Shape Matching by Curve Modelling and Alignment

CECILIA DI RUBERTO
University of Cagliari
Dept. of Math. and Comp. Sc.
Via Ospedale 72, 09124 Cagliari
ITALY
dirubert@unica.it

MARCO GAVIANO
University of Cagliari
Dept. of Math. and Computer Science
Via Ospedale 72, 09124 Cagliari
ITALY
gaviano@unica.it

ANDREA MORGERA
University of Cagliari
Dept. of Math. and Comp. Sc.
Via Ospedale 72, 09124 Cagliari
ITALY
andrea.morgera@unica.it

Abstract: Automatic information retrieval in the field of shape recognition has been widely covered by many research fields. Various techniques have been developed using different approaches such as intensity-based, model-based and shape-based methods. Whichever is the way to represent the objects in images, a recognition method should be robust in the presence of scale change, translation and rotation. In this paper we present a new recognition method based on a curve alignment technique, for planar image contours. The method consists of various phases including extracting outlines of images, detecting significant points and aligning curves. The dominant points can be manually or automatically detected. The matching phase uses the idea of calculating the overlapping indices between shapes as similarity measures. To evaluate the effectiveness of the algorithm, two databases of 216 and 99 images have been used. A performance analysis and comparison is provided by precision-recall curves.

Key–Words: Curve alignment, Information retrieval, Object recognition, Image indexing, Precision-recall.

1 Introduction

Because of recent advances in computer science, content-based retrieval has received considerable attention and, consequently, improving the technology for content-based querying systems becomes more appealing. For content-based image retrieval, of course shape contains the most attractive visual information for human perception. Four important feature components are usually considered for content-based image retrieval: color, texture, shape and spatial relationship. Among these features, shape contains the most attractive visual information for human perception. Shape recognition arises in a variety of contexts, examples of which include not only retrieval by content from image databases, but also document image analysis, automated industrial inspection, analysis of medical images as well as vision-based robotics and visual tracking. Shape recognition is also a very challenging problem because of the large degree of variations of objects in 2D images: within class variations, articulations, variations in position, viewing direction, illumination, occlusion and others. Most recognition approaches tend to restrict the representation space of objects by focusing on the main variations in the projected intensity pattern (intensity-based), on the within category variations of shape when restricted to a set of models (model-based), and on the variations of the apparent contour (shape-based). The problem of recognition when shapes are encoded by their contours is interesting for many reasons: contours are easy to obtain and can be used for shape representation regardless of shape convexity; they are one of the most commonly used shape descriptors; since they may be considered as high order curves, algorithms for contour-based recognition are potentially extendible to more generic shape classes including cursive words and hand-drawn sketches. Boundary-based representations of shapes include unorganized point sets [4, 7], curves [3, 24], and medial axis or shock graphs [9, 17, 20]. Outlines of shapes have been represented as unorganized point sets, which are typically matched using an assignment algorithm [5, 7, 13]. In [13] graduated assignment is used to match image boundary features. In [5, 14] the Hungarian method is used to match "shape contexts", where a coarse histogram of the relative location of the remaining points is used as feature. A related approach [15] uses the generalized Hausdorff distance to compare edge maps of 2D images. These methods have the advantage of not requiring ordered boundary points, but the match does not necessarily preserve the coherence of shapes and may have problems when dealing with noisy or partial data. In many object recognition and shape-based indexing applications, the objects are represented by their outline curves and matched [12, 16, 24]. Matching typically involves finding a mapping from one curve to the other that minimizes an “elastic” performance functional, which penalizes ”stretching” and ”bending” [3, 24].
minimization problem in the discrete domain is transformed into one of matching shape signatures with curvature, bending angle, or orientation as attributes [3, 12, 16]. Generally, the curve-based methods suffer from one or more of the following drawbacks: asymmetric treatment of the two curves, lack of rotation and scaling invariance, and sensitivity to articulations and deformations of parts.

Generally speaking, in order to be effective, a recognition measure should satisfy the following properties proposed by Arkin et al. [2]:

- The measure should be a metric.
- It should be invariant under translation, rotation, and scale change.
- It should be easy to compute.
- It should match intuitive notions of shape resemblance.

Region-based techniques seem effective but arduous and domain-dependent. On the contrary, representing a shape with its contour is much more efficient than with the overall region. However, the contour captured by a camera under arbitrary orientation may have been distorted by certain geometric transformations. In [19] the notion of alignment curve is introduced in order to ensure a symmetric treatment of the curves, and rotation-invariance is guaranteed by the use of intrinsic properties. Current curve alignment methods can be classified into two categories: methods based on rigid transformations [1], and those based on non-rigid deformations [3, 24]. Methods based on rigid transformations rely on matching feature points by finding the optimal rotation, translation, and scaling parameters. They are sensitive to articulations, deformations of parts, occlusion, and other variations in the object form. Methods based on non-rigid deformations model articulation and other deformations by finding the mapping from one curve to another that minimizes a performance functional consisting of stretching and bending energies. These methods suffer from one or more of the following drawbacks: asymmetric treatment of the two curves, sensitivity to sampling of the curves, lack of rotation and scaling invariance, and sensitivity to occlusion and articulations.

Our approach to find the optimal correspondence between 2D curves relies on using the intrinsic properties of the curves in an energy minimization framework by overlapping the curves and calculating the degree of alignment as a measure of similarity of the curves. In Section 2 we present the mathematical background used by the proposed alignment method.

Section 3 describes the algorithms for finding the optimal alignment and shape matching. Section 4 illustrates the proposed curve alignment performance in several applications including prototype formation, object recognition and indexing into image databases.

# 2 The mathematical formulations

In this section we write down the mathematical results that will provide the guidelines of the method described in section 3.

**Definition 2.1** (Image contour or curve). A set \( I \) of \( n \) points

\[
I = \{ P_i = (x_i, y_i) \in \mathbb{R}^2, i = 1, \ldots, n \ | \ P_1 = P_n; P_i \neq P_j, i = 1, \ldots, n - 2, j = i + 1, \ldots, n - 1 \}
\]

is said to be a image contour or curve if, for any line segment with endpoints two successive points in \( I \), the following

\[
P_i P_{i+1} \cap P_j P_{j+1} = \emptyset, i = 1, \ldots, n - 3;
\]

\[
P_i P_{i+1} \cap P_{i+1} P_{i+2} = P_{i+1}, i = 1, \ldots, n - 2
\]

holds.

