing a quadratic function in $\mathbf{x}$ with the constraint $\mathbf{x}^\mathsf{T}\mathbf{x} = 1$. The matrix G_{ij} contains the sum of outer products of gradients in cell C_{ij} and is commonly referred to as the structure tensor. We find the unique maximum using the Langrange multiplier λ, and setting $\nabla\mathbf{x}$ to zero leads to the necessary condition

$$2G_{ij}\mathbf{x} + 2\lambda\mathbf{x} = 0 \tag{9}$$

which means that we can find $\mathbf{x}$ (up to sign) as the unit eigenvector of G_{ij} corresponding to the largest eigenvalue. The eigenvalues correspond to the maximum and minimum of the quadratic functional, reflecting the distribution of gradients. Thus, a compact representation of all this information, yet not including the sign of the gradients, is given by the structure tensor G_{ij}.

In order to detect similarly oriented image edges independently of the magnitude of the edges, we store the structure tensor normalized by its Frobenius norm:

$$T_{ij} = \frac{G_{ij}}{\|G_{ij}\|_F} \tag{10}$$

We define the distance d_{ij} between two tensors T_{ij} and $\tilde{T}_{ij}$ as the Frobenius Norm of the difference between the two tensors:

$$d_{ij} = \|T_{ij} - \tilde{T}_{ij}\|_F \tag{11}$$

Finally, we define the distance between two tensor descriptors as the sum over the tensor distances in their corresponding cells:

$$\mathrm{dist}(T, \tilde{T}) = \sum_i \sum_j d_{ij} \tag{12}$$

A visualization of the resulting image descriptor is given in Figure 4.

Table 1: Grid resolutions of the descriptors ARP, EHD, HOG and Tensor. The ARP descriptor is parameterized by radial-by-angular partitions. The EHD, HOG and Tensor descriptor parameters are a rectangular grid of x-by-y image cells. The settings proposed in the original papers are marked using bold font.

size	ARP	EHD	HOG	Tensor
small	**3x8**	**4x4**	6x4	8x6
medium	10x58	12x12	12x8	16x12
large	20x115	20x20	**24x16**	**32x24**

3.5. Descriptor variants

So far, we have seen a total of 5 descriptors: ARP, EHD, EHD semiglobal, Tensor and HOG. To measure the influence of the grid resolution on retrieval performance, we define 6 variants of each descriptor: three different grid resolutions (small, medium and large) as well as for each resolution a masked and an unmasked version of the descriptor (mask not applicable to the ARP descriptor due to the use of frequency domain coefficients as the features). This results in a total of 27 descriptor variants, we list their respective parameters in Table 1.

The descriptors variants include the parameters used in the original papers; for the ARP descriptor this is the variant (small, unmasked) [26], for the EHD and EHD semiglobal the variant (small, unmasked) [31] and for the Tensor and HOG descriptor the variant (large, masked) [34]. The remaining parameter settings are chosen such that similar descriptor sizes have roughly comparable storage requirements, see Table 4.

4. Image search and clustering

In order to run our sketch-based image retrieval system and evaluate the descriptors we use a standard Apple MacPro configured with 2 Intel Xeon 2.8Ghz QuadCore processors and 32GB of main memory. When starting the system we load all descriptors into a large array which is kept in main memory. Note that all descriptors used in this paper can be considered as points in high-dimensional space. Given a query descriptor and a distance metric, searching best matches thus reduces to finding nearest neighbors in this high-dimensional space. Throughout the paper, we find *exact* nearest neighbors using linear search for evaluation of the descriptors, which we also found to be just fast enough for querying the database interactively given our

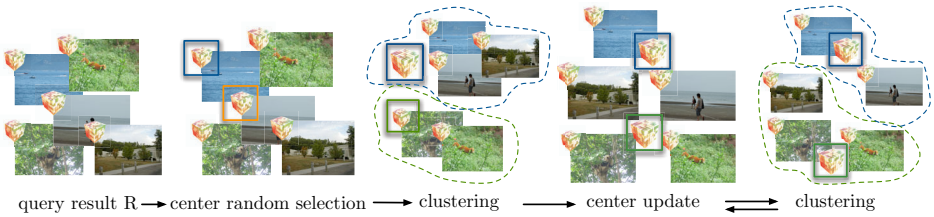

Figure 5: Partitioning query result based on dominant colors: k-means on 3D color histograms.

collection size of 1.5 million images. Note that speeding up the *exact* search is non-trivial due to the high dimensionality of the entries [47]. Instead, we give additional experimental results using an *approximate* nearest neighbor k-means tree algorithm [48, 33] and discuss in Section 7 how this algorithm could be employed to facilitate true web-scale search.

4.1. Linear search

A query is performed by computing the distances between the query descriptor and all other descriptors in the database. Then $\mathbf{R}$, the set of N (typically 50 to 100) images $\{\mathbf{I_0}, ..., \mathbf{I_{n-1}}\}$ with smallest distance to the input descriptors (typically generated from a sketch) is determined using a fixed-size priority queue of N elements. Note that the task of computing distances can be easily parallelized and we do so using a standard Map-Reduce framework.

4.2. K-means tree search

We construct a k-means tree [48] by hierarchically subdividing space using k-means clustering [49]. We decompose the whole set of descriptors into k clusters, and then recursively subdivide the resulting clusters, stopping the recursion when a cluster contains less than k descriptors. We query the resulting tree using a best-bin-first strategy [33], retrieving 2,000 potential nearest neighbors, which we sort according to their distance to the query descriptor and finally return $\mathbf{R}$, the set of N best matches. In our implementation, we use a branching factor of $k = 32$.

4.3. Clustering

Since we perform the search based on *binary* sketches, $\mathbf{R}$ may contain pictures with very different color distributions. In order to help the user

to browse this set, and before presenting $\mathbf{R}$ to him, we cluster it into k clusters (typically 5 to 10) of similarly colored images using the *k-means* algorithm. This algorithm, also known as *Lloyd clustering* [49], selects k random centers among its (possibly high-dimensional) input set and defines clusters by assigning each element in the set to its closest center. Then the algorithm updates the centers by moving them to the centroid of their relative cluster and restarts the assigment step. This procedure is repeated until a given stopping condition is satisfied (e.g. maximum number of iterations, stable clustering detected). This algorithm aims at minimizing an energy E over all clusters: when the notion of "proximity" (closer center) is modeled as the Euclidean distance in the space embedding the set, then E is the sum of squared distance of the elements to their relative center and it effectively minimizes the size of the clusters.

In our search engine, we use it with the setup illustrated in Figure 5. We start by computing a *color* descriptor $\mathbf{c_i}$ for each image of $\mathbf{R}$ which captures the dominant colors of $\mathbf{I_i}$. We then select k random images in $\mathbf{R}$, set their color descriptors as initial centers and run the k-means clustering. In practice, we define $\mathbf{c_i}$ as the color histograms of $\mathbf{I_i}$. The distance used to assign each image $\mathbf{I_i}$ to a given center (i.e. cluster) is the squared l_2 distance between these histograms. We employ three-dimensional color histograms, subdividing the RGB colorspace into $6 \times 6 \times 6$ bins. Figure 13 shows a result of query clustering.

5. Evaluation of retrieval performance

We are interested in evaluating and comparing the relative retrieval performance of the 27 descriptor variants discussed in Section 3. Our evaluation addresses the following two problems:

- How does a descriptor behave under slight affine deformations of a query sketch?
- How well does a descriptor retrieve images given queries from real users?

