PROFI

Project number:	FP6-511572
Project acronym:	PROFI
Title:	Perceptually-Relevant Retrieval of Figurative Images

Deliverable No: D11.5

Title: MPEG-7 extension report

Short description:

A region based and a contour based shape similarity method are part of the MPEG-7 standard: the Angular Radial Transform, and the Curvature Scale Space. Their description schemes are given in the standard, and repeated here. Because the MPEG-7 standard is extensible, other description schemes can be defined. This report gives a number of further description schemes for other shape description and matching methods.

The description schemes for shape description and matching may be useful for the PASCAL Visual Object Classes Challenge and for ImageCLEF, the cross-language image retrieval track which is run as part of the Cross Language Evaluation Forum (CLEF).

Due month:	M36
Delivery month:	April 2008
Partners owning:	UU
Partners contributed:	UU
Classification:	PU

Information Society Technologies

Project funded by the European Community under the "Information Society Technologies" Programme

1. Introduction

MPEG-7, also known as "Multimedia Content Description Interface", aims at providing standardized core technologies allowing description of audiovisual data content in multimedia environments [1] [2]. This is a challenging task given the broad spectrum of requirements and targeted multimedia applications, and the broad number of audiovisual features of importance in such context.

In order to achieve this broad goal, MPEG-7 standardizes:

- Descriptors (D): representations of Features, that define the syntax and the semantics of each Feature representation.
- Description Schemes (DS), that specify the structure and semantics of the relationships between their components, which may be both Ds and DSs.
- A Description Definition Language (DDL), to allow the creation of new DSs and possibly Ds, and to allow the extension and modification of existing DSs.
- System Tools, to support multiplexing of description, synchronization issues, transmission mechanisms, file format, etc.

There exist a large variety of approaches to define shape similarity measures of planar shapes, some of which are listed in the references. Since an objective comparison of their qualities seems to be impossible, experimental comparison is needed. The Motion Picture Expert Group (MPEG), a working group of ISO/IEC (see for their website at http://www.chiariglione.org/mpeg/) has defined the MPGE-7 standard for description and search of audio and visual content. A region based and a contour based shape similarity method are part of the standard. The data set created by the MPEG-7 committee for evaluation of shape similarity measures [3, 4] offers an excellent possibility for objective experimental comparison of the existing approaches evaluated based on the retrieval rate. The shapes were restricted to simple pre-segmented shapes defined by their outer closed contours. The goal of the MPEG-7 Core Experiment CE-Shape-1 was to evaluate the performance of 2D shape descriptors under change of a view point with respect to objects, non-rigid object motion, and noise. In addition, the descriptors should be scale and rotation invariant.

To compare the performance of similarity measures, we built the framework SIDESTEP – Shape-based Image Delivery Statistics Evaluation Project. The web site address is at http://give-lab.cs.uu.nl/sidestep/. Performance measures such as the number of true/false positives, true/false negative, specificity, precision, recall, negative predicted value, relative error, k-th tier, total performance, and Bull's Eye Percentage can be evaluated for a single query, over a whole class, or over a whole collection,

A region based and a contour based shape similarity method are part of the MPEG-7 standard: the Angular Radial Transform, and the Curvature Scale Space. Their description schemes are given in [2], and repeated below. Because the MPEG-7 standard is extensible, other description schemes can be defined. This report gives a number of

further description schemes for other shape description and matching methods in the following sections.

2. Shape Descriptors

2.1 Angular Radial Transform

The Radial Angular Transform is a region based shape descriptor, part of the MPEG-7 standard. The following description comes from [2].

The shape of an object may consist of either a single region or a set of regions, as well as some holes in the object as illustrated in Figure 1(c). Since the region-based shape descriptor makes use of all pixels constituting the shape, it can describe any shape, i.e. not only a simple shape with a single connected region but also a complex shape that consists of several disjoint regions as illustrated in Figure 1(d) and (e).

Figure 1: Examples of various shapes.

The region-based shape descriptor utilizes a set of ART (Angular Radial Transform) coefficients. The ART is a 2-D complex transform defined on a unit disk in polar coordinates,

$$F_{nm} = \langle V_{nm}(\rho,\theta), f(\rho,\theta) \rangle = \int_0^{2\pi} \int_0^1 V_{nm}^*(\rho,\theta), f(\rho,\theta) \rho d\rho d\theta .$$

Here, F_{nm} is an ART coefficient of order *n* and *m*, $f(\rho,\theta)$ is an image function in polar coordinates, and $V_{nm}(\rho,\theta)$ is the ART basis function. The ART basis functions are separable along the angular and radial directions, i.e.,

$$V_{nm}(\rho,\theta) = A_m(\theta) R_n(\rho) .$$

The angular and radial basis functions are defined as follows:

$$A_m(\theta) = \frac{1}{2\pi} \exp(jm\theta),$$

$$R_n(\rho) = \begin{cases} 1 & n = 0 \\ 2\cos(\pi n \rho) & n \neq 0 \end{cases}.$$

Twelve angular and three radial functions are used.
Figure 2 shows the real parts of the 2-D basis functions whose origins are at the centers of each image. The imaginary parts have similar shape to the corresponding real parts but with different phases. Note that the brighter the image, the higher the value.