We can state that an image contour or curve \( I \) is the set of the successive vertices of a closed polygon that is not self intersecting. In the sequel we denote by \( \hat{I} \) the closed polygon associated with \( I \).

**Definition 2.2** (Translation). Given \( (x_t, y_t) \in \mathbb{R}^2 \), a translation mapping \( T : \mathbb{R}^2 \rightarrow \mathbb{R}^2 \) is defined by

\[
T : \begin{cases}
  x' = x + x_t \\
  y' = y + y_t
\end{cases}
\]

**Definition 2.3** (Rotation). Given \( (x_r, y_r) \in \mathbb{R}^2 \) and \( \theta \in [0, \pi] \), a rotation mapping \( R : \mathbb{R}^2 \rightarrow \mathbb{R}^2 \) through the angle \( \theta \) and the rotation center \( (x_r, y_r) \) is defined by

\[
R : \begin{cases}
  x' = \cos(\theta) \ast (x - x_r) - \sin(\theta) \ast (y - y_r) + x_r \\
  y' = \sin(\theta) \ast (x - x_r) + \cos(\theta) \ast (y - y_r) + y_r
\end{cases}
\]

**Definition 2.4** (Scaling). Given two real numbers \( s_1 \) and \( s_2 \), a scaling mapping \( S : \mathbb{R}^2 \rightarrow \mathbb{R}^2 \) is defined by

\[
S : \begin{cases}
  x' = s_1 \ast x \\
  y' = s_2 \ast y
\end{cases}
\]

**Proposition 2.5** Let \( I' \) be a curve obtained from a curve \( I \) by a chain application \( S \ast R \ast T \) of a translation \( T \), a rotation \( R \) and a scaling \( S \) with parameters \( (x_t, y_t), (\theta, x_r, y_r) \) and \( (s_1 = s_2 = s) \), respectively.
Let $I^b$ be the image contour obtained by the chain application to $I'$ of the mapping $\text{map}_1$

$$\text{map}_1: \begin{cases} x' = x + x_c - x_c' \\ y' = y + y_c - y_c' \end{cases} \quad (2.3)$$

where $C \equiv (x_c, y_c)$ and $C' \equiv (x_c', y_c')$ denote the centroid of $I$ and $I'$, respectively. Further, let $I''$ be obtained by the application to $I^b$ of the mappings $\text{map}_2$ and $\text{map}_3$ given by

$$\begin{align*}
\text{map}_2: & \quad \begin{cases} x' = \alpha (x - x_c) - \beta (y - y_c) + x_c \\ y' = \beta (x - x_c) + \alpha (y - y_c) + y_c \end{cases} \quad (2.4) \\
\text{map}_3: & \quad \begin{cases} x' = \bar{s} (x - x_c) + x_c \\ y' = \bar{s} (y - y_c) + y_c \end{cases} \quad (2.5)
\end{align*}$$

where

$$\begin{align*}
\alpha &= \frac{(x_1 - x_c)(x_0 - x_c) + (y_1 - y_c)(y_0 - y_c)}{((x_1 - x_c)^2 + (y_1 - y_c)^2)^{0.5}((x_0 - x_c)^2 + (y_0 - y_c)^2)^{0.5}} \\
\beta &= \frac{(y_1 - y_c)(x_0 - x_c) - (x_1 - x_c)(y_0 - y_c)}{((x_1 - x_c)^2 + (y_1 - y_c)^2)^{0.5}((x_0 - x_c)^2 + (y_0 - y_c)^2)^{0.5}} \\
\bar{s} &= \frac{(x_1 - x_c)^2 + (y_1 - y_c)^2}{((x_0 - x_c)^2 + (y_0 - y_c)^2)^{0.5}}
\end{align*}$$

and $P_{\text{max}} \equiv (x_1, y_1)$ and $Q_{\text{max}} \equiv (x_0, y_0)$ the points in $I$ and $I^b$, respectively, with maximum distance from $(x_c, y_c)$, then $I'' = I$.

**Proof.** The chain application of $S \ast R \ast T$ maps the centroid $C$ of $I$ into the centroid $C'$ of $I'$ and maps $P_{\text{max}}$ into $P'_{\text{max}}$ with $P'_{\text{max}}$ the point in $I'$, with maximum distance from $C'$. The application of $\text{map}_1$ maps, through a translation, $C'$ into $C$ and the application of $\text{map}_2$ cancels the rotation done in $R$. Finally, $\text{map}_3$ maps $P'_{\text{max}}$ into $P_{\text{max}}$ by a scaling. Hence, we can state $I = I''$. \qed

**Proposition 2.6** Let $I'$ be a curve obtained from a curve $I$ by a chain application $S \ast R \ast T$ of a translation $T$, a rotation $R$ and a scaling $S$ with parameters $(x_1, y_1)$, $(\theta, x_r, y_r)$ and $(s_1, s_2)$, respectively. Let $I^b$ be the curve obtained by the application to $I'$ of the mapping $\text{map}_1$

$$\text{map}_1: \begin{cases} x' = x + x_c - x_c' \\ y' = y + y_c - y_c' \end{cases} \quad (2.6)$$

where $C \equiv (x_c, y_c)$ and $C' \equiv (x_c', y_c')$ denote the centroid of $I$ and $I'$, respectively. Further, let $I''$ be obtained by the application to $I^b$ of the mapping $\text{map}_2$

$$\text{map}_2: \begin{cases} x' = r (x - x_c) \\
y' = s (y - y_c) + y_c \end{cases} \quad (2.7)$$

with $\theta \in [0, 2\pi]$. Further, let $I''$ be obtained by the application to $I''$ of the mapping $\text{map}_3$ given by

$$\text{map}_3: \begin{cases} x' = \bar{s}_1 (x\theta - x_c) + x_c \\
y' = \bar{s}_2 (y\theta - y_c) + y_c \end{cases} \quad (2.8)$$

where $\bar{s}_1 = \frac{x_0}{x_1}$, $\bar{s}_2 = \frac{y_0}{y_1}$

with $P_x \equiv (x_1, y_c)$ and $P_y \equiv (x_c, y_1)$ the intersection points, with the smallest distance from $(x_c, y_c)$, of the boundary of $I$ with the straight lines $y = y_c$ and $x = x_c$, respectively, and $Q_x \equiv (x_0, y_c)$, $Q_y \equiv (x_c, y_0)$ the intersection points of the boundary of $I^b$ with the same straight lines. Then, there exists $\theta^* \in [0, 2\pi]$ such that $I'' = I$.