To analyze those problems we have defined a ground truth set of 43 reference images from our image database covering a wide range of different scenes, several of which we show in Figure 6. In the evaluation we measure on which rank a descriptor returns the ground truth image when querying with the sketch created from that reference image. Performing such queries for a

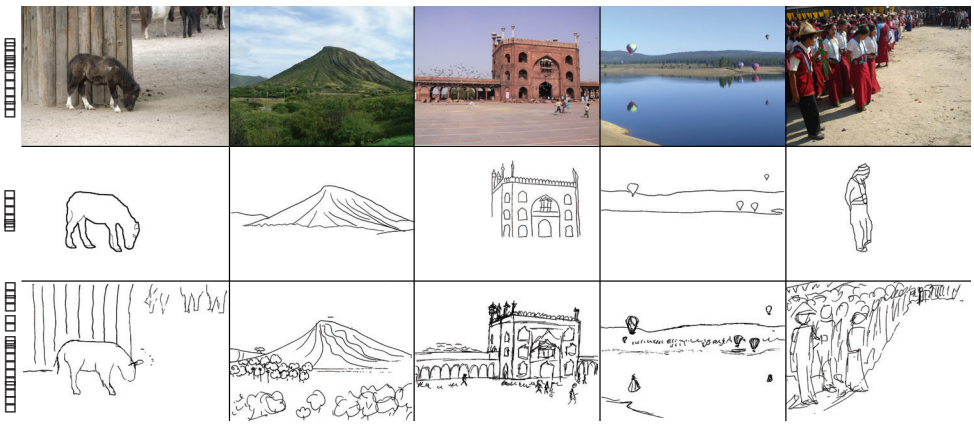

Figure 6: Shown are several of the 43 image/sketch pairs used in the formal evaluation. Top row: images from the database selected as ground truth. Second row: sketches that have been traced from the ground truth images. Third row: sketches that have been drawn from memory by participants of the evaluation. When querying with a sketch, we measure the rank of the corresponding ground truth image from the first row; we expect a good descriptor to return the ground truth image as one of the first results.

large number of input sketches with different descriptors allows for a simple direct comparison of their relative retrieval performance. For each reference image we have generated sketches using two different methods.

Sketches from tracing images. The first set of 43 sketches has been generated by three different users that we instructed to trace the most important outlines in the reference images. Specifically, the reference images were shown in a painting application and we asked the participants to trace what they considered the most important features into a second layer. The input was performed using a tablet screen and pen.

Sketches from memory. The second set of 43 sketches has been generated by nine users that we instructed to sketch a *memorized* version of the reference image. Participants were shown a printout of the reference image and then asked to sketch the memorized image using pen and paper. Participants were allowed to have a second look at the reference printout during sketching when requested in order to reduce the influence of limited short term memory on the resulting sketches. Some of the resulting sketch/image pairs are shown in Figure 6.

Benchmark definition. Using the two sets of reference sketches we have evaluated descriptor performance by querying the database for the most similar images to each sketch and finding the rank of the reference image in the resulting answer. To check robustness of the descriptors we generated queries for 9 translated, 9 scaled and 9 rotated versions of each input sketch. The translated sketches have been generated by translating the original by a factor of up to 0.1 times the width of the sketch in a random direction; scaled versions by scaling the input by a factor between 0.8 and 1.2 and rotated versions by rotating the input around its center by -20 to 20 degrees. Additionally, combinations of translation, scale and rotation were also tested, the combined transformations were generated by applying the three transformations in the order scale, rotate, translate. The detailed set of resulting parameters is listed in Table 2.

In total we generated 36 affine transformed version for each of the 43 reference sketches, gathering a total of 1548 measurements for each of the 27 descriptor variants. Additionally, we evaluated the GIST descriptor designed for example-based image retrieval [21]. First, we tested its suitability for sketch-based image retrieval using the same setup as for the sketch-based descriptors and second, we tested its invariance against affine transformations, querying with the reference images instead of the sketches.

In order to achieve robustness against outliers in retrieval rank, we use robust statistics and report median retrieval ranks for each transformed set of 43 sketches. Figures 7, 8, 9, 10 and 11 summarize the results of the evaluation graphically.

6. Results

As can be seen in Figures 13, 14 and 15 the proposed system gathers good matches for a given query sketch (all queries have been executed using the Tensor descriptor). We show a typical result of a query in Figure 15, displaying the *first* 15 matches. In Figure 14 we show a hand-picked subset of the top 50 matches for each of the three query sketches. For each sketch we show six images that match the probably intended semantics of the sketch and are considered good matches by the experimental users of our system. While the intended semantics of a sketch is not reflected in all of the answers, they still resemble the features in the user sketch; this is shown in each second row of Figure 14.

Table 2: Affine transformations applied to evaluation sketches for testing robustness against distortions. Translation is given as relative to the length in pixels of the sketch diagonal, rotation in degrees around the sketch center and scale as a factor of the sketch sidelength

Translation t	Rotation r	Scale s	Combined
0.1	-20	0.8	$s \circ r \circ t$
0.075	-15	0.85	...
0.5	-10	0.9	...
0.025	-5	0.95	...
0	0	1	...
0.025	5	1.05	...
0.05	10	1.1	...
0.075	15	1.15	...
0.1	20	1.2	...

Table 3: Timings (in $s \cdot 10^{-6}$) for comparing two descriptors with their corresponding distance metric. Descriptors using a mask in the distance computation are marked by (m). Note that using a spatially localized mask is not possible with the ARP descriptor, since its feature vector stores frequency domain coefficients.

Descriptor	ARP	EHD	EHD semiglobal	HOG	Tensor
large	7.76	3.00	51.69	3.40	3.92
medium	2.01	1.07	16.11	0.85	0.97
small	0.09	0.15	2.49	0.23	0.26
large (m)	–	4.97	44.65	6.14	4.13
medium (m)	–	2.30	19.54	1.69	1.19
small (m)	–	0.37	2.60	0.55	0.43

Table 4: Storage requirements of the descriptors ARP, EHD, Tensor and HOG (byte).

Sizes	Tensor	ARP	EHD	HOG
small	576	96	320	576
medium	2304	2320	2880	2304
large	9216	9200	8000	9216