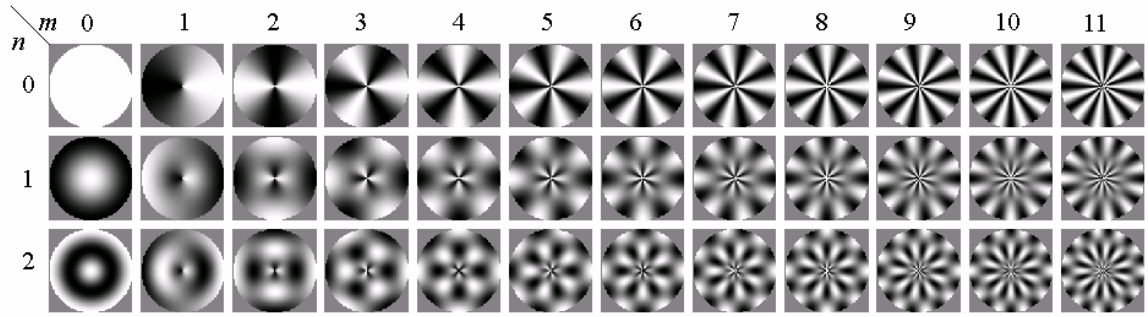


Figure 2: Real parts of the ART basis functions.

DDL representation syntax

```
<complexType name="RegionShapeType" final="#all">
    <complexContent>
      <extension base="mpeg7:VisualDType">
                  <element name="ArtDE">
                        <simpleType>
                   <restriction base="mpeg7:listOfUnsigned4Type>
                        <length value="35"/>
                   </restriction>
                </simpleType>
                </element>
        </extension>
    </complexContent>
</complexType>
```

Binary representation syntax

RegionShape {	**Number of bits**	**Mnemonic**
for(k=0; k<35; k++) {		
ArtDE[k]	4	uimsbf
}		
}		

Descriptor components semantics

ArtDE

This is an array of 35 normalized and quantized magnitudes of the shape coefficients. The normalization is performed by dividing each of them by the magnitude of the largest coefficient. The relationship between the order of k and the order of radial and angular indices (n,m) is as follows:

k	0	1	2	3	4	…	30	31	32	33	34
n	1	2	0	1	2	…	1	2	0	1	2
m	0	0	1	1	1	…	10	10	11	11	11

The reconstruction values for dequantization of ART coefficients (inverse quantization table) are listed in Table 1.

ArtDE	**Reconstruction value**
0000	0.001763817
0001	0.005468893
0010	0.009438835
0011	0.013714449
0100	0.018346760
0101	0.023400748
0110	0.028960940
0111	0.035140141
1000	0.042093649
1001	0.050043696

1010	0.059324478
1011	0.070472849
1100	0.084434761
1101	0.103127662
1110	0.131506859
1111	0.192540857

Table 1: Reconstruction value for ART coefficients.

2.2 Curvature Scale Space

The Curvature Scale Space is a contour based shape descriptor, paert of the MPEG-7 standard. The following description comes from [2].

The *object contour shape* descriptor describes a closed contour of a 2D object or region in an image or video sequence (see Figure 3).

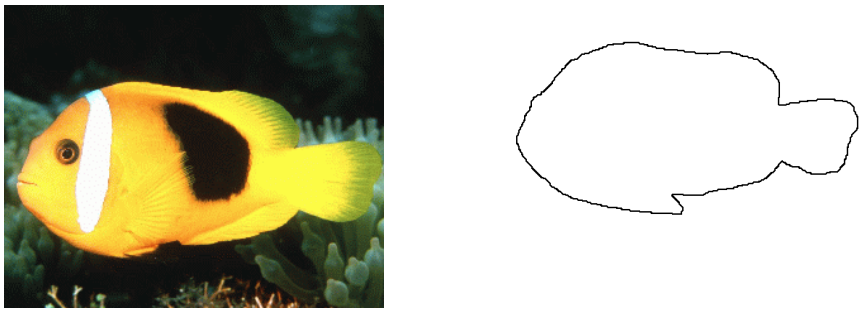


Figure 3: A 2D visual object (region) and its corresponding shape

The object contour-based shape descriptor is based on the Curvature Scale Space (CSS) representation of the contour. This representation of contour shape is very compact, below 14 bytes in size on average.
In order to create a CSS description of a contour shape, *Nsamples* equidistant points are selected on the contour, starting from an arbitrary point on the contour and following the contour clockwise. The x-coordinates of the selected *Nsamples* points are grouped together and the y-coordinates are also grouped together into two series X, Y. The contour is then gradually smoothed by repetitive application of a low-pass filter with the kernel (0.25,0.5,0.25) to X and Y coordinates of the selected *Nsamples* equidistant contour points. As a result of the smoothing, the contour evolves and its concave parts gradually flatten-out, until the contour becomes convex. The filtering process is terminated when the contour becomes convex. A so-called CSS image can be associated with the contour evolution process. Figure 4 shows the evolution of a contour and the associated CSS image. The CSS image does not have to be explicitly extracted, but is useful to illustrate the CSS representation. It is a binary image in which horizontal coordinates (*x_css*) correspond to the indices of the contour points selected to represent the contour (1,…,*Nsamples*), and vertical-coordinates (*y_css*) correspond to the amount of filtering applied, defined as the number of passes of the filter. The CSS image is made up of horizontal lines, a line defined by *y_css=k* is computed from the smoothed contour

