A Feature-Based Approach for Image Retrieval by Sketch

Yin Chan† Zhibin Lei‡ Daniel Lopresti§ S.Y. Kung†

†Department of Electrical Engineering
Princeton University
Princeton, NJ 08544

‡Division of Engineering
Brown University
Providence, RI 02912

§Matsushita Information Technology Laboratory
Panasonic Technologies, Inc.
Two Research Way, Princeton, NJ 08540

1 Introduction

Thanks to technological advancements in various areas, an ever-growing amount of digital images are becoming accessible to the general user. To take advantage of the rich information contents of these data, researchers have proposed the content-based indexing and retrieval paradigm. Various features such as colors, textures, shapes etc. have been used to characterize the contents of images to enable indexing and retrieval [1, 2, 3, 4, 5, 6, 7]. Figure 1 shows the block diagram of a widely used content-based image indexing and retrieval model that we adopt for our prototype. In the database population phase, features such as prominent edges in our case, are extracted. These features, together with the corresponding image data, are saved. At query time, the query item(image) goes through the same feature extraction step. The search engine then calculates the similarities between the query item and the database entries using the features extracted. The most similar n database entries, for some n, are then shown to the user. In this paper,

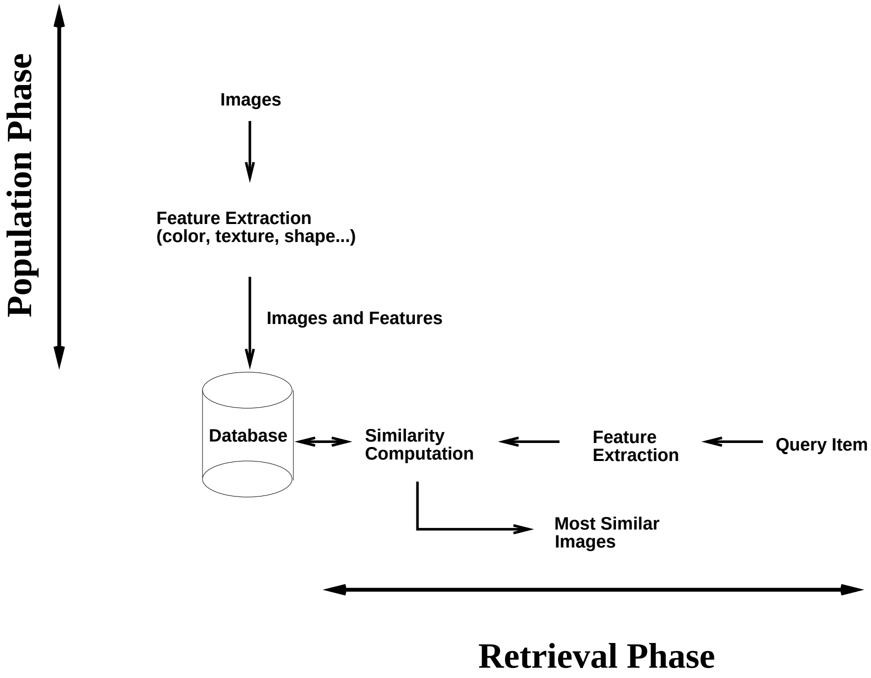


Figure 1: Block diagram of a content-based image indexing and retrieval system.

we propose an approach to automatically extract prominent edges from images as their representative features. Similarities between images are computed using individual geometric attributes of these edges such as length, angle of inclination of straight lines, and curvature etc. and relative spatial relationships between edges. We believe that in realistic query sessions, the user usually does not draw the query sketch in close correspondence to the intended images. Therefore, the emphasis of our feature-based approach is to provide robustness towards distortion in query sketches. Specifically we would like to investigate a method with graceful retrieval performance degradation as correspondence between a query sketch and its intended database image(s) decreases. We will also describe a prototype, built upon the Java applet technology, that allows the user to draw sketches and pose queries over the WWW. Preliminary results using a database of 137 color photo and computer generated images show that our feature-based approach performs better than a pixel-based method that has been adopted by various image retrieval systems [1, 2, 3].

2 Related Work

Lopresti and Tomkins have considered the problem of matching hand-drawn pictorial queries against an existing database of sketches [8]. They describe a hierarchical approach designed to exploit the temporal nature of electronic ink. The input is first segmented into strokes and vectors of descriptive features (*e.g.*, stroke length, total angle traversed) are extracted. These feature vectors are then mapped into a small set of basic types using vector quantization (VQ). The effect is to represent handwriting as a string over an alphabet of stroke types. A new string block editing algorithm [9] is then used to perform comparisons between the query and the database. In a follow-up paper, Lopresti, Tomkins, and Zhou present a more detailed analysis of the sketch-matching problem, including the results of an experiment involving five subjects, each of whom created a database of 25 sketches and 25 queries to match against it [10].

In [4], Del Bimbo *et al.* introduce an image retrieval algorithm by elastic matching of shapes and image patterns. Rectangular areas enclosing objects of interest are selected manually and subsequently intensity gradient maps for these areas are obtained. Finally, a hill-climbing optimization algorithm is applied to these gradient maps to identify the boundaries of the objects. A criterion for the selection of such rectangular areas is that the objects are quite distinct from the background.

In the ART MUSEUM project [1], Hirata *et. al.* propose a method to compute similarities of a user sketch with images in the database. In their work, color oil painting images are passed through a series of steps including size regularization, edge detection, thinning, and shrinking. The resulting abstract images are then compared with the user sketch by a template matching method. This method has been adopted by a number of content-based image/video retrieval research projects such as the IBM QBIC [2] and the work by Zhang *et al.* [3].

The subject of our present work follows naturally from the work by Lopresti *et al.*, but differs in a fundamental way in that the target database is composed of photographic images, not hand-drawn sketches. As a result, an entirely new approach is required. The database images targeted by both Del Bimbo *et al.* and Hirata *et. al.* are similar to ours. The work by Del Bimbo *et al.* differs from ours in that their semi-automatic system first relies on a human to select objects of interest and then extracts boundaries of these objects automatically. Our approach is automatic but does not seek to extract object boundaries. The work by Hirata *et. al.* is closest to ours with one major difference that a pixel-based template matching method is used to calculate the similarities between edge maps of images while our similarity computation method uses edge segments.

3 A Feature-Based Approach for Image Retrieval by Sketch

In this paper, we introduce a feature-based approach for image retrieval by comparison of salient edge features, i.e. significant edges such as long straight line and curve segments, in images. Hereafter, the term "edge segment" is used to mean either a *line* or a *curve* segment. The database images that we are targeting are static photo images, clip art, and computer generated graphics. There are a few assumptions that we rely on. First of all, there are significant edges that characterize the objects in the images. Second of all, the user has seen the target image before and wants to retrieve it and hopefully similar ones from the database. Lastly, the user can reproduce the edges with a fair amount of accuracy in terms of absolute and relative positions as well as individual attributes such as length, curvature etc.

In Section 3.1, we will illustrate the feature extraction step of our approach. Similarity computation using these features will be explained in Section 3.2. Implementation of our prototype that is based on a client/server model and utilizes the Java applet technology to enable user query from multiple platforms using the WWW will be discussed in Section 3.3. Section 3.4 will present some preliminary experimental results of our current implementation. This will be followed by a discussion of the promise as well as the limitations of our approach. Finally, we conclude this paper with a look at areas for future work.