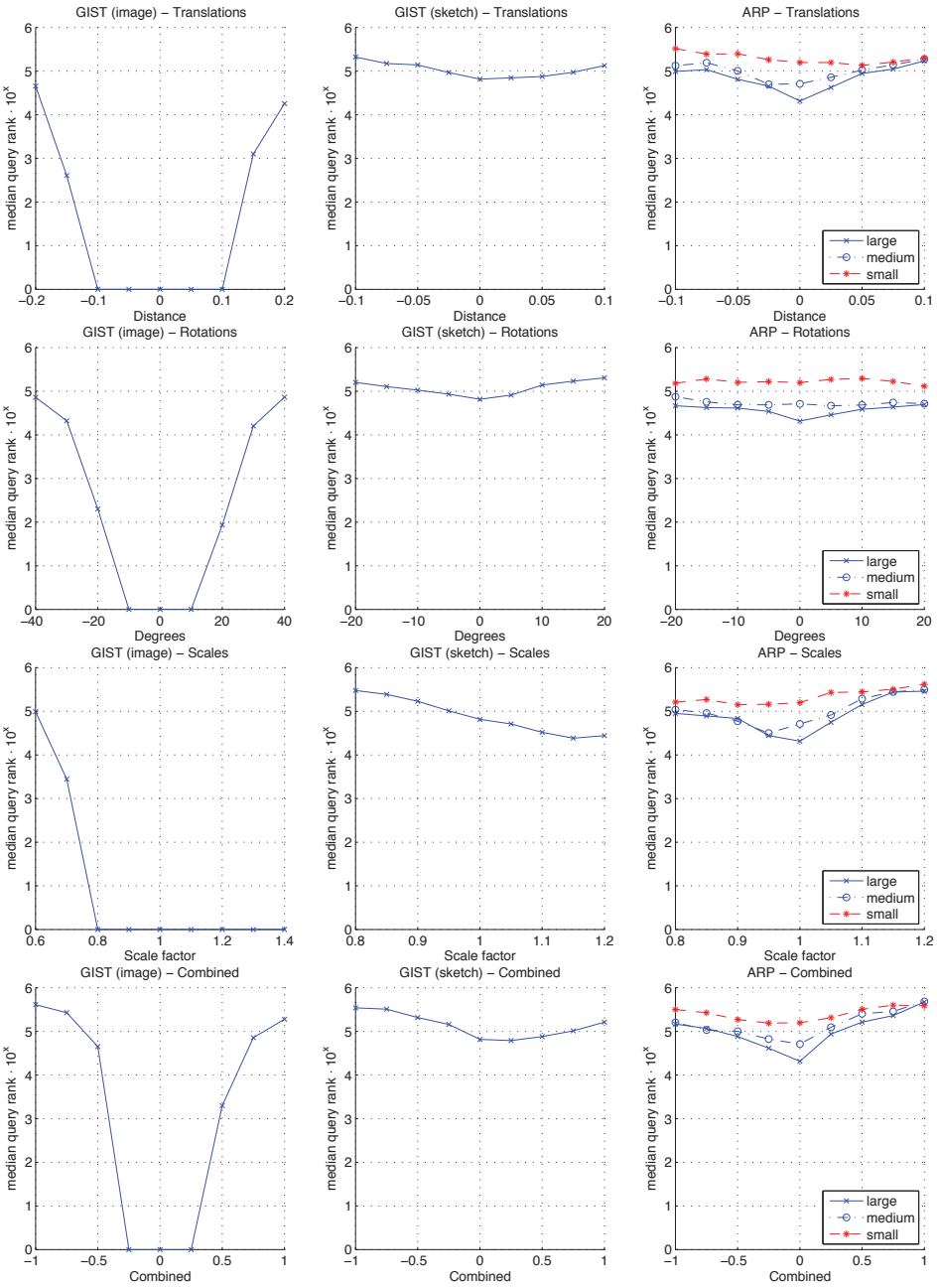

Figure 7: Evaluation results from *traced* sketches. Left to right: GIST (example based), GIST (sketch-based) and ARP. Note the different scale of the x-axis in the first column.

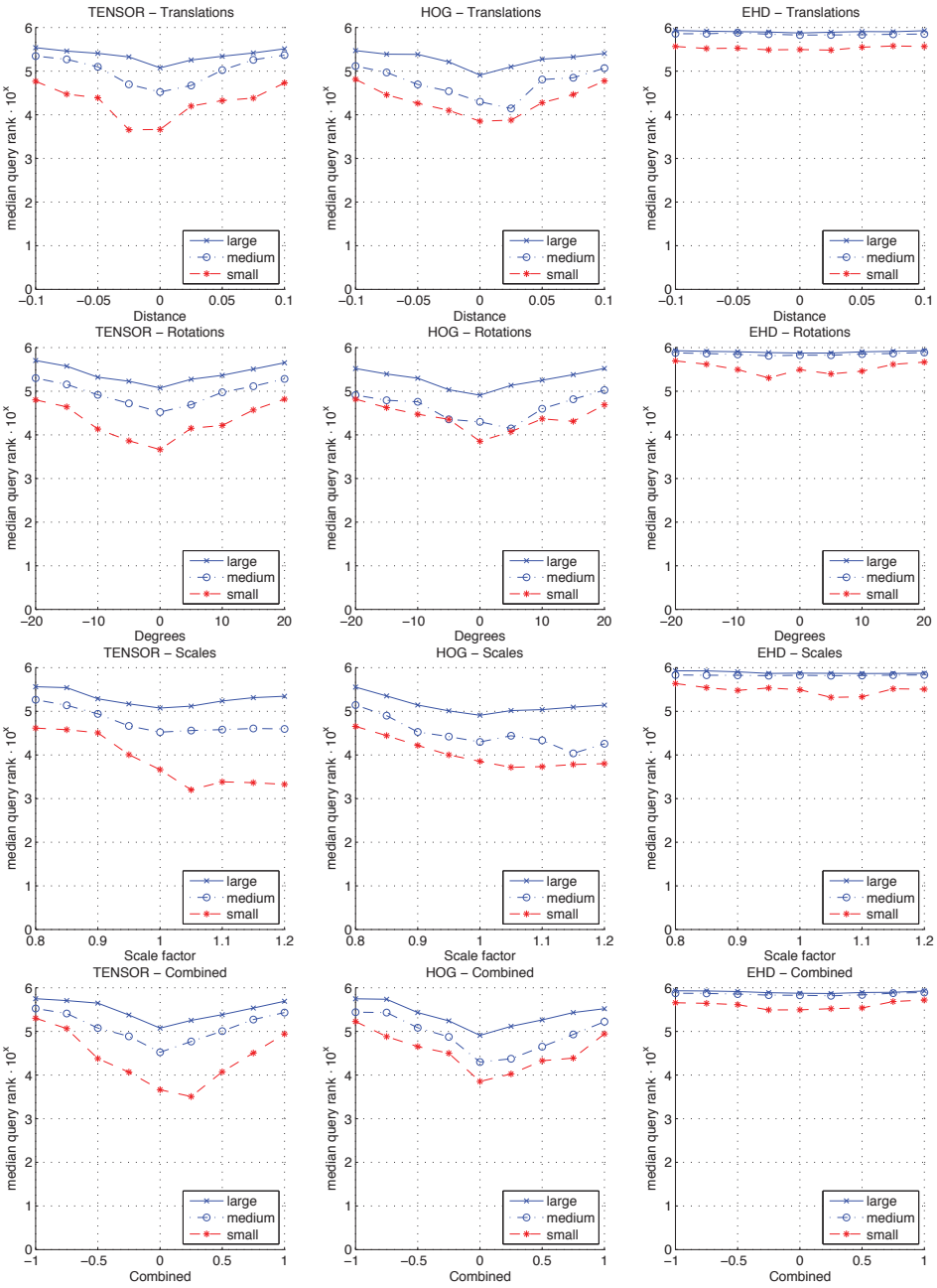

Figure 8: Evaluation results from *traced* sketches using descriptors *without* masks. Left to right: Tensor, HOG and EHD.

