AN AFFINE-INVARIANT TOOL FOR RETRIEVING IMAGES FROM HOMOGENEOUS DATABASES*

Ronald Alferez, Yuan-Fang Wang**, Long Jiao
{ronald,yfwang,jiaolong}@cs.ucsb.edu

Department of Computer Science
University of California, Santa Barbara

KEYWORDS: Content-based image retrieval, Affine, Invariants, Shape, Homogeneous database

ABSTRACT

In this paper, we examine the complexities involved in retrieving images from a database comprised of objects of very similar appearance. Such an operation requires a process that can discriminate among images at a very fine level, such as distinguishing among various species of fish. Furthermore, incidental environmental factors such as change in viewpoints and slight, nonessential shape deformation must be excluded from the similarity criteria. To this end, we propose a new method for content-based image retrieval and indexing, one that is well suited for discriminating among objects within the same class in a way that is insensitive to incidental environmental changes. The scheme comprises a global alignment and a local matching process. Affine transform is used to model the different viewpoints associated with positioning the camera, while multi-dimensional indexing techniques are used to make the global alignment scheme efficient. A local matching process based on dynamic programming allows the optimal matching of local structures using cost metrics that may ignore nonessential local shape deformation. Results show the method's ability to cancel out visual distortions caused by a changing viewpoint, and its tolerance to noise, occlusion, and slight deformations of the object.

---

* This research was supported in part by a NSF grant, IIS-9908441 and IIS-9817432.

** Corresponding author contact information: Prof. Yuan-Fang Wang, Department of Computer Science, University of California, Santa Barbara, CA 93106, phone: (805) 893-3866, fax: (805)893-8553, email: yfwang@cs.ucsb.edu

1. INTRODUCTION

Indexing and retrieval operations for image databases present a new set of challenges not often encountered in conventional databases. For instance, images stored in databases are usually inexact pieces of data, often acquired with imprecise photographic equipment, corrupted by noise, and degraded by *lossy* data compression to facilitate efficient storage. Furthermore, the search matching criteria are often based on an entirely different set of objectives than what is normally used in numeric or textual databases. While queries on conventional databases are geared toward matching some relational criteria (such as *"find entries that has a value greater than 5.0"*), the same cannot be said for image databases, which requires some analysis on a more semantic level. Consider for example, two images of the same object, say a chair, taken under different circumstances such as pose and lighting. Although the individual pixel values and various statistics of the two image arrays of the photographs can have very little resemblance to each other, almost all humans posing the query will most likely consider the two images as similar and thus a correct match. Hence, the query requires the database system to have some notion of what the images are in terms of the way humans understand them. But precisely what makes two images appear similar to humans? Even this fundamental definition of similarity remains elusive. Thus, the manner and effectiveness in which different systems measure similarity can vary widely.

In the case of imaged objects, there is a consensus that two images are similar if they are photographs of the same object, or at least the same type of object. But even identical objects can appear different when photographed under different conditions. Aside from

the individual peculiarities of objects, there can be slight nonessential deformations in their shape. Also, the pose of the same object can differ greatly between images and thus appear distorted from some viewpoints. A given object can appear in an infinite number of possible poses (or viewed from an infinite number of angles) in the final stored image. Therefore, for the retrieval to be effective, a database system must be able to disregard changes in the appearance caused by changing viewpoint and slight shape deformation.

The retrieval procedure (and consequently, the form of the query) are generally of two types; text-based methods, where the search is performed by examining textual annotations that accompany each image, or content-based methods, where the search is achieved by automatically extracting certain visual cues from the images itself and matching them to estimate similarity. It is this latter technique that is of interest in this paper.

In text-based methods, the query is intuitive, allowing the user to simply enter keywords or a short description of the images that needs to be retrieved. However, a human operator needs to manually prepare the textual annotations in advance, and the retrieval system is only as accurate as the ability of the human operator to predict likely keywords in future queries. In fact, there is a clear limitation as to how one can adequately and consistently describe any particular image. The automatic construction of these annotations requires that the computer have semantic understanding of the images in the database, something that is beyond the reach of the current technology. Moreover, textual annotations that accompany each image in the database must be maintained, and can easily become tedious and unmanageable for large databases.

The alternatives are content-based methods that rely on the automatic extraction of visual features, such as colors, geometric patterns, textures, and shapes. In many instances, the query is in the form of examples of the images that the user would like to retrieve, possibly even in the form of a rough sketch. This is especially useful in allowing the user to browse the database through relevance feedback, where the user can narrow down the search by identifying correct hits in intermediate results, which in turn serve as the new set of queries for another round of search.

A particularly interesting case is where the database is composed of images of the same class of objects, and thus the images appear very similar to each other. These are called homogeneous databases. Examples of homogeneous databases include scientific digital libraries such as the FishBase [15] featuring thousands of images of fish and the Perseus Digital Library [13] which features images of historical objects like vases and sculptures.

Most image retrieval systems are capable of distinguishing between different classes of objects in non-homogeneous databases, called *"inter-class"* retrieval. For example, these systems can effectively differentiate between fish and airplane images, because each one has its own distinct general shape. To query homogeneous database systems, on the other hand, requires the ability to discriminate among images at a finer level, such as distinguishing among the various species of fish. This is called *"intra-class"* retrieval. It is clearly more desirable to accommodate both types of retrieval. However, current systems that adopt global (or aggregate) features (such as histograms and low-ordered moments surveyed in [28][37]) capture only the general shape of a class and do not have enough descriptive power to distinguish among objects within a particular class. Thus,

queries such as retrieving pictures of rainbow trout (characterized by the shape of the body and fins) from an ensemble of fish images will fail with these types of systems, generating instead lists of images containing various species of fish.

Moreover, only a handful of the current systems are able to accommodate the visual distortions caused by incidental environment changes, and slight and nonessential deformations of the query object. Allowing global viewpoint change and local deformation increases the complexity of the search. Query images, though belonging to the class of interest, can appear different due to several factors, such as a change in the camera's viewpoint, or local shape deformations attributed to noise, occlusion, or simply peculiarities of the individual objects. It is this more challenging scenario (*that of intra-class retrieval, with invariance to viewpoint change and nonessential shape deformation*) that is the focus of this paper.

It is also worth noting that it is not an uncommon practice in current systems [27] to simply compare the query to each image in the database one by one, in a linear fashion, until the best matches are found. In these cases, the lack of an efficient search method is usually an inherent drawback of the algorithm. Such methods, however, are clearly unacceptable in today's explosion of available multimedia archives, where the size of the databases is becoming increasingly larger.

Hence, *efficient, invariant, intra-class* retrieval is an important tool for large multimedia archives. For example, a useful botanical image database application might be to help identify a leaf as belonging to a particular species, thus ascertain that it is not poisonous. Children visiting an aquarium might bring an image of a fish (taken from an unknown

viewpoint) and an automated search is performed to provide information about its particular species, along with other interesting facts such as the migration patterns. Other applications include searching on-line catalogs of tools, trademarks, etc., for similar designs and patterns. Many real-world applications require intra-class retrieval, and in a manner that is computationally efficient and immune to incidental environmental changes and slight deformations of the object. We address this problem in this paper. Other applications requiring texture and color analysis is addressed in a previous paper [2].

In order to achieve viewpoint invariance, the affine transform is used to model the different viewpoints associated with positioning the camera. In general, we extract representative feature points from the image (e.g., corner and inflection points from contours). The recovered affine parameters, derived from an efficient multi-dimensional indexing process, will allow a best global alignment between the query object and a few good candidate objects that has been narrowed down from the database.

The discrepancy between two object contours produced by such a global alignment process, in general, can be attributed to a number of factors: such as noise, and small local deformation (e.g., resulted from the flapping of tail and dorsal fins when a fish is swimming). To further refine the matching, we use a local matching process (based on dynamic programming) to determine the shape deformation. The cost in the match is used to indicate how much local deformation there is between the two objects. The cost metric can be designed in such a way (with domain specific knowledge) to discount nonessential or incidental local deformation.

We have conducted experiments using two different homogeneous databases (Fishbase and Squid database) with over 600 queries of varying degrees of viewpoint changes and shape deformation. The results are very encouraging. We were able to achieve over 98% correct retrieval rate at the expense of examining only a small fraction (less than 1%) of the search space. Furthermore, our technique is computationally efficient during runtime because a large percentage of the processing load (i.e., establishing hash tables for indexing) is done off-line. A profile reveals that over 90% of the processing time is spent on locally matching candidate objects, implying that the system can quickly narrow down the database to a few good candidates, then devote most of the processing time in providing a very detailed analysis of the object shapes for similarity matching; down to a level that would be impractical for most retrieval systems that perform linear search on the database. A performance comparison with the maxima of curvature scale space algorithm [26][27] was also performed.

The remainder of this paper is organized as follows: In the next section, we briefly discuss some background on content-based image indexing and retrieval systems, as well past research in using affine transform models and geometric hashing. In Sec. 3, we provide technical details of our proposed algorithm and is followed by some theoretical analysis in Sec. 4. Some experimental results follow in Sec. 5 to show the validity of the method, and finally Sec. 6 contains some concluding remarks.

2. REVIEW OF RELATED LITERATURE

The increasing popularity of the Internet has brought forth a surge in image and video archives. Aside from the World Wide Web itself, a wide array of images can be obtained from an extensive selection of huge stock photography databases, including Getty Images [17], Corel [12], the PressLink library [25], the Time Picture Archive Collection [5], and the now defunct Kodak Picture Exchange system (KPX) [22]. Specialized databases that exhibit homogeneous characteristics include the Squid [27] and the Fishbase [15] collections, which comprise thousands of images of fish, and the Perseus Digital Library [13], which features images of historical objects.

Managing these large volumes of images require a different set of tools than is used in conventional databases. To meet this challenge, a number of image retrieval systems have evolved, including those using text-based methods [22][13][15][25][5]. Many have also ventured into the more sophisticated content-based approach, including a few which have set the standard in content-based image retrieval; QBIC [14][30], Virage [1], and Photobook [33]. These systems use a combination of color, texture, and shape information to obtain the best possible match. Other systems that use a similar approach are the CANDID (Comparison Algorithm for Navigating Digital Image Databases) system of the Los Alamos National Laboratory [18], and NeTra [23]. Meanwhile, others have used a simpler approach, opting to concentrate on single features, such as color [6][31][29][38][40], texture [24][34], and shape [1][27][41][39].

