TRANSFORMATION INVARIANT IMAGE INDEXING AND RETRIEVAL FOR IMAGE DATABASES

T. Gevers and A.W.M. Smeulders
Faculty of Mathematics & Computer Science, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
E-mail: gevers@fwi.uva.nl

Commission III, Working Group III/IV

KEY-WORDS: image databases, constant time image retrieval, transformation invariants, image indexing, query by example, dissimilarity measure.

ABSTRACT:

This paper presents a novel design of an image database system which supports storage, indexing and retrieval of images by content. The image retrieval methodology is based on the observation that images can be discriminated by the presence of image objects and their spatial relations. Images in the database are first automatically segmented into sets of image object descriptions. Then, transformation invariant quantities of image object descriptions are derived and used as keys to index images, described by their image objects, in a hash table. A query corresponding to a sample image or sketch of image segments provided by the user on input is analyzed, invariants are computed and then used to lookup images in the table. This query processing methodology avoids exhaustive searching through the image database and offers constant time image retrieval independent of the size of the image database and the complexity of the imaging process. Performance of the proposed image database system has been evaluated by experiments for a simple application.

1 INTRODUCTION

In domains such as cartography and remote sensing, the majority of archived data is in the form of images. For the management of archived image data, an image database (IDB) system is needed which supports the analysis, storage and retrieval of images. Much attention has been paid to the problems of how to store images and to retrieve them efficiently from an IDB [2],[3],[11], for example. Many data structures have been proposed at the level of pixels, such as pixel-based [4], R-trees [8], quadtrees [12], together with their spatial processing operations. Most of the IDB systems combines these spatial processing capabilities with DBMS capabilities for the purpose of storage and retrieval of complex spatial data. Other IDB systems are still based on the paradigm to store a key-word description of the image content, created by some user on input, in a database in addition to a pointer to the raw image data. Image retrieval is then based on the DBMS capabilities. However, a different approach is required when we consider the wish to retrieve images on the basis of image objects and their spatial relations.

This paper considers a novel design of an IDB system which supports representation, indexing and retrieval of images by content. The IDB system consists of a physical and a logical database. High-resolution images in the physical database are decomposed into sets of image object descriptions which are stored in the logical database. Then, geometrical properties of the image object descriptions are derived invariant to specific transformations. These invariants are used as keys to index images, described by their image objects, in a hash table. To determine which images are to be retrieved, the query corresponding to a sample image or sketch of image segments provided by the user on input, is analyzed, invariants are computed and then used to lookup images in the table. The images are ordered with respect to their proximity to the query image and displayed for viewing. This query processing methodology avoids exhaustive searching through the image database and offers constant time image retrieval.

This paper is organized as follows. In Section 2, the image retrieval problem is formulated. Transformation invariants and their indexing functions are discussed in Section 3. The image retrieval method is proposed in Section 4 and experiments carried out on an image database of logo images are discussed in Section 5.

2 RETRIEVAL PROBLEM

This paper approaches the image retrieval problem on the basis of the observation that images can be discriminated by the presence of image objects and their spatial relations. We concentrate on the case of queries by image example where a query image or sketch of image segments is given by the user on input. Then, image retrieval is the process to compute to what extent the images in the database correspond to the query image.

The architecture of the IDB system consists of a physical image store and a logical database. The physical store contains digital photographic images taken from unknown viewpoints of rigid objects in 3-D real-world cluttered scenes. Image description is obtained for every digital image in the physical image store by applying the Canny edge detector [1] followed by a line fitting technique [13] to obtain polygonal approximation of the image object contour which is stored in the logical database. It is assumed that the image objects are distinguishable and identifiable by their contours and that the resolution of the digital image is high enough to identify those boundaries. Further, no major constraints are imposed on:

- *Transformation* : the imaging process (IDB systems should be able to deal with different transformations such as Euclidean, affine and projective transformations)
- *Generality*: the class of objects from which the images are taken from (rigid objects in 3-D real-world cluttered scenes)
- *Stability*: sensing and measurement error (the image retrieval problem is addressed in a realistic context)
- *Robustness*: incomplete data (fragmented, occluded and overlapping objects as well as multiple instances are allowed).

Let the images be represented in the logical database by a set $(L_1, L_2, ..., L_M)$ of image representations, where M is the number of images. The polygonal object descriptions of the ith image together with a variety of global image object features such as area, roundness and perimeter are all stored in L_i. Let P_j^i be the jth image object to exist in image i. Further, let P_j^i consists of $m_{P_j^i}$ vertices $\{(\vec{v}_1^{P_j^i}, \vec{v}_2^{P_j^i}, ..., \vec{v}_{m_{P_j^i}}^{P_j^i})\}$. We use the notation $[k] = k \bmod m_{P_j^i}$ to account for the cyclic nature of polygons. For the ease of exposition, the indexes i (image) and j (image object) of P_j^i will be omitted where possible.