3.1 Feature Extraction

In [11], Gray evaluated the pixel-based image retrieval technique from [1]. It was found that the pixel-based template matching method did not perform well with images having many extraneous edges after edge detection. Moreover, it was also observed that good retrieval results could only be expected when the user *remembered* the intended image well and *drew* the query sketch in close correspondence to the prominent edges of the intended database image. As will be shown in Section 3.4, our experiments using 5 subjects suggested that even if the intended image was shown to the user at the time of sketching the query, he/she tended to ignore some "details." In these scenarios, extraneous edges only serve as noise to confuse any similarity calculating algorithm. Similar to Gray's observation, our experiments

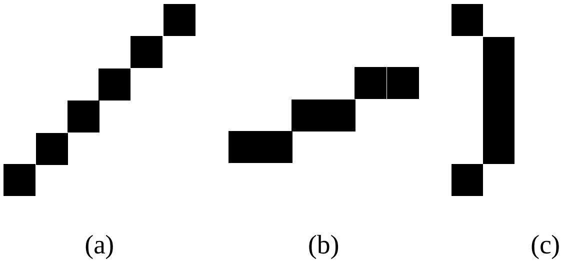


Figure 2: Pixel-based method decides that similarity between patterns (a) & (b) is the same as that between (a) and (c). They are magnified subparts of 3 segments, i.e. (a) line at 45°, (b) line at 22°, and (c) "tip" of conic curve $x + y^2 = 0$.

also gave evidence that even if the intended image had only a few prominent edges, the user did not draw the edges in such a way that the pixels in the query sketch aligned well with those of the intended image. If this happens, a pixel-based method could give counter-intuitive results. Figure 2 depicts one such situation. The three patterns are magnified subparts of three edge segments with each black block representing an enlarged pixel. Specifically, pattern (a) is part of a line segment inclined at 45° while pattern (b) represents one roughly at 22°. On the other hand, pattern (c) depicts the "tip" of a conic curve of the form $x + y^2 = 0$. While a human would decide that (a) and (b) are more similar, a pixel-based template matching method would decide that the similarity between (a) and (b) is the same as that between (a) and (c) because these two pairs have the same number of overlapping pixels.

In order to improve retrieval performance, given the observation that the user may not always sketch with good accuracy and the database images may be quite noisy, we reason that a feature-based method, which computes the similarity between two images based on salient edge segments which can be long straight line or curve segments, is desirable. One advantage of this approach is that when edge pixels are grouped into edge segments, those segments that are deemed insignificant, namely, those which are relatively short when compared with others, can be ignored in similarity computation. These short edge segments either may be noisy patterns or tend to be ignored by the user when sketching a query, as suggested by our experiments.

A second advantage is that even if the user draws the edges somewhat differently from what are intended, say slightly rotated or translated, the individual attributes of these edges such as length, angle of inclination, and curvature etc. do not change substantially. More importantly, we can use an analytic function to calculate these discrepancies

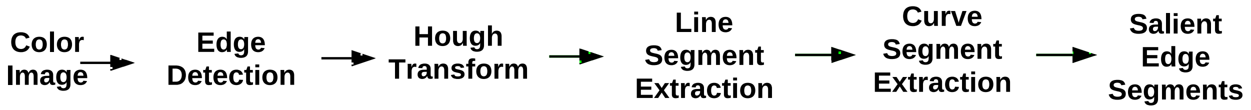


Figure 3: Processing stages for prominent edge extraction.

as part of the similarity metric. Thirdly, in contrast to unorganized edge pixels, edge segments explicitly convey structural information of the objects inside images, making it easier to compute similarities between images. Fourthly, a reduced number of features used to characterize the images facilitates faster similarity computation.

Specifically we would like our feature-base similarity computation method to provide robustness towards both reasonable distortions in query sketches and moderate noise in images. Notice that we do not intend to device an algorithm that could handle arbitrary distortions.

The goal of our feature extraction step is quite different from that required by an object recognition application. We only hope to extract prominent edge segments and make no attempt to group edges and extract contours according to different objects. The reason is that our target database images can contain overlapping objects that in turn could be embedded in slightly cluttered background. An object recognition approach that seeks to automatically extract boundaries of objects may prove difficult.

The steps to extract prominent edge segments in color images that we propose are shown in Figure 3. The edge detection step first identifies perceptually significant edge pixels. Unrelated pixels in the edge map thus obtained convey little information regarding the edges in the original image unless they are organized. Therefore the Hough transform [12] is applied to the edge map to relate edge pixels that lie on the same straight line. While the Hough transform tells us what pixels lie on which straight line, these pixels may belong to different line segments all of which lie on the same line. Hence it is the objective of the line segment extraction step to subsequently identify these line segments. Furthermore, line segments which are too short and thus may be noise or ignored by the user are eliminated. Significant curves in the original image appear as adjacent but broken line segments after this step. It is the goal of the following curve segment extraction phase to join them together. In subsequently sections, these steps will be explained in details.

p1	**p2**	**p3**
p4	**p5**	**p6**
p7	**p8**	**p9**

Figure 4: A (3×3) image region where a pseudo-Sobel operator computes color gradient at pixel $p5$.

3.1.1 Edge detection

This step is similar to what is employed in [1]. When doing edge detection, partial derivatives at each pixel are used to compute the color-change (or intensity-change in the case of a grayscale image). However, derivatives have the tendency to enhance noise in images. Therefore we use a simple extension of the Sobel operators [12] to 3-D color space for our purpose. The Sobel operators have both differencing and smoothing effects and the latter is useful to reduce noise in resulting edge maps. Figure 4 shows a (3×3) region in an image where the color gradient at pixel $p5$ is calculated. The following is the extension of the Sobel operators that we use

$$\begin{aligned} \frac{\Delta I_{p5}}{\Delta x_i} &= (p7_i + 2p8_i + p9_i) - (p1_i + 2p2_i + p3_i) \\ \frac{\Delta I_{p5}}{\Delta y_i} &= (p3_i + 2p6_i + p9_i) - (p1_i + 2p4_i + p7_i) \end{aligned}$$

where px_i is the value of pixel px in color channel $i \in \{R, G, B\}$. The magnitude of the gradient vector at pixel $p5$ is then approximated as

$$\nabla I_{p5} = \sqrt{\sum_{i \in \{R,G,B\}} {\frac{\Delta I_{p5}}{\Delta x_i}}^2 + {\frac{\Delta I_{p5}}{\Delta y_i}}^2} \tag{1}$$

which gives the rate of color change at pixel $p5$. The edge map $\mathcal{E}$ of the original color image is a gray level image formed by ∇I which captures the rate of color change, i.e.

$$\mathcal{E} = \{e(i,j) \mid e(i,j) = \nabla I_{p_{ij}}, 1 \leq i \leq H, 1 \leq j \leq W\}$$

where p_{ij} is the pixel at the location (i, j) in the original image with size $H \times W$.