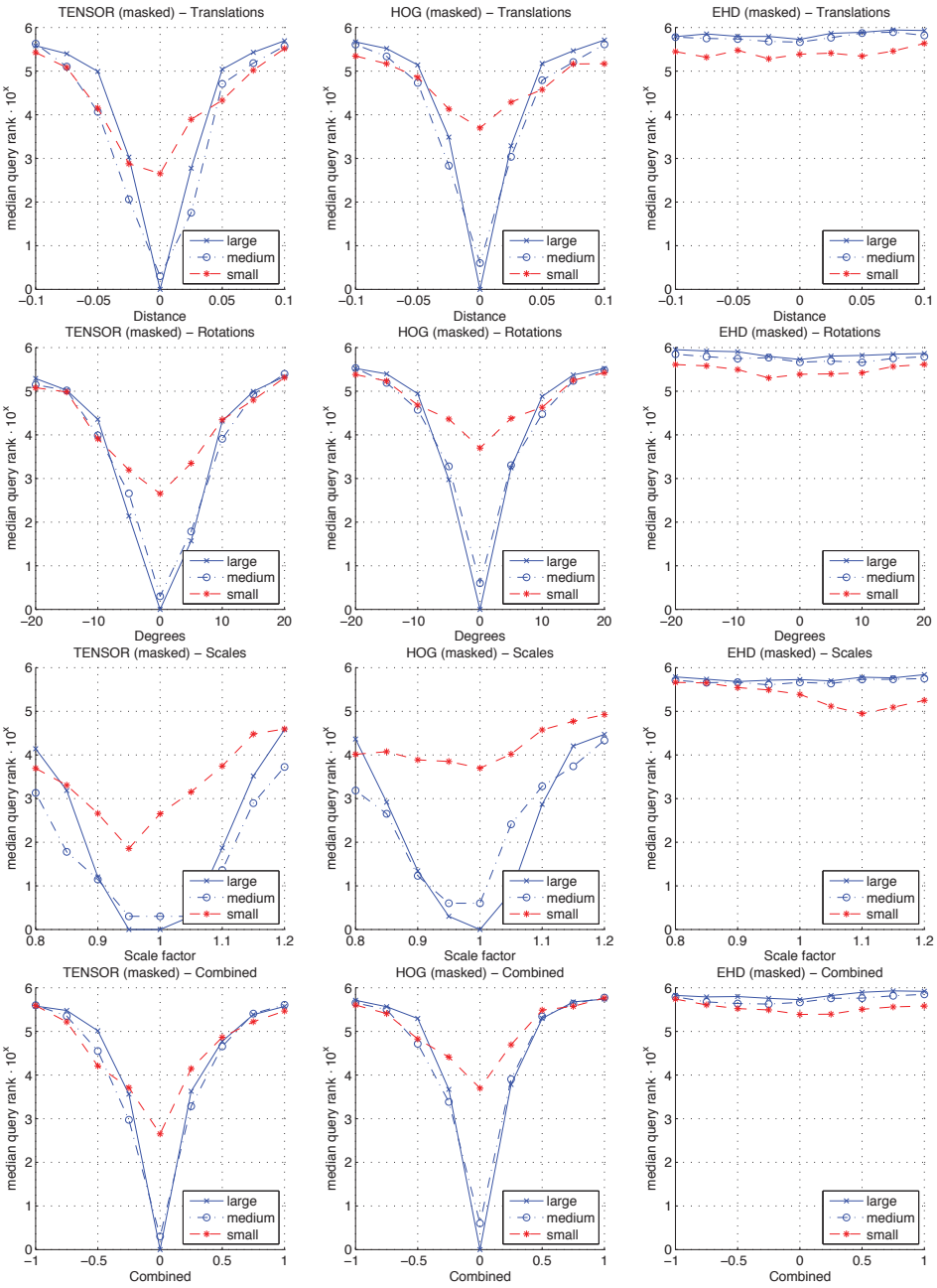

Figure 9: Evaluation results from *traced* sketches using descriptors *with* masks. Left to right: Tensor, HOG and EHD.

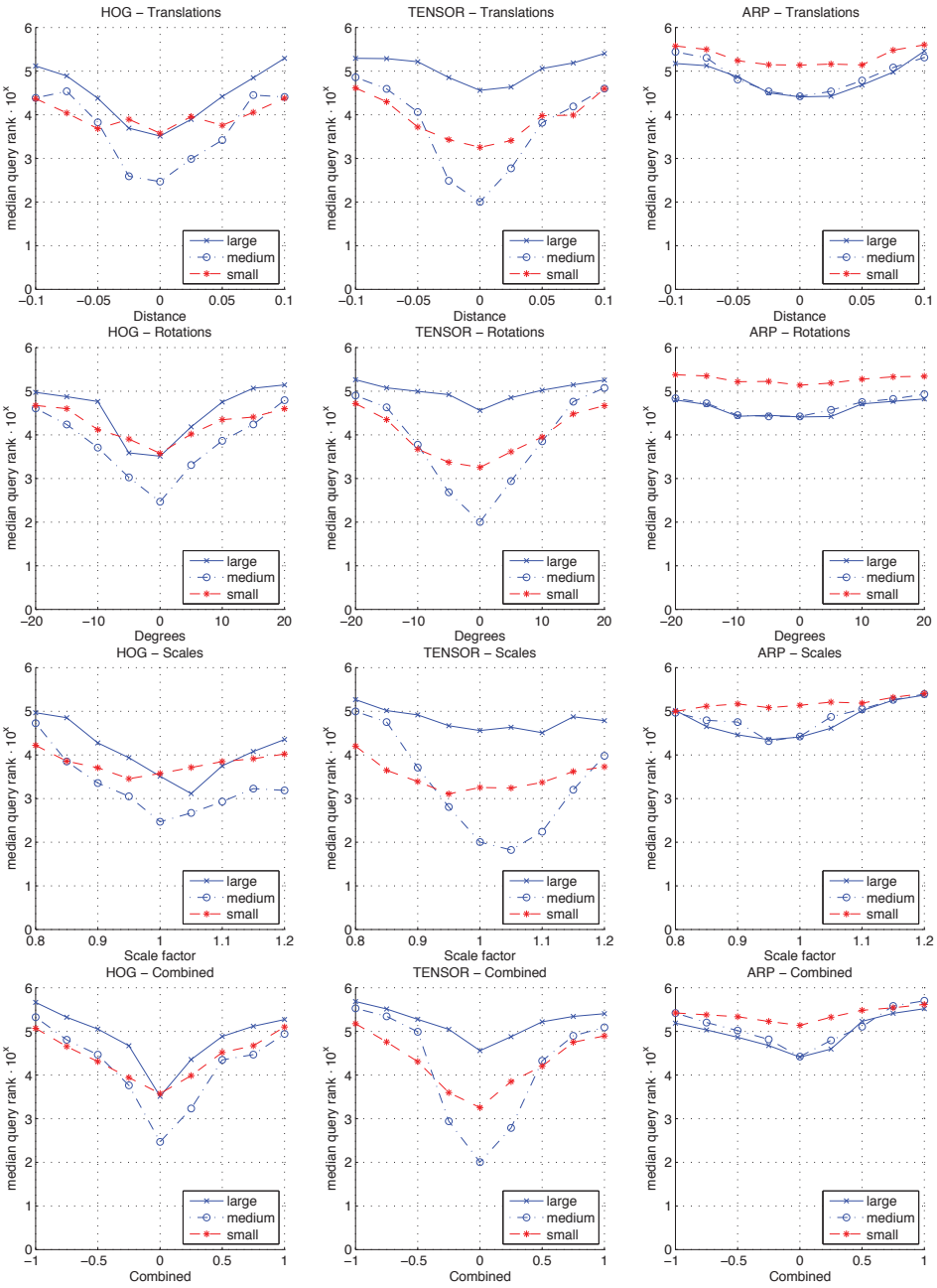

Figure 10: Evaluation results from *memory* sketches using descriptors *without* masks. Left to right: HOG, Tensor and ARP.