The computational abstraction of the present retrieval problem can be formulated as follows. Given two sets of image objects namely Q, the query image, and I, an arbitrary image in the image database, both consisting of (possibly incomplete) polygons, the retrieval problem is to determine to what extent Q and I correspond to the same set of polygons.

As an example, consider an airplane flying above the plane shown in Fig. 1.

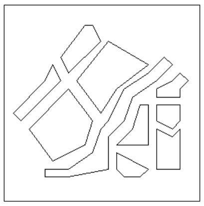

Figure 1: Polygon configuration

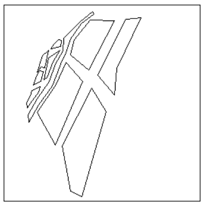

Figure 2: Perspective image of Fig. 1

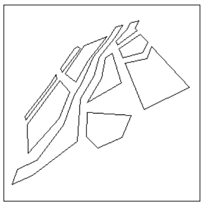

Figure 3: Perspective image of Fig. 1

Two perspective projections of this plane, taken from the airplane, are shown in Fig. 2 and Fig. 3. Imagine an IDB containing a large number of digital images including the two images (shown in Fig. 2, 3) and no other information is present (e.g. about the positions from which the images are taken). When the original polygonal configuration (shown in Fig. 1) is taken as the query image, then the image retrieval problem is to find out to what extent the images in the IDB correspond to the query image by computing a dissimilarity measure. The dissimilarity measure is used to order the images by their proximity to the query. Images with high correspondence are considered the same or similar to the query and are displayed for viewing.

A major problem in image retrieval by content is that an object in 3-D space might be seen from different points of view, resulting in different image objects, see Fig. 1, 2 and 3 for example. A simple and naive method for similarity retrieval is to perform every possible transformation of an image object to see if any of its transformed versions match the query image object. However, the search becomes overwhelmingly large for complex transformations such as the affine and projective transformation. Another approach is to derive geometrical properties of the image objects invariant to specific transformations. The invariants can be used as keys to index images, described by their image objects, in a hash table.

3 IMAGE OBJECT INDEXING

Although Q is formulated to consist of one or more polygons, it is assumed for the ease of illustration and without loss of generality that Q consist of only one polygon in the sequel of the paper.

For the purpose of efficient image retrieval, a hash table is formed where each image is indexed according to the invariants (keys) computed from their polygonal image object descriptions. Let $f()$ be defined as an indexing function. For each key, the indexing function $f()$ computes the address, where the tuple *(image,object)* is added as an entry where *image* denotes the image and *object* the image object from which the key is computed of. Depending on the imaging process, indexing functions are to be defined for the Euclidean, affine and the projective transformation.

3.1 Euclidean Indexing Function

It is known that when an object is transformed rigidly by rotation and translation, then its length is an invariant. A plane rotation can be represented by a linear transformation of 2-D coordinates:

$$x' = x\cos\theta - y\sin\theta \tag{1}$$

$$y' = x\sin\theta + y\cos\theta \tag{2}$$

The standard Euclidean distance between two points $\vec{a}$ and $\vec{b}$ is invariant under plane rotation:

$$\begin{aligned}(a_x - b_x)^2 + (a_y - b_y)^2 &= \\ [\cos\theta(a_x - b_x) - \sin\theta(a_y - b_y)]^2 &+ \\ [\sin\theta(a_x - b_x) + \cos\theta(a_y - b_y)]^2 & \end{aligned} \tag{3}$$

For polygon P with m vertices $\{(\vec{v}_1, \vec{v}_2, ..., \vec{v}_m)\}$, $f_E()$ is defined as an indexing function which is unchanged as the vertices undergo any two-dimensional Euclidean transformation:

$$f_E(\vec{v}_k, \vec{v}_{[k+1]}) = \sqrt{((v_{k_x} - v_{[k+1]_x})^2 + ((v_{k_y} - v_{[k+1]_y})^2} \tag{4}$$

where $\vec{v}_k \in \{(\vec{v}_1, \vec{v}_2, ..., \vec{v}_m)\}$.

For each P^i_j stored in the logical store the address $f_E(\vec{v}_k^{P^i_j}, \vec{v}_{[k+1]}^{P^i_j})$, for $\vec{v}_k^{P^i_j} \in \{(\vec{v}_1^{P^i_j}, \vec{v}_2^{P^i_j}, ..., \vec{v}_{m_{P^i_j}}^{P^i_j})\}$, is used to index the image object by adding the tuple *(i, j)* as an entry at the address.

3.2 Affine Indexing Function

Affine transformation of the plane can be viewed as a three-dimensional translation and rotation followed by orthographic projection plus scaling.

A 2-D affine transformation $A : \Re^2 \rightarrow \Re^2$ is a non-singular 2x2 matrix L and a translation $\vec{t} \in \Re^2$:

$$\vec{x}' = L\vec{x} + \vec{t} \tag{5}$$

where $\vec{x}', \vec{x} \in \Re^2$.