After computing the edge map $\mathcal{E}$, we compute a global threshold t_g using the mean and deviation of all the edge pixels of $\mathcal{E}$. Edge intensities that are less than t_g are eliminated to form a globally thresholded edge map

$\mathcal{E}_g = \{e_g(i,j)\}$

$$
\begin{aligned}
e_g(i,j) &= \begin{cases} 0 & e(i,j) < t_g \\ e(i,j) & otherwise \end{cases} \\
with & \\
\mu_g &= \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W} e(i,j) \\
\sigma_g &= \sqrt{\frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}(e(i,j)-\mu_g)^2} \\
t_g &= \mu_g + \frac{\sigma_g}{2}
\end{aligned}
$$

We further enhance the edge map by considering only close neighborhood information. A local threshold $t_{l(i,j)}$ is computed using the mean and deviation of pixels inside a small local window (size = 5×5 in our implementation) with $e_g(i,j)$ as the center pixel. We eliminate any pixel whose value is less than $t_{l(i,j)}$ to form the locally thresholded edge map $\mathcal{E}_l = \{e_l(i,j)\}$

$$
\begin{aligned}
e_l(i,j) &= \begin{cases} 0 & e_g(i,j) < t_{l(i,j)} \\ 1 & otherwise \end{cases} \\
with & \\
t_{l(i,j)} &= \mu_{l(i,j)} + \frac{\sigma_{l(i,j)}}{2}
\end{aligned}
$$

where $\mu_{l(i,j)}$ and $\sigma_{l(i,j)}$ are local mean and deviation. Notice that the final edge map $\mathcal{E}_l$ is a binary image.

3.1.2 Hough transform

Unorganized pixels in the edge map obtained from the previous phase convey little information regarding the structures of objects in the original image unless they are organized. Applying the Hough transform to the binary edge map, we want to identify subsets of pixels that lie on the same lines. We use the polar representation of a line i.e. $x \ \cos\ \theta \ + \ y \ \cos\ \theta \ = \ \rho$, when performing the Hough transform. For details of this transformation, please refer to [12].

3.1.3 Line segment extraction

While the Hough transform tells us what pixels lie on which straight line, these pixels may not be connected, i.e. they may belong to different line segments lying on that straight line. The objective of this line segment extraction step is two-fold. First, we want to examine the continuity of the pixels to identify subsets of pixels such that each subset contains precisely those that belong to one line segment. Second, we would like to eliminate very short line segments in an attempt to reduce extraneous edges and noise in the edge map.

Given the pixels lying on a straight line, we compare the distance between two consecutive points with a line segment separation threshold t_{ss}. Inter-pixel distance between pixels (x_i, y_i) and (x_j, y_j) that is larger than t_{ss} is taken as a discontinuity of pixels along the line between these two points which are subsequently assigned the end points of different segments. Using a line segment length threshold, t_{sl}, we eliminate any line segments that are shorter than t_{sl}.

Finally, we sort the line segments in decreasing order of their lengths and use a greedy algorithm to further eliminate line segments that are close to another longer segment with very similar orientation. The rationale behind this step is that the user tends to ignore a shorter edge in the presence of a close but longer one when their orientations are similar. This assumption seems justified by our experiments (cf. Section 3.4) with different subjects.

3.1.4 Curve segment extraction

For those prominent edge structures that are not straight lines, they will appear as many short line segments after the previous stages. The objective of the curve segment extraction step is to join these line segments to recover the prominent feature structures. A statistical grouping method is used to guide the joining process.

statistical grouping method We use a line segment's tendency or preference of joining with its neighbors as the main factor in the process of join testing. Under the assumption that the points along a set of line segments are actually generated according to some feature model (called curvelet) plus white noise, the probability that this set of line segments comes from the same curvelet M (hence needs to be grouped together) can be written as a function of this curvelet feature M and the noise model [13]. This probability value measures how well a set of data points is represented by a given curvelet structure M. It is the geometric structure of the data set itself, and not dependent on the relative location of or point orderings along the given line segments.

The goal here is to generate a set of feature structures (curvelets) that is the best or most probable representation of the original edge map. The general setup is as follows. Given a curvelet model M_j and a data set $S_k = \{p_i^k\}_{i=1}^N$, if we assume that the points in S_k are generated from the model independently, the probability that M_j generates S_k is [13, 14]:

$$P(S_k|M_j) = \frac{1}{\theta(S_k,\sigma)} e^{-\Sigma_i(dist^2(p_i^k, M_j))}$$

here, σ is the standard deviation of noise model, $dist(p_i^k, M)$ is the distance from a point p_i^k to the curvelet model M_j, and $\theta(S_k,\sigma)$ is a scale factor. The net result of the distance errors from all the data points measures the fitness of the model for the dataset.

If we further assume that different curvelets are independent, we can obtain the overall probability that edge map $S = \{S_k\}$ is generated by a set of curvelets $M = \{M_j\}$:

$$P(S|M) = P(\bigcup_k S_k | \bigcup_j M_j) = \Pi_k P(S_k|M_{j_k}) = \frac{1}{\theta(S,\sigma)} e^{-\Sigma_k \Sigma_i(dist^2(p_i^k, M_{j_k}))}$$

here, M_{j_k} is the curvelet model for the dataset S_k. $\Sigma_k \Sigma_i(dist^2(p^k{}_i, M_{j_k}))$ is the total distance error metric of the dataset under the given curvelet representation $\{M_j\}$. Our goal here is to find a best set of curvelet models that has the smallest representation error, or equivalently, is the most probable set of curvelet models that generates the given edge map.

This is a global optimization problem. To reduce the complexity of computation, we use a greedy searching algorithm and adopt a neighborhood criterion. We only search a line segment's neighboring segments for possible merging. One line segment is another's neighbor if they are close in both location and tangent direction. We start the search from the most distinctive line segment L (the longest one). If a neighboring line segment is joined to it, L's neighborhood structure is updated and the searching continues on the new neighborhood structure. When no more neighboring line segments can be joined together, L becomes a final curvelet feature and it is saved to the disk. A new search begins among those remaining line segments. This process stops when no more free line segments are available. We can run another iteration on the resulting curvelet set to improve the quality of joining.

Figure 5 shows a painting of a plant, followed by a plot of its line segments (after line segment extracting) and the results of the two joining iterations. To illustrate the joining effect, adjacent line segments or curvelets are slightly adjusted and drawn with different levels of thickness. We see that after each iteration, the total number of curvelets is reduced by about half. The result after two iterations is very satisfactory. Figure 6 is an image of airplane. Although there are many small details in the airplane body (more than 2000 small line segments to start with), the total number of curvelets after linking and grouping is still very small.

Curvelet structure models We use implicit polynomials (IP) as the curvelet feature models. There are many reasons we choose IPs as the models. They are simple and concise curve structure representations. We can express the approximate distance error of an IP model as the function of its coefficients explicitly. IP model is also a natural generalization of the line feature structure. Another important property of IP models is their algebraic invariants, which capture the shape invariant structures and are independent of the underlying coordinate system changes. It can be very useful in many feature based query applications.

An IP function of degree n is a polynomial function $f(x,y) = \sum_{i,j\geq 0, i+j\leq n} a_{ij}x^iy^j = 0$. Here a_{ij}'s are the coefficients. An IP function $f(x,y) = 0$ is a representation of a shape (object) $S = \{(x_i, y_i)|i = 1, ..., M\}$ if every point of S is on the IP model zero set $Z(f) = \{(x,y)|f(x,y) = 0\}$ [16]. The simplest IP model is the line structure whose degree is one. Circles (or conics in general) are IPs of degree two. It is customary to use IP fitting procedures to obtain IP representations for a given data set.