When comparing images for similarity, difficulties arise when the objects undergo non-intrinsic transformations such as translation and rotation. Moreover, the camera's position

may differ, resulting in a skewed projection and zooming effect. These visual distortions create unwelcome complications that must be neutralized prior to estimating image similarity.

In many such scenarios, the affine transform is a suitable model particularly because of its ability to account for an object's rigid motion, as well as the camera's change in viewpoint and/or zoom. Using a combination of translation, rotation, scaling and shear, the affine transform is often sufficient to explain the many different ways that an object can be posed. Hence, a number of research efforts have used this model to cancel out the effects of unknown camera parameters. Mokhtarian [27] used curvature scale space to establish an affine-invariant signature along the object shape boundary by computing the position of the inflection points as Gaussian smoothing is iteratively applied to the contour. Their paper is described in more detail in Sec. 5.1. Startchik [39] employed intersections between line segments, bi-tangents and cusp tangents as a representation scheme for the object shape. Other techniques using the affine model for comparing images include [1][16][4][32]. Reiss [37] and Weiss [43] provide some discussion on the use of affine invariants.

Our approach employs an efficient, affine-invariant, multi-dimensional indexing similar to that of Califano [9], and which is a generalization of the basic geometric hashing technique by Lamdan [20]. Lamdan first incorporated geometric hashing techniques to improve indexing for efficient runtime retrieval. In this technique, visual features of the images are hashed into bins for quick retrieval during runtime while simultaneously accounting for possible transformations of the query. Consequently, preprocessing time is

sacrificed for efficient runtime. Bose [7] using triangular ratios, and Lamiroy [21] using line segment intersections, both later followed on the same technique. Wang [42] also used geometric hashing in indexing polar coordinates of the feature points, and Proctor [35] used it to recognize polyhedral objects, but these methods were not affine invariant.

3. TECHNICAL DETAILS

There are two major issues we must address: canceling out the effect of a changing viewpoint through global alignment, and obtaining accurate similarity measure through local matching.

3.1 Global alignment

To achieve viewpoint invariance, the affine transform is used to model the different viewpoints associated with positioning the camera. The affine transform model has the ability to account for an imaged object's rigid motion, as well as the camera's change in viewpoint and/or zoom. Using a combination of translation, rotation, scale, and shear, this model is often sufficient to explain the many different ways that an object can be posed (or conversely, the many different ways the camera can be positioned). It is also a generally accepted approximation to the more general perspective projection, as long as the perspective distortion is not too severe [20]. Mathematically, an affine transformation of a point $[x,y]^T$ is defined by

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \mathbf{AX} + \mathbf{T} \qquad \text{(Eq. 1)}$$

for some nonsingular matrix **A**, and a_{ij}, t_x, $t_y \in \Re$. Such a transformation preserves a number of characteristics in an image. For instance, straight lines remain straight and parallel lines remain parallel. On the other hand, many significant measurements are lost in the transformation. For example, distances are not preserved, nor are angles between line segments. This visual distortion in the image is primarily what makes it difficult to compare images with one another.

At the heart of our global alignment algorithm is a well-known geometric property on affine transform, which states that the ratio of the area of two triangles is constant under an affine transformation [37]. That is, given any two triangles Δ_1 and Δ_2,

$$\frac{\alpha(\Delta_1)}{\alpha(\Delta_2)} = \frac{\alpha(\Delta'_1)}{\alpha(\Delta'_2)} \qquad \text{(Eq. 2)}$$

where α denotes the area and Δ' is some affine transformation of Δ. Assuming that we can effectively decompose an image into a small set of representative feature points, we can use the fact that a correspondence of only three ordered points uniquely determines an affine transformation between images, thus allowing us to recover the pose of the object. However, the number of possible triplet correspondences between two sets of n points is in the order of n^6. Since only one correspondence is needed, the search space is reduced to n^3. Therefore, assuming v is the time to verify a similarity match between two images, and there are m models in the database, the total search time for a given query

will be in the order of O(n^3mv), which can be prohibitively high, especially with very large databases.

Therefore, we need to make the process efficient, achieved by employing ideas from multi-dimensional indexing [9] and geometric hashing [20] techniques that have gained popularity over the past decade. The main idea is to build a hash table that can be processed off-line without prior knowledge of the query image. A set of affine-invariant features (or signatures) (Eq.2) is extracted from each image in the database, and is used as indexes into the hash table to enter the same image code in the different bins. During the search phase, the same affine-invariant features (or signatures) are extracted from the query image, and used as indexes into the hash table, with the expectation that similar objects will index to the same set of bins in the hash table. Hence, by inspecting only relevant bins in the hash table during the search phase, and ranking the images based on the number of times they are retrieved from the hash table, huge numbers of unlikely candidates from the database are immediately filtered out, allowing for a closer inspection of only a small number of good candidates.

Since the hash table is built only once and well in advance of any search task, the time needed to construct it becomes immaterial. As will be shown in Sec. 3.1.3, however, the actual search time can be reduced to the order of n*v, if we assume a negligible number of collisions in the bins.

3.1.1 Pre-processing stage

A multi-dimensional hash table is constructed, and is filled in with encoded information from each image in the database. Each image is reduced to a set of representative feature points, which are carefully chosen such that the same set of points are highly likely to present in the query image (e.g., corner and inflection points) even with viewpoint change and shape deformation.

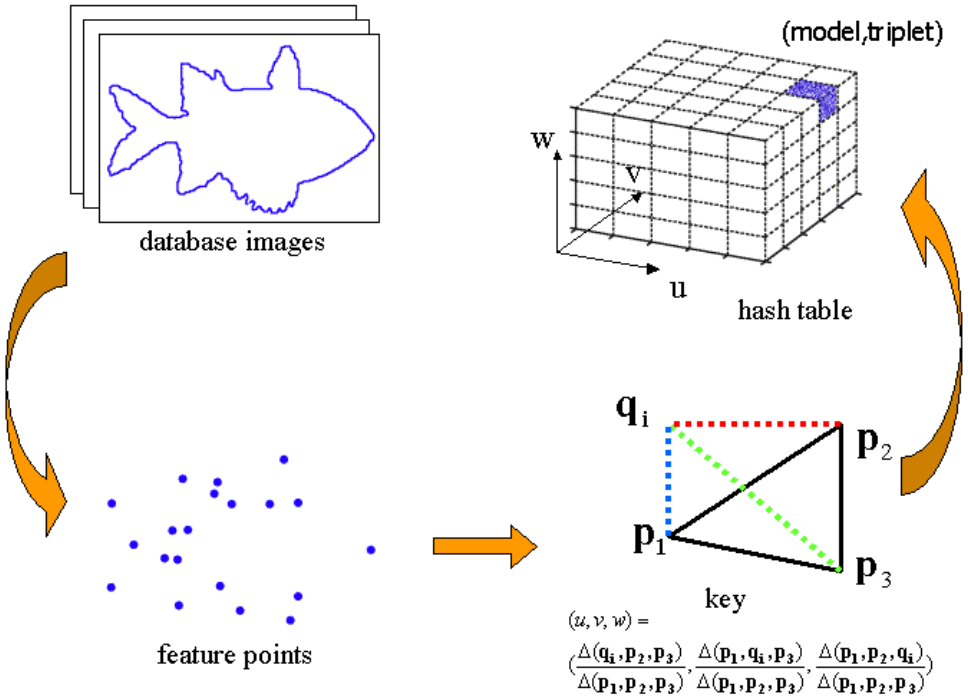


Figure 3-1. Multi-dimensional index construction

For each image M in the database represented by m feature points, and for each ordered triplet (P_1,P_2,P_3) of M,

1. Compute a set of 3-tuple index keys, described in Sec.3.1.2.
2. At the bins designated by each 3-dimensional index key, the entry $(M,(P_1,P_2,P_3))$ is added.

Figure 3-1 summarizes the hash table construction process.

3.1.2 Computing the hash index key

Given a triangle $\Delta P_1P_2P_3$ (using three feature points as vertices), we use the remaining m-3 feature points Q_i in M to compute a set of ratios that are invariant to an affine transformation (see Eq.2). We present two variations.

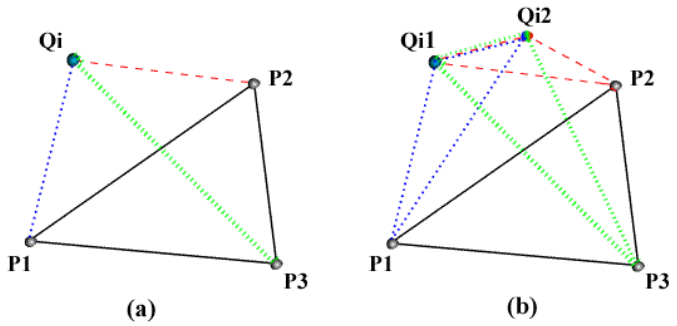


Figure 3-2. Two ways to compute the hash index key.

In Figure 3-2a, the hash index keys are computed as follows. For each point Q_i, compute the areas of $\Delta P_1P_2Q_i$, $\Delta P_1P_3Q_i$ and $\Delta P_2P_3Q_i$, which become part of the first, second, and third dimensions of a 3-dimensional key, respectively. The hash index key corresponding to point Q_i, with α denoting area, will be

$$\left(\frac{\alpha(\Delta P_1 P_2 Q_i), \alpha(\Delta P_1 P_3 Q_i), \alpha(\Delta P_2 P_3 Q_i)}{\alpha(\Delta P_1 P_2 P_3)} \right), \qquad \text{(Eq. 3)}$$

which is quantized according to a fixed bin size (discussed later in Sec. 4). This results in *m*-3, 3-tuples that are affine-invariant.

An alternate version is shown in Figure 3-2b. For each unordered pair of points Q_{i1} and Q_{i2}, we compute the areas of $\Delta P_1 Q_{i1} Q_{i2}$, $\Delta P_2 Q_{i1} Q_{i2}$ and $\Delta P_3 Q_{i1} Q_{i2}$. As before, the hash index key corresponding to the unordered pair $\{Q_{i1}, Q_{i2}\}$ will be the quantized components of

$$\left(\frac{\alpha(\Delta P_1 Q_{i1} Q_{i2}), \alpha(\Delta P_2 Q_{i1} Q_{i2}), \alpha(\Delta P_3 Q_{i1} Q_{i2})}{\alpha(\Delta P_1 P_2 P_3)} \right), \quad \text{(Eq. 4)}$$

resulting in ${}^{m\text{-}3}\mathbf{C}_2$ (order of m^2) 3-tuples that are affine-invariant.