Known property of the 2-D affine transformation is:

- Given four points $\vec{a}$, $\vec{b}$, $\vec{c}$ and $\vec{d}$ and let $\vec{s}$ be their intersection point, then

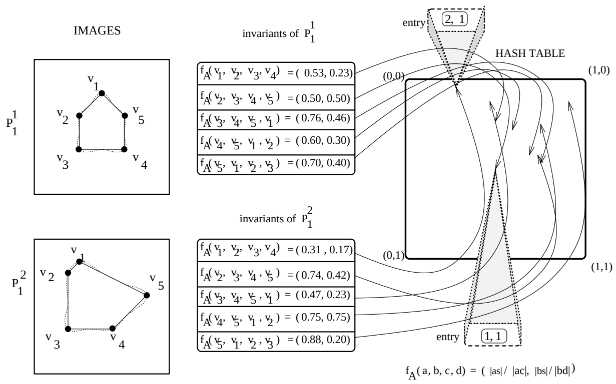

Figure 4: Indexing scheme

$$\frac{|\vec{a}\vec{s}|}{|\vec{a}\vec{b}|} = \frac{|f_A(\vec{a})f_A(\vec{s})|}{|f_A(\vec{a})f_A(\vec{b})|} \tag{6}$$

$$\frac{|\vec{c}\vec{s}|}{|\vec{c}\vec{d}|} = \frac{|f_A(\vec{c})f_A(\vec{s})|}{|f_A(\vec{c})f_A(\vec{d})|} \tag{7}$$

where $|\vec{a}\vec{b}|$ denotes the length of segment $\vec{a}\vec{b}$.

For polygon P, the indexing function $f_A()$ is defined as:

$$f_A(\vec{v}_k, \vec{v}_{[k+1]}, \vec{v}_{[k+2]}, \vec{v}_{[k+3]}) = \left(\frac{|\vec{v}_k\vec{s}|}{|\vec{v}_k\vec{v}_{[k+2]}|}, \frac{|\vec{v}_{[k+1]}\vec{s}|}{|\vec{v}_{[k+1]}\vec{v}_{[k+3]}|}\right) \tag{8}$$

The method to compute the index key of vertex $\vec{v}_k \in \{(\vec{v}_1, \vec{v}_2, ..., \vec{v}_m)\}$ of P is as follows:

- For vertices $\vec{v}_k$, $\vec{v}_{[k+1]}$ and $\vec{v}_{[k+2]}$, find vertex $\vec{v}_{[k+l]}$, $l \geq 3$, such that lines $\vec{v}_k\vec{v}_{[k+2]}$ and $\vec{v}_{[k+1]}\vec{v}_{[k+l]}$ intersect.
- Compute intersection point $\vec{s}$ for segment $\vec{v}_k\vec{v}_{[k+2]}$ with segment $\vec{v}_{[k+1]}\vec{v}_{[k+l]}$.
- Compute key values $\frac{|\vec{v}_k\vec{s}|}{|\vec{v}_k\vec{v}_{[k+2]}|}$ and $\frac{|\vec{v}_{[k+1]}\vec{s}|}{|\vec{v}_{[k+1]}\vec{v}_{[k+l]}|}$.

Thus, for each P^i_j the address is computed and the tuple (i, j) is added as an entry $f_A(\vec{v}_k^{P^i_j}, \vec{v}_{[k+1]}^{P^i_j}, \vec{v}_{[k+2]}^{P^i_j}, \vec{v}_{[k+3]}^{P^i_j}) = (i, j)$, for $\vec{v}_k^{P^i_j} \in \{(\vec{v}_1^{P^i_j}, \vec{v}_2^{P^i_j}, ..., \vec{v}_{m_{P^i_j}}^{P^i_j})\}$.

As a motivating example, imagine a database of two images each described by one image object. The images are represented and indexed by their image object polygonal approximations P^1_1 and P^2_1, see Fig. 4.

3.3 Projective Indexing Function

For the projective case, the issue arises which geometrical properties of the shape of a planar object are invariant under a change in the point of view. Perspective projections are often used as approximations to the camera imaging process [9]. From the classical projective geometry we have the following theorem [5], [14]:

- Theorem 1: Consider an object O in $\Re^3$, five points of which lie in a plane. Let $\vec{a}$, $\vec{b}$, $\vec{c}$, $\vec{d}$ and $\vec{e}$ be that five points, no three collinear; let $\vec{a_1}$, $\vec{b_1}$, $\vec{c_1}$, $\vec{d_1}$ and $\vec{e_1}$ be their images on the image plane. Extend rays from $\vec{a}$ to each other point. Let $\theta_1, \theta_2, \theta_3$ be the the angles at $\vec{a}$ between $\vec{ab}$ and $\vec{ac}$, $\vec{ac}$ and $\vec{ad}$, and $\vec{ad}$ and $\vec{ae}$ respectively. Let θ_i^1 be the corresponding angles in the image plane. Then:

$$\frac{\sin(\theta_1+\theta_2)\cdot\sin(\theta_2+\theta_3)}{\sin(\theta_2)\cdot\sin(\theta_1+\theta_2+\theta_3)} = \frac{\sin(\theta_1^1+\theta_2^1)\cdot\sin(\theta_2^1+\theta_3^1)}{\sin(\theta_2^1)\cdot\sin(\theta_1^1+\theta_2^1+\theta_3^1)} \tag{9}$$

independent of the projection viewpoint.