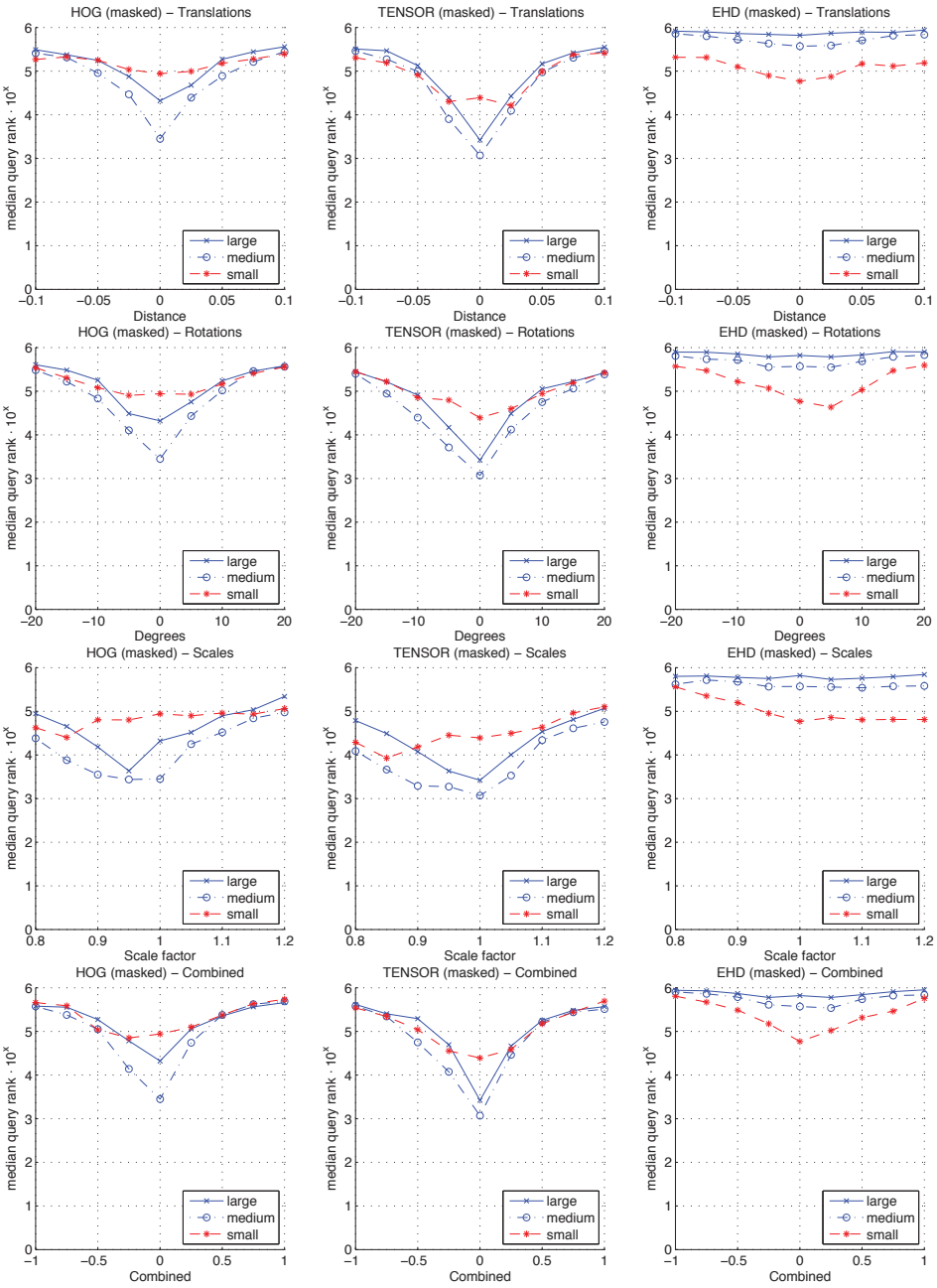

Figure 11: Evaluation results from *memory* sketches using descriptors *with* masks. Left to right: HOG, Tensor and EHD.

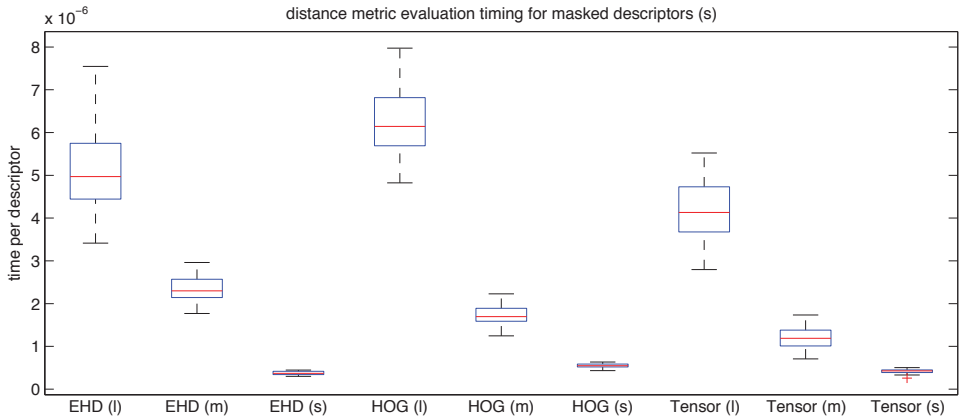

Figure 12: Timings for computing the distance between two masked descriptors (each in the variant large (l), medium (m) and small (s)). Timings are measured for each descriptor by querying using all 43 example sketches. The variance in the timing of single descriptor results from the fact, that the 43 query sketches each have different numbers of cells that are masked out.

Table 5: Timings of the computation of a single descriptor (ms).

Sizes	ARP	EHD	Tensor	HOG
small	10.4	6.5	15.2	9.3
medium	9.3	7.9	15.6	9.3
large	9.1	6.5	15.3	10.2

Figure 13: Answer of the proposed system to the sketch shown in the middle. The first 50 matches are clustered into 6 clusters. Note that the result set contains a very high percentage of trees as probably desired by the user.

6.1. Interactive sketch-based retrieval

We have implemented a simple user interface for drawing and editing sketches that allows to interactively query our database of 1.5 million images. While we have not performed any user study, the use of simple outlines for querying the database has been intuitive to use in our experiments. Typical response times of the system are in the order of a few seconds using linear search, depending on the size of the descriptor chosen.

6.2. Computational performance

We have measured median query time under linear search by performing queries with all 43 sketches used in the evaluation. The result is illustrated in Figure 12 for the Tensor, HOG and EHD descriptor, a detailed listing of the timings of all variants is given in Table 3. Note that we have normalized the measured timings to the time needed for computing the distance between a single pair of descriptors since this is the operation, that has to be performed in the inner loop of any nearest-neighbor search algorithm. The timings have been measured on a single core; in our interactive query system we use a parallelized distance computation using a standard Map-Reduce framework and achieve a speedup roughly in the number of cores available.

The query timings are similar among the descriptors for comparable descriptor sizes; smaller descriptors (storing less data) can be searched faster.