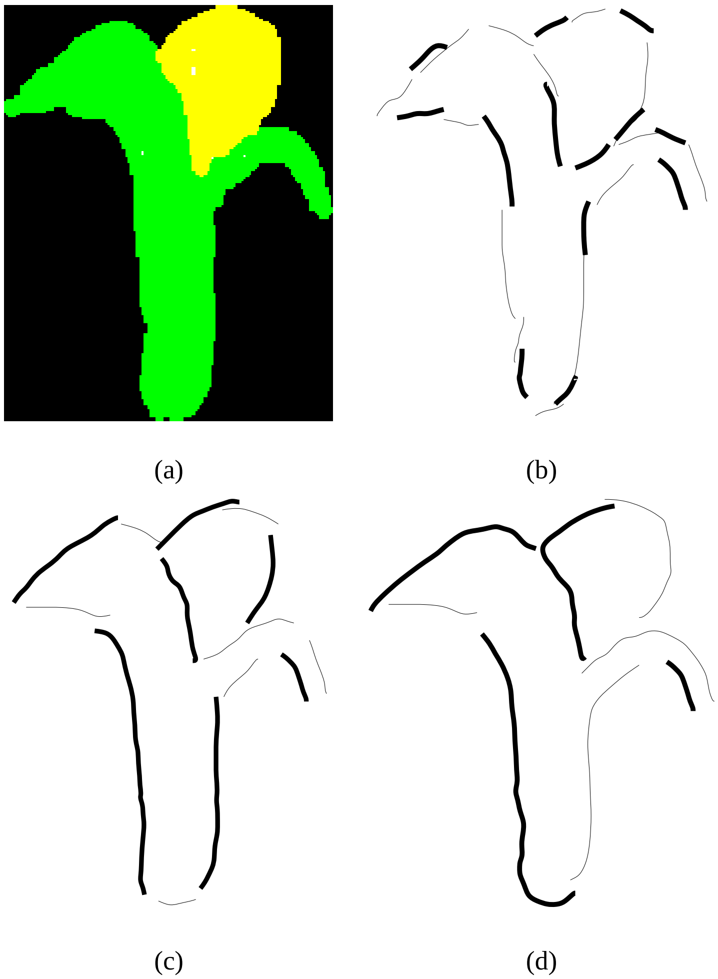
The IP fitting problem is usually set up as follows. Given a data set $\mathbf{Z} = \{z_i = (x_i, y_i)|i = 1, .., N\}$, find the best IP model of certain degree that minimizes the average squared distance from the data points to the zero set of the polynomial $Z(f)$. An iterative process is needed to solve for the exact distance from a point to the zero set of an implicit polynomial because there is no explicit expression for this distance. A commonly used first order distance approximation is [15, 13]:

$$d(z_i, z(f)) = \frac{|\ f(z_i)\ |}{||\ \nabla f(z_i)\ ||} \tag{2}$$

Once the IP model is given, this can be calculated directly for each point in a data set. The average squared distance is:

$$\bar{d}^2 = \frac{1}{N}\sum_{i=1}^{N} d^2(z_i, Z(f)) = \frac{1}{N}\sum_{i=1}^{N}\frac{|\ f(z_i)\ |^2}{||\ \nabla f(z_i)\ ||^2} \tag{3}$$

Equation 3 is the error metric we use for the IP curvelet models.



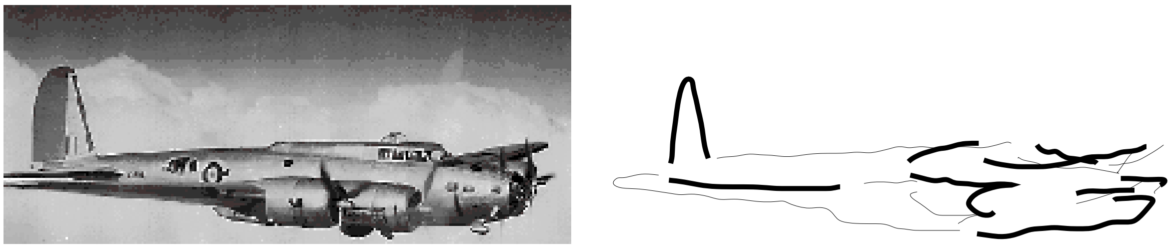
(a)

(b)

(c)

(d)

Figure 5: (a) original plant image. (b) result of line segment extraction. (c) curvelets after first join iteration. (d) curvelets after second join iteration.



(a) (b)

Figure 6: (a) original b17 aircraft image. (b) curvelets after two join iterations.

3.2 Similarity Computation

Assuming that we are given two sets of features, i.e. curvelets, one from a database image and the other from the query item, we calculate the similarity between these images using curvelets modeled by first degree IPs in our current implementation. A similarity computation algorithm using higher degree curvelets are being developed. Nevertheless, the experimental results in Section 3.4 already show the promise of our feature-based approach.

The similarity between two images is computed by calculating the similarities between pairs of curvelets from the two images. Given two sets of curvelets

$$C_Q = \{c_q(x,y)\} \quad \& \quad C_{DB} = \{c_{db}(x,y)\} \cup \{null\}$$

where $\{c_q(x,y)\}$ and $\{c_{db}(x,y)\}$ are the sets of curvelets corresponding to the query and database images (the reason for the inclusion of $\{null\}$ in C_{DB} will be apparent later), and $c(x,y)$ is a curvelet with centroid at (x,y). We first calculate the dissimilarity $\overline{\mathcal{S}}(C_Q, C_{DB})$ is calculated as

$$\overline{\mathcal{S}} = \min_{(M,N)} \sum_{\Delta x=-M}^{M} \sum_{\Delta y=-N}^{N} \sum_{c_q(x,y)\in C_Q} \min_{(m,n)} \sum_{\delta x=-m}^{m} \sum_{\delta y=-n}^{n} \Phi[c_q(x,y), \widetilde{c_{db}}(x+\Delta x+\delta x, y+\Delta y+\delta y), (\delta x, \delta y)]$$

where $\Phi[c_q, c_{db}, (\delta x, \delta y)]$ is a mapping

$$\Phi \,:\, C_Q \times C_{DB} \times \mathcal{R}^2 \rightarrow \mathcal{R}$$

and

$$\widetilde{c_{db}}(\tilde{x},\tilde{y}) = \begin{cases} c_{db}(\tilde{x},\tilde{y}) & if \quad c_{db}(\tilde{x},\tilde{y}) \in \{c_{db}(x,y)\} \\ null & otherwise \end{cases}$$

We calculate the dissimilarity between two first degree curvelets c_q and $\widetilde{c_{db}}$ using their lengths, angles of inclination, δx, and δy as follows

$$\Phi[c_q, c_{db}, (\delta x, \delta y)] = \begin{cases} del(c_q) & if\ c_{db} = null \\ \begin{bmatrix} & [L(c_q) - L(c_{db})]/max[L(c_q), L(c_{db})] \times \alpha \\ + & \min[|A(c_q) - A(c_{db})|\,, \pi - |A(c_q) - A(c_{db})|] \times \beta \\ + & (\delta x/W_{img}\ +\ \delta y/H_{img}) \times \gamma \end{bmatrix} & otherwise \end{cases}$$

where α, β, and γ are constants, $L(c)$ is the length of c, $A(c)$ is the angle of inclination of c from $-\frac{1}{\pi}$ to $\frac{1}{\pi}$, W_{img} and H_{img} the width and height of the database image respectively, and $del(c)$ the cost of deleting c which is a linear function of its length.