So the cross ratio of sines between five points in a plane is projective invariant.

The indexing function $f_P()$ for polygon P is defined as:

$$f_P(\vec{v}_k, \vec{v}_{[k+1]}, \vec{v}_{[k+2]}, \vec{v}_{[k+3]}, \vec{v}_{[k+4]}) = \frac{\sin(\theta_1+\theta_2)\cdot\sin(\theta_2+\theta_3)}{\sin(\theta_2)\cdot\sin(\theta_1+\theta_2+\theta_3)} \tag{10}$$

where $\theta_1, \theta_2, \theta_3$ are the angles at $\vec{v}_k$ between $\vec{v}_k\vec{v}_{[k+1]}$ and $\vec{v}_k\vec{v}_{[k+2]}$, $\vec{v}_k\vec{v}_{[k+2]}$ and $\vec{v}_k\vec{v}_{[k+3]}$, and $\vec{v}_k\vec{v}_{[k+3]}$ and $\vec{v}_k\vec{v}_{[k+4]}$ respectively.

The method to compute the cross ratio of vertex $\vec{v}_k \in \{(\vec{v}_1, \vec{v}_2, ..., \vec{v}_m)\}$ of P is as follows:

- Extend straight lines going through $\vec{v}_k$ and each of the following four vertices $\vec{v}_{[k+1]}$, $\vec{v}_{[k+2]}$, $\vec{v}_{[k+3]}$ and $\vec{v}_{[k+4]}$.
  If two lines overlap, $\vec{v}_{[k+4]}$ is replaced by the first following vertex $\vec{v}_{[k+l]}$, $l \geq 5$, such that the lines do not overlap.
- Take the line from $v_j^{P_i}$ to $v_{[j+1]}^{P_i}$ as the first line. Rotate the first line in a counterclockwise manner. The first three lines encountered in this way are regarded as the other three lines.
- Calculate the angles θ_1, θ_2 and θ_3 between the four lines and compute the projective invariant key value $\frac{\sin(\theta_1+\theta_2)\cdot\sin(\theta_2+\theta_3)}{\sin(\theta_2)\cdot\sin(\theta_1+\theta_2+\theta_3)}$.

Again for each P_j^i the address is computed and the tuple (i,j) is added as an entry $f_P(\vec{v}_k^{P_j^i}, \vec{v}_{[k+1]}^{P_j^i}, \vec{v}_{[k+2]}^{P_j^i}, \vec{v}_{[k+3]}^{P_j^i}, \vec{v}_{[k+4]}^{P_j^i}) = (i,j)$.

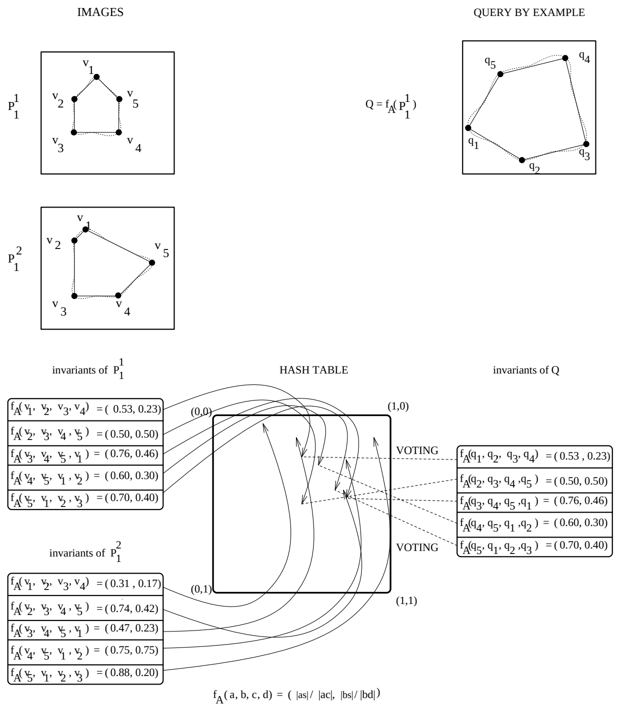

Figure 5: Voting scheme

4 IMAGE RETRIEVAL

For each indexing function a separated hash table is created. All images, described by their image objects, are indexed off-line. The actual on-line image retrieval consists of two separated stages: (1) computation of candidate solutions by fast filtering (2) verification of each candidate solution.

4.1 Generating Candidate Solutions

The user is allowed to specify query Q and viewing transformation (i.e. Euclidean, affine or projective transformation) on input. The polygonal description of the query image is computed and invariant values (i.e. invariant to the transformation as specified by the user) are calculated. For each key, indexing function $f()$ determines the address. Then for each tuple (i,j) appearing at that address, a vote is given, see Fig. 5, for the affine case.