**Proof.** The chain application of $S \ast R \ast T$ maps the centroid $C$ of $I$ into the centroid $C'$ of $I'$. The application of $T \ast R$ can be rewritten as an application of a rotation $R'$ through a suitable angle $\theta'$ with the centroid of $I$ as rotation center and of a suitable translation $T'$. The transformation $\text{map}_1$ maps $C'$ into $C$. The application of $\text{map}_2$ with angle $-\theta'$ and the scaling in $\text{map}_3$ moves $I''$ to the original curve. \qed

In order to compare shapes, we give now a definition that will provide an overlapping index between two curves $I$ and $I'$. Such an index measures the degree of alignment or overlapping of two curves and it does represent a measure of similarity between two shapes.
Definition 2.7 (Overlapping index 1). Let \( I \equiv (P_{i}, i = 1, \ldots, n) \) and \( I' \equiv (P'_{j}, j = 1, \ldots, m) \) be two curves and \( \bar{I} \) and \( \bar{I}' \) the associated polygons with vertices \( I \) and \( I' \), respectively. Let
\[
\text{ind}_1(I, I') = \max_{i=1, \ldots, n} \left( \frac{\text{area}(\bar{I} - \bar{P})}{\text{area}(\bar{I})}, \frac{\text{area}(\bar{P} - \bar{I})}{\text{area}(\bar{I})} \right)
\]
where \( \text{area}(\bar{I}) \) stands for the area of the set \( \bar{I} \) and \( \text{area}(\bar{P}) \).

Fig.1 explains the definition, the slightly dark region and the darker region contribute to define \( \text{ind}_1 \). That value goes to zero as the areas of both regions reduce to zero, i.e., the regions are identical. In the sequel we shall make use of a second definition of the overlapping index that will turn out to be computationally less expensive.

Definition 2.8 (Overlapping index 2). Let \( I \equiv (P_{i}, i = 1, \ldots, n) \) and \( I' \equiv (P'_{j}, j = 1, \ldots, m) \) be two image contours and \( \bar{I} \) and \( \bar{I}' \) the associated polygons. The overlapping index \( \text{ind}_2 \) is defined as
\[
\text{ind}_2(I, I') = \max_{i=1, \ldots, n} \left( \frac{\max_{j=1, \ldots, m} (\text{distance}(P_{i}, \bar{S}(\bar{I})))}{\text{distance}(\bar{P}_{j}, \bar{S}(\bar{I})))} \right)
\]
where \( \bar{S}(S) \) denotes the boundary of the set \( S \) and distance refers to the euclidian distance.

In Fig.2 \( A \in I' \) and \( B \in I \) are the points with the maximum distance from \( I \) and \( I' \) respectively. The maximum of these distances gives \( \text{ind}_2 \). Note that given two image contours \( I \) and \( I' \) such that \( \text{ind}_2(I, I') = 0 \) then the associated polygons fully overlap, i.e., \( \bar{I} = \bar{I}' \); that is not true if \( \text{ind}_2(I, I') = 0 \). The following example
\[
I = \{(-1, 1), (0, 2), (0, 0.5), (1, 1), (1, -1), (0, -2), (0, -0.5), (-1, -1), (-1, 1)\}
\]
\[
I' = \{(-1, 1), (0, 0.5), (0, 2), (1, 1), (1, -1), (0, -2), (0, -0.5), (-1, -1), (-1, 1)\}
\]
shows that. Whenever we don’t make reference to a specific definition, by \( \text{ind}(I, I') \) we shall denote any overlapping index between \( I \) and \( I' \).

3 The alignment method for shape matching

The mathematical definitions given in section 2 suggest the main features of our method: to compare two shapes \( I \) and \( I' \) we assume that \( I' \) is derived from \( I \) by some linear transformations. That is in order to compare two curves \( I \) an \( I' \) we proceed as follows:

1. overlap by a translation \( I \) and \( I' \) so that \( C = C' \), with \( C \) and \( C' \) being the centroids of \( I \) and \( I' \), respectively;
2. carry out a rotation on \( I' \) with rotation center \( C \);
3. perform a scaling;
4. calculate the overlapping index \( \text{ind}(I, I') \).

We assume that two cases can take place: \( I \) and \( I' \) may differ either (a) by a equal scaling along both axis, either (b) by non equal scaling along the axis.

To handle cases (a) and (b) we shall consider two algorithms \( OA_1 \) and \( OA_2 \); in the first case we rewrite the four steps sketched above as

**Algorithm OA_1 (single scaling)**

1. find centroids \( C \) and \( C' \) and translate \( I \) and \( I' \) so that their centroids are the axis origin \( O \equiv (0, 0) \);
2. find points \( P_{\text{max}} \) and \( P'_{\text{max}} \) in \( I \) and \( I' \), respectively, whose distance from \( O \) is maximum;
3. rotate \( I \) by the angle \( (P_{\text{max}} - O)^T (P'_{\text{max}} - O) / (\|P_{\text{max}}\| \|P'_{\text{max}}\|) \);
4. scale \( I \) by \( (\|P_{\text{max}}\|/\|P'_{\text{max}}\|) \).
5. calculate \( \text{ind}(I, I') \).