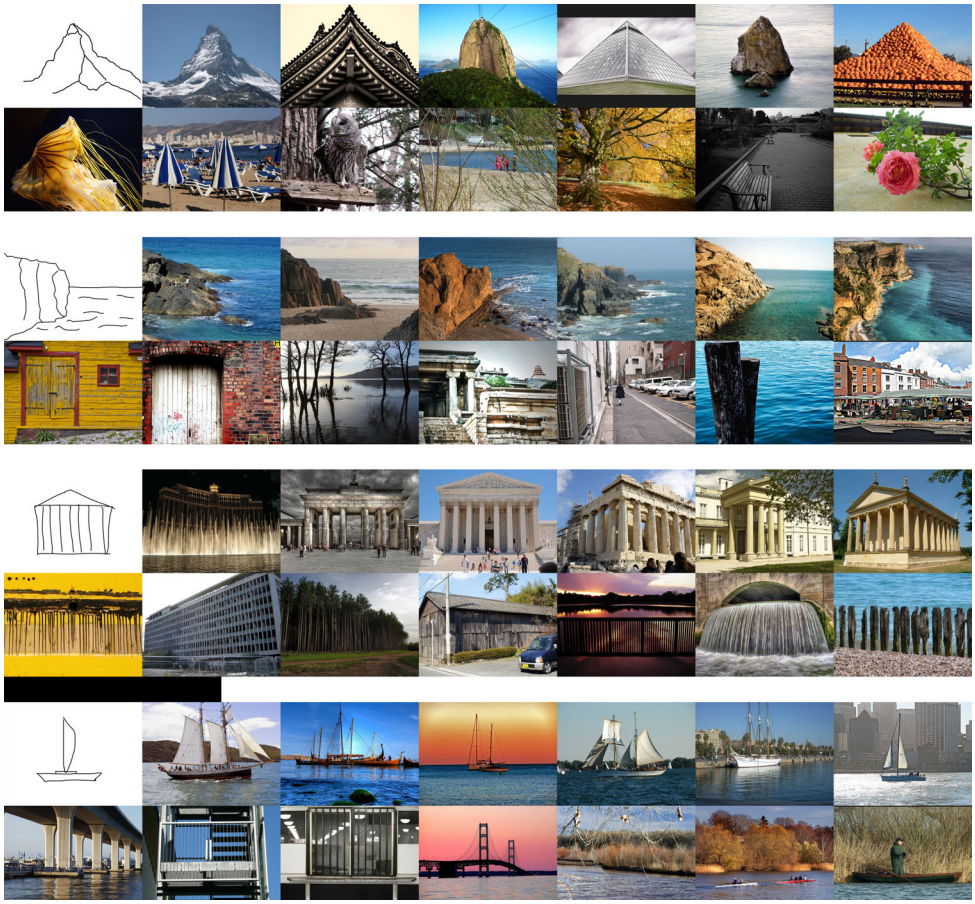

Figure 14: Shown is a hand-picked subset of the results when querying the database for 50 images matching the sketches on the top left of a row. In the top row we show "expected" results, and rather "unexpected" results in the corresponding bottom row.

Figure 15: The first 15 matches (left to right, top to bottom) for the query on the left. The query sketch has been generated from an image in the database, which has been ranked 5,474. Note that the first matches provide a very reasonable answer to the user query, meaning that a low rank of the image used to generate the sketch does not imply the descriptor failed.

The masked version of the descriptors require slightly more time than their unmasked equivalents, probably due to the additional memory accesses needed. Note that the EHD semiglobal descriptor requires an order of magnitude longer for comparison. This is due to the fact, that we compute the semiglobal histograms during runtime as suggested by Won et al. [31]. At the expense of additional memory, this computation could easily be done in the preprocessing step, which seems to be necessary for larger image databases.

Additionally, we have evaluated search performance of the k-means tree algorithm. An approximate search using the HOG medium descriptor takes 6 ms, querying for 2,000 potential nearest neighbors using a best-bin-first strategy and selecting the N best matches from that set. This is more than two orders of magnitude faster than linear search – but also has some potential drawbacks which we discuss in Section 7.

The descriptors considered in this paper are all very efficient to compute, we achieve timings between 6.9 and 15.6 ms for extracting a single descriptor from a (one megapixel) input image (corresponds to 67 to 144 descriptors per second). This timing includes the time for loading the image from harddisk and decoding it (JPEG compression).

6.3. Evaluation of retrieval performance

We have performed an extensive experimental evaluation that allows to concisely assess descriptor performance, gathering a total of $50,000$ measurements. The setup is described in Section 5, the results of this evaluation are shown graphically in Figures 7, 8, 9, 10 and 11. As expected, images are most likely recovered from a sketch if the sketch is in the right position, scale, and orientation, however, we see that small amounts of transformation are tolerable (the ARP descriptor is rotation invariant if rotated by integer multiples of its angular binning angle). Our results do not show any significant advantage of using the semiglobal version of the EHD descriptor over the standard EHD descriptor which employs only local histograms. We therefore only show plots for the standard EHD descriptor in Figures 8, 9 and 11.

Evaluation of traced sketches. In Figures 7, 8 and 9 we can see that both the HOG and the Tensor descriptor provide significantly better retrieval performance than the ARP and the EHD descriptor, with the Tensor descriptor slightly outperforming the HOG descriptor. Both the ARP descriptor and the unmasked versions of the Tensor and HOG descriptor perform comparably, however, all versions of the EHD descriptor perform rather poorly given our evaluation setup. We believe that this is due to the edge extraction techniques applied. While the ARP descriptor uses Canny edges (without directionality information) and both the Tensor and HOG descriptor use the squared magnitude of gradient as the edge information, the EHD descriptor employs a simple edge counting technique, all edges – independently of their magnitude – have the same weight. We can also conclude that – at least for the set of sketches used in our evaluation – the baseline descriptors ARP, EHD and the extended semiglobal version of the EHD descriptor do not scale well to our large database size. We believe that this is – in part – due to the absence of using a mask in the distance computation. For sketches with large blank areas (in our experience, users typically draw sparse sketches), descriptors *without* a mask essentially try to retrieve images that have the same blank areas, the matching of the actual contours thus gaining less influence. To quantify this effect we have generated masked and unmasked versions for all descriptors where applicable. The evaluation results for descriptors *without* masks are shown in Figure 8, the corresponding masked versions in Figure 9. The results for all descriptors clearly show that using a mask for sketch-based image retrieval is essential for good retrieval per-

formance. In Figure 7 we show that the GIST descriptor [21] (a popular descriptor designed for example-based retrieval) can – as expected – not be directly applied to sketch-based image retrieval. Interestingly, when used for example-based retrieval, the GIST descriptor is more affine-invariant than sketch-based descriptors (see Figure 7, left). We discuss possible reasons and conclusions that can be drawn from this observation in Section 7.

Evaluation of memory sketches. In Figures 10 and 11 we show evaluation results for sketches created from memory. All descriptors perform slightly worse than for traced images, we attribute this to the additional abstraction, transformation and distortion introduced into the sketches during the memorization, drawing and digitization process. Consistent with the results from traced sketches, the ARP and EHD descriptors are outperformed by the Tensor and HOG descriptor, with the Tensor again slightly outperforming the HOG descriptor. The best performing descriptor (Tensor medium) is still able to place roughly 50 % of the reference images in the top 100 results – note that the top 100 images correspond to only 0.0067 % of our collection size.