Basically, we treat the set of query curvelets as a whole entity where the relative spatial relationships[1] among them are fixed. Then this set is shifted within a $(2M + 1) \times (2N + 1)$ window in the database image. This allows for the case where the user sketch is a translated version of the intended one. For each query curvelet $c_q(x, y)$, this global shift moves it to the coordinates $(x + \Delta x, y + \Delta y)$ in the database image. c_q is then allowed to move within a small $(2m + 1) \times (2n + 1)$ window. The reason is that even if the location of the whole set of query curvelets is close to the target image curvelets, centroids of individual curvelets may deviate. The local shift of individual cruvelets allows for this type of imprecision. This movement gives yet a pair of new coordinates $(x + \Delta x + \delta x, y + \Delta y + \delta y)$ in the database image.

If there is no curvelet in the database image with centroid at $(x + \Delta x + \delta x, y + \Delta y + \delta y)$, the matching cost is assigned to the cost of deleting the query curvelet $del(c_q)$ which is a linear function of its length $L(c_q)$. Otherwise, the cost is a function of the lengths and angles of the two curvelets as well as the local shift vector $(\delta x, \delta y)$. As can be seen from our similarity function, we currently penalize local shift but not global one. To account for other types of distortion of the user sketch, we could also replace the shift operators by affine operators that will take into account rotation, scaling, and sheering. However, we have not investigated the performance of such an implementation.

Notice that in calculating $\overline{S}$, we accumulate, for each query curvelet c_q, the dissimilarity between c_q and some $\widetilde{c_q} \in C_{DB}$, hence $\{\widetilde{c_q}\} \subseteq C_{DB}$. To better reflect the similarity between the query and database images, we penalize all curvelets in the database image that do not have a matching query curvelet, i.e. the set $C_{DB} \setminus \{\widetilde{c_q}\}$. Therefore the

[1] relative centroid displacements between first degree curvelets in our current implementation

similarity between the two images is

$$\mathcal{S} = \mathcal{K} - \left[\overline{\mathcal{S}} + \xi \sum_{c \in C_{DB} \setminus \{\widetilde{c_q}\}} del(c) \right]$$

where $\mathcal{K}$ is an arbitrarily large constant to make $\mathcal{S}$ a positive quantity. The reason that the sum of the costs of deleting these non-matching database curvelets be scaled by constant ξ, where $0 < \xi < 1$, is because we believe that deleting a database curvelet should incur less penalty than deleting a query one. Having a query curvelet matching no corresponding database curvelet means that the user drew one but it does not appear in the database image. On the other hand, the absence of a query curvelet in correspondence to a database one may only mean that the user forgets or does not care about this curvelet(edge). We would like to allow partial sketches with little penalty so that query on subparts of an image is possible. Throughout our experiments, we observed that our subjects tended to ignore some less important edges and this suggested that placing less penalty on deleting database curvelets may be an appropriate strategy.

3.3 Prototype Implementation

3.3.1 WWW-based user interface

In order to provide an environment where the user can pose queries regardless of location and platform, we elect to implement a WWW-based retrieval by sketch prototype as shown in Figure 7. On the client side, the user can download an applet which captures the sketch. It then sends the data to the server via the *HTTP* **POST** method. On the server side, these sketch data are subsequently passed to a CGI program which is the search engine. Query results are then saved as a HTML page on the server machine and the *URL* sent back to the applet that has been busy-waiting. A call by the applet upon receiving the result *URL* to the browser that contains it to load the result HTML page completes a query cycle. Figure 8 depicts a query session and Figure 9 shows the corresponding query result HTML page that displays the 10 most similar images in our database. The use of an HTML page as a means to present retrieval results free us from complicated window display manipulation.

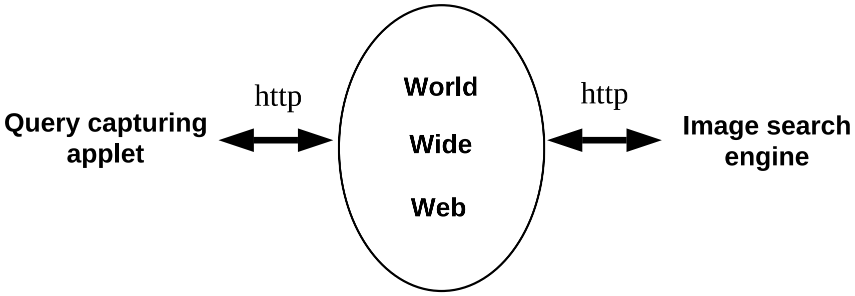


Figure 7: WWW-based image retrieval by sketch system.

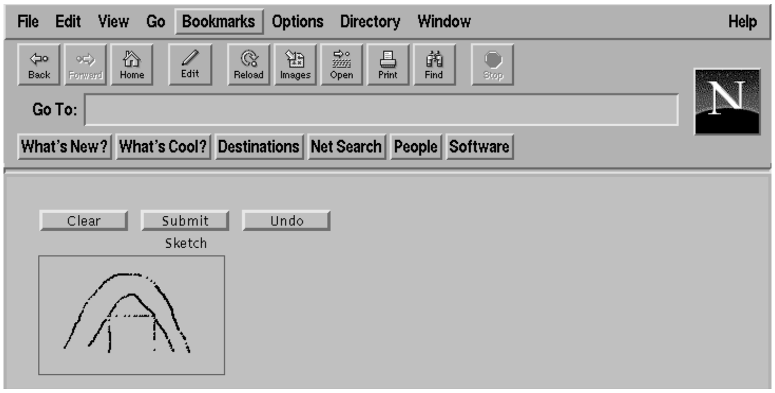
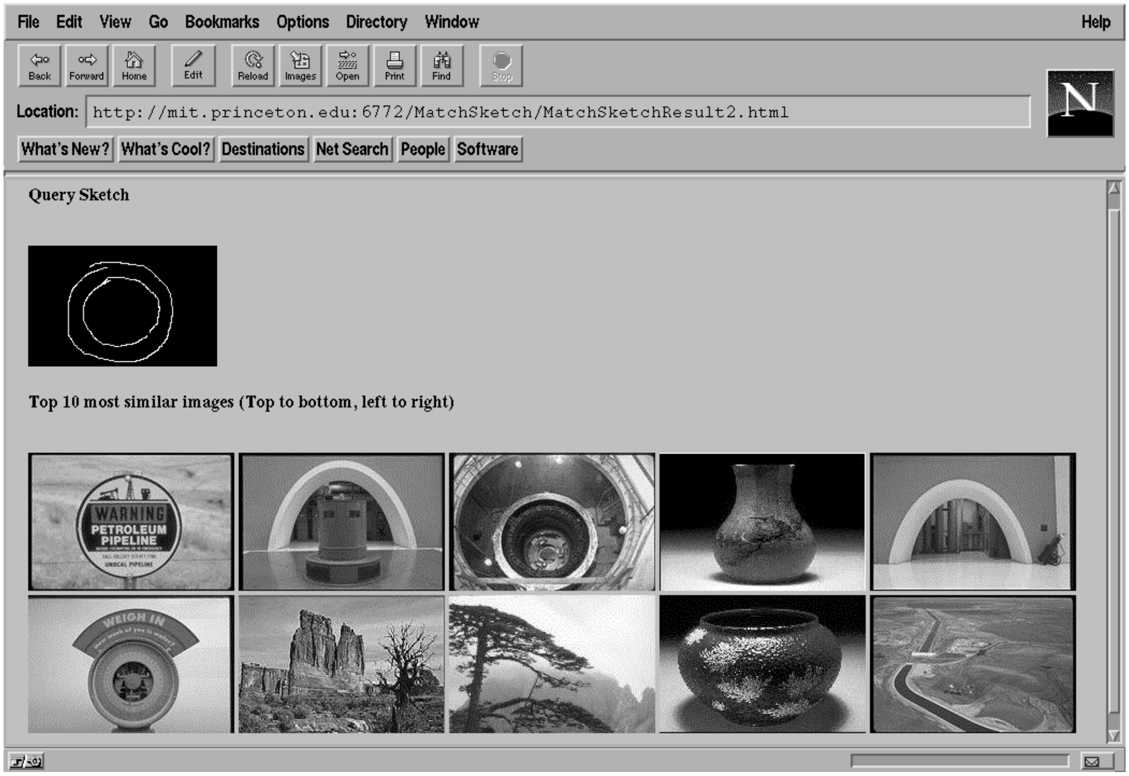