resulting from *k*-passes of the filter. For each smoothed contour, the zero-crossings of its curvature function are computed. Contour curvature function zero-crossing points separate concave and convex parts of the contour. Each zero-crossing is marked on the horizontal line corresponding to the smoothed contour (*y_css=k*) and at the *x_css* location corresponding to the position of this zero-crossing along the contour. The CSS image has characteristic peaks. The coordinate values of the prominent peaks (*x_css*, *y_css*) in the CSS image are extracted. The peaks are ordered based on decreasing values of *y_css*, transformed using a non-linear transformation and quantized. In addition, the eccentricity and circularity of the contour are also calculated, quantized and stored.

In Figure 4, the left column shows the original contour and the contour at two different stages of the smoothing evolution process, after 20 and 80 passes of the filter. In the right column, the CSS image obtained from the evolution is shown. The contour smoothed by 20 passes of the filter has 8 curvature zero crossings (A, B, …, H), and these points are used to "construct" the CSS image at horizontal line *y_css=20*. The peaks in the CSS representation are shown in the figure.

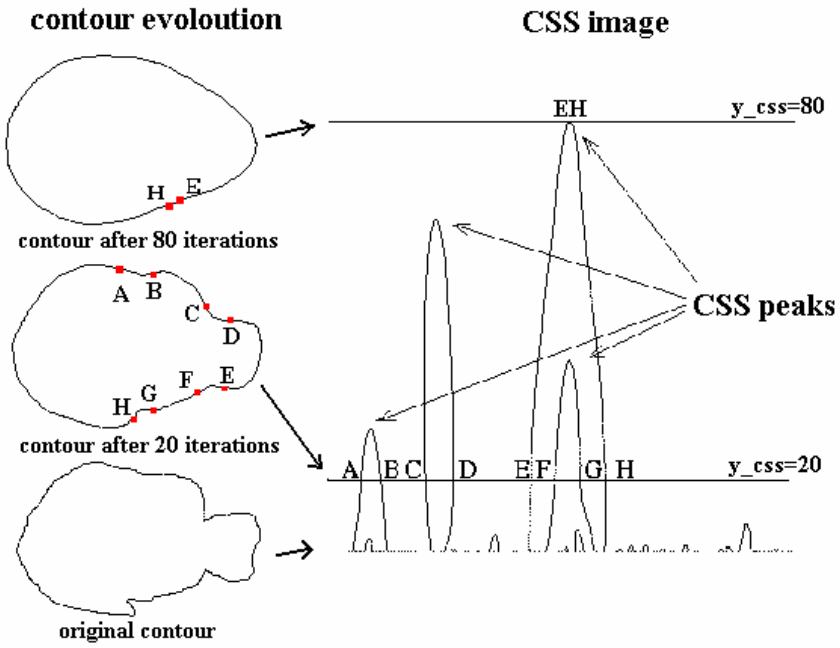


Figure 4: CSS Image Formation.

DDL representation syntax

```
<complexType name="ContourShapeType" final="#all">
      <complexContent>
            <extension base="mpeg7:VisualDType">
            <element name="GlobalCurvatureVector"
                    type="mpeg7:curvatureVectorType"/>
            <element name="PrototypeCurvatureVector"
                    type="mpeg7:curvatureVectorType" minOccurs="0"/>
```

```
                  <element name="HighestPeak" type="mpeg7:unsigned7"/>
                  <element name="Peak" maxOccurs="62">
                        <complexType>
                              <element name="xpeak"
type="mpeg7:unsigned6"/>
                              <element name="ypeak"
type="mpeg7:unsigned3"/>
                        </complexType>
                  </element>
                  <attribute name="numberOfPeaks"
type="mpeg7:unsigned6"/>
            </extension>
      </complexContent>
</complexType>
<simpleType name="curvatureVectorType">
      <restriction base="mpeg7:listOfUnsigned6Type">
            <length value="2"/>
      </restriction>
</simpleType>
```