If Q, consisting of n vertices, lays unoccluded in image i corresponding to image object j, then n votes are given to tuple (i,j). Even in the case that the image object lays occluded in the image, the image object is still detectable. A vote accumulator selects those image objects for which the number of votes exceeds a voting threshold. The threshold must be defined relative to the number of vertices n of Q, because the number of vertices determines the number of votes. Let t_r be a relative threshold, then the voting threshold t_v is defined as:

$$t_v = t_r * n \tag{11}$$

where $t_r \in (0, 1)$ is the amount of tolerated occlusion.

4.2 Verifying Candidate Solutions

The next step is to verify candidate solutions to be an instance of Q, because the filtering technique is not necessarily a conclusive retrieval method but a procedure to select efficiently promising candidate solutions. A candidate solution scoring a large number of votes implies that the number of similar invariant values is sufficient but does not necessarily conclusively induce that the order in which they occur is held. Consequently, a candidate solution may have a different shape then Q. Therefore, to avoid false candidate solutions a verification step is necessary.

Consider the Euclidean transformation consisting of a translation and rotation. A translation and rotation transformation is uniquely defined by the transformation of two points [10]. Therefore, each combination of two vertices can be chosen to represent all other vertices.

More precisely, let P be a polygon selected by the vote accumulator for verification. Again P consists of m vertices $\{(\vec{v}_1, \vec{v}_2, ..., \vec{v}_m)\}$, see Fig 6. Let Q be the sample query polygon consisting of n vertices $\{\vec{q}_1, \vec{q}_2, ..., \vec{q}_n\}$, see Fig. 8.

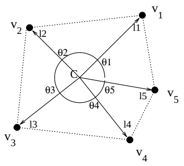

Figure 6: Image polygon P

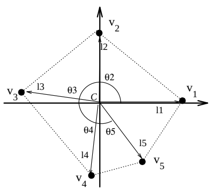

Figure 7: P transformed by basis $(\vec{v}_0, \vec{v}_1)$

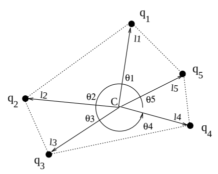

Figure 8: Query sample polygon Q

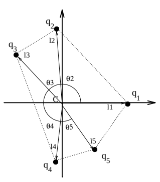

Figure 9: Q transformed by basis $(\vec{q}_0, \vec{q}_1)$

Let P be transformed in the 2-D Euclidean plane spanned by $(\vec{v}_0, \vec{v}_1)$, see Fig. 7, where v_0 is defined as the centroid C of P. The same is done for Q for basis $(\vec{q}_0, \vec{q}_1)$, where $\vec{q}_0$ is the centroid of Q, see Fig.

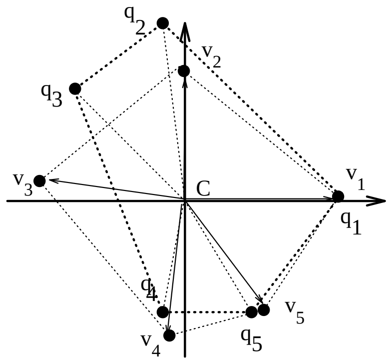

Figure 10: Euclidean canonical frame

9. By mapping $(\vec{v}_0, \vec{v}_1)$ and $(\vec{q}_0, \vec{q}_1)$ to two corresponding vertices of an unit square, a canonical frame is obtained where the coordinates of all other vertices are Euclidean invariant, see Fig. 10.

The method is suitable for other transformations by changing the number of basis vertices. 3 basis vertices are enough for the 2-D affine transformation and 4 basis vertices are needed for a projective transformation [6].

Let k be the number of basis vertices needed for the desired transformation. To compare P and Q, k most similar invariant value pairs between the two sequences of invariant values of P and Q are determined by taken the least square error measure. In this way, k vertex pairs, from which the most similar invariant value pairs are computed, are obtained. The canonical frame is obtained by mapping the k basis vertex pairs to the vertices of a unit square. All other vertices are transformed with respect to the k-tuple basis, see Fig. 10 as an example of the Euclidean case.

To allow vertex occlusion, a mapping function $g()$ is defined which refers to pairs of vertices which are close enough with respect to their length and direction:

$$g(\vec{v}_i) = \begin{cases} h(\vec{v}_i), & \text{if} \quad \rho(L(\vec{v}_i), L(\vec{q}_i)) < t_l \text{ and} \\ & \quad \rho(\theta(\vec{v}_i), \theta(\vec{q}_i)) < t_\theta, h(\vec{v}_i) = \vec{q}_i \\ \emptyset & \end{cases} \tag{12}$$

where t_l and t_θ are thresholds, $L(\vec{v}_i)$ denotes the distance of vertex $\vec{v}_i$ to the origin of the canonical frame and $\theta(\vec{v}_i)$ the angle, where $\rho(.,.)$ is the L_2 norm.