The choice of which variation to use is completely domain-specific, the most noticeable difference being in the number of entries produced by each method.

3.1.3 Retrieval of candidates

After the off-line construction of the hash table, the system searches for a given query image, represented by *n* feature points,

1. Select an arbitrary ordered triplet (P_1, P_2, P_3) from the set of points of the query image, and compute a set of 3-tuple index keys for this triplet, as described in Sec.3.1.2.
2. Check the entries in each bin designated by the computed index keys, and accumulate the image/triplet information encoded within. To account for noise

and deformations of the object, neighboring bins within a certain radius should also be included.

3. Compute a histogram for the occurrence of each accumulated (image, triplet) pair, and select the peak values as suitable candidates for the correct image and triplet values in the database corresponding to the query image and triplet (P_1,P_2,P_3), respectively.
4. Since three ordered points uniquely determines an affine transformation between two images, recover the affine parameters $\boldsymbol{A}$ and $\boldsymbol{T}$ (Eq. 1) which transforms the candidate triplet to the triplet (P_1,P_2,P_3). Immediately exclude affine transforms that unrealistically distort the image, by using the condition number of the 2x2 matrix of $\boldsymbol{A}$ (Eq. 1), $||\boldsymbol{A}\ ||*||\boldsymbol{A}^{-1}||$, as an estimate. The condition number measures how nearly singular a matrix is.
5. Apply $\boldsymbol{A}$ and $\boldsymbol{T}$ to all feature points (and possibly the whole image itself) in the database image M to obtain M', canceling out the effect of any affine transformation.
6. Verify that M' is a suitable match for the query image, using a suitably designed similarity metric, as described in Sec. 3.2. If no (image, triplet) candidate pair is a suitable match, or if a refinement of the search results is desired, return to step 1 for another triplet.

3.2 Local matching

Since M' (from Sec. 3.1.3, step 5) and the query image are now properly aligned in the same affine coordinate frame, any visual distortion caused by a change in viewpoint,

including translation, rotation, scale, and shear, has been canceled out. The two images can thus be conveniently compared, point-by-point, using a similarity metric suitably designed for the particular domain.

Conventionally, the measure of similarity is defined to be the summation of some cost function (such as squared Euclidean distances) between matched points (with corresponding penalties for unmatched points), among all possible pairings of the points between the two contours. However, this does not have to be the only choice. If we are interested in qualitative similarity, e.g., the two contours should have similar twists-and-turns characteristics, (for example, maple leaves are not going to be identical in shape and size, but will have very similar visual characteristics that are described by a pattern grammar or a production rule for that species), then a metric measuring the number and ordering of corner and inflection points will suffice, without insisting that the corner and inflection points be at exactly the same locations. Domain specific knowledge, (e.g., downplaying the dissimilarity of fin and tail positions when a fish is swimming) can also be brought in to emphasize/de-emphasize certain characteristics. By carefully designing the cost function, nonessential local deformation can be de-emphasized.

The underlying assumption from Sec. 3.1.3, step 3, is that the correspondence of three points, (P'_1,P'_2,P'_3) to (P_1,P_2,P_3), is already known. Hence, we can break the problem down into matching the three contour segments, $P'_1P'_2$, $P'_2P'_3$, and $P'_3P'_1$ with the corresponding segments from the other contour P_1P_2, P_2P_3, and P_3P_1, respectively.

Consider matching m points s_i of segment S to n points s'_j of segment S'. The ordering of the points along the contour is assumed preserved. That is, if point s_i corresponds to s'_j,

then a point s_k can only correspond to points s'_l, for $k < i$ and $l < j$, or for $k > i$ and $l > j$. To obtain an optimal solution, a dynamic programming algorithm is used, described as follows. A table T is constructed, and filled according to the following rules:

$$T[i,0] = i * \rho$$

$$T[0,j] = j * \rho \qquad \text{(Eq. 5)}$$

$$T[i,j] = \min (T[i-1,j-1] + \delta(s_i,s_j), T[i-1,j] + \rho, T[i,j-1] + \rho)$$

where ρ is the penalty imposed for failing to match a point, and δ is the cost function between two points, typically the squared Euclidean distance. An entry T[*i,j*] represents the optimal value in matching the first *i* points in *S* with the first *j* points in *S′*. The optimal solution for the whole contour segment is thus found by computing for T[*m,n*]. As the table is filled-in diagonally from T[0,0] to T[m,n], the operation can easily be visualized as going through each contour segment, point-by-point, starting with the first pair of points. At each turn, there is choice of whether to match the two points, declare the point of *S* as unmatched, or declare the point in *S′* as unmatched. As the optimal solution is found, point correspondence for all points is easily established by tracing the decision made at each turn.

4. THEORETICAL ANALYSIS

We consider two scenarios that contribute to the failure of the proposed scheme. First, the base triangle chosen in Sec. 3.1.3, step 1 for the query image must be consistent with the

one used for the models in the construction of the hash table. Different choices of bases result in distinct recordings in the hash tables, which are not comparable. Therefore, unless there is an identical choice of bases, the scheme will result in a ***catastrophic failure***, as similarity matching will not be possible.

Since it is generally not possible to guarantee that the same base triangle will be used, the solution is in selecting *multiple* such base triangles for *database* images to construct hash tables (because this process can be performed off-line in advance and is not time critical) and use only a *single* base triangle for *query* images to ensure efficient query processing[1]. The next logical question to ask is how many such base triangles are needed? With n feature points, there are theoretically $\binom{n}{3} = O(n^3)$ choices of base triangles. However, recording all these base combinations significantly clutters a hash table and is expensive even for an off-line process. Our first analysis is thus to compute the minimum number of such base triangles needed for database images in order to guarantee with a certain confidence (a tunable parameter) that at least one of these triangles will be the one used for the query images.

Second, we examine the case of ***remediable failure***, as computing the hashing keys can also be prone to precision errors. (E.g., feature positions extracted from images are affected by the imprecision in line and corner detection and image noise.) Furthermore, the affine model is only an approximation to true camera distortion. All these induce error in the computed hash indices. Fortunately, such error is not fatal, as we can deal

[1] Note that in real experiments, more than one base triangle can be used to generate object signatures for query images– if time permits. Hence, our anlaysis is for the worst case assuming that only a single base is used. When multiple bases are used for query images as well, the chance of having the same base used for both database and query images can only increase and we can only do better than what is predicted in the theoretical anlaysis.

with them by probabilistically estimating the deviation in the index keys – given the error in feature localization. The error estimate is used in determining how an index key should be recorded. In the following, we present a more thorough analysis.

4.1 Catastrophic Failure Analysis

To avoid catastrophic failure, more than one base is chosen for computing the hash tables in the database images. In practical implementations, the number of bases used should be minimized. We sort the areas of all possible bases by size and record the signature of an image in the hash tables using only the largest k bases. For the query image, we again sort possible base triangles by size but compute the hashing key for only *the largest* base for efficiency. Catastrophic error is avoided if the largest base in the query image happens to be one of the k largest bases that were used in creating the hash tables for the database images. The question then is: in order to assert with a certain confidence c that at least one such base also exists in the query image, how many bases should we use for the database images?

There are three possible sources of error that might make a top k base absent from the query image: (1) the area becomes smaller which drops it out of the top k ranking. Since all triangular areas are scaled uniformly by an affine transform, theoretically this cannot happen, unless errors in image processing randomly displaces feature points and alters triangle size. However, we consider this unlikely if the random perturbation in feature point location is small relative to the size of a triangle, and if the base triangle is well conditioned, or equilateral, (this point is studied further in Sec. 4.2; (2) a feature point used as a vertex for a top k triangle is missing due to occlusion and image processing error; and (3) extraneous features are accidentally added in the query images.

In our application, database images are usually acquired under standard imaging conditions that allows the ingest and catalog operations to be automated. For example, to construct a database of airplane images, many books on civil and military aircrafts are available with standard front, side, and top views taken against a uniform or uncluttered background. (The above is also true for applications in botany and marine biology.) This allows the contours of the objects of interest to be extracted automatically or with the aid of standard tools such as the flood fill mask in Photoshop. Hence, we assume that database images are processed relatively free of error.

On the other hand, query images are usually taken under different lighting and viewing conditions (a situation that we address in the paper). Objects of interest can be embedded deeply in cluttered background that makes automated extraction difficult, if not impossible. Here, our intention is not to provide a foolproof, fully automated solution to this difficult segmentation problem. Instead, we enlist the help of the user to specify the object of interest. A query-by-sketch or a "human-in-the-loop" type solution with an easy-to-use graphics interface and segmentation aids are perfectly adequate and do not impose undue burden on the user. This proved to be feasible in our experiments.

With the above setup in mind, we argue that while it is possible features can be missing from query images due to occlusion (case 2 above), it is not likely that extraneous features are added (case 3 above) because the user should know what she is looking for and can correct error in segmentation not to include extraneous feature points.

Our analysis will hence focus on case 2 above. In this case, a base triangle will simply not be computed if one or more of its vertex is missing in the query image. Assuming that there are a total of n feature points to start with and any one of the features may

disappear from the query image with a probability p because of occlusion and image processing errors, the question is then how many bases (k) do we need to use to guarantee a certain level of confidence?