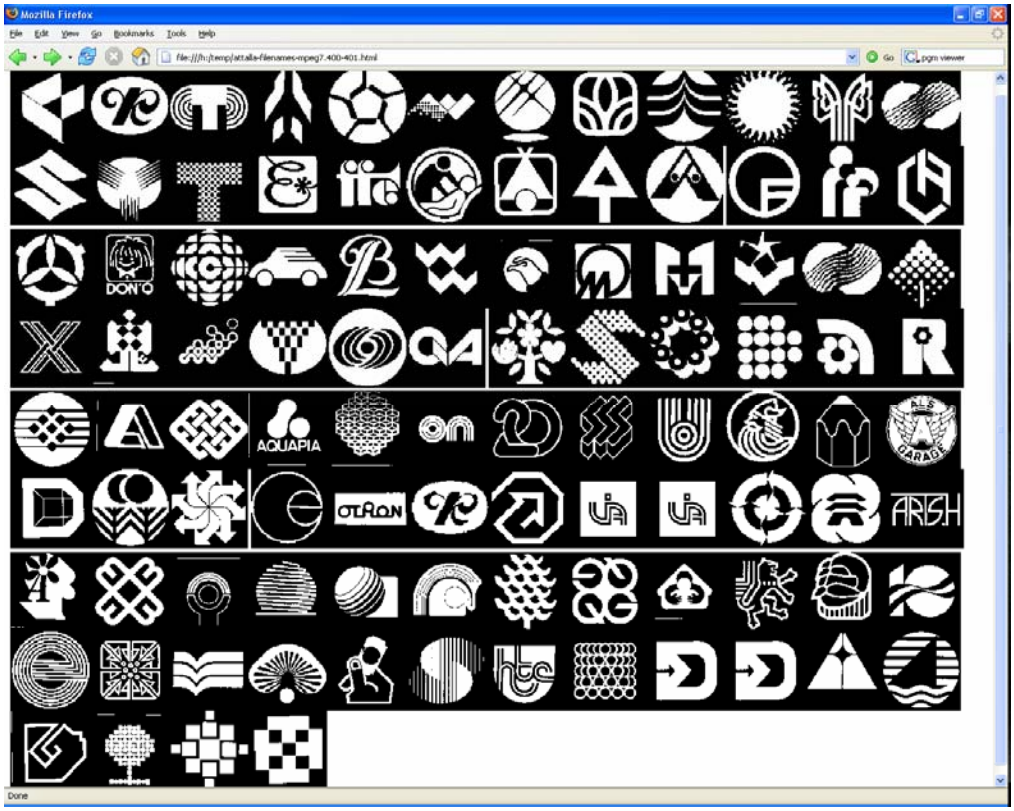
Binary representation syntax

ContourShape {	**Number of bits**	**Mnemonic**
numberOfPeaks	6	uimsbf
GlobalCurvatureVector	2*6	uimsbf
if (numberOfPeaks != 0)		
PrototypeCurvatureVector	2*6	uimsbf
HighestPeakY	7	uimsbf
for (k=1; k<numberOfPeaks; k++) {		
PeakX[k]	6	uimsbf
PeakY[k]	3	uimsbf
}		
}		

2.3 Contour to Centroid Triangulation

The contour-to-centroid triangulation [5] first picks the farthest point fromthe centroid of the shape and use it as the start point of segmenting the contour. It then divides the contour into n equal length arcs, where n can be between 10 and 75, and considers the triangles connecting the endpoints of these arcs with the centroid. It builds a shape descriptor by going clockwise over all triangles, and taking the left interior contour angle, the length of the left side to the centroid, and the ratio contour segment length to contour arc length. To match two descriptors, the triangle parameters are compared to the correspond triangle of the other descriptor, as well as to its left and right neighbor, thereby achieving some form of elastic matching.

On the MPEG-7 image set, this methods works well, as reported in [5], and confirmed by the SIDESTEP system. However, it works not that well on the UKPTO image set, which has a higer complexity.



Contour to Centroid retrieval results on the UKPTO image collection.

DDL representation syntax

```
<complexType name="ContourShapeType" final="#all">
      <complexContent>
            <extension base="mpeg7:VisualDType">
                  <element name="StoredFeature1" type="Segment"
occurs="1*numberOfSegments"/>
                  <element name="StoredFeature2" type="Segment"
occurs="2*numberOfSegments"/>
                  <element name="StoredFeature3" type="Segment"
occurs="3*numberOfSegments"/>
                  <attribute name="numberOfSegments"
type="mpeg7:unsigned5" minInclusive="10" maxInclusive="25"/>
            </extension>
      </complexContent>
</complexType>

<complexType name="Segment">
      <complexContent>
            <element name="StartAngle" type="Angle"/>
            <element name="EndAngle" type="Angle"/>
```

```
            <element name="StartLengthWeight" type=" mpeg7:double"/>
            <element name="EndLengthWeight" type=" mpeg7:double"/>
            <element name="Smoothness" type="mpeg7:double" />
      </complexContent>
</complexType>

<simpleType name="Angle">
      <element name="Value" type="mpeg7:double" minInclusive="0"
maxInclusive="180">
</simpleType>
```

2.4 Shape Context

Shape context [6] is a method that first builds a shape representation for each contour point, using statistics of other contour points 'seen' by this point in quantized angular and distance intervals. The obtained view of a single point is represented as a 2D histogram matrix. To compute a distance between two contours, the correspondence of contour points is established that minimizes the distances of corresponding matrices.