Figure 8: A query session where the user downloads the sketch capturing applet from the server and composes the sketch query.



Location: http://mit.princeton.edu:6772/MatchSketch/MatchSketchResult2.html

Query Sketch

Top 10 most similar images (Top to bottom, left to right)

Figure 9: A sketch result HTML page that displays the 10 images from the database having prominent edges most similar to those in the query sketch. Similarity decreases left to right, top to bottom.

3.4 Experimentation

To test the effectiveness of our feature-based sketch matching approach, we implemented the pixel-based matching method in [1] for comparison purposes. The current database consists of 137 color photo and computer generated images. All of these images have prominent edges. However, in order to investigate the ability of our feature-based approach in dealing with images whose edge maps are moderately noisy, there are a small subset of such images inside the database.

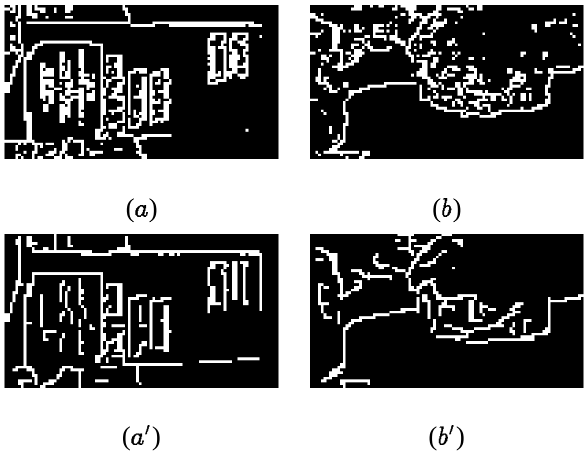
3.4.1 Pixel-based matching method

We adopt the matching method introduced in [1] with two minor modifications in accordance to the characteristics of our database. Firstly, since the database that we are using are photo and computer generated images while theirs were oil paintings, the edges in our image collection are not as sharp in general and the resulting edge maps can be fairly noisy. We therefore, after edge detection, pass the edge maps through our line segment extraction step with finer Hough transform resolution and smaller line segment length threshold t_{sl} in an attempt to reduce noise. As can be seen in Figure 10 which compares two edge images with one processed by the line extraction step and the other one not, prominent edges are well preserved while much noise is eliminated.

Secondly, we reason that for a 1-to-0 pixel mismatch, it means a user draws something but the database image does not have the corresponding pixel while a 0-to-1 mismatch may mean that the user does not remember or does not care. Therefore we believe that the penalty for the first type should be larger than that of the second. In our experiments, we used -30 and -3 as the respective penalties. In [1], the ratio between a 1-to-1 pixel match to a mismatch is 10 : (-3) and a 0-to-0 match carries $1/10$ weight of a 1-to-1 match. Accordingly, in our experiments, we used 100 and 10 as the 1-to-1 and 0-to-0 match scores respectively.

3.4.2 Empirical results

Robustness of similarity computation towards query sketch distortions is the focus of our work. There are various possible factors that affect the user's ability to draw the query sketch in close correspondence to what is intended. These could include how well his/her memory works, how well one can draw, how easy it is for the user to maneuver the drawing hardware etc. Throughout the course of our investigation, mice were used as the drawing tool because of



(a) (b)

(a') (b')

Figure 10: (a) and (b) are edge maps of two images after edge detection, thresholding, thinning, and shrinking while (a') and (b') are edge maps with extraneous edges removed.

its wide-spread availability.

We conducted two sets of experiments. The objective of the first set was to concentrate on people's ability to draw and the effectiveness of the two similarity calculating algorithms with the variable of human memory removed. With the second set of experiments, we hoped to investigate the robustness of the two algorithms when the memory factor was taken into account.

The 16 images that were used as target images are shown in Figure 11. Some have only a few clearly perceptible contours and uniform background while some have more details and noisy background. Five subjects were asked to sketch the edges of these target images while the images were shown to them. One observation was that they tended to ignore some of the shorter edges when the target image had more details. Figure 12 depicts the edge maps of two target images with their respective sketches.

Moreover, even if they were asked to draw the edges with care, sketched edges did not align well with those in the images. Other observed deviations of the sketches from the actual edge maps include skewed or displaced contours (edges) of objects, edges that the users drew but missing from the respective edge maps. This latter deviation arises from the fact that humans have a strong object recognition ability and hence could easily trace the contours of objects using different types of information not available to an edge detection algorithm which merely uses color gradients to

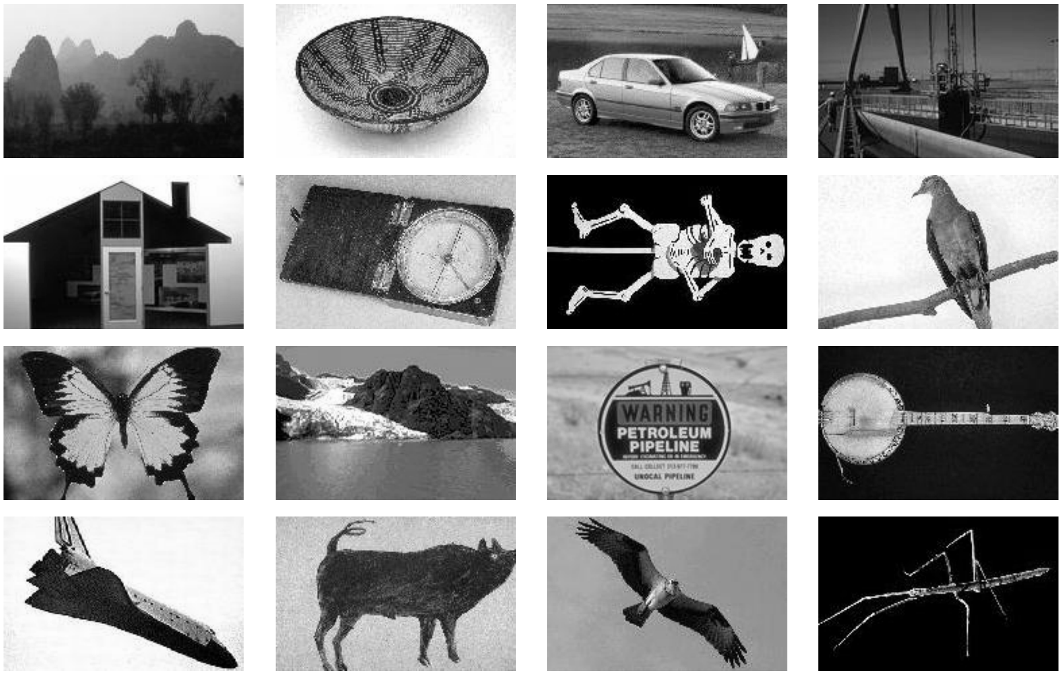


Figure 11: 16 images from a database of 137 color photo and computer generated images as targets in our query experiments.

accomplish its task.