6.4. Additional observations

While in the objective evaluation we can recognize a slight advantage of the Tensor descriptor over the HOG descriptor, the users of our experimental system indeed seemed to prefer the query results generated by the tensor descriptor. We thus used the Tensor descriptor in all our examples, unless stated otherwise.

We also found that the rank of the sketched image is of only limited value in the context of very large databases: a good descriptor is not supposed to discriminate objects belonging to the *same* class. If many images similar to the sketch are contained in the database, it is unlikely that the reference image is found in the first few results. We demonstrate this effect for a query that has essentially not been successful according to our evaluation metric, i.e. the rank of the original image is very large. Figure 15 shows the 15 images with smallest distance to the sketch input shown (using the Tensor descriptor). While the original image is not among these 15 images (in fact the rank in this case is 5,474) most images in this set clearly show a resemblance to the input. This leads us to the conclusion that additional to the experimental evaluation a user study is also desirable, letting real users rate query results.

7. Conclusions & discussion

We have presented an interactive system for sketch-based image retrieval on a large database of over one million images and shown in a thorough experimental evaluation that the Tensor and HOG descriptors clearly outperform other existing approaches under the evaluation metrics applied in this paper.

Shape based retrieval. Note that while the resulting search engine retrieves images that resemble the *shapes* depicted in a query sketch, the results do not necessarily match the (probably intended) semantics of a query. This however would indeed be a desirable property of a retrieval system and we can envision systems that use higher level descriptors containing additional semantic information in addition to the low level image features currently used. Additionally, we see further potential for all descriptors by applying advanced edge extraction techniques, such that human sketching style gets mimicked more closely [50, 51]. Then, defining an improved version of e.g. the ARP descriptor would be straightforward, using the improved sketch lines instead of Canny lines.

Evaluation setup. While tracing images certainly is not a realistic way of generating input for an actual query, it helps eliminating the influence of human performance from the evaluation process and thus making the evaluation more objective. Assuming that a traced sketch closely resembles the shapes in the reference image, we can now expect that a good descriptor retrieves the reference image as one of the top results and objectively compare descriptors based on retrieval ranks. This also allows the evaluation to focus on how robust a descriptor is against affine transformations by querying with affine transformed versions of the sketch. Asking users to draw sketches from memory certainly is more natural in terms of query input. However, in the context of benchmarking, this would introduce a range of problems: instead of solely measuring descriptor performance, human drawing and memorization skills would influence the evaluation. We have therefore evaluated descriptors using a) traced feature lines, thus mostly eliminating human influence on retrieval performance and as a compromise using b) memory sketches while allowing subjects to have a second look at the reference image. We openly admit that sketching the feature lines seen in an image completely from memory would probably not work, in the sense of retrieving the reference image corresponding to that sketch in the top ranks. Note that this is no indication about

actual system performance, as the system might well return relevant images, just not the one we had initially presented. Indeed, all our examples as well the accompanying video show that the system works in the sense of providing matches that fit sketches that were drawn without any image as a template.

Retrieval performance. While the system and descriptors have shown very promising results, there are still many opportunities for further improvement: reducing the memory footprint of the image descriptors by using e.g. quantization or learning a compact binary code such that pairwise descriptor distances are preserved [52] would help using the system with even larger databases. In the same spirit, out-of-core search would allow running it on smaller machines with limited main memory. While for our database size (1.5 million images) the performance of a linear search was not a limitation, larger databases could certainly make use of faster searches, employing e.g. approximate nearest neighbor techniques [33, 32]. All descriptors covered in this paper are applicable to standard nearest neighbor search techniques and we have shown that by using a k-means tree [48, 33] the search can indeed be sped up by several orders of magnitude. It is clear that employing an approximate nearest neighbor algorithm always requires a trade-off between speed and precision (what fraction of the true nearest neighbors are returned?). While faster retrieval is certainly always desirable, it remains to be evaluated how much influence – if any – the approximate search has on retrieval quality. Due to its hierarchical structure, the k-means tree also could be a promising approach for facilitating retrieval on true web-scale collections: the sub-trees could be distributed over a large number of computing nodes and queried independently, merging individual query results on a master-node to form the final result.

Invariance to affine transformations. While all descriptors used in this paper can be efficiently computed and evaluated, they only provide limited invariance to similarity transformations (with the notable exception of the ARP descriptor which is invariant to rotations that are integer multiples of its binning angle). We believe that such deficits in a descriptor can be overcome by exploiting the variety provided by a *large* image database and support this claim with the results shown in Figures 13, 14 and 15. There are several interesting observations to be made from our evaluation: first, invariance to affine transformations is tightly coupled to the resolution of the local subdivision scheme. This can be clearly seen in the evaluation, descriptors using

coarser subdivision grids have a better chance of retrieving transformed versions of the original sketch. The GIST descriptor which only uses 4×4 local cells but stores a rich representation of the image content in each cell outperforms the sketch-descriptors regarding affine invariance. This suggests that a similar strategy could also be successful for sketch-based descriptors – we can see the HOG descriptor as a good candidate, storing a rich representation of e.g. gradient orientation and curvature in rather coarse cells. Note that orientation histograms can always be made completely rotation invariant by constructing them relative to a well-defined local frame instead of a fixed global frame [44]. Exploiting the shift-invariance property of the Fourier transform could be an interesting approach to make descriptors translation invariant, in a similar manner used to make the ARP descriptor rotation invariant. While wavelet transforms are unstable under translations of the signal, shiftable multiscale transforms [53] are specifically designed to yield translation invariant signal decompositions. Finally, a user study assessing how much invariance actually is desirable in a sketch-based retrieval system certainly would be extremely helpful.

An interesting final observation is the dependence of the system on the database content. Our database contains relatively few objects in a simple frontal view (i.e. the front side of a house, the side view of a car). However, most users tend to sketch objects from these points of view and will find that only few images match their sketch – simply because there are no objects in the database with silhouettes as sketched by the user.

[1] A. Smeulders, M. Worring, S. Santini, A. Gupta, R. Jain, Content-based image retrieval at the end of the early years, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (12) (2000) 1349–1380.

[2] R. Datta, D. Joshi, J. Li, J. Z. Wang, Image retrieval: Ideas, influences, and trends of the new age, ACM Computing Surveys 40 (2) (2008) 1–60.

[3] J. J. Koenderink, A. J. van Doorn, C. Christou, J. S. Lappin, Shape constancy in pictorial relief, in: Object Representation in Computer Vision II, 1996, p. 151.

[4] D. D. Hoffman, M. Singh, Salience of visual parts, Cognition 63 (1997) 29–78.

[5] F. Cole, A. Golovinskiy, A. Limpaecher, H. S. Barros, A. Finkelstein,

T. Funkhouser, S. Rusinkiewicz, Where do people draw lines?, ACM Transactions on Graphics 27 (3) (2008) 88:1–88:11.

[6] J. Elder, Are edges incomplete?, International Journal of Computer Vision 34 (2) (1999) 97–122.

[7] A. Orzan, A. Bousseau, H. Winnemöller, P. Barla, J. Thollot, D. Salesin, Diffusion curves: a vector representation for smooth-shaded images, in: ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008), Vol. 27, 2008, pp. 1–8.

[8] N. Chang, K. Fu, Query-by-pictorial-example, in: The IEEE Computer Society's Third International Computer Software and Applications Conference, 1979. Proceedings. COMPSAC 79, 1979, pp. 325–330.