Because any measurement made in the canonical frame is invariant to the specific transformation, the error in length and direction is computed as follows:

$$\epsilon_l = \sum_{g(\vec{v}_i) = \vec{q}_i, \vec{q}_i \neq \emptyset} \rho(L(\vec{v}_i), L(\vec{q}_i)) \tag{13}$$

$$\epsilon_\theta = \sum_{g(\vec{v}_i) = \vec{q}_i, \vec{q}_i \neq \emptyset} \rho(\theta(\vec{v}_i), \theta(\vec{q}_i)) \tag{14}$$

The total error distance is:

$$\epsilon_t = \frac{\epsilon_l}{\eta \cdot t_l} + \frac{\epsilon_\theta}{\eta \cdot t_\theta} \quad (15)$$

where η is the number of vertices pairs contributing to the error.

The verification method, consisting of computing the dissimilarity measure between P and Q in the canonical frame, is as follows:

- 1 k basis vertices of P and Q are obtained by selecting k pairs which produce the most similar subsequence of invariant values.

- 2 Map those k basis pairs to the vertices of a unit square. All other vertices are transformed with respect to the k-tuple basis.

- 3 Compute ϵ_t.

- 4 If ϵ_t is below a threshold then P is considered as a solution and its dissimilarity measure to the sample query Q is expressed by ϵ_t.

For the purpose of image retrieval, the correspondence measure ϵ_t is used to order the images by their proximity to the query, where images with a low dissimilarity measure are considered the same or similar to the query. Finally, the images are displayed for viewing.

4.3 Stability

Due to the uncertainty in location data obtained from real sensing devices, it is important to analyze the effects of this uncertainty with respect to indexing functions. Noise and error cause variation in the invariant values yielding incorrect mapping of a query image to an image object. If the invariants are very sensitive to position error, then for real image data the voting scheme will be unusable.

The method to estimate the effects of error on the indexing functions are discussed in [7].

If it is assumed that the query image is noise free, then it can be shown that the error in $f_E()$, $f_A()$ and $f_P()$ is bounded. Therefore, $f()$ can be interpreted as a range of addresses because noise and error cause variations in the key values. For determining whether an image object correspond to a query image, the range of invariant values is used to select all image objects that fall within the index range. In this way, it is assured that indexing for the correct image object is not lost. The need to access a range of hashes yields an increased number of wrong candidate solutions. However, these wrong candidate solutions will be discarded in the verification step. In the case of a large index ranges, it makes sense to skip those exceeding a certain threshold, because large index ranges will introduce a relative large amount of wrong candidate solutions.

4.4 Generality and Robustness

As discussed above, the class of objects (*generality*), the proposed IDB system is able to cope with, is the class of rigid objects in 3-D space.

Another important issue is whether the IDB system is able to deal with incomplete data (*robustness*), because objects in real-world cluttered scenes may be hidden from view (e.g. partial occlusion and overlapping objects). Because a number or sequence of transformation-invariant quantities for subparts of each image object is computed, the image, described by its image objects, is still retrievable even if the object is partly hidden from view.

4.5 Complexity

The computational cost of generating the candidate solutions is independent of the size of the image database. Therefore, the query processing methodology, proposed in this paper, avoids exhaustive searching through the entire image database and offers constant time candidate image retrieval. However, the time complexity of the verification step is linear $O(n)$ where n is the number of vertices of Q. Because the verification step is only executed on a small set of candidate solutions, its computational cost is relatively cheap with respect to the complexity of the entire image retrieval problem.

Because the on-line image retrieval method (indexing is done off-line) can be computed very efficient due to the hash table and voting scheme (i.e. computing the hash $f()$ of Q and giving votes to each P appearing there), the method is very efficient and is able to be executed at high speeds allowing real-time image retrieval for a large image database even for complex transformations.

5 EXPERIMENTS

In this section, a simple application is discussed to illustrate the nature of the image retrieval method and its use in real-world situations. Experiments have been carried out on a SPARC-10 station with UNIX as operating system.

Consider an IDB containing images of logo objects. The set of logo objects from which the images have been taken from are shown in 11. Logo object P_i is given in terms of its vertices in a counterclockwise order, where v_1 denotes the starting vertex. Two logo images extracted from the image database are shown in Fig. 12 and 13. The imaging process yielding images Fig. 12 and 13 taken from the set of 2-D planar logo objects in 3-D space, is by an orthographic (affine transformation) and perspective projection, respectively. For the image in Fig. 12, affine invariant values have been computed for the image objects, see Fig. 16. These invariants have been used as keys to