Studying the overall distortions of the sketches drawn by different subjects, we found that Subject 3 drew with most precision. Subjects 2, 4, and 5 sketched with less accurate pixel correspondence but edges in sketches were still close to what were in the intended database images. Comparatively, sketches from Subject 1 appeared most distorted.

In Figure 13, we plot the ranks of these 16 target images among the 137 database images for the 5 subjects. It can be observed that the feature-based method generally performs better than the pixel-based approach in terms of ranking of target images.

When the sketches were drawn with close correspondence to the intended images, as in the case of sketches from Subject 3, both algorithms gave good retrieval results, with the feature-based method performing better. When sketches were drawn with somewhat less precision, as in the case of sketches by Subjects 2, 4, and 5, the feature-based method still provided good performance. However, the pixel-based technique's effectiveness degraded much more.

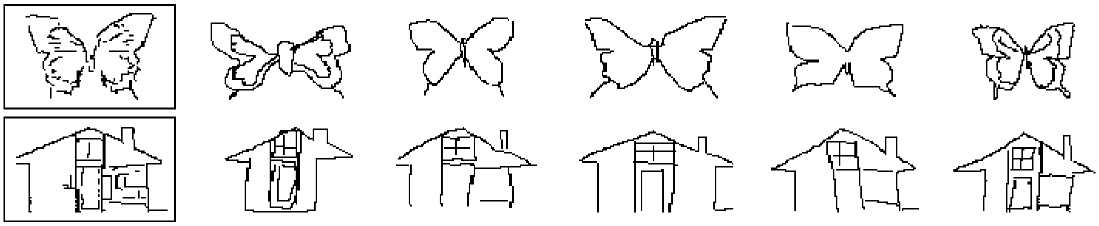


Figure 12: Edge maps of two images (left most on each row) and their respective sketches by 5 subjects.

When the sketch precision further deteriorated, as in the case of sketches from Subject 1, retrieval effectiveness of both methods declined more. Although its performance was still “better” than that of the pixel-based method overall, the feature-based technique demonstrated some erratic retrieval results, having one target ranked at 30 and two ranked around 80, out of a total of 137 images. The reason was that even though the feature-based method provided some tolerance towards distortions in sketches, it failed to capture the similarities between first degree curvelets when the distortions were too large. We believe that at this level of sketch distortion, a matching technique that makes use of higher degree curvelets which capture more global structural information would be more robust.

When a user wishes to find an image from the database, very often he/she does not have the intended image shown before him/her. He/she may have seen it and only remembers roughly what the image may look like. The second set of experiments conducted was to investigate the effectiveness of the two algorithms when the human memory variable was considered. We asked Subjects 3 and 5 to remember those test images when they performed the first sketch session. One day later, they were asked to draw what they remembered.

Generally speaking, sketches from the second session were more distorted compared to their counterparts from the first session. Subject 3 seemed to remember the images better than Subject 5 and hence his sketches showed closer correspondence to what were intended. Among other deviations, Subject 5 drew a 160° -rotated and shrunk version of the object in the 12^{th} test image. His sketches exhibited a higher degree of distortion when compared those by those by Subject 1 from the first session.

Figure 14 shows the retrieval results of their sketches. Due to more distortions of the sketches and as expected, ranks of sketches from the second session are lower for both pixel-based and feature-based similarity computation methods, when compared with their results using sketches from the two subjects’ first sketch session.

The feature-based method demonstrated better retrieval performance using sketches by Subject 3. However, when examining the results using sketches from by Subject 5, we could not decide which method gave better results. At a high level of sketch distortion, neither the feature-based method nor the pixel-based method performed reasonably. We believe that the feature-based should use higher curvelets in order to improve its tolerance towards this level of distortions.

4 Conclusions and Future Work

In this paper, we introduce a feature-based approach for image retrieval by sketch. Using edge segments as features which are modeled by implicit polynomials, we hope to provide a similarity computation method that is robust towards user query sketch distortions. We report some preliminary results of the first phase of our work in this paper. From these experimental results, we could see that the feature-based method, which currently uses first degree curvelets, generally performs better than the pixel-based method. It appears more robust and tolerant towards distortion in the sketches. We attribute this quality to the fact that it uses more structure information to compute the similarities between images.

However, in cases where the degree of distortion in sketches deteriorates further, it would give erratic retrieval results. We believe that at this level of sketch distortion, higher degree curvelets that capture even more structure information of images should be used to improve the retrieval performance of the feature-based method. The investigation of such a technique is the objective of our immediate future work.

References

[1] K. Hirata and T. Kato, “Query by Visual Example”, in *Advances in Database Technology EDBT ’92, Third International Conference on Extending Database Technology*, March 1992.

[2] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, “Query by Image and Video Content: The QBIC System,” in *Computer*, pp. 23–32, September, 1995.