Denote A_i as the event that the base triangle i chosen for a database image also exists in the query image, $1 \le i \le k$. Then it is easily seen that confidence c is

$$c = P(\bigcup_{i=1}^{k} A_i) \qquad \text{(Eq. 6)}$$

Denote S_m as an event that exactly m out of the top k triangles are preserved, or $S_m = A_{i1} \cap A_{i2} \cap \cdots A_{im},\ 1 \le i1, i2, ..., im \le k$. Eq. 6 can be expanded to

$$\begin{aligned} c &= P(\bigcup_{i=1}^{k} A_i) \\ &= \sum_{i=1}^{k} P(A_i) - \sum_{i2 \ne i2} P(A_{i1} \cap A_{i2}) + \cdots + (-1)^{k+1} \bigcap_{i=1}^{k} P(A_{i1} \cap A_{i2} \cap \cdots A_{ik}) \\ &= \sum_{m=1}^{k} (-1)^{m+1} \sum_{S_m} P(A_{i1} \cap A_{i2} \cap \cdots A_{im}) \end{aligned}$$

The critical part is then to compute $P(A_{i1} \cap A_{i2} \cap \cdots A_{im})$. Define $\delta(m,s)$ as the number of ways for constructing m triangles using a total of s points. We can then compute

$$P(A_{i1} \cap A_{i2} \cap \cdots A_{im}) = \frac{1}{\binom{n}{3}^m} \sum_{s=3}^{\min(3m,n)} \binom{n}{s} \delta(m,s)(1-p)^s$$

Or that m base triangles survive in query images depends on all the vertices used to construct these m triangles are preserved $(1-p)^s$, weighed by the number of ways that these m triangles can be constructed using these vertices $\delta(m,s)$, and possible ways of choosing these vertices out of a total of n $\binom{n}{s}$ vertices , for all possible s. The

reason we divide the above expression by $\binom{n}{3}^m$ is that it is the total number of ways of selecting m triangles out of n points, and $\delta(m,s)$ can be shown to be

$$\delta(m,s) = \binom{s}{3}^m - \binom{s}{s-1}\binom{s-1}{3}^m + \cdots + (-1)^{s-3}\binom{s}{3}\binom{3}{3}^m = \sum_{j=3}^{s}(-1)^{s-j}\binom{s}{j}\binom{j}{3}^m .$$

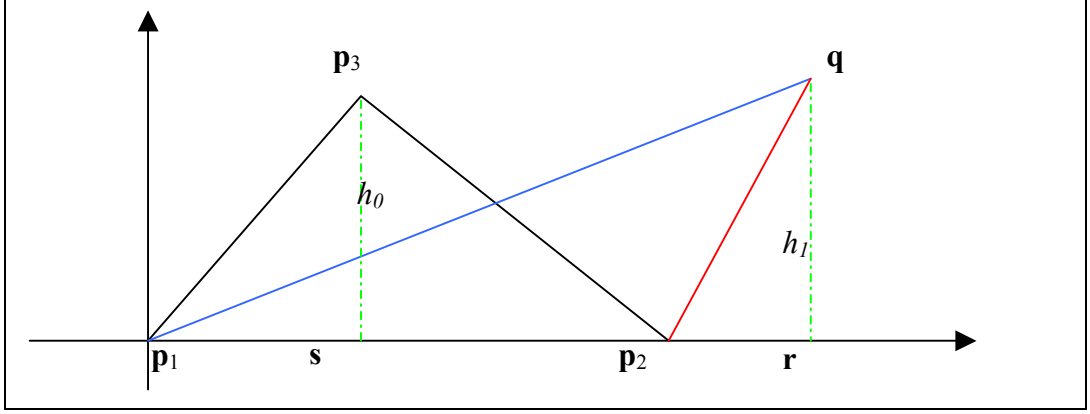
To simplify the above analysis, we assume that two base triangles can be the same. In fact, we will never select two base triangles with identical vertexes. The confidence we get above is higher than that we need. I.e., we use the above equations to estimate the number of base triangles needed given a confidence level and the result is an over-estimate of the base triangles needed. Hence, we can use the estimated value safely.

Table 1. Estimated number of bases needed

c \ ***p***	0.05		0.1		0.15		0.2	
k	***n***= 20	***n***= 30	20	30	20	30	20	30
80%	1	1	2	2	2	2	3	3
90%	2	2	3	2	3	3	4	4
95%	2	2	3	3	5	4	6	6
99%	4	3	6	5	8	7	11	9

Table 1 shows the estimated number of base triangles (k) needed with different confidence levels (c) and feature missing probability (p) for 20 and 30 (n) total feature points, respectively. As can be seen, the number of base triangles does not have to be very large.

Figure 4-1. Distribution of key values in geometric hashing



4.2 Remediable Error Analysis

The feature positions, as detected from the query image, will not confirm precisely those predicted by the affine model. The error occurs because the true image formation process is only approximately affine, line and corner detection can be imprecise, image noise can be present, etc. This kind of error will cause the index key value to perturb away from the theoretical affine prediction. Hence, instead of recording a key in just one bin in the hash table, it is also recorded in several adjacent bins, weighed by the probability that the key might be perturbed into such bins.

Assume that the error in localizing a feature in a query image is of a normal distribution centered on the predicted feature location (that predicted by the affine model) with variance σ. Recall that the key value is composed of the ratio of two triangles' areas. And

these two triangles always share one side. Without loss of generality, these two triangles can be arranged such that the shared edge aligns with the x-axis (Fig. 3). Furthermore, we can place one point of the shared edge at the origin and the other on the positive x-axis[2]. Then the key becomes the ratio of the vertical distances $\frac{|\mathbf{qr}|}{|\mathbf{p}_3\mathbf{s}|}$ of $\mathbf{q}$ and $\mathbf{p_3}$ to that common side. Furthermore, we denote the theoretical affine predicted values for $|\mathbf{p_3s}|$ as h_0 and for $|\mathbf{qr}|$ as h_1.[3] Let the coordinates of $\mathbf{p}_i$ be (x_i, y_i), $i=1,2,3$; and that of $\mathbf{q}$ (x_4, y_4), as predicted by the affine model. Then

$$\mathbf{r} = \frac{x_4}{x_2}(\mathbf{p}_2 - \mathbf{p}_1) + \mathbf{p}_1 = t_1(\mathbf{p}_2 - \mathbf{p}_1) + \mathbf{p}_1,$$

where $t_1 = \frac{x_4}{x_2}$. If we ignore the fact that t_1 is actually a random variable and use its mean position $t_1 = \frac{x_4}{x_2}$, then the error in $\mathbf{r}$ is normal with zero mean, and variance $\sigma_r = \sqrt{t_1^2 + (1-t_1)^2}\sigma$ [18]. Define $h_1' = |\mathbf{q'r'}|$. That is,

$$h_1' = \sqrt{(x_{q'} - x_{r'})^2 + (y_{q'} - y_{r'})^2}$$
$$= \sqrt{(x_q + \varepsilon_{x_q} - x_r - \varepsilon_{x_r})^2 + (y_q + \varepsilon_{y_q} - y_r - \varepsilon_{y_r})^2},$$

where $\mathbf{q}$', $\mathbf{r}$' are transformed points (*with error*) in query image, and ε_{x_q} is the deviation between x_q (affine predicted value) and x_q' (with error). And we know $x_q = x_r$, $y_q - y_r = h_1$. So

$$h_1' = \sqrt{(\varepsilon_{x_q} - \varepsilon_{x_r})^2 + (h_1 + \varepsilon_{y_q} - \varepsilon_{y_r})^2}$$
$$= (h_1 + \varepsilon_{y_q} - \varepsilon_{y_r})\sqrt{1 + \left(\frac{\varepsilon_{x_q} - \varepsilon_{x_r}}{h_1 + \varepsilon_{y_q} - \varepsilon_{y_r}}\right)^2}.$$
$$\approx h_1 + \varepsilon_{y_q} - \varepsilon_{y_r} + \frac{(\varepsilon_{x_q} - \varepsilon_{x_r})^2}{2(h_1 + \varepsilon_{y_q} - \varepsilon_{y_r})}$$

The error is then

[2] This can be accomplished by translation and rotation, which represents a rigid motion and does not change the configuration of the two triangles in anyway.

[3] In this section, we use unprimed symbols for quantities predicted based on the theoretical affine model, while primed aymbols for real, observed quanitites.in an image

$$\varepsilon_{h_1'} = \varepsilon_{y_q} - \varepsilon_{y_r} + \frac{(\varepsilon_{x_q} - \varepsilon_{x_r})^2}{2(h_1 + \varepsilon_{y_q} - \varepsilon_{y_r})} \approx \varepsilon_{y_q} - \varepsilon_{y_r} + \frac{(\varepsilon_{x_q} - \varepsilon_{x_r})^2}{2h_1}.$$

We know that if a variable has a normal distribution $x \sim N(0, \sigma^2)$, then the distribution of its square is a Gamma distribution $x^2 \sim \Gamma(\alpha = \frac{1}{2}, \lambda = \frac{1}{2\sigma^2})$, with expectation $E(x^2) = \frac{\alpha}{\lambda}$, and variance $\sigma_{x^2}^2 = \frac{\alpha}{r^2}$ [18]. Then it can be shown that the mean of |**qr**| is $\overline{h_1} = h_1 + (1 + t_1^2 + (1 - t_1)^2)\sigma^2 / 2h_1$, and the variance is $\sigma_{h_1} = (1 + t_1^2 + (1 - t_1)^2)\sigma^2 + (1 + t_1^2 + (1 - t_1)^2)^2 \sigma^4 / h_1$. We summarize the means and variances of relevant variables in Table 2.

Table 2. Variance and mean

	ε_{y_q}	ε_{y_r}	$\frac{(\varepsilon_{x_q} - \varepsilon_{x_r})^2}{2h_1}$	$\varepsilon_{h_1'}$
Mean	0	0	$\frac{(1+t_1^2+(1-t_1)^2)\sigma^2}{2h_1}$	$\frac{(1+t_1^2+(1-t_1)^2)\sigma^2}{2h_1}$
Var	σ^2	$(t_1^2+(1-t_1)^2)\sigma^2$	$\frac{(1+t_1^2+(1-t_1)^2)\sigma^4}{h_1}$	$(t_1^2+(1-t_1)^2)\sigma^2 + \frac{(1+t_1^2+(1-t_1)^2)\sigma^4}{h_1}$

Similarly, we can show that mean of |**p3s**| is $\overline{h_o} = h_o + (1 + t_o^2 + (1 - t_o)^2)\sigma^2 / 2h_o$, and the variance is $\sigma_{h_o} = (1 + t_o^2 + (1 - t_o)^2)\sigma^2 + (1 + t_o^2 + (1 - t_o)^2)^2 \sigma^4 / h_o$. The key will be put into all the bins between $\frac{\overline{h_1} - \sigma_{h_1}}{\overline{h_0} + \sigma_{h_0}}$ (min) and $\frac{\overline{h_1} + \sigma_{h_1}}{\overline{h_0} - \sigma_{h_0}}$ (max), properly weighed by their probability. The same analysis applies to all three keys. Fig.4 shows the range of keys when $\sigma_{h_0} = \sigma_{h_1} = h_0 \cdot 5\%$ (generally, $\sigma = h_0 \cdot 1\%$). As can be seen, the uncertainty of key values is relatively uniform and predictable from the plot.