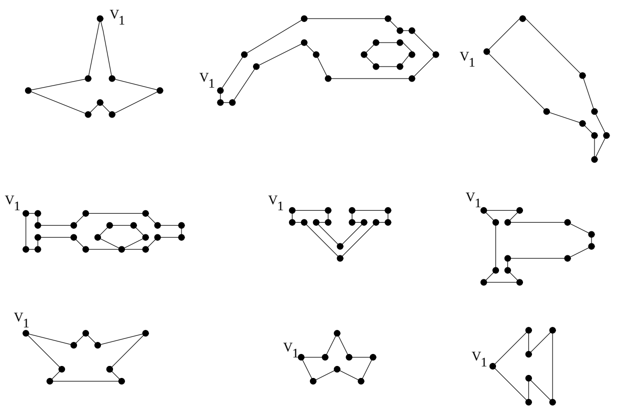

Figure 11: The set of logo objects from top left to bottom right: P_1, P_2,...,P_9

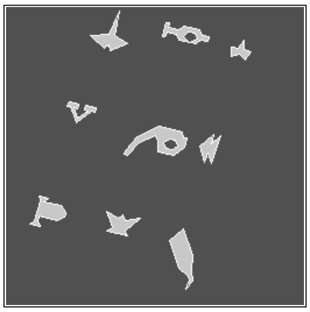

Figure 12: Logos affine transformed

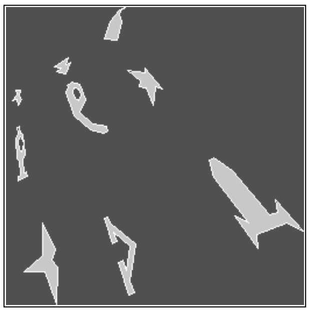

Figure 13: Logos perspective transformed

index the image, described by its image objects, in a hash table. Then, a query image object, corresponding to P_6, has been provided by the user. To determine which images are to be retrieved, the query object has been analyzed, invariants are computed and then used to lookup images in the table. The affine invariant values of the query object are shown in Fig. 14.

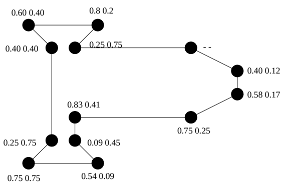

Figure 14: Affine invariants

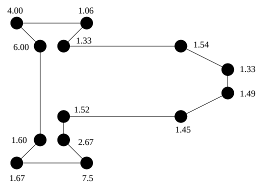

Figure 15: Projective invariants

Notice that because of substantial sensor noise, invariant values computed from the the image object corresponding to the query object differs slightly from those of the query object.

Projective invariant values are computed for the image objects to exist in the image shown in Fig. 13, see Fig. 17. Again these invariants have been used as keys to index the image, described by its image objects, in a hash table. The query image object provided by the user is shown in Fig. 15.

The running time for the logo image database containing over more than 100 images including those shown in Fig. 12 and 13 was about 0.6 seconds. The image retrieval system correctly retrieved all images, containing image object P_6, and no false matches were obtained.

AFFINE	v_1	v_2	v_3	v_4
P_1	- -	0.58 0.84	0.95 0.37	- -
P_2	0.57 0.14	0.39 0.28	0.70 0.43	0.94 0.35
P_3	0.82 0.39	0.78 0.18	0.07 0.11	- -
P_4	0.78 0.71	0.92 0.40	0.5 0.33	0.25 0.46
P_5	0.93 0.11	- -	0.99 0.73	- -
P_6	0.62 0.42	0.42 0.37	0.24 0.73	0.73 0.75
P_7	- -	0.36 0.56	0.93 0.33	- -
P_8	0.81 0.54	- -	0.40 0.66	0.94 0.50
P_9	0.97 0.50	- -	0.25 0.74	0.62 0.57
	v_5	v_6	v_7	v_8
P_1	0.19 0.41	0.89 0.69	- -	0.82 0.63
P_2	0.32 0.06	0.60 0.37	0.81 0.39	0.75 0.54
P_3	0.87 0.15	0.28 0.24	0.49 0.50	0.82 0.59
P_4	0.58 0.40	0.93 0.53	- -	0.44 0.57
P_5	0.30 0.25	0.53 0.52	0.75 0.75	0.75 0.25
P_6	0.60 0.06	0.07 0.40	0.81 0.39	0.77 0.25
P_7	0.69 0.53	- -	0.77 0.33	- -
P_8	- -	0.46 0.66	- -	0.23 0.56
P_9	- -	- -	0.66 0.97	
	v_9	v_{10}	v_{11}	v_{12}
P_1				
P_2	0.98 0.07	- -	0.45 0.22	0.80 0.37
P_3				
P_4	0.84 0.13	- -	0.63 0.48	0.84 0.23
P_5	0.02 0.25	0.71 0.94	0.92 0.23	0.35 0.24
P_6	0.55 0.16	0.42 0.15	- -	0.80 0.21
P_7	0.43 0.45			
P_8				
P_9				
	v_{13}	v_{14}	v_{15}	v_{16}
P_1				
P_2	0.85 0.55	0.86 0.41		
P_3				
P_4	0.55 0.40	0.00 0.25	0.34 0.22	0.48 0.46
P_5	0.48 0.48	0.75 0.78		
P_6	0.84 0.20			
P_7				
P_8				
P_9				