Matching the shapes on their shape contexts can be done by finding the minimum distance from one shape context in set A to another point in set B, see figure 4, left. However, the result will be that all points of A correspond to the best fitting points of B. To obtain symmetry the, min(d(A,B),d(B,A)) could be taken. However this would double the computations. Another approach would be to throw away the best point found from set B; this would mean less histogram matching along the way. Unfortunately at the end the set B becomes small and points from set A have to resort to Shape Contexts with large distances, see figure 4, middle. The Hungarian method works on balanced bipartite graphs, and not all shapes have the same amount of points available. When the amount of points is different then dummy distances (points) are added to the list of distances, the use of dummy values is also mentioned in the article. Afterwards the dummy distances are removed from the point matching result, see figure 4, right.

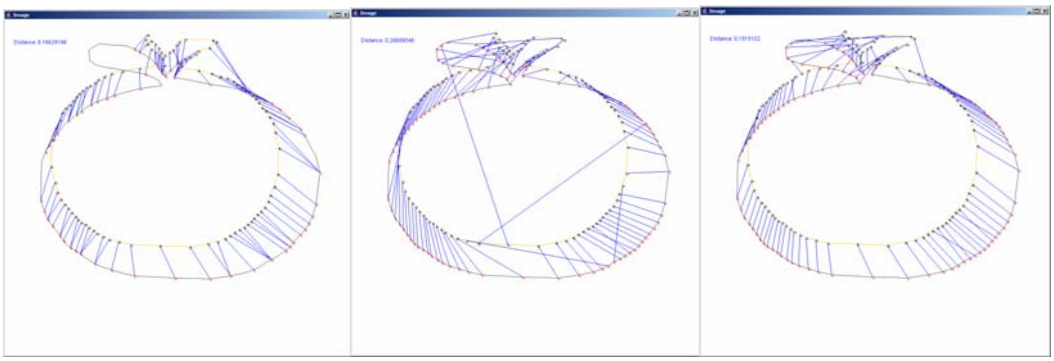


Image 1, 2, and 3

DDL representation syntax

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
 <xs:element name="mpeg7">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="shape"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="shape">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="descriptors"/>
   </xs:sequence>
   <xs:attribute name="name" use="required" type="xs:string"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="descriptors">
  <xs:complexType>
   <xs:sequence>
    <xs:element maxOccurs="unbounded" ref="descriptor"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="descriptor">
  <xs:complexType>
   <xs:sequence minOccurs="1">
    <xs:element maxOccurs="unbounded" ref="ring"/>
   </xs:sequence>
   <xs:attribute name="centerX" use="required" type="xs:unsignedInt"/>
   <xs:attribute name="centerY" use="required" type="xs:unsignedInt"/>
   <xs:attribute name="type" use="required" type="xs:string"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="ring">
  <xs:complexType>
   <xs:sequence minOccurs="1">
    <xs:element maxOccurs="unbounded" ref="wedge"/>
   </xs:sequence>
   <xs:attribute name="number" use="required" type="xs:unsignedInt"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="wedge">
```

```
   <xs:complexType>
    <xs:attribute name="number" use="required" type="xs:unsignedInt"/>
    <xs:attribute name="value" use="required" type="xs:unsignedInt"/>
   </xs:complexType>
  </xs:element>
</xs:schema>
```

2.5 Curve Alignment

The curve alignment method [7] to compare curves only looks at bendings and the lengths of the segments of the curves. The distance between bending and length of the curves is minimized:

$$| ds_1 - ds_2 | + R | d\theta_1 - d\theta_2 |$$

The first term gives the length, and the second the bending. Changing the value of R affects the extend to which the bending counts.
Matching of the curves is done by the dynamic programming algorithm. Each time a maximum of three segments is compared to two segments of the other, as in the template of figure 5, and the value is computed as computed as follows:

$$d(i, j) = \min_{k,l} [d(i-k, j-l) + \delta([i-k, i][j-l, j])]$$

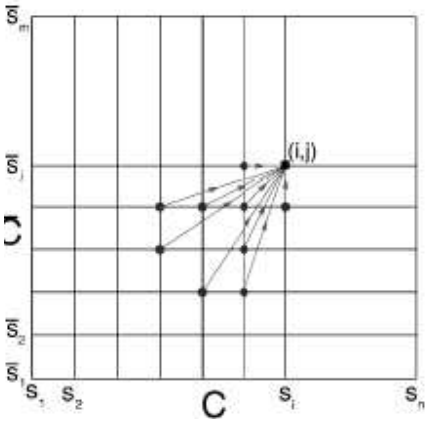


Figure 5: dynamic programming matrix.

DDL representation syntax

```
<mpeg7>
 <complexType name="ContourShapeType" final="#all">
  <complexContent>
   <extension base="mpeg7:VisualDType">
    <element name="curve">
     <simpleType name="listOflengthAngle" base="segment" derivedBy="list"/>
```