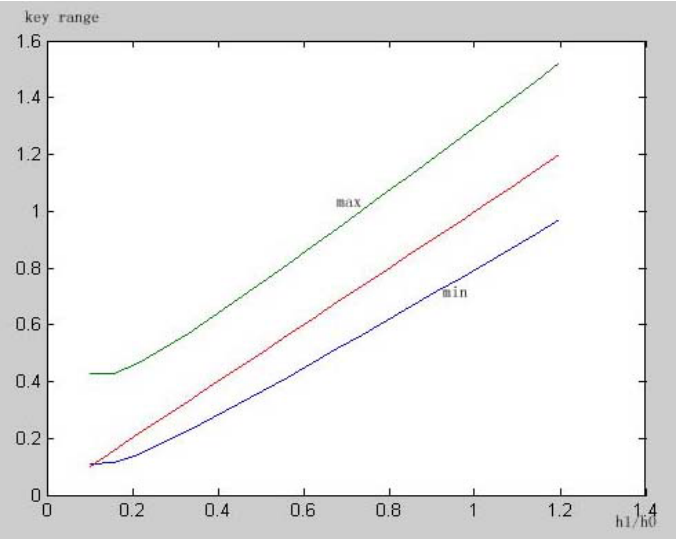


Figure 4-2. Range of keys.

5. EXPERIMENTAL RESULTS

We present results obtained from two different databases, the FishBase [15] and the SQUID [27] database, and provide an empirical analysis of querying the latter. The silhouettes of the imaged objects were extracted, and both the corner points and points of inflection along the object contour were identified as prominent landmarks. It should be noted that although these feature points are not affine invariant in a strict sense, they are likely to remain unchanged in an affine transformation with a distortion that is not too severe and the viewpoint is not incidental (e.g. any 2D pattern looked edge-on reduces to a line segment with no corner or inflection points; we use the condition number to make sure that this degeneracy does not happen). Hash index keys were computed using the second variant (Eq. 4), and used in the construction of a 4-dimensional hash table, the additional fourth dimension being a combination of the type of each point in the triplet. In particular, we classified each feature point as being one of the following six types: a

corner forming a convex/concave angle that is a member/non-member of the convex hull (a total of 4 types), and a point of inflection which goes in one direction or the another (a total of 2 types). We stored entries only for the ten largest (in terms of triangular area) triplets in each model. For query images, again we sort base triangles by size. The theoretical analysis presented in Sec. 4.1 allows us to use a single largest triangular base for query images. To improve query precision and if time permits, we can also iterate among the larger triplets in the query image in a manner similar to database images. Dynamic programming was used (Sec. 3.2) to optimally measure the amount of local deformation between the query image and the candidate images. The cost function δ (in Eq. 5) used was the squared distance between corresponding points.

As a preliminary test, search results were obtained from a small subset of the FishBase [15] database, shown in Figure 5-1. Fifty images were used, the contours of which were traced semi-automatically with the aid of an "edge-seeking" selection tool. Notice the deformation of the fins in the query image as the fish moves in water. Preliminary tests indicate that the method is adept at retrieving very similar objects.

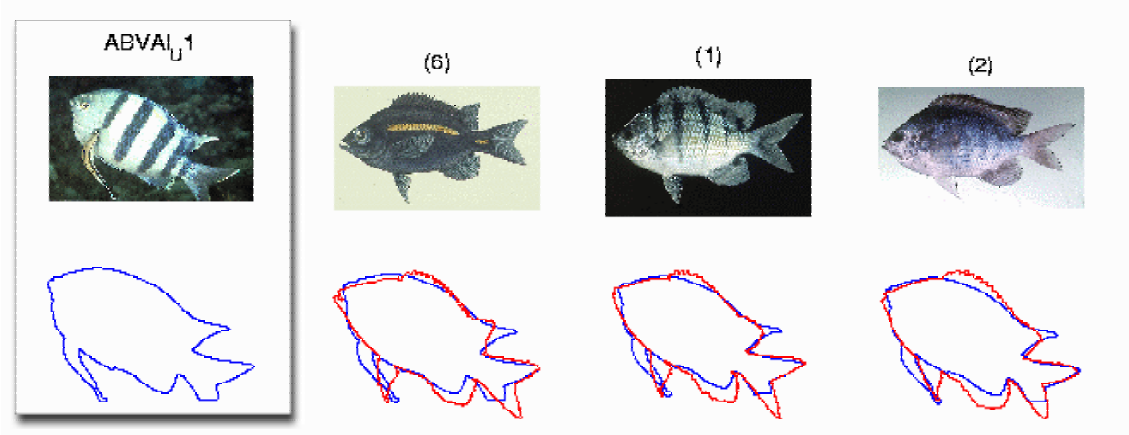


Figure 5-1. Results from querying the Fishbase database.

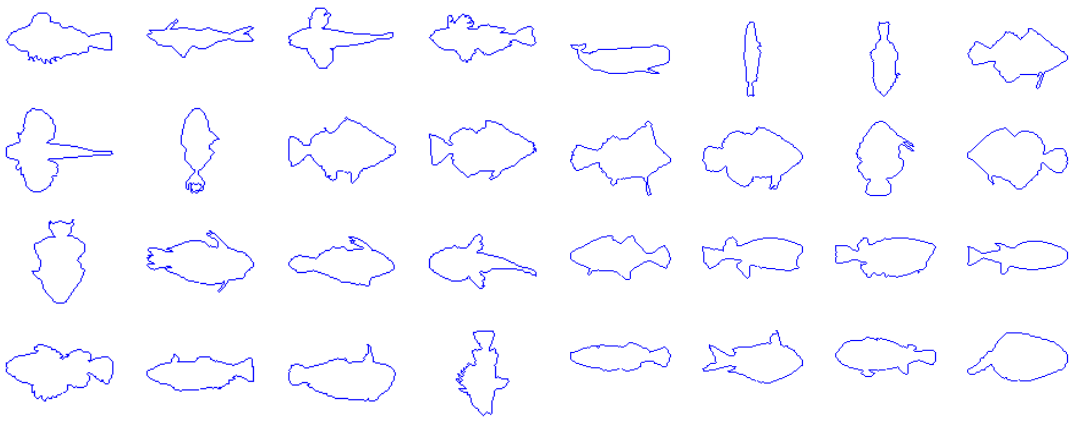


Figure 5-2. A sampling of the SQUID database, consisting of 1,100 images of marine animal contours.

We then tested the method on a larger scale. Figure 5-2 shows a sampling from the SQUID database that was used in [27] to study curvature scale space. (The next section provides a performance comparison.) The database consists of 1,100 images of marine animal contours, many of which appear very similar to each other. (Unfortunately, the actual scanned images are not provided due to copyright restrictions, so for the SQUID database, the contour *is* the image itself). Homogeneous databases such as these require painstakingly slow comparisons between the query image and the model images in order to distinguish among the many similar images in the database. Such comparisons, however, demand too much processing power when performed over each model in the entire database. The solution, therefore, is to quickly identify a few promising candidate models and perform meticulous comparisons only on this small set of hopefuls.

An analysis involving 650 queries on the SQUID database was performed. To obtain measurable results, query images were formed from the model database itself. Model images were randomly selected and applied varying levels of affine distortion (which is similar to the visual distortion caused by changing the camera's viewpoint). In order to prevent degeneracy of the image caused by severe distortion, the set of affine parameters was limited to a condition number of at most two. Furthermore, the image was re-sampled to destroy any remaining semblance of point correspondence between the model image and the query image. The preprocessing time to include 1,100 image models in the hash table is approximately 12 minutes on a 900Mhz Pentium machine. Figure 5-3 displays a histogram that reflects the retrieval performance of our method. Of 650 queries, 98% correctly found the exact model image that was used to form the distorted query, 91% of which was selected as the topmost result. Most of the 2% of the queries that failed can be attributed to distortion that was severe enough to cause the corner/inflection points to be displaced by a large amount or even cause certain feature points to be missed or added.

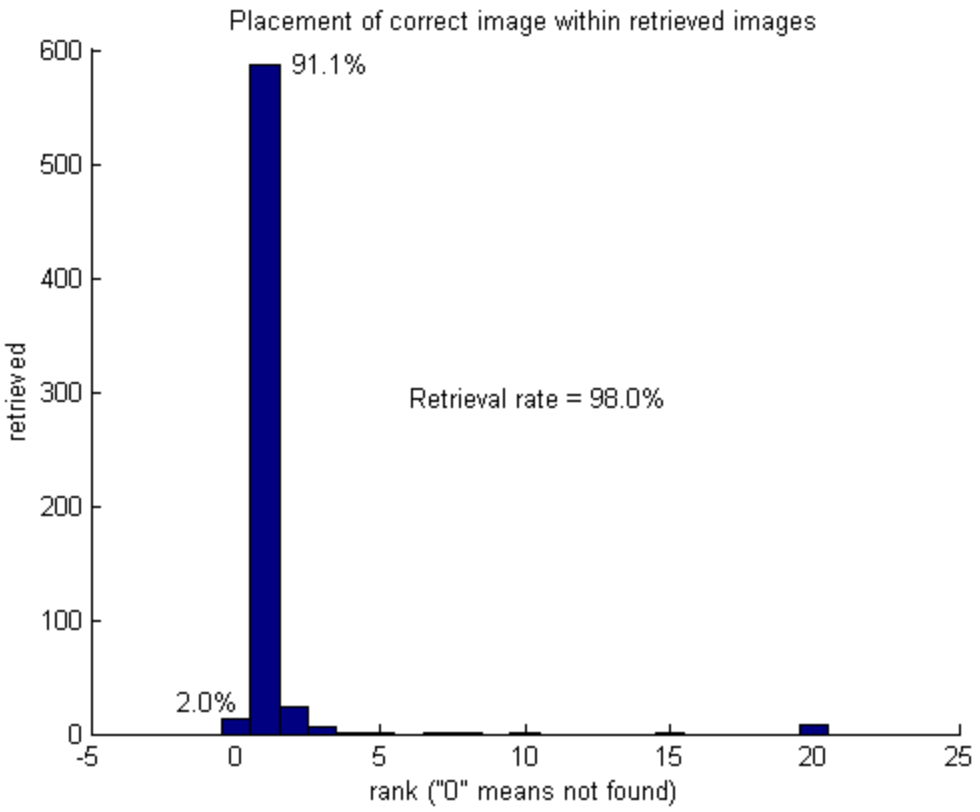


Figure 5-3. A histogram of how the correct original image was ranked among the retrieved images. Out of 650 queries, 98% succeeded in finding the original image, 91.1% of which was ranked 1st in the results.

