Representation and Description

- After we have acquired, enhanced, restored and segmented an image, as described in the previous lectures, we need to represent and describe it (for further processing).

- Representation can be based on:
  1. the external characteristics of its regions (for example, edges or contours, that describe regions’ shapes) and
  2. the internal characteristics of its regions (for example, average and/or minimum/maximum grayvalue, grayvalue variance, also called “busyness”, colour or texture).
• In addition to representing an image’s regions, we need a way to describe the regions (for example, a contour can be described in terms of its length, local edge orientation and the number of concavities in the contour).

• We would like the region descriptors to be reasonably insensitive to changes in size, orientation, or location for comparison purposes.

**Representation Schemes**

• Below we present a number of the more common techniques. Typically, these techniques compact the data into a form suitable for further processing.
Chain Codes

- A boundary can be represented as a sequence of straight line segments. Each line segment has an orientation based on 4- or 8-connectivity of boundary pixels. Figure 1 shows the directions for 4- and 8-connectivity.

Figure 1: 4- and 8-connectivity chain code directions.
• Since digital images are acquired on a grid with equal spacing in the $x$ and $y$ dimensions (the pixels), a chain code can be generated by following a boundary in clockwise (or counter-clockwise) order and assigning a direction to the line segment between each 2 adjacent boundary points. Generally 2 problems result:

1. The chain code is quite long and

2. small disturbances along the boundary (caused by noise or less than perfect segmentation) can cause changes in the code not related to the regions shape.

• Figure 2a shows a sampled boundary while Figure 2b shows the same boundary re-sampled. Re-sampling with a larger grid can solve the above
2 problems: the chain code is shorted and the boundary shape is effectively smoothed out. Figure 2c and 2d show the 4-connected and 8-connected chain codes for the re-sampled boundary. The black dot in Figures 2c and 2d is the chain code starting point. Note that the chain codes are the shortest allowable 4-connected or 8-connected boundaries of the region.
Figure 2: (a) a digital boundary, (b) the boundary re-sampled on a larger grid, (c) the 4-connected chain code and (d) the 8-connected chain code.
• The 4-connected chain code is

0033333323221211101101

while the 8-connected chain code is

076666553321212.

Note how the accuracy of the chain code depends on the size of the sampling grid.

• The chain codes depend on the starting point, different starting points yield different chain codes. We can normalize a chain code, making it independent of its starting point, by treating it a circular sequence of the direction numbers and redefine starting point so that the number formed
by the sequence is of minimum (or maximum) magnitude. The 4- and 8-connected chain codes given above are already normalized in this fashion.

- We can normalize for rotation by using the first differences of the chain code instead of the code itself. When the differences are negative we add that negative difference to 4 or 8, depending on the connectivity, to get a positive integer in the ranges $[0, 3]$ or $[0, 7]$. The first difference is computed using the difference between the last and first numbers in the chain code.

- Thus, the 4-connected chain code:

$$003333323221211101101$$
has differences:

1 0 -3 0 0 0 0 1 -1 1 -1 1 0 0 1 -1 0 1 -1

or the difference chain code:

1010000131310013013,

where $-3$ has become 1 and $-1$ has become 3.

- The 8-connected chain code:

076666553321212

has differences:
2 -7 1 0 0 0 1 0 2 0 1 1 -1 1 -1

or the difference chain code:

211000102011717,

where -7 has become 1 and -1 has become 7.

- Theoretically, if two difference chain codes for the same shaped regions (at different orientation) is normalized for starting position then they should be equal.

- Finally, size normalization can be accomplished by re-sampling with a different (unknown?) grid before computing the chain code (find the correct grid size by “trial and error”?).
Polygonal Approximations

- All digital boundaries can be represented perfectly by a polygon comprised of segments (edges) between each adjacent set of boundary points. In general, however, we want to approximate the boundary shape with a polygon containing as few polygonal segments (edges) as possible.

- Suppose one wants to represent shapes with polygons of minimal perimeter. Figure 3a shows a boundary enclosed by square cells (determined by the sampling grid). If we think of the cell boundaries as comprising two walls for the inside and outside boundaries for the enclosure then, if the boundary were treated as a “rubber band” and allowed to shrink, it would take on the shape of the polygon of minimum perimeter, as shown in
Figure 3b. The maximum error of any point on the minimum perimeter polygon is the diameter of a square cell, $\sqrt{2}d$, where the cell dimensions are $d \times d$. If each cell is forced to be centered on its corresponding boundary pixel then this error is reduced by $\frac{1}{2}$.