In the second case we rewrite the four steps as follows:

**Algorithm OA_2 (double scaling)**

1. find centroids \( C \) and \( C' \) and translate \( I \) and \( I' \) so that their centroids are the axis origin \( O \equiv (0, 0) \); let \( I^a \) and \( I^b \) be the new curves;
2. let \( \theta(i) \ i = 1, \ldots, n \) a vector of \( n \) points equally spaced in \( [0, 2\pi] \);
3. find \( \theta(i^*) \) such that
\[
\text{ind}(I^a, I'^{\theta(i^*)}) = \min \text{ind}(I^a, I'^{\theta(i)})
\]
with \( I'^{\theta} \) obtained as result of a chain application of 2.7, and 2.8 to \( I^b \).
3.1 Shape representation and modelling

In our approach shapes are represented by their contours. Given a set \( S \) of assigned shapes in order to find all the similar shapes in \( S \) or that share a similar contour, we consider a model \( M \) which all the shapes of \( S \) must be compared to. Such a model is made up of one or more curves \( m_i, i = 1 \ldots m \) and we measure the shape similarity of an element \( I \) of \( S \) to \( M \) by calculating the overlapping indices between \( I \) and each element \( m_i \) of \( M \). The smallest value of these indices is assumed as similarity measure of \( I \) to the model and is denoted by \( \text{ind}(I, M) \):

\[
\text{ind}(I, M) = \min_{i=1 \ldots m} \text{ind}(I, m_i)
\]

The shape models have been drawn from real image contours by detecting the most significant points of the contours and joining them by straight lines. There exist a wide variety of ways to achieve a polygonal approximation of a contour [6, 11, 22]. The significant points are used both to reserve the shape of curve and to reduce the amount of data. In order to make the method more efficient, the elements of \( M \) will be approximated contours, usually, of real images. Indeed, this allows to define a image contour of a model by a number of points smaller than the number of points that would define the original image. In Fig.4 two real image contours and approximated contours are depicted. In Fig.4(a) the original contour, represented by a smaller solid line, is defined by 379 points while the approximated contour, represented by a larger solid line, is defined by 40 points. In Fig.4(b) we have 302 points and 30 points, respectively. In our experiments we have detected the dominant points both manually and automatically.

Let’s describe our automatic approach for dominant point detection [10]. In order to reach a complete set of dominant points of a given a contour, we need of an initial group of starting points. By the result of the four vertex theorem [8] we choose to locate four initial points by searching for local maxima and minima in the signature of the shape. A point is considered a maximum peak if it has the maximal value, and was preceded (to the left) by a value lower by \( \Delta \). In our case we use \( \Delta = 0.5 \). In such way, searching the minimal values, we find minimum peaks. We use the first and second maximum and minimum peaks so we have four starting points for our method.

After the initial points setup the method builds a set of dominant points in the following way:

1) Let \( D = \{ d_i, \ i = 1 \ldots m \} \) the initial dominant points set. For each pair of points \( d_id_{i+1}, \ i = 1, 2 \ldots m \) where \( d_{i+1} \) is a neighbor of \( d_i \) (modulo \( m \)) it is possible to calculate the sum of distance from each contour point \( p_j \) \( j = 1 \ldots n \) between \( d_i \) and \( d_{i+1} \) and the line that connects the two points:

\[
\text{dist}_j = \frac{|(x_{i+1} - x_i)(y_{j} - y_i) - (x_i - x_{j})(y_{i+1} - y_i)|}{\sqrt{((x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2)}}
\]

where \( (x_{i+1}, y_{i+1}), (x_i, y_i) \) and \( (x_j, y_j) \) are coordinates of \( d_{i+1}, d_i \) and \( p_j \) respectively. A global distance is calculated as:

\[
T = \sum_j \text{dist}_j, \ j = 1 \ldots n \quad (3.9)
\]

2) In order to find dominant points the value \( T \) of the previous step is compared to a threshold \( s \) defined as \( s = p * l \) where \( p \) is an input parameter of the algorithm and \( l \) is the number of points of the line that connects \( d_i \) and \( d_{i+1} \). If \( T > s \) we add an intermediate point between \( d_i \) and \( d_{i+1} \) looking for the point with maximum distance to the chord. The current set of points and the new dominant points added are the set of points for the next iteration.

3) The algorithm stops when, after two following iterations, the number of the current set of points is equal to the number of the previous set.

Given a dominant points set \( D \) we adopt a refinement technique to suppress dominant point which are near enough between them by a predefined threshold \( \tau = 0.007 \).

3.2 Shape matching and timing analysis

We now give a remark allowing to reduce substantially the computation of the overlapping index 2.

In comparing an element \( I \) of \( S \) to all elements in \( M \), the calculation of \( \text{ind}_2(I, M) \) has to be carried out by calculating first \( \text{ind}_2(I, m_1) \) and then \( \text{ind}_2(I, m_i), i = 2 \ldots m \). Once a
ind₂(I, mj), j ∈ {1, ..., m} has been carried out we can start to compute ind₂(I, mi), i > j.
However, if for a point (x̄, ȳ) ∈ I the overlapping index ind₂((x̄, ȳ, ȳ̂(mi))) is greater than or equal to ind₂(I, mj) then ind₂(I, mi) will be greater or equal to ind₂(I, mj). As a consequence we don’t need to complete the calculation of ind₂(I, mi). This feature will be present in the implementation of our algorithms.

Finally, for algorithm OA₁ we shall consider an option that carries out a sort of suboptimization procedure. In comparing contour I to the contour of each element mj of M we perform rotations of mj through angles θ(i) and calculate the best ind(I, mj).

The algorithm OA₁ is now written in pseudo-code language. It calls four procedures: find_centroid, find_max_point, rotate, and find_ind and, eventually, uses option_A.

Algorithm, single scaling (OA₁) m = number of contours in model M.

```
input_model: [x̄, ȳ, x̄', ȳ'; S];
[x̄', ȳ'] = find_centroid(x̄, ȳ, x̄', ȳ'; S);
translate model centroids to (0,0)
for j = 1 : m
  x̄(j) = x̄(j) - x̄;
  ȳ(j) = ȳ(j) - ȳ;
end
[x₁, y₁, n₁] = input_data I ∈ S;
[x̄₁, ȳ₁] = find_centroid(x₁, y₁, 1, 1);
translate image centroid to (0,0)
for i = 1 : n
  xᵢ = xᵢ - x̄;
  yᵢ = yᵢ - ȳ;
end
best = ∞;
for i = 1 : m
  n = n_s(i, m);
  x = xₗ(n, 1, n).
  y = yₗ(i, m, 1);
  best(buf) = find_ind(x, y, x₁, y₁, n₁, best)
  if best(buf) < best best = best(buf); end
end
Find_ind.
```

is given by

\[
x_c = \frac{1}{6A} \sum_{i=0}^{n-1} (x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i)
\]

\[
y_c = \frac{1}{6A} \sum_{i=0}^{n-1} (y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i)
\]

\[
A = \frac{1}{2} \sum_{i=0}^{n-1} (x_i y_{i+1} - x_{i+1} y_i).
\]

Find_max_point. This function reads the coordinates of a contour I and, then, finds and returns the index of the element of I whose distance from the (0,0) is the largest one.