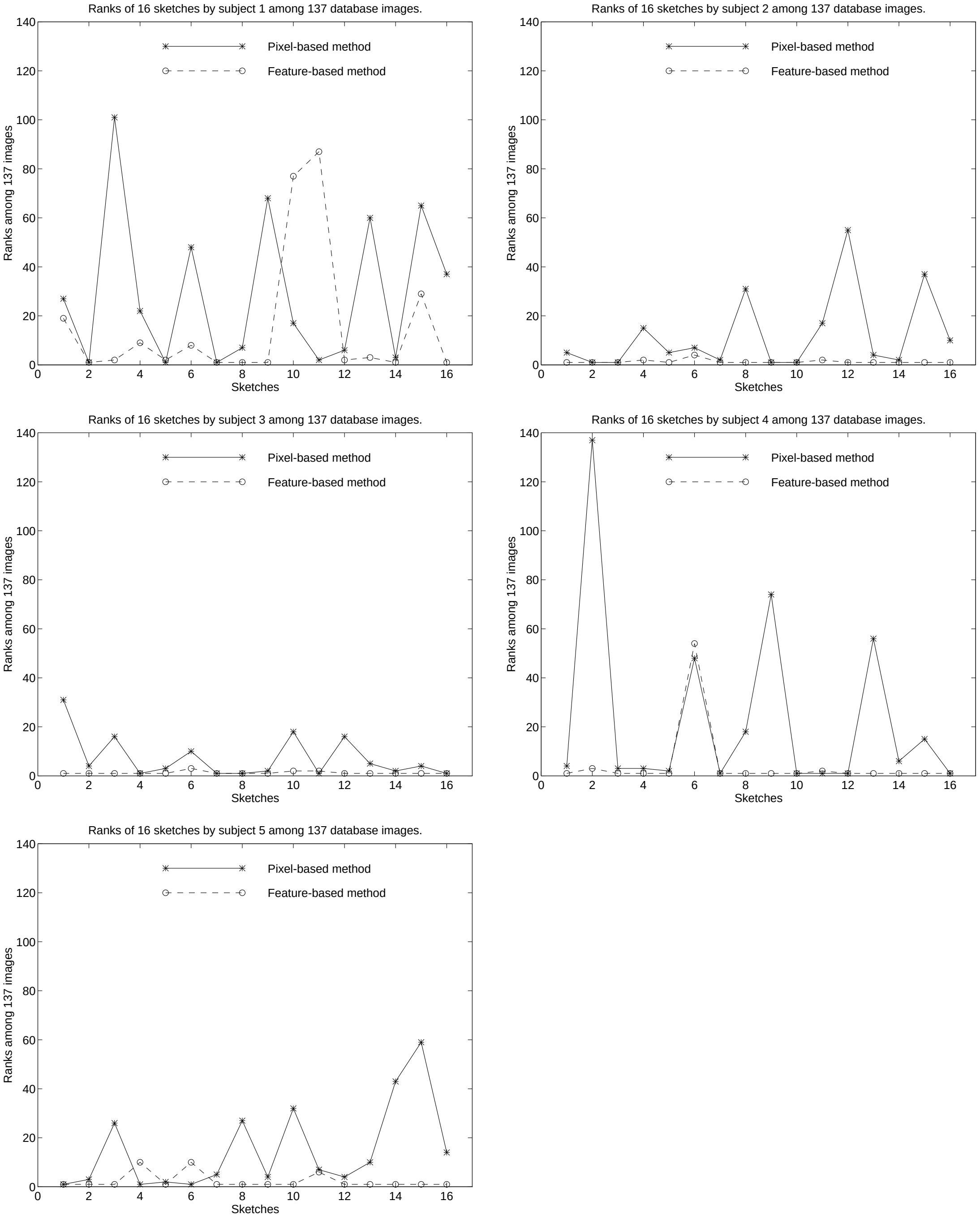


Figure 13: Ranks of 16 target images among 137 database images for 5 subjects. Sketches were drawn with target images being shown to the subjects.

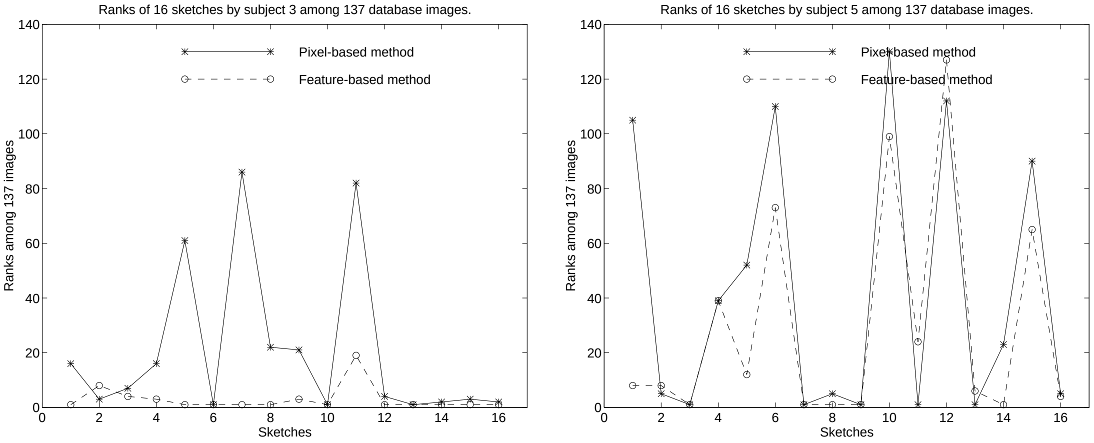


Figure 14: Ranks of 16 target images among 137 database images for 2 subjects. Sketches were drawn without target images being shown to the subjects.

[3] H.J. Zhang, C.Y. Low, S.W. Smoliar, and J.H. Wu, “Video Parsing, Retrieval and Browsing: An Integrated and Content-Based Solution,” in *Proceedings of ACM Multimedia ’95*, pp. 15–24.

[4] A. Del Bimbo, P. Pala, and S. Santini, “Image Retrieval by Elastic Matching of Shapes and Image Patterns,” in *Proceedings of IEEE Multimedia ’96*, pp. 215–218.

[5] J.R. Smith and S.F. Chang, “VisualSEEK: a fully automated content-based image query system,” in *Proceedings of ACM Multimedia ’96*, pp. 87–98.

[6] C.E. Jacobs, A. Finkelstein, and D.H. Salesin, “Fast Multiresolution Image Querying,” in *Proceedings of ACM SIGGRAPH ’95*, pp. 277–286.

[7] W. Hsu, T.S. Chua, and H.K. Pung, “An Integrated Color-Spatial Approach to Content-Based Image Retrieval,” in *Proceedings of ACM Multimedia ’95*, pp. 305–313.

[8] D. Lopresti and A. Tomkins, “Temporal-Domain Matching of Hand-Drawn Pictorial Queries,” in K.L. Simner, C.G. Leedham, A.J.W.M. Thomassen, editors, *Handwriting and drawing Research: Basic and Applied Issues*, IOS Press 1996, pp. 387–401.

[9] D. Lopresti and A.Tomkins, “Block Edit Models for Approximate String Matching,” in *Proceedings of the Second Annual South American Workshop on String Processing*, pp. 11–26.

[10] D. Lopresti, A.Tomkins, and J.Y. Zhou, “Algorithms for Matching Hand-Drawn Sketches,” in *Proceedings of the Fifth International Workshop on Frontiers in Handwriting Recognition*, pp.233–238.

[11] R.S. Gray, “Content-based Image Retrieval: Color and Edges”, in *Proceedings of the 1995 Dartmouth Institute for Advanced Graduate Studies: Electronic Publishing and the Information Superhighway*, pp. 77–90.

[12] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, Addision Wesley, 1992.

[13] D.B. Cooper and Z. Lei, “On Representation and Invariant Recognition of Complex Objects Based on Patches and Parts,” in *Springer Lecture Notes in Computer Science series*, M. Hebert, J. Ponce, T. Boult, A. Gross, editors, pp. 139-153, 1995.

[14] Z. Lei, D. Keren and D.B. Cooper, “Computationally Fast Bayesian Recognition of Complex Objects Based on Mutual Algebraic Invariants,” *1995 IEEE International Conference on Image Processing*, Washington, D.C., October 1995

[15] Z. Lei, M.M. Blane, and D.B. Cooper, ”3L Fitting of Higher Degree Implicit Polynomials,” *3rd IEEE Workshop on Applications of Computer Vision*, Sarasota, FL, December 1996.

[16] Z. Lei and D.B. Cooper, ”Implicit Polynomial Based Geometric Shape Modeling and Recognition,” *3rd International Workshop on Visual Form*, Capri, Italy, May 1997.