Figure 3 (a) A boundary enclosed by square cells and (b) the minimum perimeter polygon.
• One **Merging** technique merges points along a boundary until a least squares error line fit exceeds a preset threshold. When this happens, the parameters of the line are saved, the error is reset to 0 and merging starts again. The collection of lines at the end of this merging process represent the boundary. The main problem with this method is that the line endpoints do not necessarily correspond to inflections such as corners. Thus if a long straight line were being processed and a corner was encountered, a number of points pass the corner would have to be processed before the line fit error would exceed the threshold. The result would be that there is no polygonal line endpoint at the corner.

• Another approach is boundary segment **splitting**: divide a segment into two parts recursively until some criterion is satisfied for each part. For
example, one criterion is that the maximum perpendicular distance of a boundary point to the line segment joined by its endpoints not exceed a present threshold. If it does, the maximum point become a vertex and the initial line segment is subdivided into 2 segments. For a closed boundary, the best starting endpoints are the two furthest points on the boundary.

Figure 4. (a) An original boundary, (b) the boundary subdivided into segments based on maximum perpendicular distance, (c) the resulting line segments and (d) the resulting polygon.
• Figure 4a shows a closed boundary while Figure 4b shows the boundary divided by a line segment between its furthest points, \( a \) and \( b \). Lower and upper boundary points, \( c \) and \( d \), have the longest perpendicular distances from the line \( ab \). Hence \( ab \) is split into segments \( ad \) and \( db \) for the lower part of the boundary and into \( ac \) and \( cb \) for the upper part of the boundary. Figure 4c shows that this is indeed a good approximation to the boundary. This approach has the advantage that polygonal vertices tend to correspond to the boundary’s inflection points.

Signatures

• A signature is a 1D functional representation of a 2D boundary.

• One of the simplest ways to represent it is to plot distance from the cen-
troid to the boundary as a function of angle. Figure 5 show the signatures of a circle and a square.

Figure 5: (a) and (b): the signatures of a circle and a box.
• Normalization with respect to rotation requires us to find the same starting point. One way to do this is to select the point furthest from the centroid (assuming this is unique) and using it as the starting point. In general, one could simply shift one signature against another until a match occurs or the signature has been shifted through 360° and no match has been found.

• Normalization with respect to scaling is simple: scale two signatures so that the their minimum and maximum values are 0 and 1 respectively. Then two differently sized regions with the same shape will have the same signatures.
Boundary Segments

- Decomposition of the boundary into significant segments is useful if the boundary has significant concavities.

- The **convex hull**, $H$, of an arbitrary set, $S$, of points is the smallest convex set containing $S$. The set difference, $H - S$, is called the **convex deficiency** set of $S$, denoted as $D$.

- Figure 6a shows an object (set $S$) and its convex deficiency (shaded region). The object boundary can be partitioned by following the contour of $S$ and marking the points at which a transition is made from into or from out of a component of a convex deficiency. Figure 6b shows the result. In principle, this scheme is independent of region size and
orientation.

Figure 6 (a) A region $S$ and its convex deficiency (shaded) and (b) the partitioned boundary.
• Description of an image might be comprised of on image regions’ areas, their convex deficiency areas, the number and location of the partitions of each regions, etc.

**Skeletons**

• This method allows a region to be reduced to a graph via skeletoning or thinning.

• The skeleton of an object’s image is defined by the **medial axis transform** (MAT). If a region, \( R \), has boundary, \( B \), than we find points \( p \) in \( R \) such that \( p \) has two or more closest boundary points. In this case \( p \) belongs to the medial axis or skeleton of \( R \). The concept of “closest” depends on the distance measure used and different distance measures
can produce very different results. Figure 7 shows some MATs using Euclidean distance.

Figure 7: Three simple examples of MATs
Figure 8 (a) and (b) the first two iteration results of a thinning operation and (c) the final result.
• Note how a small change in the border of 7a, shown in 7b, produces a large change in the MAT.

• MAT computation is computationally expensive; algorithms to do it typically iteratively delete edge points of a region subject to the following constraints:

  1. do not remove endpoints,

  2. do not break connectedness and

  3. do not cause excessive erosion of the region.

This type of algorithm can be likened to the way a “prairie fire” burns out (these are the locations of the MAT).
• Figures 8a and 8b show the result of the first and second thinning iterations on a simple region while Figure 8c shows the final result.

• Figure 9 shows another MAT for a more complex object.

