Efficient Affine-Invariant Similarity Retrieval[*]

Sofie Haesevoets Bart Kuijpers
Hasselt University, Transnational University of Limburg

Peter Revesz
University of Nebraska-Lincoln

Abstract

We describe novel image similarity measures that are affine-invariant. Based on these new similarity measures, we further describe efficient affine-invariant image indexing algorithms. We present also our experiments on a preliminary data set with 400 *images.*

1 Introduction

Different images of the same object are *affine-invariant* transformations of each other under the so-called *weak perspective assumption* [20]. Hence recognizing that a new image and a stored image show the same object requires an *affine-invariant similarity measure* between pairs of images. Computer vision researchers already proposed several affine-invariant similarity measures between pairs of pictures, for example the *minimum Hausdorff distance measure* [5], the *geometric hashing* [25] technique, and least squares distance-based similarity measures [12]. However, none of these measures leads to efficient indexing.

Database researchers also already proposed some efficient image indexing methods based on the properties such as *shape* [13], *color histograms* [22, 23], *attributed relational graphs* [18], and *image compression coefficients* [1, 21]. However, none of these properties is affine-invariant.

We propose, for the first time, affine-invariant similarity measures that allow efficient indexing. Both of our measures extract shape features of an image in the same way, but they differ regarding color feature extraction. For color feature extraction, we propose a novel measure called *primary color ratio measure*. This measure computes the ratios of the sums of the red, respectively green and blue color values of all image pixels. We also implement the *rainbow color ratio measure* first proposed in [19]. Here, N colors are defined a priori, and the ratios of the areas of the image occupied by each color are computed. Both measures use Euclidean distance between vectors of ratios to compare images. For shape feature extraction, we propose an affine triangulation method for spatial data. This method is based on the computation of barycentric points in convex polygons obtained by the so-called *sketch* of the image. A spatial triangulation method is called *affine-invariant*, if whenever it is applied to two spatial figures A and B (e.g., the two stars in Figure 1) that can be mapped to each other by an affinity α of the plane, also the resulting triangulations can be mapped to each other by that same affinity α (see the triangulation in Figure 1). In the remainder, we always consider affinities of the plane.

The outline of this article is as follows. In Section 2 we describe two methods for color feature extraction. We propose an affine-invariant triangulation in Section 3. In Section 4 we propose a comparison measure for pictures combining both color and shape information. The experimental results are explained in Section 5. Section 6 gives a conclusion and discusses further work.

2 Affine-invariant color feature extraction

We start with the description of two methods for color feature extraction. The *primary color ratio measure* computes the ratios of the sum of the red, respectively green and blue color values of all pixels, whereas the *rainbow color ratio measure* defines N colors a priori and computes the ratios of the areas of the picture filled with each color. We show that both measures are affine-invariant. We assume that each image is represented as a set of colored points. Most pixel-based image representations allow 256 different shades of green, red and blue.

[*]This research was partially funded by the European Union under the FP6-IST-FET programme,Project n. FP6-14915, GeoPKDD: Geographic Privacy-Aware Knowledege Discovery and Delivery, and by the Research Foundation Flanders (FWO-Vlaanderen), Research ProjectG.0344.05. The third author was also supported by a Humboldt Research Fellowship from the Alexander von Humboldt Foundation.

2.1 The primary color ratio similarity measure

Let R_A (respectively G_A, B_A) be the sum of all red (respectively green, blue) color values of the pixels of an image A. Then the ratios $\frac{R_A}{G_A}$ and $\frac{B_A}{G_A}$ (if $G_A \neq 0$)[1] are independent of each other. Hence, we can describe each image A as a two-dimensional vector $V_A = (\frac{R_A}{G_A}, \frac{B_A}{G_A})$, which we call the *primary color ratio vector*.

We now prove that the primary color ratios are affine-invariant.

Property 2.1 Let R_A, G_A and B_A be the sums of the red, green and blue pixel values of image A. Let α be an affinity. Then $R_{\alpha(A)} = |\alpha|R_A$, $G_{\alpha(A)} = |\alpha|G_A$ and $B_{\alpha(A)} = |\alpha|B_A$, where $|\alpha|$ is the determinant of the transformation matrix of α. $\square$

Property 2.2 Let R_A, G_A and B_A the sums of the red, green and blue pixel values of image A. Let α be an affinity. Then all ratios of two of these values are affine-invariant. $\square$

Definition 2.1 Let V_A and V_B be the primary color ratio vectors of images A and B, respectively. Then the *primary color ratio similarity measure* defines the distance between A and B as the distance between the two-dimensional vectors V_A and V_B. $\square$

Remark that any distance measure can be used. In our implementation we use the Euclidean distance between the primary color ratio vectors.

2.2 The rainbow color ratio similarity measure

The rainbow color ratio similarity measure was first introduced in Chapter 21 of [19]. From all possible red, green and blue values of the image pixels, a large set of colors of the rainbow can be defined. Suppose we classify each part of an image as having one of N different colors. The ratio of the total areas of any pair of colors is affine-invariant [19].

Suppose that we have a set of N different colors. We can derive $M = N - 1$ ratios that are independent of each other, e.g. by fixing one color and use it as the denominator for all ratios. We can describe each image as a vector of M values where the i^{th} entry corresponds to the value of the i^{th} rainbow color ratio.

[1]It is very unlikely that all three values R_A, G_A and B_A are zero for some image A. This would mean that the image is completely black. Therefore, we can always find one of R_A, G_A and B_A that is not zero and use this value in the denominator for computing the ratios. We assume that each image has some green in it.