Some results showing the query and the retrieved images are shown starting in Figure 5-4. The boxed images on the top-left are the query images. (Unfortunately, the actual scanned images are unavailable). The top nine search results are shown, arranged in row-major, with the first one being the most similar to the query image. The upper sub-rows depict the retrieved images from the database, identified by the image number in parentheses. The lower rows show the best global alignment that was found between the query image and the corresponding database image. Such results are typical for this

database, affirming the proposed method's ability to retrieve highly similar objects, while at the same time, disregarding varying viewpoints and nonessential shape deformation.

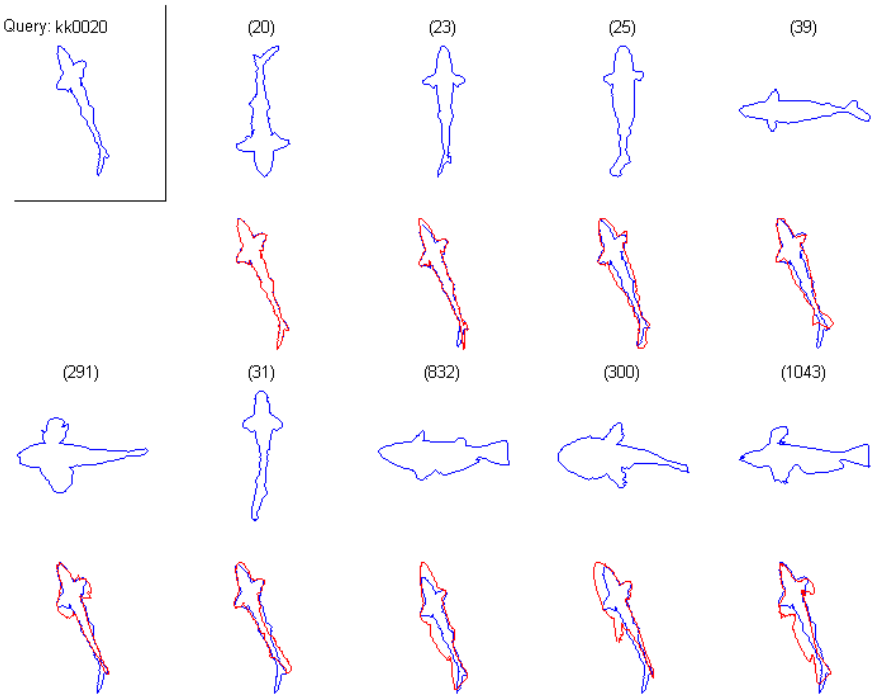


Figure 5-4. Retrieved images for query kk0020 (top-left). The upper rows are the database images; the lower rows are the computed global alignment.

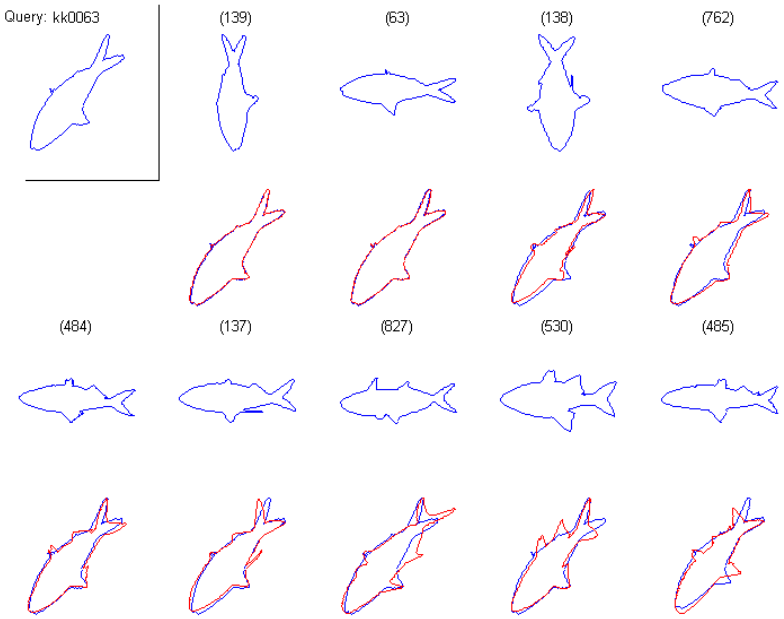


Figure 5-5. Retrieved images for query kk0063 (top-left). The upper rows are the database images; the lower rows are the computed global alignment.

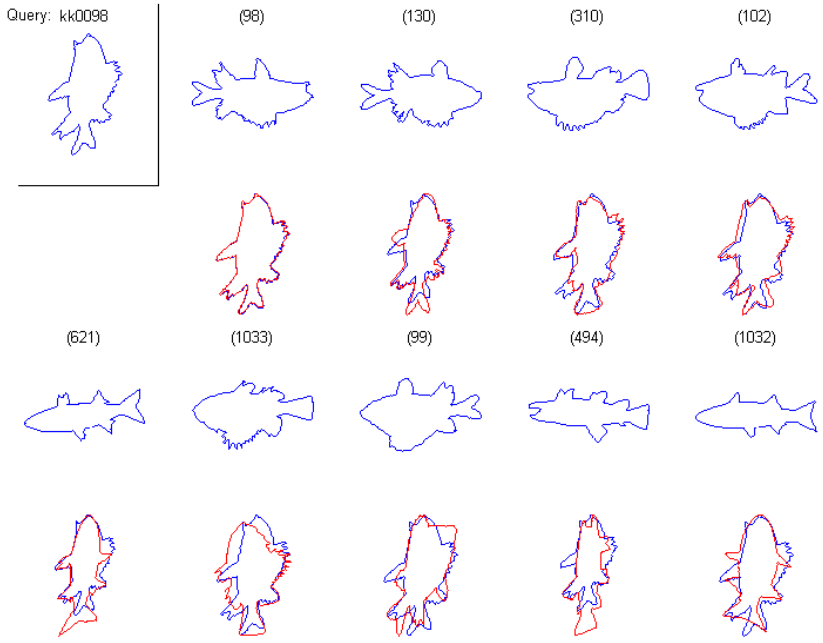


Figure 5-6. Retrieved images for query kk0098 (top-left). The upper rows are the database images; the lower rows are the computed global alignment.

Subclasses within the database were identified by a human expert to serve as ground truths for correct retrieval of relevant images. Figure 5-7 shows samples of such groups that were established. Recall and precision measurements were computed on the same 650 queries on the SQUID database. *Recall* is computed as the number of relevant images retrieved divided by the actual number of relevant images in the database. *Precision* is computed as the number of relevant images retrieved divided by the total number of images retrieved. Figure 5-8 graphically shows the relationship between recall and precision on this set of queries.

One of the attractive features of our method is that only a small portion of the entire search space is ever inspected. Figure 5-9 shows the retrieval rate, precision, and recall with respect to the amount of search space covered. It can be seen that even with only 0.001 % of the entire search space examined, the retrieval rate is already over 90%. The relative decline in precision is a clear indication that it is able to get most of the correct relevant matches early on, and only with the need to increase the number of relevant matches is it forced to be more permissive in allowing false positives in later stages.

Figure 5-10 shows the retrieval rate, precision, and recall with respect to time elapsed, as conducted on a 900Mhz Pentium machine. A profile of the processing time indicates that over 90% of the processing is spent on the detailed local matching that is performed after the two images have been aligned. As expected therefore, the curves for the retrieval rate, precision, and recall, exhibit the same behavior as in Figure 5-9. That is, its processing time is directly proportional to the size of examined search space, and that the time for identifying the potential candidates for verification (i.e., accessing the hash table) is

constant or negligible. As affirmed by the time profile of the processes, the scheme facilitates the quick identification of potential candidates so that a slow, careful, and detailed analysis can be performed on only these few candidates.

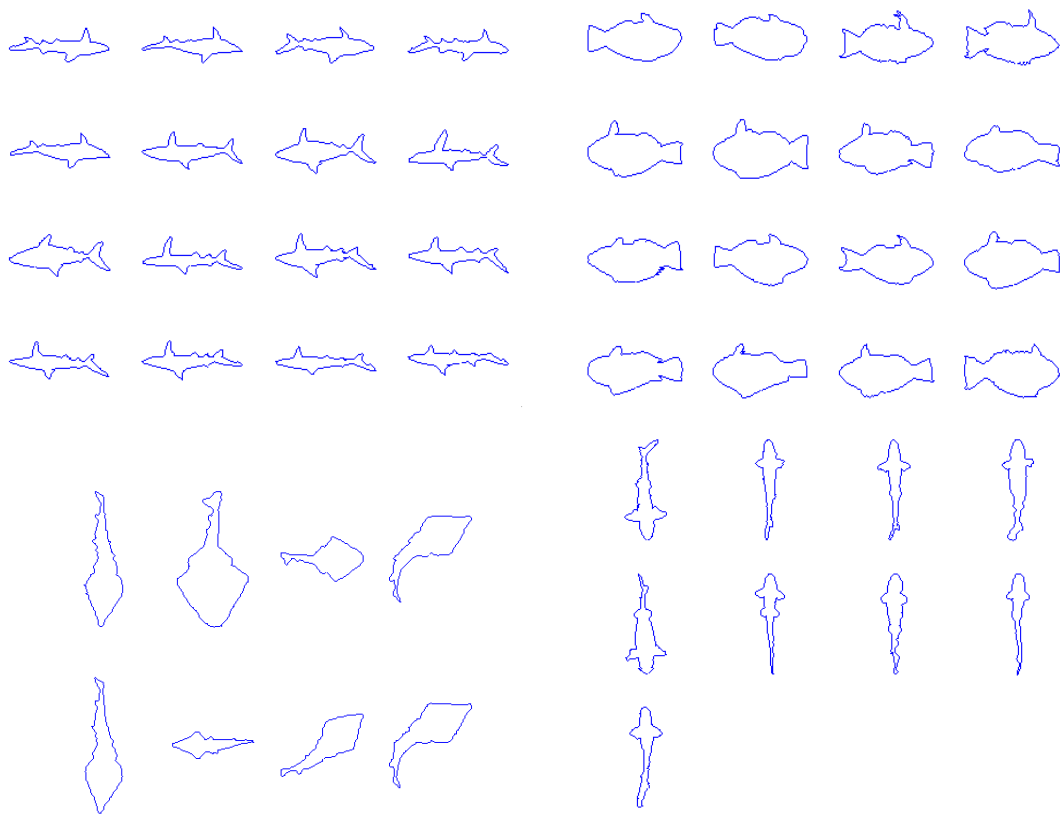


Figure 5-7. A sample of groupings that were identified in the SQUID database.