Rotate_scale. This function reads the coordinates of a contour I, then applies to I a transformation so that it is rotated by the angle

\[
θ = \frac{(x(nn), y(nn))^T (x(1nn1), y(1nn1))}{\| (x(nn), y(nn)) \| \| (x(1nn1), y(1nn1)) \|}
\]

and scaled by

\[
sca = \frac{\| (x(nn), y(nn)) \|}{\| (x(1nn1), y(1nn1)) \|}.
\]

It returns the transformed I.

Find_ind. This function reads the coordinates of a contours I and I’ and a real value best, then calculates and returns the overlapping index ind(I, I’) between the two contours. If ind(I, I’) is ind₂(I, I’), the procedure is stopped as soon as any point P of I has a distance from I’ greater than best either any point P of I’ has a distance from I greater than best.

Option_A for OA₁. It consists of the following program segment:

```
% option: find the best rotation with rot_no
% a positive integer
rot_no = 2 * π/(rot_no - 1) + 2 * πi;
for j = 1 : rot_no
  x = cos(teta(j)) * x_start - sin(teta(j)) * y_start;
  y = sin(teta(j)) * x_start + cos(teta(j)) * y_start;
  best(buf) = find_ind(x, y, n, sca * x₁, sca * y₁, n₁, best);
  if best(buf) < best best = best(buf); end
end
```

Option A

We give the main features of the procedures called by the OA₁ algorithm.

Find_centroid. This function reads the contours of the m images that made up the model. Then it calculates and returns their centroids values. The centroid \((x_c, y_c)\) of \(I \equiv \{P_i \equiv (x_i, y_i), i = 1, ..., n\}\)
\begin{align*}
  x_{s}(j,:) &= x_{s}(j,:) - x_{b}(j); \\
  y_{s}(j,:) &= y_{s}(j,:) - y_{b}(j);
\end{align*}

end

\begin{align*}
  [x_{1}, y_{1}, n_{1}] &= \text{input} \_ \text{data} \ I \ \in \ S; \\
  [x_{b1}, y_{b1}] &= \text{find} \_ \text{centroid}(x_{1}, y_{1}, n_{1}, 1); \\
  \text{translate} \ \text{image} \ \text{centroid} \ \text{to} \ (0,0) \\
  x_{1} &= x_{1} - x_{b1}; \\
  y_{1} &= y_{1} - y_{b1}; \\
  \text{best} &= \infty; \\
  \text{for} \ i \_ m = 1: m \\
  n &= n_{s}(i \_ m); \\
  x &= x_{s}(i \_ m, 1: n); \\
  y &= y_{s}(i \_ m, 1: n); \\
  \text{best} \_ \text{buf} &= \text{find} \_ \text{ind}(x, y, n, x_{1}, y_{1}, n_{1}, \text{best}) \\
  \text{if} \ \text{best} \_ \text{buf} < \text{best} \ \text{best} &= \text{best} \_ \text{buf}; \ \text{end} \\
  x_{1} &= \text{find} \_ \text{zero}(x_{1}, y_{1}, n_{1}); \\
  y_{1} &= \text{find} \_ \text{zero}(y_{1}, x_{1}, n_{1}); \\
  \text{find the} \ \text{best} \ \text{rotation} \\
  \text{teta} &= [0: 2 \pi / (\text{rot} \_ \text{no} - 1): 2 \pi ]; \\
  x_{\_ \text{start}} &= x; \\
  y_{\_ \text{start}} &= y; \\
  \text{for} \ j = 1: \text{rot} \_ \text{no} \\
  x &= \cos(\text{teta}(j)) \times x_{\_ \text{start}} - \sin(\text{teta}(j)) \times y_{\_ \text{start}}; \\
  y &= \sin(\text{teta}(j)) \times x_{\_ \text{start}} + \cos(\text{teta}(j)) \times y_{\_ \text{start}}; \\
  x_{0} &= \text{find} \_ \text{zero}(x, y, n); \\
  y_{0} &= \text{find} \_ \text{zero}(y, x, n); \\
  \text{scal1} &= \text{abs}(x_{0} / x_{1} \_ 0); \\
  \text{scal2} &= \text{abs}(y_{0} / y_{1} \_ 0); \\
  \text{best} \_ \text{buf} &= \text{find} \_ \text{ind}(x, y, n, \text{scal1} \times x_{1}, \text{scal2} \times y_{1}, \text{best}); \\
  \text{if} \ \text{best} > \text{best} \_ \text{buf} \ \text{best} &= \text{best} \_ \text{buf}; \ \text{end} \\
\end{align*}

\textbf{output:} best.

\textbf{Find} \_ \text{zero}. \ \text{This reads} \ \text{the} \ \text{coordinates} \ \text{of} \ \text{a} \ \text{contour} \ \I \equiv (x, y) \ \text{then} \ \text{calculates} \ \text{and} \ \text{returns} \ \text{the} \ \text{smallest} \ \text{distance} \ \text{from} \ \text{O} \ \equiv (0, 0) \ \text{to} \ \text{the} \ \text{intersection} \ \text{of} \ \text{the} \ \text{boundary} \ \text{of} \ \I \ \text{with} \ \text{the} \ x \ \text{axis}.