The *rainbow color ratio similarity measure* is defined as follows [19]:

Definition 2.2 Let V_A and V_B be the vectors of the rainbow color ratios in images A and B, respectively. Then the distance between A and B is defined as the Euclidean distance between V_A and V_B. $\square$

The primary color ratio measure treats the red, green and blue color values independently, but is a very global method. The rainbow color ratio measure results in a higher-dimensional feature vector, hence the vectors are possibly more sparsely distributed. However, the definition of colors out of their red, green and blue values implies that they are not completely independent from each other. In Section 5.1 we compare both methods experimentally.

2.3 Indexing feature vectors

The primary color ratio measure and rainbow color ratio measures described above construct a feature vector from an image. There exist several efficient techniques for indexing multidimensional points. In general, spatial access methods (SAM) can be divided into two classes: data-driven and space-driven. Examples of data-driven techniques are $R-$trees [9], R^*-trees [2] and Hilbert-$R-$trees [14]. Those are all based on a hierarchy of minimum bounding rectangles (MBRs). Quad trees [7] are an example of a space driven indexing technique. Quad trees, in the case of two-dimensional points, recursively partition the plane into four equal parts, until each part only contains one element. Quad trees are generally used when the feature vectors are two-dimensional, but they can be extended for higher dimensions too.

3 Affine-invariant triangulation of spatial data

We believe that for invariant image retrieval not only color information is relevant. Therefore, we need a technique for affine-invariant shape matching. For this purpose, we introduce a novel affine-invariant triangulation method. If we want to test whether an image is in the database, or is an affine distortion of an other image in the database, we triangulate the image, and look for images that have the same (up to a threshold) triangulation.

We assume a figure to be a union of (possibly overlapping) simple polygons. A simple polygon is a region enclosed by a single, closed polygonal line that does not intersect itself, i.e. it contains no holes. We allow polygons to degenerate into line segments or points.

We define the *sketch* of a figure as the collection of boundary line segments of that figure. The sketch of an image can be computed in time $O(n^2 logn)$ using the well-known *plane sweep technique*, as described in [15]. Figure 1 shows at the left a star-shaped image which is the union of four triangles (the triangles are indicated with different colors). The center image is the triangulation of the sketch of the star-shape.

Definition 3.1 Let A be an image. A collection of triangles[2] $\{T_1, \ldots, T_m\}$ in $\mathbf{R}^2$ is a *triangulation of* A if the *interiors* of different T_i are disjoint and if the union $\cup_{i=1}^{m} T_i$ equals A.

The interior of a triangle is defined to be its topological interior. The interior of a line segment is defined to be the segment without endpoints, and the interior of a point is defined to be the point itself. $\square$

In Figure 1, two stars with their respective triangulations are shown. Triangulations, as defined here, basically *partition* images. Indeed, a triangulation of an image is a partition of this image into triangles, but this is not a partition in the mathematical sense, as the elements of the partition may have common boundaries, but their topological interiors do not intersect. For spatial data, it is customary to allow the elements of a partition to share boundaries (see for example [6]).

A *spatial triangulation method* is a procedure (or function) that on input of an image produces a triangulation of this spatial image.

Next, we define what it means for such methods to be affine-invariant.

Definition 3.2 A spatial triangulation method $\mathcal{M}_S$ is called *affine-invariant* if and only if for any two images A and B, for which there is an affinity $\alpha : \mathbf{R}^2 \to \mathbf{R}^2$ such that $\alpha(A) = B$, also $\alpha(\mathcal{M}_S(A)) = \mathcal{M}_S(B)$. $\square$

3.1 An affine-invariant triangulation method

Next we describe an affine-invariant triangulation method AT_S for images and prove its correctness, improving the triangulation method of [16]. Our algorithm is based on the fact that any set of lines partitions the plane in a set of convex polygons. This partition is affine-invariant.

Given an image, we compute first the sketch of the image. Then we compute the intersection points of all lines through one of the boundary segment in the sketch. We keep only those intersection points that are part of the original image. By connecting all pairs of consecutive intersection points on each boundary segment, we obtain a set of line segments isolating a set of convex polygons in the plane.

Each polygon is either part of at least one polygon of the original image, or part of the convex closure of it. Each line segment belongs to at most two polygons. If we traverse each polygon in an *a priori* chosen direction, each line segment is only traversed once in each direction. After we find all polygons, we triangulate those that are not triangles yet by connecting their center of mass to each corner point. Figure 1 shows in the center the triangulation of a star shape.

We now give the algorithm more formally. In the following, we will denote points in the plane by bold letters $\mathbf{p}, \mathbf{q}, \ldots$ and vectors between two points in the plane by $\overrightarrow{\mathbf{pq}}, \overrightarrow{\mathbf{qr}}, \ldots$. We will denote the length of the vector $\overrightarrow{\mathbf{pq}}$ by $|\overrightarrow{\mathbf{pq}}|$. The line segment between the points $\mathbf{p}$ and $\mathbf{q}$ will be represented $\overline{\mathbf{pq}}$. The line through the points $\mathbf{p}$ and $\mathbf{q}$ will be denoted $L_{\mathbf{p},\mathbf{q}}$. We sometimes refer to $L_{\mathbf{p},\mathbf{q}}$ as the *carrier* of the line segment $\overline{\mathbf{pq}}$ or the vector $\overrightarrow{\mathbf{pq}}$. Real numbers that represent coordinates of points in the plane will have index 1 (resp. 2) if they represent the first (resp. second) spatial coordinate of a point with respect to the standard coordinate system.