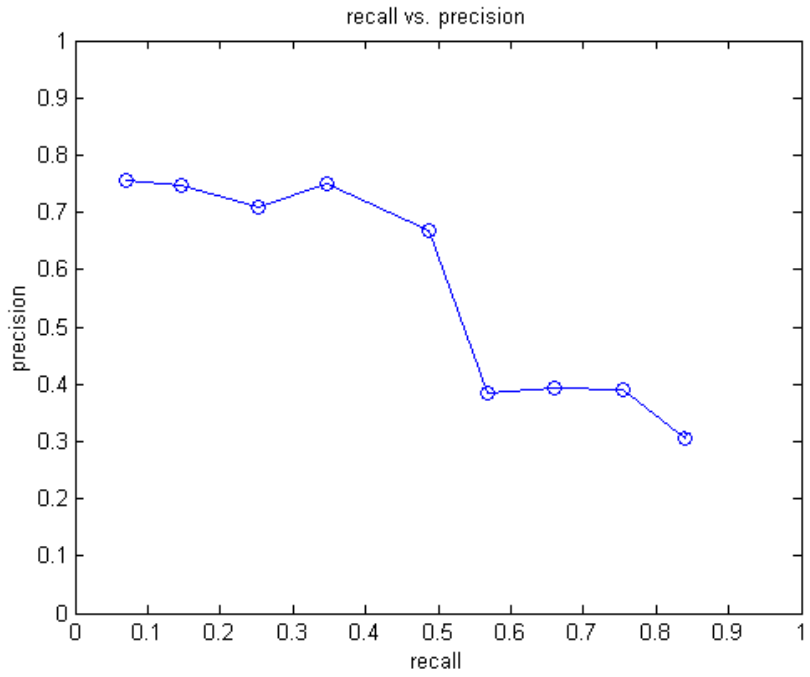


Figure 5-8. Recall versus precision, out of 650 queries.

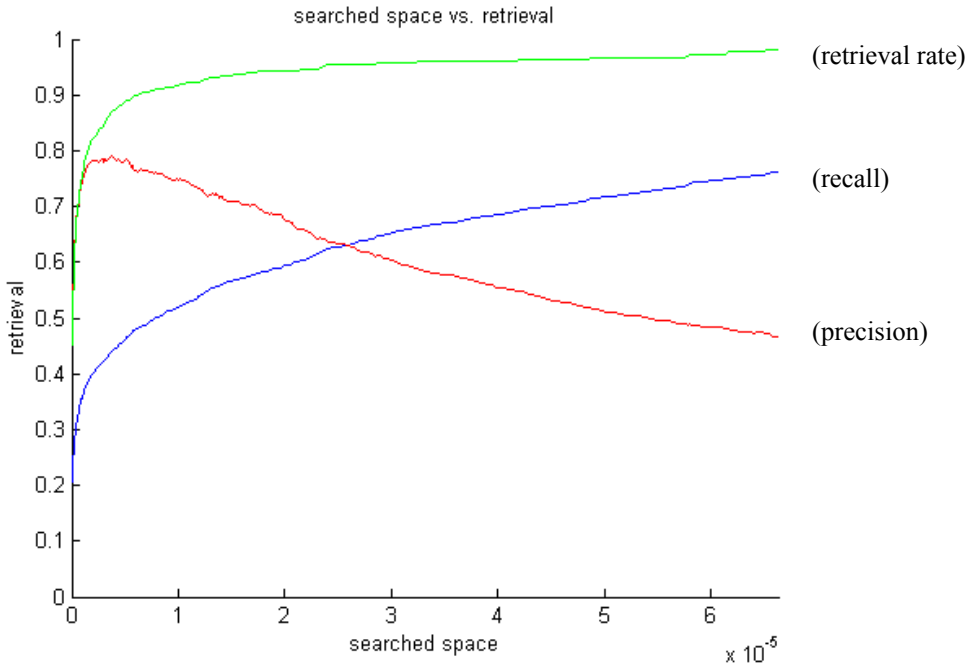


Figure 5-9. Retrieval rates (green), recall (blue), and precision (red), with respect to the percentage of the search space that was examined.

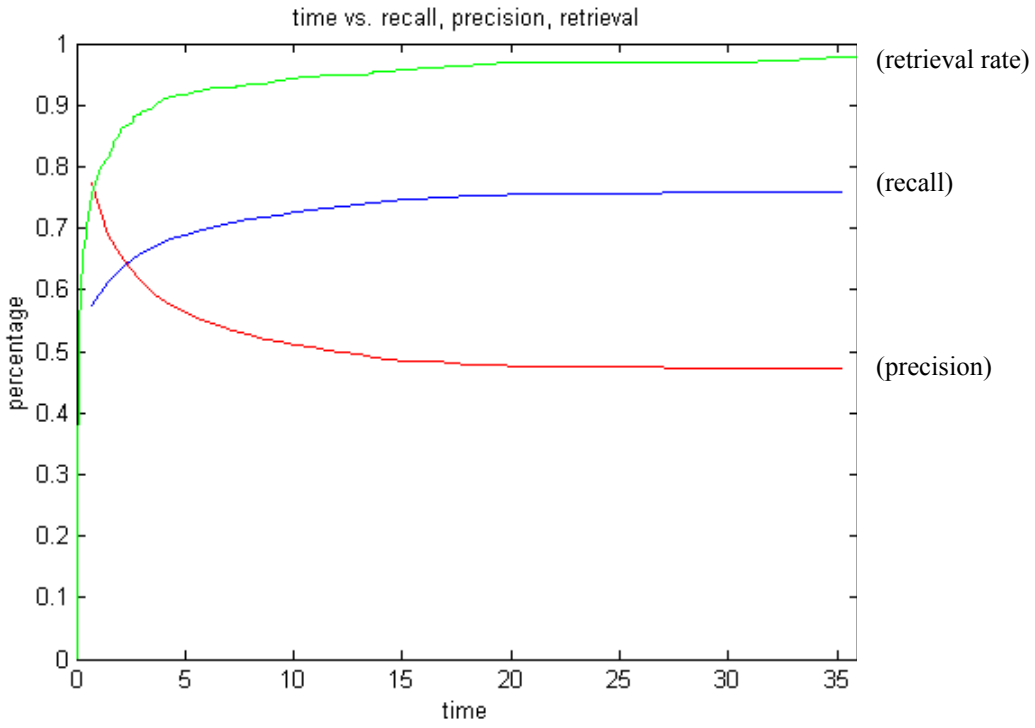


Figure 5-10. Retrieval rates (green), recall (blue), and precision (red), with respect to processing time (in seconds).

Note that even though contours were used to represent these objects and select the feature points, and shape similarity metrics were used to verify the match, it was not necessarily the case. The feature points could very well have been selected from the raw image itself, and a different image similarity metric used, such as one that uses color/texture information. The decision to rely on contours in this case was based on available data, and its suitability to this domain.

5.1 Comparison to the CSS algorithm

We compared the performance of our method to the Maxima of Curvature Scale Space (CSS) algorithm proposed by Mokhtarian, et.al. [27] which has been selected for

MPEG-7 standardization. Briefly, the CSS image of a curve is represented as a binary image on a (u, σ) plane, where u is an approximation of the normalized (affine) arc length and σ is the width of the Gaussian kernel used to smoothen the curve. The CSS image is formed by smoothing the curve with increasing values of σ, and plotting the points corresponding to the locations of zero crossings (inflection points) on the curve. Every image in the database is then represented with the locations of its major CSS contour maxima, which is affine invariant. A comparison with Fourier descriptors and moment invariants is also provided in [26].

In order to compare performance results, we tested our method using the same experimental parameters and methodology described in the CSS paper [26]. Using the same dataset of image contours, 500 images were used to generate a database of 5000 objects; Each of the original 500 images was transformed by rotating the image between 20˚ and 180˚ with 20˚ intervals (forming 9 new images each, for a total of 5000 images), and deforming each using the shearing matrix $\begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix}$, where k is the shear ratio. The experiments were conducted on three different databases using shear ratio values k = 1.0, 2.0, and 3.0 (see Figure 5-11), with the original 500 images as input queries. Furthermore, the images were re-sampled along its contour to remove any point correspondence from the transformation.

The success rate for a query is defined as $m/m_{\max}$ x 100, where m is the number of outputs which are the actual transformed versions of the input, and $m_{\max}$ is the maximum possible value of m. The success rate is observed for the first n outputs, ranging from 2 to 40, and in each case, the average success rate of the system for all 500 inputs queries is computed.

Figure 5-11. Distortion on the image (from left to right) using a shear ratio of k = 0,1,2, and 3.

The high value of the shear ratio that was used in the experiment extremely distorts the image, up to a point that corner and inflection points no longer become reliable features. Corner and inflection points are resistant to affine transformations, but are not completely invariant, since angles and distance are not preserved. Slight changes in the contour can easily be confused with noise. Hence, a more appropriate feature point selection is needed.

In this experiment, we use the convex hull to serve as feature points since it is affine invariant and is unaffected by changes in angles and distance. The convex hull remains the same regardless of the level of distortion in the image.

Figure 5-12 shows the result of the CSS algorithm on the three databases, using affine length parameterization. For the first ten outputs, the success rate dips to 98.3% for k=1.0 and down to 97% for k = 3.0. In contrast, our method performed flawlessly, with a 100% success rate regardless of the shear ratio, k, or the number of observed ouputs n.

The advantage of using our method lies in its flexibility in allowing for the selection of appropriate features depending on the conditions, such as the convex hull, which is not possible with the CSS algorithm. Furthermore, the features need not be dictated by the parameterization of the contour, thus making the method applicable to a wider range of databases. The CSS algorithm requires that the boundary shape of the object in the image is known. In our method, the same restriction is preferred, but not required, in order for

the system to function. Hence, our method can work not just on databases of object contours, but also on any image as long as feature points of interest can be located in the image. The only consequence will be a larger hash table, caused by the formation of more triplet bases (see Sec. 3.1.1) that are not actually part of the object of interest (e.g., the background).