[9] K. Hirata, T. Kato, Query by visual example - content based image retrieval, in: Proceedings of the 3rd International Conference on Extending Database Technology: Advances in Database Technology, Springer-Verlag London, UK, 1992, pp. 56–71.

[10] Y. Chan, Z. Lei, D. Lopresti, S. Kung, A feature-based approach for image retrieval by sketch, in: SPIE Storage and Retrieval for Image and Video Databases II, 1997.

[11] R. Kumar Rajendran, S.-F. Chang, Image retrieval with sketches and compositions, in: IEEE International Conference on Multimedia and Expo, Vol. 2, 2000, pp. 717–720.

[12] D. Lopresti, A. Tomkins, Temporal domain matching of hand-drawn pictorial queries, in: Proc. of the Seventh Conf. of The Intl. Graphonomics Society, 1995, pp. 98–99.

[13] A. Jain, A. Vailaya, Image retrieval using color and shape, Pattern Recognition 29 (8) (1996) 1233–1244.

[14] T. Hurtut, Y. Gousseau, F. Schmitt, F. Cheriet, Pictorial analysis of line-drawings, in: Porc. International Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging, 2008.

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

[16] S. Sclaroff, Deformable prototypes for encoding shape categories in image databases, Pattern Recognition 30 (4) (1997) 627–641.

[17] S. Matusiak, M. Daoudi, T. Blu, O. Avaro, Sketch-based images database retrieval, Lecture notes in computer science (1998) 185–191.

[18] F. Mokhtarian, A. Mackworth, A theory of multiscale, curvature-based shape representation for planar curves, IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (8) (1992) 789–805.

[19] H. H. S. Ip, A. K. Y. Cheng, W. Y. F. Wong, J. Feng, Affine-invariant sketch-based retrieval of images, in: Proceedings of the International Conference on Computer Graphics, IEEE Computer Society, 2001, p. 55.

[20] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, P. Yanker, Query by image and video content: The QBIC system, IEEE Computer 28 (9) (1995) 23–32.

[21] A. Oliva, A. Torralba, Modeling the shape of the scene: A holistic representation of the spatial envelope, International Journal of Computer Vision 42 (3) (2001) 145–175.

[22] C. Carson, S. Belongie, H. Greenspan, J. Malik, Blobworld: Image segmentation using expectation-maximization and its application to image querying, IEEE Trans. Pattern Analysis and Machine Intelligence 24 (8) (2002) 1026–1038.

[23] A. Torralba, R. Fergus, W. T. Freeman, 80 million tiny images: a large database for non-parametric object and scene recognition, IEEE Trans. Pattern Analysis and Machine Intelligence 30 (11) (2008) 1958–1970.

[24] C. E. Jacobs, A. Finkelstein, D. H. Salesin, Fast multiresolution image querying, in: Proceedings of SIGGRAPH 95, 1995, pp. 277–286.

[25] J. Z. Wang, G. Wiederhold, O. Firschein, S. X. Wei, Content-based image indexing and searching using daubechies' wavelets, Int. J. on Digital Libraries 1 (4) (1997) 311–328.

[26] A. Chalechale, G. Naghdy, A. Mertins, Sketch-based image matching using angular partitioning, IEEE Transactions on Systems, Man and Cybernetics, Part A 35 (1) (2005) 28–41.

[27] T. Sikora, The mpeg-7 visual standard for content description-an overview, IEEE Transactions on Circuits and Systems for Video Technology 11 (6) (2001) 696–702.

[28] A. Yamada, M. Pickering, S. Jeannin, L. Cieplinski, J.-R. Ohm, M. Editors, Mpeg-7 visual part of experimentation model version 8.0, ISO/IEC JTC1/SC29/WG11/N3673 (2000) 1–82.

[29] B. Manjunath, P. Salembier, T. Sikora, Introduction to MPEG-7: multimedia content description interface, John Wiley & Sons Inc, 2002.

[30] A. Chalechale, Content-based retrieval from image databases using sketched queries, Ph.D. thesis, University of Wollongong (2005).

[31] C. Won, D. Park, S. Park, Efficient use of mpeg-7 edge histogram descriptor, Etri Journal 24 (1) (2002) 23–30.

[32] A. Andoni, P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, Communications of the ACM 51 (1) (2008) 117–122.

[33] M. Muja, D. G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration, in: International Conference on Computer Vision Theory and Applications, 2009, pp. 331–340.

[34] M. Eitz, K. Hildebrand, T. Boubekeur, M. Alexa, A descriptor for large scale image retrieval based on sketched feature lines, in: Symposium on Sketch-Based Interfaces and Modeling, 2009, pp. 29–36.

[35] H. Knutsson, Representing local structure using tensors, in: The 6th Scandinavian Conference on Image Analysis, Oulu, Finland, 1989, pp. 244–251.

[36] J. E. Kyprianidis, J. Döllner, Image abstraction by structure adaptive filtering, in: Proc. EG UK Theory and Practice of Computer Graphics, 2008, pp. 51–58.

[37] D. Lowe, Object recognition from local scale-invariant features, in: International Conference on Computer Vision, Vol. 2, 1999, pp. 1150–1157.

[38] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: International Conference on Computer Vision & Pattern Recognition, Vol. 2, 2005, pp. 886–893.

[39] P. Shilane, P. Min, M. Kazhdan, T. Funkhouser, The Princeton shape benchmark, in: Shape Modeling International, Vol. 105, 2004, p. 179.

[40] M. Eitz, K. Hildebrand, T. Boubekeur, M. Alexa, PhotoSketch: a sketch based image query and compositing system, in: SIGGRAPH 2009: Talks, 2009.

[41] T. Chen, C. M. Ming, P. Tan, A. Shamir, S.-M. Hu, Sketch2photo: Internet image montage, ACM Transactions on Graphics, (Proceedings SIGGRAPH ASIA 2009) 28 (5).

[42] M. Swain, D. Ballard, Color indexing, International Journal of Computer Vision 7 (1) (1991) 11–32.

[43] J. Canny, A computational approach to edge detection, IEEE Transactions on Pattern Analysis and Machine Intelligence (1986) 679–698.

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

[45] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.

[46] S. Di Zenzo, A note on the gradient of a multi-image, Computer Vision, Graphics, and Image Processing 33 (1) (1986) 116–125.

[47] K. S. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is "nearest neighbor" meaningful?, in: International Conference on Database Theory, 1999, pp. 217–235.

[48] K. Fukunaga, P. M. Narendra, A branch and bound algorithms for computing k-nearest neighbors, IEEE Trans. Computers 24 (7) (1975) 750–753.

[49] S. Lloyd, Least squares quantization in pcm, IEEE Transactions on Information Theory 28 (2) (1982) 129–137.

[50] D. Martin, C. Fowlkes, J. Malik, Learning to detect natural image boundaries using local brightness, color, and texture cues, IEEE Transactions on Pattern Analysis and Machine Intelligence (2004) 530–549.

[51] H. Kang, S. Lee, C. Chui, Coherent line drawing, in: Proceedings of the 5th international symposium on Non-photorealistic animation and rendering, 2007, pp. 43–50.

[52] A. Torralba, R. Fergus, Y. Weiss, Small codes and large image databases for recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.

[53] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, D. J. Heege, Shiftable multiscale transforms, IEEE Transactions on Information Theory 38 (2) (1992) 587–607.