Figure 9 A complex thinning example (of a running man).
Fourier Descriptors

Figure 10: A digital boundary and its representation as a complex sequence. The points \((x_0, y_0)\) and \((x_1, y_1)\) are shown (arbitrarily) as the first 2 points in the sequence.
• Figure 10 shows a $K$ point boundary in the $xy$-plane. Coordinate pairs $(x_0, y_0), (x_1, y_1), (x_2, y_2), ..., (x_{k-1}, y_{k-1})$ are encountered in traversing the boundary in counterclockwise order. We can denote each point $(x_k, y_k)$ as $s(k) = [x(k), y(k)]$ for $k = 0, 1, 2, ... K - 1$. Moreover, each pair can be treated as a complex number so that $s(k) = x(k) + jy(k)$. for $k = 0, 1, 2, ..., K - 1$. This reduces a 2D problem to a 1D problem if you think of complex numbers as 1D.

• The discrete Fourier transform of $s(k)$ is:

$$a(u) = \frac{1}{K} \sum_{k=0}^{K-1} s(k) e^{-j2\pi uk/K}$$

for $u = 0, 1, 2, ..., K - 1$. The complex coefficients $a(u)$ are called the Fourier descriptors of the boundary. The inverse Fourier transform of
these coefficients restores \( s(k) \):

\[
s(k) = \sum_{u=0}^{K-1} a(u) e^{j2\pi uk/K}
\]

for \( k = 0, 1, 2, \ldots, K - 1 \).

- Suppose, instead of all the Fourier descriptors, only the first \( P \) coefficients are used. Thus \( a(u) = 0 \) for \( u > P - 1 \). The resulting approximation to \( s(k) \) is:

\[
\hat{s}(k) = \sum_{u=0}^{P-1} a(u) e^{j2\pi uk/K}
\]

for \( k = 0, 1, 2, \ldots, K - 1 \). Although only \( P \) terms are used to obtain each component of \( \hat{s}(k) \), \( k \) still ranges from 0 to \( K - 1 \). That is, the same number of points exists in the approximate boundary nut not as many terms are used in the reconstruction of each point.
• Remember that high frequency components in the Fourier Transform account for fine detail and low frequency components account for the global shape. Thus the smaller $P$ becomes the more fine detail is lost on the boundary.
Figure 11: Examples of the reconstruction of a square from the Fourier descriptors. $P$ is the number of Fourier coefficients used in the reconstruction of the boundary.
• Figure 11 shows a square consisting of $K = 64$ points and the results of reconstruction this boundary for various values of $P$. $P$ has to be more than 8 before the reconstructed boundary looks more like a square than a circle. At $p = 56$ the corner points begin to “break out” of the sequence. At $P = 61$ the curves begin to straighten. Thus a few lower order coefficients capture the gross shape but many higher order terms are needed to accurately define sharp features such as corner and straight lines.

• Shape descriptors should be independent (insensitive) as much as possible to translation, rotation, scale changes and starting point (where points are begun to be processed). Fourier descriptors are not insensitive to these geometrical changes but the changes can be related to simple trans-
formations on the descriptors.

- For rotation we just multiply $s(k)$ by $e^{j\theta}$ to rotate by angle $\theta$:

$$s_r(u) = \frac{1}{K} \sum_{k=0}^{K-1} e^{j\theta} e^{-j2\pi k/K} = a(u)e^{j\theta}$$

for $u = 0, 1, 2, \ldots, K - 1$.

- Using $\Delta_{xy} = \Delta x + j\Delta y$ so that for this amount of translation $s_t(k) = s(k) + \Delta_{xy}$ means

$$s_t(k) = [x(k) + \Delta x] + j[y(k) + \Delta y].$$

Translation has no effect on the Fourier descriptors.

- If we scale the boundary points by $\alpha$, i.e. $s_s(k) = \alpha s(k)$, then its Fourier transform is simply $\alpha a(u)$. 
Transformation | Boundary | Fourier Descriptor
---|---|---
Identity | $s(k)$ | $a(u)$
Rotation | $s_r(k) = s(k)e^{j\theta}$ | $a_r(u) = a(u)e^{j\theta}$
Translation | $s_t(k) = s(k) + \Delta_{xy}$ | $a_t(k) = a(u) + \Delta_{xy}\delta(u)$
Scaling | $s_s(k) = \alpha s(k)$ | $a_s(u) = \alpha a(u)$
Starting Point | $s_p(k) = s(k - k_0)$ | $a_p(u) = a(u)e^{-j2\pi k_0 u/K}$

Table 1: Some basic properties of Fourier descriptors.

- Finally, $s_p(k) = s(k - k_0)$, means redefining the sequence as

$$s_p(k) = s(k - k_0) = s(k - k_0) + jy(k - k_0)$$

which merely changes the starting point of the sequence from $k = 0$ to $k = k_0$. Table 1 summarizes the basic properties of Fourier descriptors.