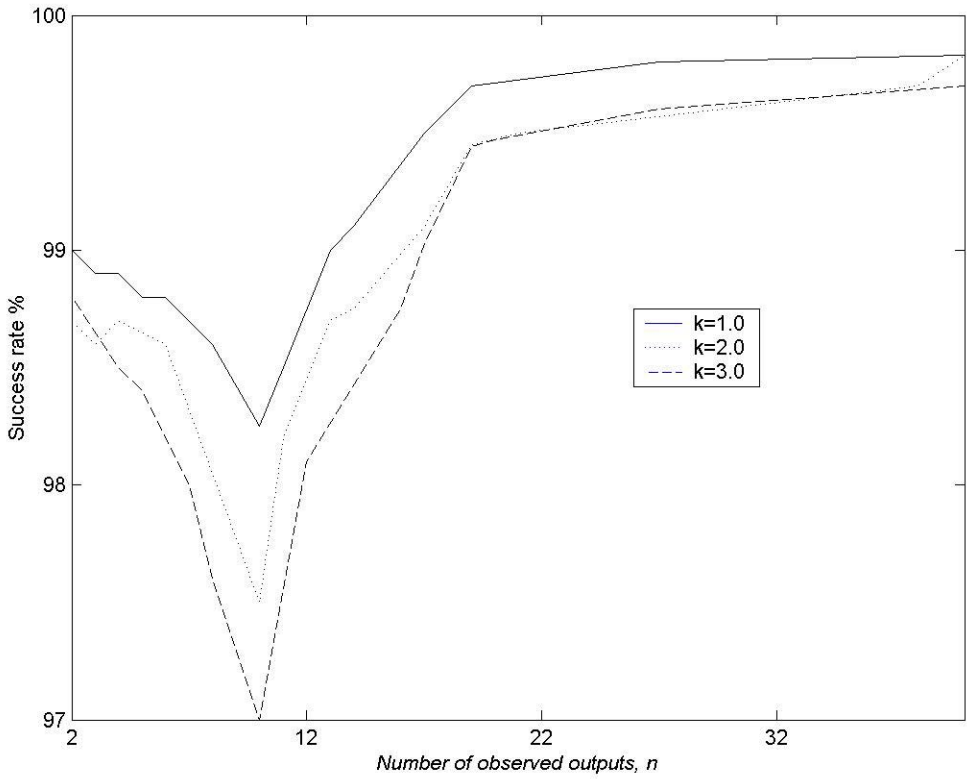


Figure 5-12. Results of the CSS algorithm, displaying success rates. In comparison, our proposed method, using the same experimental parameters, showed 100% success rates regardless of the values for *n* and *k*.

6. CONCLUSION

We presented a method for efficiently retrieving images from a homogeneous database. Aside from being able to discriminate among images at a very fine level, the technique is insensitive to changes in viewpoint and slight, nonessential shape deformation. A theoretical analysis of the failure modes was examined, and the method's performance was compared with a similar and popular algorithm. Experiments show promising results, validating the method's ability to handle such types of scenarios.

7. REFERENCES

[1] S.L. Adler, R. Krishnan, "Similarity and Affine Normalization of Partially Occluded Planar Curves Using First and 2nd Derivatives", *Pattern Recognition* (31), No. 10, October 1998, pp. 1551-1556.

[2] R. Alferez and Y. F. Wang, "Geometric and Illumination Invariants for Object Recognition"**,** *IEEE Trans. Pattern Analysis and Machine Intelligence*, June 1999.

[3] J.R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Jain, and C.F. Shu, "The Virage image search engine: an open framework for image management," *Proc. IS&T/SPIE Symposium on Electronic Imaging*, vol. 2670, pp.76-87, 1996.

[4] I.A. Bachelder and S. Ullman, "Contour Matching Using Local Affine Transformations", *Computer Vision and Pattern Recognition.* 1992 (798-801).

[5] L. Bielski, "The image database of the future begins," *Advanced Imaging* 10(10): pp.26-91, 1995.

[6] E. Binaghi, I. Gagliardi, and R. Schettini, "Indexing and fuzzy logic-based retrieval of colour images," *Proc. IFIP Working Conf. on Visual Database Systems II, Elsevier Science Publishers*, pp.79-92, 1992.

[7] S.K. Bose, K.K. Biswas and S.K. Gupta, "Model based object recognition – the role of affine invariants", *Artificial Intelligence in Engineering*. 1996 (227-234).

[8] R. Brunelli, and O. Mich, *Proceedings of ICME 2000, IEEE International Conference on Multimedia and Expo*, New York City, July 30 - August 2, 2000.

[9] A. Califano, and R. Mohan, "Multidimensional Indexing for Recognizing Visual Shapes," *IEEE Trans. on Pattern Analysis and Mach. Intelligence*, vol. 16, no. 4, pp. 373-392, April 1994.

[10] M. La Cascia, and E. Ardizzone, "JACOB: just a content-based query system for video databases," In Proceedings, *ICASSP-96*, Atlanta, Georgia, 1996.

[11] S.K. Chang, C.W. Yan, D.C Dimitroff, and T. Arndt, "An intelligent image database system" *IEEE Trans. SE* 14: 681-688, 1988.

[12] Corel Corporation, Ottawa, Canada.

[13] G. Crane, Editor, The Perseus Digital Library, Tufts University, February 2001.

[14] M. Flickner, et al., "Query by image and video content: the QBIC system," *IEEE Computer*, 28(9): 23-32, 1995.

[15] R. Froese, and D. Pauly, Editors, FishBase 2000, International Center for Living Aquatic Resources Management (ICLARM), February 2001.

[16] S. Frydrychowicz, "A New Approach to Affine Transform Invariant Shape Matching", Visual Form 1991(267-274).

[17] Getty Images, Inc., Seattle, Washington.

[18] N. J. Hastings and J. B. Peacock, Statistical Distribution, John Wiley & Sons, Toronto, 1975.

[19] P.M. Kelly, and T.M. Cannon, "Efficiency issues related to probability density function comparison," *Proc. IS&T/SPIE Symposium on Electronic Imaging*, vol. 2670, pp.42-49, 1996.

[20] Y. Lamdan, J.T. Schwartz and H.J. Wolfson, "Affine Invariant Model-Based Object Recognition, *IEEE Trans. on Robotics and Automation*, vol. 6, no. 5, October 1990.

[21] B. Lamiroy and P. Gros. "Rapid object indexing and recognition using enhanced geometric hashing". *In Proc. Of the 4th European Conf. On Computer Vision.* England, April 1996. (59-70).

[22] J. Larish, "Kodak's still picture exchange for print and film use," *Advanced Imaging* 10(4): 38-39, 1995.

[23] W. Y. Ma and B. S. Manjunath, "NETRA: A Toolbox for navigating large image databases", *Proc. IEEE Intl. Conf. Image Processing (ICIP),* Santa Barbara, October 1997.

[24] B. S. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of large image data"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (Special Issue on Digital Libraries)*, Vol. 18 (8), August 1996, pp. 837-842.

[25] M. Martucci, "Digital still marketing at PressLink," *Advanced Imaging*, 10(4): 34-36, 1995.

[26] F. Mokhtarian, S. Abbasi, "Shape similarity retrieval under affine transforms," *Pattern Recognition* (special issue on Shape Representation and Similarity for Image Databases), Volume 35, Number 1, pp. 31-41, 2002.

[27] F. Mokhtarian, S. Abbasi, and J. Kittler, "Robust and Efficient Shape Indexing through Curvature Scale Space," *Proc. 6th Brit. Mach. Visual Conf.*, pp 53-62, September 1996.

[28] J. Mundy, and A. Zisserman, Editors, *Geometric Invariance in Computer Vision*, MIT Press, Cambridge, MA, 1992.

[29] A. Nagasaka, and Y. Tanaka, "Automatic video indexing and full-video search for object appearances," *Visual Database System, II, IFIP*, Elsevier Science Publishers, pp.113-127, 1992.

[30] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin, "The QBIC project: Querying images by content using color, texture and shape," *Proc. SPIE*, 1908:173-187, 1993.

[31] V.E. Ogle, and M. Stonebraker, "Chabot: retrieval from a relational database of images," *IEEE Computer*, 28:40-48, 1995.

[32] E.J. Pauwels, T. Moons, L.J Van Gool, P. Kempenaers, A. Oosterlinck, "Recognition Of Planar Shapes Under Affine Distortion", *International Journal of Computer Vision* (14), No. 1, January 1995, pp. 49-65.

[33] A. Pentland, R.W. Picard, and S. Sclaroff, "Photobook: tools for content-based manipulation of image databases," *Proc. SPIE*, vol.2185, pp.34-47, 1994.

[34] R.W. Picard, T. Kabir, and F. Liu, "Real-time recognition with the entire Brodatz texture database," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, NY, pp.638-639, 1993.

[35] S. Proctor and J. Illingworth. "ForeSight: Fast object recognition using geometric hashing with edge-triple features." *Inter. Conf. On Image Processing.* 1997.

[36] F. Rabitti, and P. Stanchev, "GRIM-DBMS: a graphical image database management system," *Proc. IFIP Working Conf. on Visual Database Systems*, pp.415-430, 1989.

[37] T. Reiss, *Recognizing Planar Objects Using Invariant Image Features*, Springer-Verlag, Berlin, 1993.

[38] J.R. Smith and S.F. Chang. "Visualseek: a fully automated content-based image query system". *In Proceedings of ACM Multimedia 96*, pages 87--98, Boston MA USA, 1996.

[39] S. Startchik, R. Milanese, C. Rauber and T. Pun. “Planar shape databases with affine invariant search”. *First IAPR Int. Workshop on Image Databases and Multi-Media Search (IDB-MMS'96)* August 1996, Amsterdam, NL, 202-209.

[40] M.J. Swain, “Interactive indexing into image database,” *Proc. SPIE*, vol.1908, pp.193-197, 1993.

[41] K. Tanabe, and J. Ohya, “A similarity retrieval method for line drawing image database,” *Progress in Image Analysis and Processing*, Chichester: Wiley, pp.138-146, 1989.

[42] J. Wang, W. Chang and R. Acharya. “Efficient and Effective Similar Shape Retrieval”. *International Conference on Multimedia Computing and System. Vol.1* 1999: 875-879.

[43] I. Weiss. “Geometric Invariants and Object Recognition”. *International Journal of Computer Vision.* 1993. (10) 3: 207-231.