```
      </element>
    </extension>
  </complexContent>
 </complexType>
 <complexType name="segment">
   <element name="segmentLength" type="positiveDouble"/>
   <element name="segmentAngle" type="positiveDouble"/>
 </complexType>
 <simpleType name="positiveDouble">
  <restriction base="double">
    <minInclusive value="0.0"/>
  </restriction>
 </simpleType>
</mpeg7>
```

2.6 Moment Invariants

Moment invariants [8] are based on Hu's invariant moments (Hu, 1962). Moments can be calculated by multiplying the x coordinate, raised to a power p, and y coordinates, raised to a power q, of an image by the value of the image at that coordinate. From these image moments normalized central moments can be calculated. These values can be calculated by dividing Hu's moment by the p=0, q=0 moment of the image raised to the power lambda, where lambda equals 1/2 * (p + q) + 1 for p + q = 2,3,4...

The central moment of order (p,q) are defined by:

$$\mu_{pq} = \int\int (x - \bar{x})^p (y - \bar{y})^q \, dxdy$$

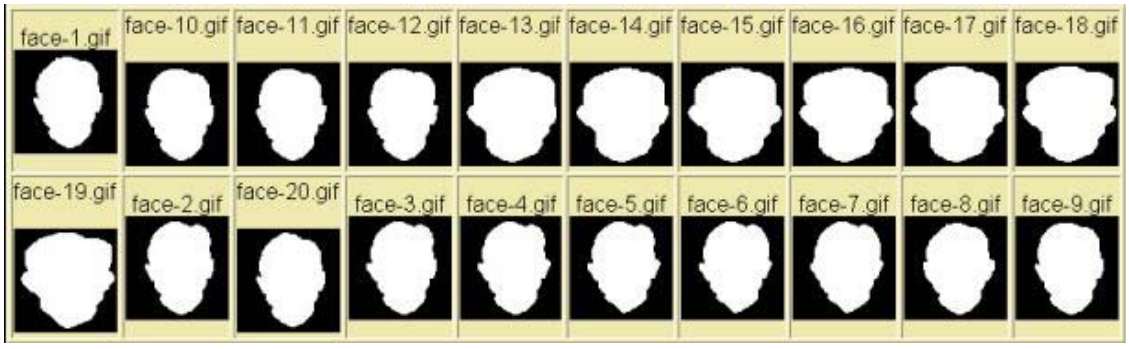


Figure 6: retrieval result of the MPEG-7 class Face based on momen invariants.

DDL representation syntax

Incorporation moment invariant in MPEG 7 can be done using the following DDL:

```
<complexType name="InvariantMoments" final="$\#$all">
  <complexContent>
    <sequence minOccurs="7" maxOccurs="7">
      <element name="inv_moment" type="float" />
    </sequence>
  </complexContent>
</complextType>
```

Binary representation syntax

InvariantMoments {	**Number of bits**	**Mnemonic**
for (k=0; k<7; k++) {		
Float[k]	32	fsbf
}		
}		

2.7 Convex Part Correspondence

Convex parts correspondence [9] is based on an optimal correspondence of contour parts of both compared shapes. The correspondence is restricted so that at least one of element in a corresponding pair is a maximal convex contour part. Since the correspondence is computed on contours simplified by a discrete curve evolution, the maximal convex contour parts represent visually significant shape parts. This correspondence is computed using dynamic programming.

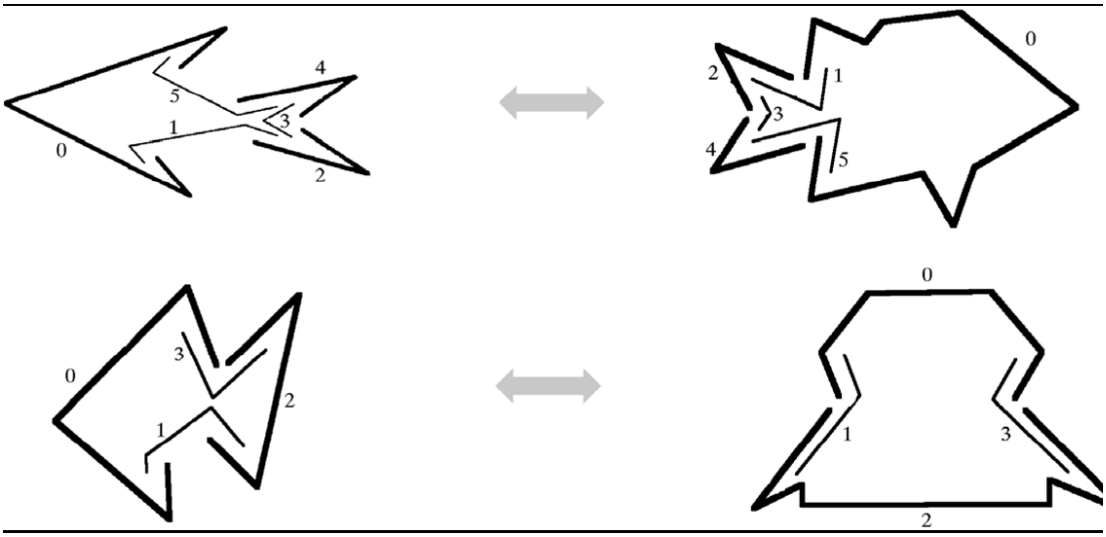


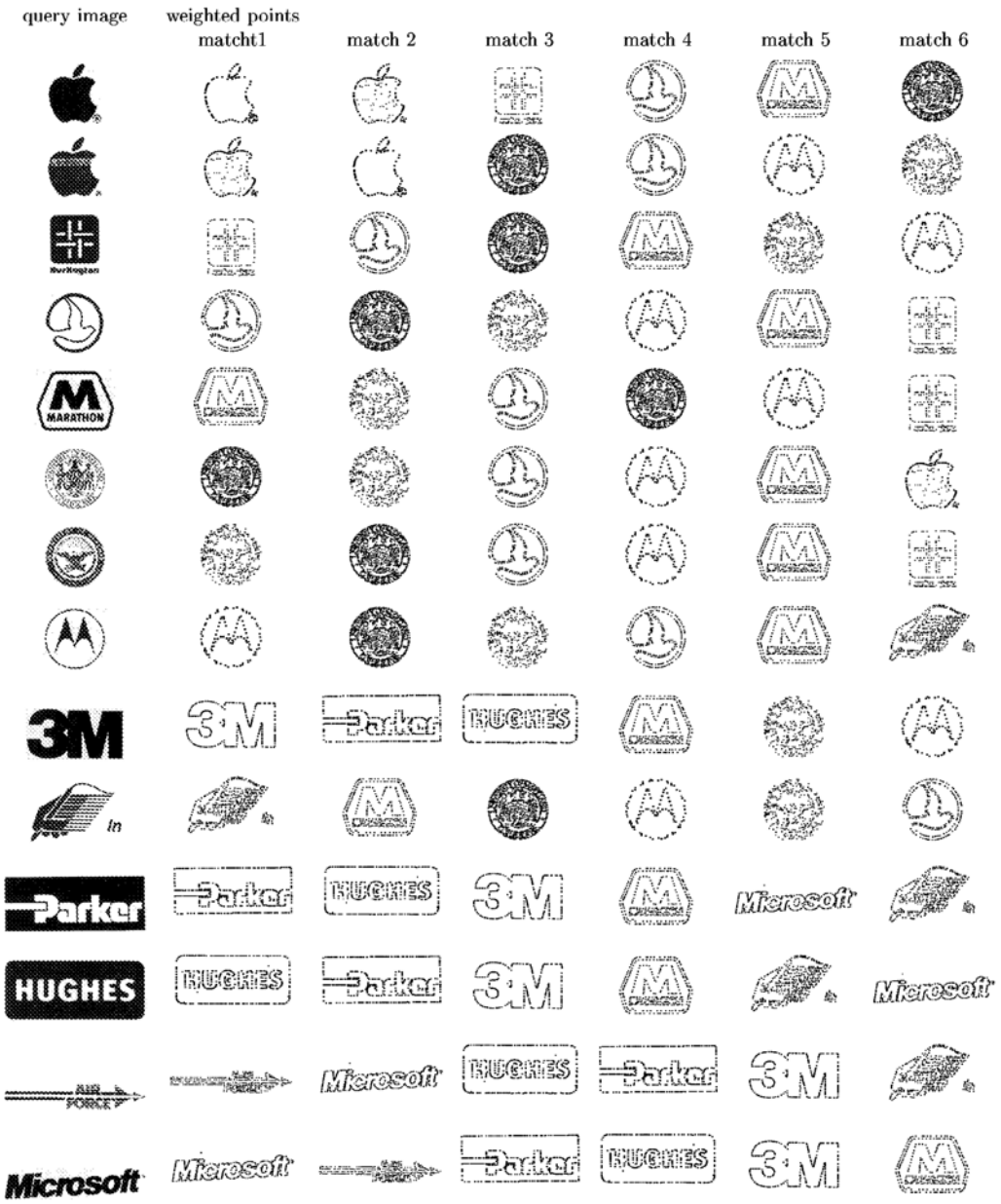
Figure 9: convex part correspondence.

DDL representation syntax

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns="http://chipzero.net/convexparts"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
targetNamespace="http://chipzero.net/Latecki" elementFormDefault="qualified">
  <xs:element name="Mpeg7" final="#all">
    <xs:complexType>
      <xs:sequence>
        <!-- For each part -->
        <xs:element name="part" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence minOccurs="1" maxOccurs="unbounded">
              <xs:element name="length" type="xs:float"/>
              <xs:element name="heading" type="xs:float"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

2.8 Proportional Transportation Distance

The Proportional Transportation Distance (PTD) is an extension of the Earth Movers Distance (EMD). The EMD calculates the minimum amount of work that is needed to fill a set of holes with a set of piles. The distances and sizes (weights) of the holes and piles can differ. However, the EMD is a metric only if the total weight of the piles and holes is equal. The PTD is a pseudo metric, all metric properties hold except for positivity. As opposed to the EMD the weights con't need be equal. With the PTD the weight difference is distributed proportionally. To rewrite the EMD into the PTD a normalization needs to be done, as described in [10]. This is done by summing up all weights for a point set and then dividing each weight by the total sum. Now the total sum for both point sets is 1, so the EMD can be applied to the sets but without positivity.



Fjigure 10: Proportional Transportation Distance retrieval on the UMD logo database.

DDL representation syntax