Figure 16: Table of measurements for the affine image

6 CONCLUSIONS

We presented a novel design of an IDB system which supports representation, indexing and retrieval of images by content. The architecture of the IDB system consists of a physical image store and a logical database. Image description is automatically obtained for every digital digital image in the physical image store by applying image processing and pattern recognition techniques. The image description include polygonal approximation of the image object contour

PROJ.	v_1	v_2	v_3	v_4
P_1	1.62	1.25	1.37	1.14
P_2	1.05	1.67	2.54	10.99
P_3	-	4.10	1.55	1.90
P_4	2.33	1.09	-	5.96
P_5	14.50	-	8.62	-
P_6	3.49	6.60	1.59	1.61
P_7	9.69	6.01	1.22	1.44
P_8	7.83	-	1.95	-
P_9	2.21	1.17	1.12	2.16
	v_5	v_6	v_7	v_8
P_1	1.62	5.48	1.64	5.42
P_2	5.17	1.59	1.11	1.55
P_3	2.42	2.93	1.41	1.60
P_4	1.08	1.47	1.36	1.22
P_5	2.55	1.06	1.84	3.33
P_6	7.40	2.98	1.51	1.44
P_7	5.10	1.96	3.17	4.16
P_8	1.31	4.59	4.57	1.31
P_9	2.11	1.64	7.00	
	v_9	v_{10}	v_{11}	v_{12}
P_1				
P_2	4.25	1.18	1.49	1.19
P_3				
P_4	2.02	5.17	-	12.59
P_5	1.15	1.17	1.05	2.45
P_6	1.49	1.27	1.65	1.39
P_7	7.00			
P_8				
P_9				
	v_{13}	v_{14}	v_{15}	v_{16}
P_1				
P_2	1.43	1.70		
P_3				
P_4	16.97	1.23	2.66	1.18
P_5	1.09	3.70		
P_6	1.06			
P_7				
P_8				
P_9				

Figure 17: Table of measurements for the projective image

together with a variety of global image object features. Geometrical properties of the image object descriptions are then derived invariant to specific transformations. These invariants are used as keys to index images, described by their image objects, in a hash table. The query consists of a sample image or sketch of image segments provided by the user on input. To determine which images are to be retrieved, the query is analyzed, invariants are computed and then used to lookup images in the hash table. A dissimilarity measure is proposed to order the images by their proximity to the query. Images with high correspondence are considered the same or similar to the query and are displayed for viewing. The query processing methodology avoids exhaustive searching through the image database and offers constant time image retrieval. Image retrieval can be done in real-time for large image databases independent of the complexity of the imaging process.

Experiments show excellent performance, even in noisy images, and makes the image retrieval methodology a promising one.

References

[1] Canny, J., *A Computational Approach to Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, pp. 679-698, 1986.

[2] Chang, S. K., Shi, Q. Y. and Yan, C. W., *Iconic Indexing by 2-D Strings*, IEEE Trans. Pattern Anal. Machine Intell., vol. 9, no. 3, 1987, pp. 413-428.

[3] Chang, S. K. and Jungert, E., *A Spatial Knowledge Structure for Image Information Systems using Symbolic Projections*, Proc. Fall Joint Comp. Conf., Dallas, TX, 1986, pp. 79-86.

[4] Chock, M., Cardenas, A., F. and Klinger, A., *Database Structure and Manipulation Capabilities of the Picture Database Management System (PICDMS)* , IEEE Trans. Pattern Anal. Machine Intell., vol. 6, no. 4, 1984, pp. 484-492.

[5] Coxeter, H. S. M., *Projective Geometry* , Univ. of Toronto Press, Toronto, 1974.

[6] Delone, B. N. and Raikov, D. A., *Analytic Geometry* , Vol. 2, Moscow, 1949.

[7] Gevers, T. and Smeulders A. W. M., *On the Stability of Affine and Projective Invariant Indexing*, In preparation.

[8] Guttman, A., *R-trees: a Dynamic Index Structure for Spatial Searching* , Proc. ACM-SIGMOD Int. Conf. Management of Data, June 18-21, 1984, pp. 47-57.

[9] Haralick, R. M., *Using Perspective Transformation in Scene Analysis*, CGIP, vol. 13, pp. 191-221, 1980.

[10] Klein, F., *Elementary Mathematics from an Advanced Standpoint; Geometry*, Macmillan, NY, 1925.

[11] Lee, S. Y. and Hsu, F. J., *Picture Algebra for Spatial Reasoning of Iconic Images represented in 2D C-string*, Pattern Recognition Letters, 12, 1991, pp. 425-435.

[12] Samet, H., *The Quadtree and Related Data Structures*, ACM Computer Surveys, vol. 16. no. 2, 1984, pp. 187-260.

[13] Sharaiha, Y. M. and Christofides, N., *An Optimal Algorithm for the Straight Segment Approximation of Digital Arcs* , CVGIP: Graphical Models and Image Processing, Vol. 55, No. 5, pp. 397-407, 1993.

[14] Veiblen, O. and Young, J. W., *Projective Geometry* , Ginn. Boston, 1910.