\section{Experimental results}

In this section we illustrate our shape recognition approach with some experiments. We have evaluated the matching ability of the proposed method by indexing two databases of shapes, both constructed by Kimia’s group [18, 17]. The two proposed algorithms, \textit{OA},\textunderscore \textit{1} and \textit{OA},\textunderscore \textit{2}, have been first tested on a set \textit{S} of 216 images \textit{I}_i, i = \ldots, 216 (see Fig.5). The database consists of twelve shapes each from eighteen families (or classes) for a total of 216 shapes. In order to simplify the presentation of our procedure, we first assume that we want to identify the shapes of just one family \textit{F} \ \in \ \textit{S}; \ for instance, we want to identify all screwdrivers. We assume \textit{F} \ \text{made up} \ of \ n_F = n_2 - n_1 + 1 \ \text{images} \ \text{and} \ \textit{F} = \{I_j, j = n_1, \ldots, n_2\}, \ \text{where} \ \textit{I}_j \ \text{is the} \ j^{th} \ \text{image} \ \text{in} \ \textit{S}. \ \text{The indexing process is achieved by first defining a model} \ M \ \text{for the family and then by comparing each element of} \ \textit{S} \ \text{with the model and calculating the overlapping indices} \ \textit{ind}(I_j, M), j = 1, \ldots, 216. \ \text{The overlapping indices are then sorted in ascending order and stored as a vector named} \ \textit{res} \_ \textit{sort}. \ \text{The algorithm is considered fully effective if the overlapping indices of the images in} \ \textit{F} \ \text{are in the first} \ n_F \ \text{positions of} \ \textit{res} \_ \textit{sort}. \ \text{Further, whenever this does not take place, we consider how much the positions of the overlapping indices of the images in} \ \textit{F} \ \text{which are not in the first} \ n_F \ \text{positions are far from the} \ n_F \ \text{position. Hence, we evaluate the effectiveness of the algorithm by defining two accuracy values} \ \textit{acc} \_ \textit{1} \ \text{and} \ \textit{acc} \_ \textit{2}. \ \text{Given}

\begin{align*}
  \textit{F} &= \{j | n_1 \leq j \leq n_2, \ \textit{pos}(j) \leq n_F\}, \\
  \textit{E} &= \{j | n_1 \leq j \leq n_2, \ \textit{pos}(j) > n_F\}, \\
  \textit{acc} \_ \textit{1} &= \frac{\text{card}(\textit{F})}{n_F}, \\
  \textit{acc} \_ \textit{2} &= \frac{\sum_{j \in \textit{E}} (\text{pos}(j) - n_F)}{\text{card}(\textit{F})}
\end{align*}

where \text{by pos}(j) \ \text{and card}(\textit{F}) \ \text{we denote} \ \text{the position} \ \text{of} \ \textit{I}_j \ \text{in} \ \textit{res} \_ \textit{sort} \ \text{and} \ \text{the cardinality} \ \text{of} \ \textit{F} \ \text{respectively. In Fig.7 we show the results obtained by comparing} \ \text{the set} \ \textit{S}_1 \ \text{with the model made up of the two images given in Fig.6. The filled circles represent contours in the family to be identified; the empty ones represent those not in the family. In such a case all the contours in the family are identified (acc} \_ \textit{1} = 0, \textit{acc} \_ \textit{2} = 0). \ \text{We now present the overall results of our experiments. The set} \ \textit{S}_1 \ \text{is partitioned into 18 families} \ F_1, i = 1, \ldots, 18: \ \text{bones, glasses, and so on. Since our task is to identify the shapes} \ \text{of} \ \text{each family} \ (or} \ \text{class}, \ \text{we define four sets} \ M_1, \ M_1, \ M_2, \ M_2, \ \text{and} \ M_2, \ \text{each made up of 18 models, one for each family of images to identify. For each family, both the models in} \ M_1, \ \text{and} \ M_1, \ \text{consist of one image model}; \ \text{each model in} \ M_1, \ \text{is an approximated contour chosen such that its mean overlapping} \ \textit{ind}_1 \ \text{calculated with respect to all the images of the family to be identified provides the minimum value. In the same way we have chosen the models in} \ M_2, \ \text{but calculating} \ \textit{ind}_2. \ \text{That is, only one image model has been selected as the most representative approximated contour of a family with respect to} \ \textit{ind}_1 \ \text{or to} \ \textit{ind}_2. \ \text{The models in} \ M_2, \ \text{and}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{Fig6.png}
\caption{A model made up of two image contours.}
\end{figure}
Figure 5: A database of 216 shapes: set $S^1$.

Figure 7: Plot of the overlapping indices of all images.

$M_{2.2}$ consist of from one to two images. They have been chosen trying to minimize the overlapping indices $\text{ind}_1$ and $\text{ind}_2$. In our case we have


$$M_{1.2} = \{1, 13, 25, 37, 57, 67, 89, 102, 109, 123, 135, 145, 168, 173, 187, 195, 207\}$$

$$M_{2.1} = \{3, 13, 25, 44, 54, 56, 67, 68, 75, 82, 89, 103, 109, 128, 129, 142, 146, 152, 168, 173, 180, 186, 190, 193, 197, 208, 216\}$$

$$M_{2.2} = \{1, 13, 30, 32, 38, 42, 49, 52, 67, 75, 84, 89, 102, 109, 123, 125, 135, 147, 155, 168, 171, 175, 187, 195, 201, 205, 207\}.$$  

Each element in the model sets corresponds with the shape number in the image database. We run algorithms $OA_1$ and $OA_2$ with various settings:

1. the overlapping index can be $\text{ind}_1$ or $\text{ind}_2$;
2. the contour image of $S$ may be transformed by a chain application of the map $R \ast S \ast T$ with $R$ a rotation through an angle $\theta$ chosen at random in the interval $[0, \pi]$, $S$ a scaling and $T$ a translation;
3. the option A in $OA_1$ may be or not be present; whenever is present $\text{rot}_\text{no} = 40$;
4. $OA_2$ is run with $\text{rot}_\text{no} = 40$.

The scaling $S$ is assumed equal along both axis for $OA_1$, while is not equal in the case of $OA_2$.

The results of the numerical experiments referring to $OA_1$ are summarized in tables 4.1 and 4.2; while those referring to $OA_2$ are in 4.3 and 4.4. The acc$_1$ and acc$_2$ appearing in the tables are mean values of the values calculated for each family.

We now report the results gathered by testing algorithms $OA_1$ and $OA_2$ for another set of images.
Table 4.2: Results of algorithm $OA_1$ with models of up to two images (set $S^1$).

<table>
<thead>
<tr>
<th>transf</th>
<th>overl. index</th>
<th>rot_no</th>
<th>model set</th>
<th>acc_1</th>
<th>acc_2</th>
<th>time (min.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$M_{2,1}$</td>
<td>0.93</td>
<td>6.92</td>
<td>35</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>40</td>
<td>$M_{2,1}$</td>
<td>0.89</td>
<td>7.37</td>
<td>926</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>$M_{2,2}$</td>
<td>0.97</td>
<td>10.83</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>40</td>
<td>$M_{2,2}$</td>
<td>0.92</td>
<td>3.96</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$M_{1,1}$</td>
<td>0.79</td>
<td>13.71</td>
<td>32</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>40</td>
<td>$M_{1,1}$</td>
<td>0.86</td>
<td>10.07</td>
<td>919</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>$M_{2,2}$</td>
<td>0.80</td>
<td>21.67</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>40</td>
<td>$M_{2,2}$</td>
<td>0.92</td>
<td>8.11</td>
<td>24</td>
</tr>
</tbody>
</table>

Table 4.3: Results of algorithm $OA_1$ with models of one image (set $S^1$).