```
<complexType name="SegLineType">
  <complexContent>
    <extension base="mpeg7:VisualDType">
      <element name="SegID" type="mpeg7:integer" />
      <element name="SegVertex" minOccurs="0" maxOccurs="unbounded">
        <complexType>
          <element name="xCoord" type="mpeg7:unsigned7" />
          <element name="yCoord" type="mpeg7:unsigned7" />
        </complexType>
      </element>
      <attribute name="isClosed" type=" mpeg7:boolean" />
    </extension>
  </complexContent>
</complexType>

<complexType name=”SegmentFile”>
  <complexContent>
    <element name=”line” type=”SegLineType” minOccurs=”0”
maxOccurs=”unbounded” />
  </complexContent>
</complexType>

<complexType name="CoreContourLine">
  <complexContent>
    <extension base="mpeg7:VisualDType">
      <element name=”CoreClassName” type=”mpeg7:string” />
      <element name=”CoreCoord” minOccurs=”0” maxOccurs=”unbounded”>
        <complexType>
          <element name="xCoord" type="mpeg7:unsigned7" />
          <element name="yCoord" type="mpeg7:unsigned7" />
        </complexType>
      </element>
      <attribute name=”numOfCoords” type=”mpeg7:integer” />
    </extension>
  </complexContent>
</complexType>
```

2.9 Skeleton Turning Angle

One representation for shapes is the Turning Angle Function (TAF) which is invariant under rotation, scaling and translation. Traditionally the TAF was applied directly to polygonal shapes which can be extracted from an image, but it can also be applied to straight skeletons which can be derived from these polygons.

The TAF of a skeleton is generated in the same fashion as the TAF of a polygon. We walk around the boundary of the skeleton and record the angle of an edge with respect to the horizontal at each turn we take.

DDL representation syntax

```
<comlexType name="TurningAngleFunctionType" final="#all">
 <complexContent>
  <extention base="mpeg7:VisualDType">
   <element name="Leg">
    <complexType>
     <element name="leg_y" type="mpeg7:unsigned8"/>
     <element name="leg_length" type="mpeg7:unsigned8"/>
    </complexType>
   </element>
   <attribute name="numberOfLegs" type="mpeg7:unsigned8"/>
  </extention>
 </complexContent>
</complexType>
```

Binary representation syntax

TurningAngleFunction {	**Number of bits**	**Mnemonic**
numberOfLegs	8	uimsbf
for (k=1; k<numberOfLegs; k++) {		
LegLength[k]	8	uimsbf
LegY[k]	8	uimsbf
}		
}		

Descriptor components semantics

- numberOfLegs: This attribute specifies the number of legs (horizontal and vertical line segments) as defined by the Turning Angle Function.
- LegLength, LegY: These elements represent the parameters of the Legs that are defined by the Turning Angle Function.

3. References

[1] Akio Yamada, Mark Pickering, Sylvie Jeannin, Leszek Cieplinski, Jens Rainer Ohm, Munchurl Kim (Eds). MPEG-7 Visual part of eXperimentation Model Version 9.0. ISO/IEC JTC1/SC29/WG11/N3914, 2001.

[2] Multimedia content description interface - Part 3 Visual. Leszek Cieplinski, Munchurl Kim, Jens-Rainer Ohm, Mark Pickering, Akio Yamada (eds). ISO/IEC JTC1/SC29/WG11/N4062, 2001.

[3] M. Bober, J. D. Kim, H. K. Kim, Y. S. Kim, W.-Y. Kim, and K. Muller. Summary of the results in shape descriptor core experiment. ISO/IEC JTC1/SC29/WG11/MPEG99/M4869, 1999.

[4] L. J. Latecki, R. Lakaemper, U. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. Proc. CVPR, 2000, 424-429.

[5] Emad Attalla, Pepe Siy. Robust shape similarity retrieval based on contour segmentation polygonal multiresolution and elastic matching. Pattern Recognition 38, 2005, pp. 2229–2241.

[6] Serge Belongie, Jitendra Malik, Jan Puzicha. Shape Matching and Object Recognition Using Shape Context. IEEE PAMI, 24(24), 2002, pp 509-522.

[7] Thomas B. Sebastian, Philip N. Klein, Benjamin B. Kimia. On Aligning Curves. IEEE PAMI 25 (1), 2003, pp. 116-125.

[8] Babu M. Mehtre, Mohan S. Kankanhalli, Wing Foon Lee. Shape Measures for Content Based Image Retrieval: a Comparison. Information Processing and Management 33(3), 1997, pp. 319-337.

[9] L. J. Latecki, R. Lakaemper. Shape Similarity Measure Based on Correspondence of Visual Parts. IEEE PAMI 22, 2000, 1185-119

[10] Panos Giannopoulos, Remco C. Veltkamp. A Pseudo-Metric for Weighted Point Sets. In: Proceedings European Conference on Computer Vision (ECCV 2002), LNCS 2352, Springer, 2002, 715-730.