The input of this triangulation algorithm is an image A that is represented as a finite union of polygons p_i where each p_i is given by a series of corner points $\mathbf{p_{i,1}}, \mathbf{p_{i,2}}, \ldots, \mathbf{p_{i,k}}$ $(i = 1, ..., n)$, i.e., $A = \cup_{i=1}^{n} p_i$.

In the following description we leave out the special cases where input polygons are line segments or points, due to space restrictions. It only requires little adaptation to include these cases. Polygons that are degenerated to points are added to the output directly. Lines are returned after Step 5 of the algorithm.

Algorithm 3.1 *AT_S(input snapshot A):*

Let Out *be a set of triples of points. Let $\mathcal{P}$ be a set of convex polygons. A convex polygon is represented as a list of directed line segments, represented as pairs of points.*

Let A' be a list of line segments. Let $\mathcal{C}$ be a list of elements of the form $(\overrightarrow{\mathbf{pq}}, x)$. Here, x points to a list **points($\overrightarrow{\mathbf{pq}}$)***, who's elements are composed of a point and a real number. The elements of the lists* **points($\overrightarrow{\mathbf{pq}}$)** *are sorted by this real number.*

Step 0. *Choose an affine coordinate system $(\mathbf{e_0}, \mathbf{e_1}, \mathbf{e_2})$ for the plane. We define the oriented angle between the*

[2]We mean filled triangles; we allow a triangle to degenerate into a line segment or point.

vectors $\overrightarrow{\mathbf{e_0e_1}}$ and $\overrightarrow{\mathbf{e_0e_2}}$ to have a positive *orientation*[3].

$\mathcal{C} := NULL$
$A' := NULL$
$\mathcal{P} := \emptyset$
$\mathbf{Out} := \emptyset$

Step 1. *Compute the sketch of A. Store all boundary line segments of the sketch in the list A'.*

Step 2. *For each element $\overline{\mathbf{pq}}$ of A' do the following:*

$\mathcal{C} := \mathcal{C} \cup \{(\theta(\overline{\mathbf{pq}}), \emptyset)\}$, where $\theta(\overline{\mathbf{pq}})$ equals $\overrightarrow{\mathbf{pq}}$ if the oriented angle between $\overrightarrow{\mathbf{e_0e_1}}$ and $\overrightarrow{\mathbf{e_0u}}$, where $\mathbf{u} = \mathbf{q} - \mathbf{p} + \mathbf{e_0}$, is both positive and between 0 and π, and $\overrightarrow{\mathbf{qp}}$ otherwise.

Step 3. *Sort the vectors of $\mathcal{C}$ by ascending positive angle with the vector $\overrightarrow{\mathbf{e_0e_1}}$.*

Step 4. *For every pair of vectors $\overrightarrow{\mathbf{pq}}$ and $\overrightarrow{\mathbf{rs}}$ of $\mathcal{C}$, compute the intersection point $\mathbf{t}$ of $L_{\mathbf{pq}}$ and $L_{\mathbf{rs}}$.*

If $\mathbf{t}$ exists, compute the cross-ratios[4] *$c_1 = CR(\mathbf{p}, \mathbf{t}, \mathbf{q})$ and $c_2 = CR(\mathbf{r}, \mathbf{t}, \mathbf{s})$. If either c_1 or c_2 is between 0 and 1 (bounds included), then add $(\mathbf{t}, c_1)$ to $\mathbf{points}(\overrightarrow{\mathbf{pq}})$ and $(\mathbf{t}, c_2)$ to $\mathbf{points}(\overrightarrow{\mathbf{rs}})$, such that the elements of $\mathbf{points}(\overrightarrow{\mathbf{pq}})$ and $\mathbf{points}(\overrightarrow{\mathbf{rs}})$ are sorted by cross-ratio.*

Step 5. *For every pair of consecutive vectors $\overrightarrow{\mathbf{pq}}$ and $\overrightarrow{\mathbf{rs}}$ of $\mathcal{C}$ such that the points $\mathbf{p}, \mathbf{q}, \mathbf{r}, \mathbf{s}$ are all collinear, do the following:*

- *For each element $(\mathbf{t}, c)$ of $\mathbf{points}(\overrightarrow{\mathbf{rs}})$ such that there is not already an element $(\mathbf{t}, d)$ of $\mathbf{points}(\overrightarrow{\mathbf{pq}})$, add the element $(\mathbf{t}, c')$ to $\mathbf{points}(\overrightarrow{\mathbf{pq}})$, where $c' = CR(\mathbf{p}, \mathbf{t}, \mathbf{q})$ (in a sorted way).*
- *Remove the vector $\overrightarrow{\mathbf{rs}}$ from $\mathcal{C}$.*

Step 6. *Traverse the list $\mathcal{C}$. For each element $\overrightarrow{\mathbf{pq}}$ of $\mathcal{C}$, append a new element containing the vector $\overrightarrow{\mathbf{qp}}$ to the end of C. The list $\mathbf{points}(\overrightarrow{\mathbf{qp}})$ consists of all elements of $\mathbf{points}(\overrightarrow{\mathbf{pq}})$, but sorted in reverse order.*

We now define "making one pass through $\mathcal{C}$, starting from the vector $\overrightarrow{\mathbf{pq}}$ and the point $\mathbf{t}$" *as follows:*

Start with the element containing $\mathbf{t}$ *in* **points($\overrightarrow{\mathbf{pq}}$)** *and visit all later elements of* **points($\overrightarrow{\mathbf{pq}}$)**. *Then, for all further elements $\overrightarrow{\mathbf{rs}}$ of $\mathcal{C}$, visit all points in* **points($\overrightarrow{\mathbf{rs}}$)**. *If the end of $\mathcal{C}$ is reached, restart at the beginning of $\mathcal{C}$, until the element $\overrightarrow{\mathbf{pq}}$ is reached again.*

Step 7. *Now, for each element $\overrightarrow{\mathbf{pq}}$ and for each point $\mathbf{p_j}$ of the points $\mathbf{p_0}, \mathbf{p_1}, \dots, \mathbf{p_k}$ in the list* **points($\overrightarrow{\mathbf{pq}}$)** *do the following:*

Make one pass through $\mathcal{C}$, starting from $\overrightarrow{\mathbf{pq}}$ and the point $\mathbf{p_j}$. Meanwhile, try to find a cycle of directed line segments $\overline{\mathbf{p_jq_0}}, \overline{\mathbf{q_0q_1}}, \dots, \overline{\mathbf{q_lp_j}}$, where $l \geq 2$. Each such line segment $\overline{\mathbf{q_vq_w}}$ consists of a pair of consecutive points in some list **points($\overrightarrow{\mathbf{rs}}$)** *and is the last unmarked line segment starting with $\mathbf{q_v}$ and different from $\overline{\mathbf{q_vq_{v-1}}}$. If such a cycle of line segments is found, add it to the set $\mathcal{P}$, and place a mark on the first point of each line segment, to prevent it from being used more than once.*

Step 8. *For each element $Q = \{\overline{\mathbf{p_0p_1}}, \overline{\mathbf{p_1p_2}}, \dots, \overline{\mathbf{p_kp_0}}\}$ of $\mathcal{P}$, check if there exists a polygon p_i in A such that Q is a subset of p_i. If not, remove Q from $\mathcal{P}$.*

Step 9. *For each element $Q = \{\overline{\mathbf{p_0p_1}}, \overline{\mathbf{p_1p_2}}, \dots, \overline{\mathbf{p_kp_0}}\}$ of $\mathcal{P}$ do the following:*

If $k = 2$, then $\mathbf{Out} := \mathbf{Out} \cup \{(\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2})\}$, else compute the center of mass $c = \frac{1}{k}\sum_{i=1}^{k} \mathbf{p_i}$ of the points $\mathbf{p_0}, \mathbf{p_1}, \dots, \mathbf{p_k}$ and let $\mathbf{Out} := \mathbf{Out} \cup \bigcup_{i=1}^{k}\{(\mathbf{p_i}, \mathbf{c}, \mathbf{p_{(i+1)\mathit{Mod}(k+1)}})\}$. Return **Out**. □

We now state that the triangulation algorithm is correct, affine-invariant and effectively computable. The proofs are omitted due to space restrictions.

Proposition 3.1 The procedure AT_S, given in Algorithm 3.1, returns on input a snapshot A an affine-invariant triangulation of A. □

Proposition 3.2 Let A be a union of k polygons $p_1, p_2, \dots, p_k$. Let the number of corner points of polygon p_i be n_i. Define n as the sum $n_1 + n_2 + \dots + n_k$. The procedure AT_S, given in algorithm 3.1, returns on input A, a set of $O(n^6)$ triangles in $O(n^6)$ time. □

We introduced a novel affine-invariant triangulation method. In Section 4, we investigate the possibility of using this technique for shape matching. Indeed, two images containing the same shape, up to an affinity, will have the same triangulations.

Now, we propose a distance measure for images that combines color and shape information.

[3]The orientation of the angle between the vectors $\overrightarrow{\mathbf{pq}}$ and $\overrightarrow{\mathbf{pr}}$ can be determined by computing the *sign* of the expression $q_1r_2 - q_2r_1 + p_2r_1 - p_1r_2 + p_1q_2 - p_2q_1$. The function *sign* maps real numbers to the set $\{+1, -1, 0\}$.

[4]The cross ratio $CR(\mathbf{p}, \mathbf{q}, \mathbf{r})$ of three collinear points $\mathbf{p}$, $\mathbf{q}$, $\mathbf{r}$ is $\frac{|\overrightarrow{\mathbf{pq}}|}{|\overrightarrow{\mathbf{pr}}|}$.

4 Affine-invariant recognition of images using combined color and shape information

In previous sections, we introduced an affine-invariant comparison method for both color and shape information of an image. We now propose a two-level indexing structure based on both methods. First, a color ratio measure is used to group all images with comparable color schemes (up to some threshold). Within such a group of similar images, shape information is used for more refined comparison.

We now describe in more detail the implementation of the measures. We used Intel's openCV library for image processing functions. The test set consists of colored drawings of birds, all having a white background. The images are obtained from *http://www.clipart.com*. The test set consists of 292 different images, and 10 of them have 9 distortions. The distortions include three rotations, two scalings, two sheer transformations and two reflections. Figure 2 shows an image and its distortions.

4.1 Implementation of the primary color and rainbow color ratio measures

The implementation of the primary color ratio measure is straightforward. We used the ratios $\frac{R}{G}$ and $\frac{B}{G}$. For the rainbow color ratio measure, we defined 9 colors: red, green, blue, yellow, turquoise, purple, white, gray and black. Their respective areas are denoted R, G, B, Y, T, P, W, Gr and Bl respectively. For each image, we take out the background. Let I be the total area of the picture without its background. The rainbow color ratio vector of each image consists of the ratios $\frac{R}{I}$, $\frac{G}{I}$, $\frac{B}{I}$, $\frac{Y}{I}$, $\frac{T}{I}$, $\frac{P}{I}$, $\frac{Gr}{I}$ and $\frac{Bl}{I}$.

For both methods, we do not take into account the pixel values of the background of the image.

4.2 Implementation of the shape comparison measure based on the affine triangulation

As we use digital images, we don't have an exact description of the shape by means of boundary line segments. We extract the boundary information of an image and compute the triangulation in the following way:

Algorithm 4.1 Step 1. *The image is converted into black and white to abstract from all color information and get the shape silhouette;*

Step 2. *The shape boundary is detected using a Canny edge detector operation. This method, proposed by Canny [3], is considered the most efficient method for edge detection. It is implemented in the openCV library;*

Step 3. *From this boundary, lines are extracted using the Simple Hough Transform (SHT) [24]. This is a popular method for extracting geometric primitives. Each line in an image corresponds to a point* (ρ, θ) *in the Hough space such that the equation of the line is* $\rho = x cos(\theta) + y sin(\theta)$. *Given a threshold* τ, *the Hough transform returns* $(\rho, \theta)-$*pairs corresponding to lines that contain at least* τ *boundary points.*

Step 4. *The lines are then clipped against the image frame and* merged.

The nature of the Hough transform is such that it returns a lot of redundant lines. If the image contains a strong line, then it is detected as a bundle of lines that vary only slightly. Ideally, there would be only a limited amount of lines. Otherwise, the triangles resulting from the triangulation become very small and the chance that two different images match increases because of the small triangles. Therefore, we merge the lines until there are at most L left. The input of the merging procedure is a set of line segments. First, all lines receive weight one. The weight of line segment $\overline{\mathbf{p},\mathbf{q}}$ *is denoted* $W(\mathbf{p},\mathbf{q})$. *A threshold is initialized on one pixel. Repeatedly this threshold is increased until no more than L lines are left. For each value of the threshold, all pairs of line segments* $\overline{\mathbf{pq}}$ *and* $\overline{\mathbf{rs}}$ *are tested whether their end points are closer together than the threshold value. If so, the pair of segments is replaced by one segment* $\overline{\mathbf{uv}}$ *that is computed as follows:*

$$\mathbf{u} = \frac{\mathbf{W}(\mathbf{p},\mathbf{q})}{\mathbf{W}(\mathbf{p},\mathbf{q}) + \mathbf{W}(\mathbf{r},\mathbf{s})}\mathbf{p} + \frac{\mathbf{W}(\mathbf{r},\mathbf{s})}{\mathbf{W}(\mathbf{p},\mathbf{q}) + \mathbf{W}(\mathbf{r},\mathbf{s})}\mathbf{r}$$

and

$$\mathbf{v} = \frac{\mathbf{W}(\mathbf{p},\mathbf{q})}{\mathbf{W}(\mathbf{p},\mathbf{q}) + \mathbf{W}(\mathbf{r},\mathbf{s})}\mathbf{q} + \frac{\mathbf{W}(\mathbf{r},\mathbf{s})}{\mathbf{W}(\mathbf{p},\mathbf{q}) + \mathbf{W}(\mathbf{r},\mathbf{s})}\mathbf{s}.$$

Step 5. *The four segments bounding the image frame are added and the triangulation is computed as described in Algorithm 3.1.*

In our experiments, we reduce the number of lines L to 10. Figure 3 shows a parrot and the lines which result using Algorithm 3.1.

Because we are dealing with very approximate boundary information, it is not robust to compare the exact topological information of the triangulations of the two images. Instead, we compute the color ratio vectors of the N biggest polygons (i.e., we don't compute the last step of the triangulation algorithm). As relative areas are affine-invariant, it is true that if $p_1, p_2, \ldots, p_N$ are the N biggest polygons in the triangulation of image A, and α is an affine transformation, then $\alpha(p_1), \alpha(p_2), \ldots, \alpha(p_N)$ are the biggest polygons of the triangulation of image $\alpha(A)$.

Finding a one-to-one mapping between the biggest polygons would have to deal with the case that several polygons have Reni, we take the weighted average of the N feature vectors for each image, and compare those. We call those vectors *weighted color ratio vectors*. The weight of each polygon is the ratio of the area of that polygon and the total area of the three biggest polygons (hence the weight is also an affine invariant). If N is large, then there could be a large difference in the size of the biggest and the smallest polygons. If N is small, then in general there is only a small difference and in that case the weight may be even eliminated without a significant loss of performance. The shape similarity measure is defined as the distance (e.g., the Euclidean distance) between two weighted color ratio vectors.

5 Experimental results

5.1 Comparison of the color ratio methods

We compare the primary color ratio and the rainbow color ratio measure. We compare the distribution of the values of the distances between all pairs of images. Also, we test for each of the 10 images of which there are distortions in the database, how accurate all distortions can be found using each method.

We first compare the distribution of the values of the distances between all pairs of images. Figure 4 shows this distributions for each method. The horizontal axis shows the range of the distance values. The vertical axis shows the number of image pairs that have a distance within a certain range. It appears that the rainbow color ratio measure has a smaller range of distance values, they are all smaller than 1.3. For the primary color ratio measure, the values range between zero and 2.7 (for clarity, in the graph of Figure 4, all values greater than 2 are put in one category).

We now compute, for each of the 10 test images, the distances to all other images and sort the other images by that distance. We count the maximal distance between each image and its 9 distortions, and the number of images (called *rank*) we have to return in order to get all relevant matches, i.e. the 9 distortions. These calculations are shown in the table below, for both the primary color ratio and the rainbow color ratio. Figure 5 shows for both measures the closest images for a parrot, corresponding to image one in the table. The upper row is the result for the primary color ratio measure, the lower row is the rainbow color ratio measure. We see that the primary color ratio measure has better performance. The average distance of the primary color ratio measure is only half that of the rainbow color ratio measure. If we compute the efficiency of both methods we have that the rainbow color ratio measure has an efficiency of 60%, where the efficiency of the primary color ratio is 89%.

	primary		rainbow		combined	
id	rank	dist.	rank	dist.	rank	scan
1	11	0.0486	14	0.2172	9	14
2	9	0.1793	10	0.9633	9	10
3	9	0.0293	11	0.0931	9	11
4	9	0.0096	21	0.1203	10	21
5	10	0.0274	9	0.0337	9	10
6	9	0.0619	10	0.0919	9	10
7	16	0.0212	31	0.1511	9	31
8	9	0.1078	23	0.1027	9	23
9	10	0.0413	10	0.0411	9	10
10	9	0.0253	10	0.0664	9	10
max	16	0.1793	31	0.2172	10	31
avg	10.1	0.0552	14.9	0.1018	9.1	15

If we combine all information, we can say that the primary color ratio measure both spreads the distances over a bigger range (although the feature vectors have a lower dimension than those of the rainbow color ratio method) and puts the distortions of the same image closer to each other. So the primary color ratio measure can distinguish better between images that are distortions of each other and images that have a similar color scheme but a different shape.

In Figure 5, we can see that the extra images found by both color methods are different. Images 7 and 10 of the first are different from images 7, 10, 11 or 12 from the second row. This leads to the novel idea of combining both the primary color ratio measure and the rainbow color measure by taking the intersection of the nearest neighbors found by the two methods. In the rainbow and primary color ratio measure, we still need to distinguish the relevant from the nonrelevant images in the output. This can be done either by the user of the system, or by some automatic method, for example an affine-invariant shape comparison method.

We first search the matching images by means of both the rainbow color ratio and the primary color ratio, and then reduce the output using the following algorithm:

Algorithm 5.1 Step 1. Let L_1 be the output of the primary color ratio measure, containing m images. [5] Let L_2 be the output of the rainbow color ratio measure, containing m images. Let L'_1, L'_2 and **Out** be lists of image ID's, initialize both lists to the empty list.

Step 2. Sequentially scan L_1 and L_2 in parallel. The first time, take 2 elements of L_2 such that we always read one element ahead in that list. Add each element that had been viewed to L'_1 or L'_2, corresponding to the list it originates from. Each time an element from both lists is read, check of L'_1 and L'_2 contain any common elements. If so, add the common element to **Out** and remove it from both L'_1 and L'_2.

Step 3. Return **Out**.

In the Table above, we show again the number of elements we have to return in order to obtain 100% relevance. We can see that, except for image number 4 (i.e. in 90% of the cases), the first 9 elements of the intersection are the 9 distortions of the image (column 6). Column 7 shows how far we have to scan both lists before the number of intersections in column 6 of the corresponding row are found. This is the same as the maximum of the ranks for the primary and the rainbow color measure. The efficiency of the combination is significantly better than the efficiency of the primary color ratio method. We obtain that 98.9% of the returned images is relevant. In Figure 6, the performance of all three methods is combined into one graph.

5.2 Comparing shapes

As we already mentioned in the previous subsection, when the system returns the best matched to the user, either the user or some other technique needs to separate the relevant from the nonrelevant images. We now look if we can use the shape comparison method to do this separation. For the primary color ratio method we apply the shape comparison on the 16 nearest neighbors found by the primary color ratio method. For the rainbow color ratio method we search amongst the 31 nearest neighbors.

[5]The m is the number of images returned that guarantees that 100% of the relevant images are found. In our previous experiments $m = 31$ when using the rainbow color ratio method.

The following table shows the experimental results for the shape measure. We computed the distance for $N = 1$, i.e. the biggest polygon only is considered.

	distance	rainbow color rank	primary color rank
1	0.1938	13	10
2	0.3986	16	11
3	0.4059	30	15
4	0.5903	29	10
5	0.5936	30	16
6	0.2126	10	10
7	0.2898	24	9
8	0.1999	23	12
9	0.7869	30	16
10	0.6930	31	15
max:	0.7869	31	16
avg:	0.4364	23.6	12.4

It seems that the shape comparison cannot add any value to the rainbow color ratio and primary color ratio measures. There are two explanations for this. First, the primary color and rainbow color ratio measures perform very well. Secondly, the boundary detection method used (i.e., the Hough transform) is not very accurate.

Without more accurate boundary detection algorithms, it seems that the applications of the affine-invariant triangulation lie mainly in the field of constraint databases [17, 19]. Here, the exact equations that define the boundary of figures are known.

6 Conclusion and further work

Our experients show that the primary color ratio measure clearly outperforms the rainbow color ratio measure. It would be interesting to also investigate whether the primary color ratio measure is robust against images saved at different compressing qualities, or taken in varying lightning conditions. Also, more tests need to be done to test recognition of images when objects are only partially visible, for example due to occlusion.

We also plan to study *affine-invariant queries* [10, 8, 11, 4], whose results are not affected by affine transformations of the input spatial data.

References

[1] R. P. A. Pentland and S. Sclaroff. Photobook: Content-based manipulation of image databases. *International Journal of Computer Vision*, 18:133–254, 1996.

[2] N. Beckman, H.-P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: An efficient and robust method for points and rectangles. *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 322–331, 1990.

[3] J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[4] J. Chomicki, S. Haesevoets, B. Kuijpers, and P. Revesz. Classes of spatiotemporal objects and their closure properties. *Annals of Mathematics and Artificial Intelligence*, 39(4):431–461, 2003.

[5] G. K. D.P. Huttenlocher and W. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:850–863, 1998.

[6] M. Erwig and M. Schneider. Partition and conquer. In *Proceedings of the 3rd International Conference on Spatial Information Theory*, volume 1329 of *Lecture Notes in Computer Science*, pages 389–408. Springer, 1997.

[7] R. A. Finkel and J. L. Bentley. Quad trees: A data structure for retrieval of composite keys. *Acta Informatica*, 4(1):1–9, 1974.

[8] F. Geerts, S. Haesevoets, and B. Kuijpers. A theory of spatio-temporal database queries. In G. Ghelli and G. Grahne, editors, *Database Programming Languages, 8th International Workshop, DBPL 2001*, volume 2397 of *Lecture Notes in Computer Science*, pages 198–212. Springer, 2002.

[9] A. Guttman. R-trees: a dynamic index structure for spatial searching. *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 47–57, 1984.

[10] M. Gyssens, J. V. den Bussche, and D. V. Gucht. Complete geometrical query languages. In *Proceedings of the 16th ACM Symposium on Principles of Database Systems*, pages 62–67. ACM Press, 1997.

[11] S. Haesevoets. *Modelling and Querying Spatio-temporal Data*. Ph.d. thesis, Hasselt University, 2005.

[12] M. Hagedoorn and R. C. Veldkamp. Reliable and efficient pattern matching using an affine invariant metric. *Indernational Journal of Computer Vision*, 31:203–225, 1999.

[13] H. V. Jagadish. A retrieval technique for similar shapes. *Proc. ACM SIGMOD International Conference on Management of Data*, pages 208–217, 1991.

[14] I. Kamel and C. Faloutsos. Hilbert R-tree: An Improved R-tree using Fractals. In *Proceedings of the Twentieth International Conference on Very Large Databases*, pages 500–509, Santiago, Chile, 1994.

[15] M. O. M. de Berg, M. van Kreveld and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 2000.

[16] G. Nielson. A characterization of an affine invariant triangulation. In G. Farin, H. Hagen, and H. Noltemeier, editors, *Geometric Modelling, Computing Supplementum 8*, pages 191–210, 1993.

[17] J. Paredaens, G. Kuper, and L. Libkin, editors. *Constraint databases*. Springer-Verlag, 2000.

[18] E. G. M. Petrakis and C. Faloutsos. similarity searching in large databases. *IEEE Transactions on Knowledge and Data Engineering*, 9:435–437, 1997.

[19] P. Revesz. *Introduction to Constraint Databases*. Springer-Verlag, 2002.

[20] L. Roberts. Machine perception of three-dimensional solids. *J.T. Tippet, editor, Optical and Electro-optical Information Processing*, 1965.

[21] R. M. S. Ravela. Retrieving images by similarity of visual appearance. *IEEE Workshop on Content-based Access of Image Databases*, pages 311–347, 1997.

[22] M. Stricker and M. Orengo. Similarity of color images. *Proc. SPIE Conference on Storage and Retrieval for Image and Video Databases III*, 2420:381–392, 1995.

[23] M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, 7:11–32, 1991.

[24] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Halli, Inc., 1998.

[25] J. S. Y. Lamdan and H. Wolfson. Affine-invariant model-based object recognition. *IEEE Journal of Robotics and Automation*, 6:578–589, 1990.

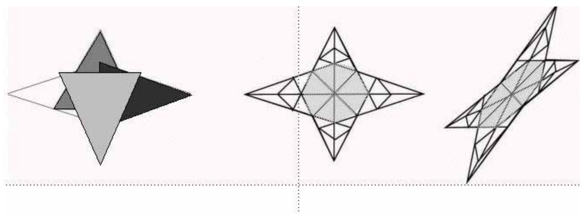

Figure 1. An example of an affine-invariant triangulation.

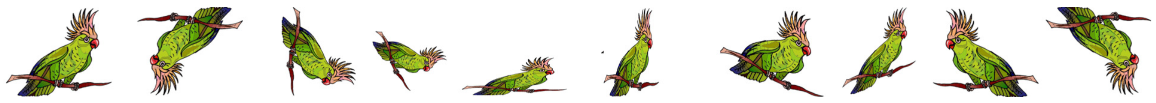

Figure 2. An example of an image representing a cockatiel and its distortions.

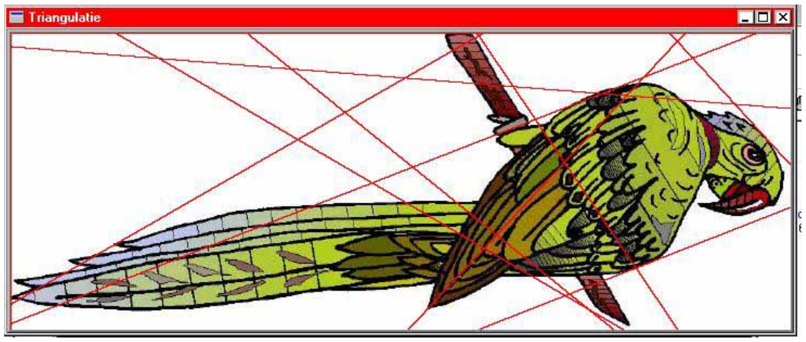

Figure 3. The detected lines after reduction.

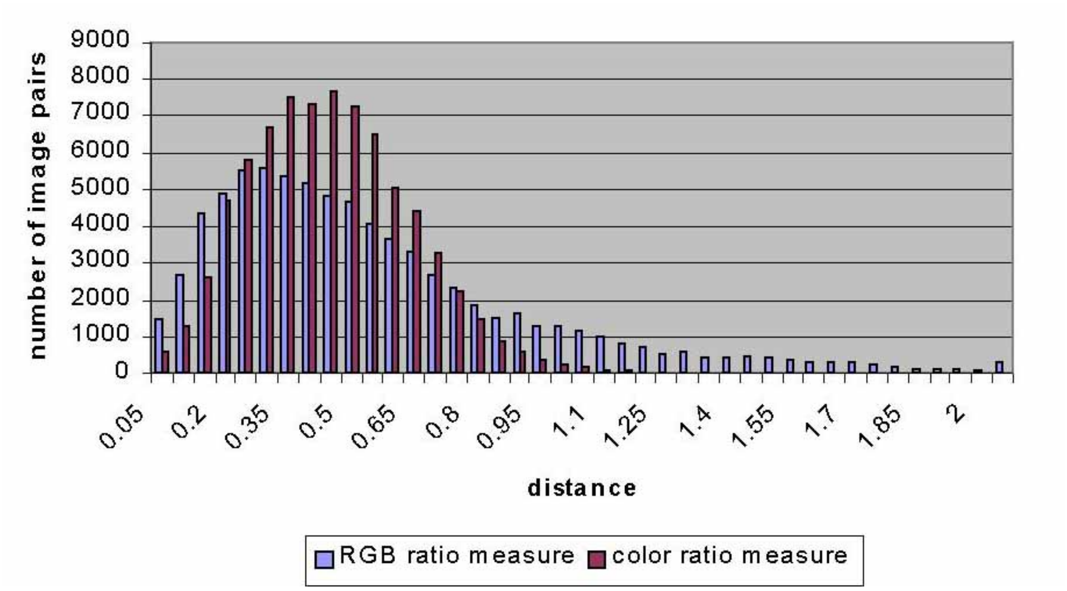

Figure 4. Comparison of both color measures: distribution of the distance values.

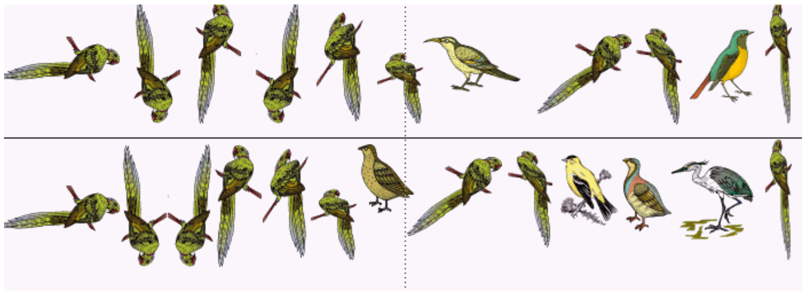

Figure 5. Comparison of the both color measures: closest images.

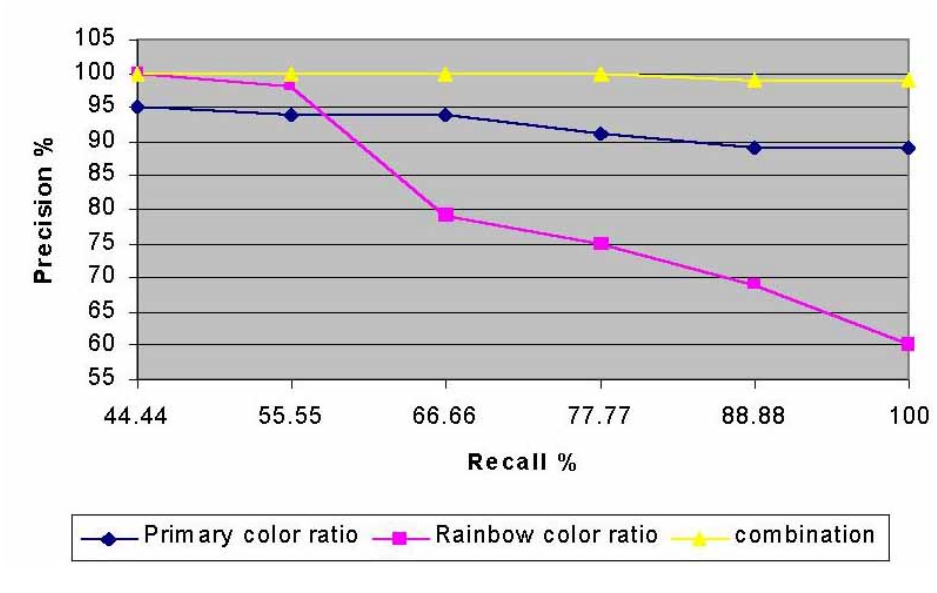

Figure 6. Comparison of the performance of the three color measures.