<table>
<thead>
<tr>
<th>transf</th>
<th>overl. index</th>
<th>rot_no</th>
<th>model set</th>
<th>acc_1</th>
<th>acc_2</th>
<th>time (min.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>40</td>
<td>$M_{1,1}$</td>
<td>0.80</td>
<td>24.20</td>
<td>637</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>40</td>
<td>$M_{1,1}$</td>
<td>0.84</td>
<td>14.61</td>
<td>24</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>40</td>
<td>$M_{1,1}$</td>
<td>0.69</td>
<td>39.90</td>
<td>637</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>40</td>
<td>$M_{1,1}$</td>
<td>0.79</td>
<td>29.48</td>
<td>19</td>
</tr>
</tbody>
</table>

Table 4.4: Results of algorithm $OA_2$ with models of up to two images (set $S^1$).

<table>
<thead>
<tr>
<th>transf</th>
<th>overl. index</th>
<th>rot_no</th>
<th>model set</th>
<th>acc_1</th>
<th>acc_2</th>
<th>time (min.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$M_{1,1}$</td>
<td>0.84</td>
<td>6.33</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>40</td>
<td>$M_{1,1}$</td>
<td>0.87</td>
<td>7.50</td>
<td>131</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>$M_{1,2}$</td>
<td>0.86</td>
<td>8.78</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>40</td>
<td>$M_{1,2}$</td>
<td>0.83</td>
<td>13.80</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$M_{1,1}$</td>
<td>0.71</td>
<td>10.03</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>40</td>
<td>$M_{1,1}$</td>
<td>0.84</td>
<td>9.57</td>
<td>131</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>$M_{1,2}$</td>
<td>0.65</td>
<td>11.14</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>40</td>
<td>$M_{1,2}$</td>
<td>0.82</td>
<td>13.40</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 4.5: Results of algorithm $OA_1$ with models (mod=1) of one image (set $S^2$).

and

$S^2$, again created by Kimia’s group [18, 17](Fig.8). There are 99 shapes in this database from nine families, which were collected from a variety of sources: fish, and sea-animals from Farzin Mokhtarian, “greebles” from Mike Tarr, and tools from Stan Sclaroff. Eleven shapes are included from each family so that shapes have variations in form, and as well transformations such as occlusion, articulation and missing parts. The numerical experiments have been carried out as for the set $S^1$. In such a case the model set are given by

$M_{1,1} = \{3, 17, 30, 39, 51, 56, 72, 80, 99\}$

$M_{1,2} = \{3, 17, 32, 38, 53, 56, 76, 87, 98\}$

An overall analysis of the results given in the tables allows to state that the overlapping algorithm is effective in identifying the images of each family. Furthermore, the results highlight the features of each setting taken into account. We summarize the main points in the following remarks

1. the use of overlapping index 1 does produce any clear advantage with respect to that of index 2,
but the computer execution time is dramatically favorable to the use of index_2;

2. option A gives usually best results but is computationally expensive;

3. the use of models made up of more images improves considerably the algorithm efficiency.

In order to compare the performance of the algorithms the precision recall curve has been used. Precision and recall are two classical performance evaluation metrics in the field of information retrieval. As described above results of the comparison of shapes are sorted in ascending order and stored in a vector. We have obtained a ranking of database images based on similarity with a given model; a particular item in the ranking is considered as relevant if the image belongs to the class of the model, otherwise it is not relevant.

Building a precision-recall curve requires another important parameter, called scope. It is defined as the number of retrieved items from the top of the ranking list. We have set a scope size value equal to 12 for the set \( S^2 \) and 11 for the set \( S^1 \). This choice is due to the fact that in [18] the summary of rank-ordered results is given only for the first 12 and 11 shapes, respectively. For each family, given a sample shape and the list of all the images, we have created automatically. We have repeated the experiments by using our automatic method for dom-

\[
\begin{array}{cccccc}
\text{transf} & \text{overl.} & \text{rot_no} & \text{model} & \text{acc}_1 & \text{acc}_2 & \text{time min.} \\
\text{index} & \text{set} & & & & \\
0 & 1 & 0 & M_{1,1} & 0.97 & 4.44 & 8 \\
0 & 1 & 1 & M_{2,2} & 0.94 & 5.33 & 232 \\
0 & 2 & 1 & M_{2,2} & 0.96 & 8.44 & 0 \\
0 & 2 & 2 & M_{2,2} & 0.85 & 15.58 & 8 \\
1 & 1 & 0 & M_{2,1} & 0.86 & 6.07 & 8 \\
1 & 1 & 1 & M_{2,1} & 0.92 & 4.72 & 232 \\
1 & 2 & 0 & M_{2,2} & 0.80 & 11.91 & 0 \\
1 & 2 & 1 & M_{2,2} & 0.84 & 14.00 & 6 \\
\end{array}
\]

Table 4.6: Results of algorithm \( O A_1 \) with models (mod=2) of up to two images (set \( S^2 \)).

\[
\begin{array}{cccccc}
\text{transf} & \text{overl.} & \text{rot_no} & \text{model} & \text{acc}_1 & \text{acc}_2 & \text{time min.} \\
\text{index} & \text{set} & & & & \\
0 & 1 & 0 & M_{1,1} & 0.76 & 13.92 & 135 \\
0 & 2 & 1 & M_{2,1} & 0.73 & 15.83 & 4 \\
1 & 1 & 0 & M_{1,1} & 0.63 & 22.33 & 138 \\
1 & 2 & 0 & M_{2,1} & 0.64 & 29.00 & 4 \\
\end{array}
\]

Table 4.7: Results for algorithm \( O A_2 \) with models (mod=1) of one image (set \( S^2 \)).

\[
\begin{array}{cccccc}
\text{transf} & \text{overl.} & \text{rot_no} & \text{model} & \text{acc}_1 & \text{acc}_2 & \text{time min.} \\
\text{index} & \text{set} & & & & \\
0 & 1 & 0 & M_{1,1} & 0.84 & 7.83 & 240 \\
0 & 2 & 0 & M_{2,1} & 0.82 & 10.24 & 6 \\
1 & 1 & 0 & M_{2,1} & 0.70 & 16.83 & 242 \\
1 & 2 & 0 & M_{2,2} & 0.71 & 20.86 & 8 \\
\end{array}
\]

Table 4.8: Results for algorithm \( O A_2 \) with models (mod=2) of up to two images (set \( S^2 \)).

\[
\begin{array}{cccccc}
\text{transf} & \text{overl.} & \text{rot_no} & \text{model} & \text{acc}_1 & \text{acc}_2 & \text{time min.} \\
\text{index} & \text{set} & & & & \\
0 & 1 & 0 & M_{1,1} & 0.86 & 6.54 & 0.29 \\
0 & 2 & 1 & M_{1,2} & 0.88 & 6.44 & 3.00 \\
1 & 2 & 0 & M_{1,2} & 0.77 & 16.71 & 0.31 \\
1 & 2 & 1 & M_{2,2} & 0.87 & 5.54 & 3.6 \\
\end{array}
\]

Table 4.9: Results for algorithm \( O A_1 \) with models of one image (set \( S^1 \)) created automatically.

\[
\begin{array}{cccccc}
\text{transf} & \text{overl.} & \text{rot_no} & \text{model} & \text{acc}_1 & \text{acc}_2 & \text{time min.} \\
\text{index} & \text{set} & & & & \\
0 & 2 & 0 & M_{1,2} & 0.96 & 8.22 & 0.35 \\
0 & 2 & 1 & M_{2,2} & 0.94 & 6.64 & 4.11 \\
1 & 2 & 0 & M_{2,2} & 0.85 & 18.71 & 0.37 \\
1 & 2 & 1 & M_{2,2} & 0.93 & 5.87 & 4.59 \\
\end{array}
\]

Table 4.10: Results for algorithm \( O A_1 \) with models of up to two images (set \( S^1 \)) created automatically.

\[
\begin{array}{cccccc}
\text{transf} & \text{overl.} & \text{rot_no} & \text{model} & \text{acc}_1 & \text{acc}_2 & \text{time min.} \\
\text{index} & \text{set} & & & & \\
0 & 2 & 0 & M_{1,2} & 0.85 & 17.07 & 3.37 \\
1 & 2 & 0 & M_{1,2} & 0.77 & 38.35 & 3.43 \\
\end{array}
\]

Table 4.11: Results for algorithm \( O A_2 \) with models of one image (set \( S^1 \)) created automatically.
We have proposed a new recognition method based on a curve alignment technique. The method consists of various phases including extracting outlines of images, detecting significant points and aligning curves. The dominant points can be manually detected or by using a very fast and efficient approach described in subsection 3.1. The last phase uses the idea of calculating the overlapping indices between shapes as similarity measures. The proposed algorithm has given satisfying results in terms of accuracy, although a reasonable performance degradation in the \( OA_2 \) variation, compared to similar approaches. Such results justify the computational effort required to shapes matching. Also, by using an automatic method for creating approximated models we can achieve a very impressive improvement in terms of required computation in spite of a negligible lack of accuracy.

Table 4.12: Results of algorithm \( OA_2 \) with models of one image (set \( S^1 \)) created automatically.

<table>
<thead>
<tr>
<th>transf</th>
<th>overl. index</th>
<th>rot no Opt. A</th>
<th>model set</th>
<th>acc(_1)</th>
<th>acc(_2)</th>
<th>time min.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>40</td>
<td>( M_{1,2} )</td>
<td>0.87</td>
<td>9.76</td>
<td>5.15</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>40</td>
<td>( M_{1,2} )</td>
<td>0.82</td>
<td>19.92</td>
<td>5.34</td>
</tr>
</tbody>
</table>

Table 4.13: Results of algorithm \( OA_1 \) with models of one image (set \( S^2 \)) created automatically.

<table>
<thead>
<tr>
<th>transf</th>
<th>overl. index</th>
<th>rot no Opt. A</th>
<th>model set</th>
<th>acc(_1)</th>
<th>acc(_2)</th>
<th>time min.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>( M_{1,2} )</td>
<td>0.82</td>
<td>9.56</td>
<td>0.8</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>40</td>
<td>( M_{1,2} )</td>
<td>0.77</td>
<td>9.74</td>
<td>0.47</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>( M_{1,2} )</td>
<td>0.74</td>
<td>11.55</td>
<td>0.7</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>40</td>
<td>( M_{1,2} )</td>
<td>0.77</td>
<td>9.89</td>
<td>0.49</td>
</tr>
</tbody>
</table>

Table 4.14: Results of algorithm \( OA_1 \) with models of up to two images (set \( S^2 \)) created automatically.

<table>
<thead>
<tr>
<th>transf</th>
<th>overl. index</th>
<th>rot no Opt. A</th>
<th>model set</th>
<th>acc(_1)</th>
<th>acc(_2)</th>
<th>time min.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>( M_{2,2} )</td>
<td>0.91</td>
<td>6.89</td>
<td>0.9</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>40</td>
<td>( M_{2,2} )</td>
<td>0.89</td>
<td>9.91</td>
<td>1.11</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>( M_{2,2} )</td>
<td>0.87</td>
<td>6.56</td>
<td>0.9</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>40</td>
<td>( M_{2,2} )</td>
<td>0.89</td>
<td>6.66</td>
<td>1.13</td>
</tr>
</tbody>
</table>

Table 4.15: Results of algorithm \( OA_2 \) with models of one image (set \( S^2 \)) created automatically.

<table>
<thead>
<tr>
<th>transf</th>
<th>overl. index</th>
<th>rot no Opt. A</th>
<th>model set</th>
<th>acc(_1)</th>
<th>acc(_2)</th>
<th>time min.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>40</td>
<td>( M_{1,2} )</td>
<td>0.80</td>
<td>12.30</td>
<td>0.57</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>40</td>
<td>( M_{1,2} )</td>
<td>0.66</td>
<td>24.07</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 4.16: Results of algorithm \( OA_2 \) with models of one image (set \( S^2 \)) created automatically.
in accuracy in few circumstances.

References:


