• Motion segmentation & motion tracking
• Boundary tracking
• Chain codes
• Minimum perimeter polygons
• Signatures
Accumulative Difference Image (ADI) for a moving rectangle.

\[ A_k(x, y) = \begin{cases} 
A_{k-1}(x, y) + 1 & \text{if } |K(x, y) - f(x, y, k)| > T \\
A_{k-1}(x, y) & \text{otherwise}
\end{cases} \]

The accumulator is incremented if the absolute difference between the reference frame and each successive frame is above a threshold. We can also have positive and negative accumulator images.

\[ P_k(x, y) = \begin{cases} 
P_{k-1}(x, y) + 1 & \text{if } |K(x, y) - f(x, y, k)| > T \\
P_{k-1}(x, y) & \text{otherwise}
\end{cases}, \quad N_k(x, y) = \begin{cases} 
N_{k-1}(x, y) + 1 & \text{if } |K(x, y) - f(x, y, k)| < -T \\
N_{k-1}(x, y) & \text{otherwise}
\end{cases} \]

FIGURE 10.59 ADIs of a rectangular object moving in a southeasterly direction. (a) Absolute ADI. (b) Positive ADI. (c) Negative ADI.
How do you construct a static reference image if something is always in motion?
• People Counter
Moore Boundary Tracking Algorithm [1968]:
1. Start with the uppermost, leftmost foreground point \( b_0 \) in the image. Let \( c_0 \) be the west neighbor of \( b_0 \). Going clockwise from \( c_0 \) identify the first non-zero neighbor \( b_1 \) of \( b_0 \). Let \( c_1 \) be the background point immediately preceding \( b_1 \) in the sequence.
2. Let \( b = b_1 \) and \( c = c_1 \).
3. Let the 8-neighbors of \( b \) starting at \( c \) and proceeding clockwise be denoted as \( n_1, n_2, ..., n_8 \). Find the first \( n_k \) which is foreground (i.e., a “1”).
4. Let \( b = n_k \) and \( c = n_{k-1} \).
5. Repeat steps 3 and 4 until \( b = b_0 \) and the next boundary point found is \( b_1 \). The sequence of \( b \) points found when the algorithm stops is the set of ordered boundary points.

Assumptions:
1. The image is binary where 1=foreground and 0=background
2. The image is padded with a border of 0’s so an object cannot merge with the border.
3. There is only one object in this example.

**FIGURE 11.1** Illustration of the first few steps in the boundary-following algorithm. The point to be processed next is labeled in black, the points yet to be processed are gray, and the points found by the algorithm are labeled as gray squares.
The Moore boundary following algorithm repeats until $b = b_0$ and the next boundary point found is $b_1$.

This prevents errors encountered in spurs. Simply finding $b_0$ again is not enough. For example, in this figure the algorithm would start at $b_0$, find $b$, and then come back to $b_0$ stopping prematurely with only $b = b_0$. However, the next boundary point found should be the circled point which is NOT $b_1$. Adding “and the next boundary point found is $b_1$” will cause the algorithm to traverse the lower part of the object.
A Freeman chain code represents a boundary as a connected sequence of straight line segments of specified direction and length.
Chain Codes

Using original pixels usually results in a code which is too long and subject to noise.

Resample original image on a larger grid to reduce code size.

Arbitrarily start chain code here

8-direction chain coded boundary

8-connected code: 0766666453321212
Chain Codes

4-connected code:
10103322

Arbitrarily start chain code here

One practice is to rotate (pick start) to get smallest integer code.

4-connected difference code:
33133030

(START CODE HERE)
Between end and start of boundary

Chain codes can be normalized for rotation by using first differences.

To compute the first difference go in counter-clockwise direction and find number of 90° rotations of direction.

1->0 3-90° rotations ccw
0->1 1-90° rotation ccw
1->0 3-90° rotations ccw
0->3 3-90° rotations ccw
3->3 0-90° rotations ccw
3->2 3-90° rotations ccw
2->2 0-90° rotations ccw
2->1 3-90° rotations ccw
**Figure 11.5** (a) Noisy image. (b) Image smoothed with a 9 × 9 averaging mask. (c) Smoothed image, thresholded using Otsu’s method. (d) Longest outer boundary of (c). (e) Subsampled boundary (the points are shown enlarged for clarity). (f) Connected points from (c).
The goal is to represent the shape in a given boundary using the fewest possible number of sequences.

FIGURE 11.6 (a) An object boundary (black curve). (b) Boundary enclosed by cells (in gray). (c) Minimum-perimeter polygon obtained by allowing the boundary to shrink. The vertices of the polygon are created by the corners of the inner and outer walls of the gray region.
The boundary cells from the previous slide enclose the circumscribed shape.

Traverse the 4-connected boundary of the circumscribed shape.

Concave vertices on this boundary have “mirrors” on the outer boundary. The boundary is described by inner convex and outer concave vertices.

**FIGURE 11.7** (a) Region (dark gray) resulting from enclosing the original boundary by cells (see Fig. 11.6). (b) Convex (white dots) and concave (black dots) vertices obtained by following the boundary of the dark gray region in the counterclockwise direction. (c) Concave vertices (black dots) displaced to their diagonal mirror locations in the outer wall of the bounding region; the convex vertices are not changed. The MPP (black boundary) is superimposed for reference.
MPP Observations:
1. The MPP bounded by a simply connected cellular complex is not self-intersecting.
2. Every convex vertex of the MPP is a W vertex, but not every W vertex of a boundary is a vertex of the MPP.
3. Every mirrored concave vertex of the MPP is a B vertex, but not every B vertex of a boundary is a vertex of the MPP.
4. All B vertices are on or outside the MPP, and all W vertices are on or inside the MPP.
5. The uppermost, leftmost vertex in a sequence of vertices contained in a cellular complex is always a W vertex of the MPP.
Let \( a=(x_1,y_1), b=(x_2,y_2), \) and \( c=(x_3,y_3) \)

\[
A = \begin{bmatrix}
  x_1 & y_1 & 1 \\
  x_2 & y_2 & 1 \\
  x_3 & y_3 & 1 
\end{bmatrix}
\]

\[
\text{sgn}(a,b,c) = \text{det}(A) = \begin{cases} 
  > 0 & \text{if } (a,b,c) \text{ is a counterclockwise sequence} \\
  = 0 & \text{if } (a,b,c) \text{ are collinear} \\
  < 0 & \text{if } (a,b,c) \text{ is a clockwise sequence}
\end{cases}
\]
Definitions:
Form a list whose rows are the coordinates of each vertex and whether that vertex is W or B. The concave vertices must be mirrored, the vertices must be in sequential order, and the first uppermost, leftmost vertex $V_O$ is a W vertex. There is a white crawler ($W_c$) and a black crawler ($B_c$). The $W_c$ crawls along the convex W vertices, and the $B_c$ crawls along the mirrored concave B vertices.
MPP Algorithm:
1. Set $W_C = B_C = V_O$

2. 
   (a) $V_K$ is on the positive side of the line $(V_L, W_C)$ \( \text{sgn}(V_L, W_C, V_K) > 0 \)
   (b) $V_K$ is on the negative side of the line $(V_L, W_C)$ or is collinear with it \( \text{sgn}(V_L, W_C, V_K) \leq 0 \);
   $V_K$ is on the positive side of the line $(V_L, B_C)$ or is collinear with it \( \text{sgn}(V_L, B_C, V_K) > 0 \)
   (c) $V_K$ is on the negative side of the line $(V_L, B_C)$ \( \text{sgn}(V_L, B_C, V_K) < 0 \)

If condition (a) holds the next MPP vertex is $W_C$ and $V_L = W_C$; set $W_C = B_C = V_L$ and continue with the next vertex.

If condition (b) holds $V_K$ becomes a candidate MPP vertex. Set $W_C = V_K$ if $V_K$ is convex otherwise set $B_C = V_K$. Continue with next vertex.

If condition (c) holds the next vertex is $B_C$ and $V_L = B_C$.

Re-initialize the algorithm by setting $W_C = B_C = V_L$ and continue with the next vertex after $V_L$.

3. Continue until the first vertex is reached again.
The fundamental concept is to move the crawlers along the perimeter, calculate the curvatures, and determine if the vertex is a vertex of the MPP.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_0$</td>
<td>(1,4)</td>
<td>W</td>
<td>$V_0$</td>
<td>WC curvature</td>
<td>$V_0$</td>
</tr>
<tr>
<td>$V_1$</td>
<td>(2,3)</td>
<td>B</td>
<td>$V_0$</td>
<td>BC curvature</td>
<td>$V_0$</td>
</tr>
<tr>
<td>$V_2$</td>
<td>(3,3)</td>
<td>W</td>
<td>$V_0$</td>
<td>$V_L$, $W_C$, $V_1$</td>
<td>0</td>
</tr>
<tr>
<td>$V_3$</td>
<td>(3,2)</td>
<td>B</td>
<td>$V_0$</td>
<td>$V_L$, $W_C$, $V_2$</td>
<td>0</td>
</tr>
<tr>
<td>$V_4$</td>
<td>(4,1)</td>
<td>W</td>
<td>$V_0$</td>
<td>$V_L$, $W_C$, $V_3$</td>
<td>&lt;0</td>
</tr>
<tr>
<td>$V_5$</td>
<td>(7,1)</td>
<td>W</td>
<td>$V_0$</td>
<td>$V_L$, $W_C$, $V_4$</td>
<td>&lt;0</td>
</tr>
<tr>
<td>$V_6$</td>
<td>(8,2)</td>
<td>B</td>
<td>$V_0$</td>
<td>$V_L$, $W_C$, $V_5$</td>
<td>&gt;0</td>
</tr>
<tr>
<td>$V_7$</td>
<td>(9,2)</td>
<td>B</td>
<td>$V_0$</td>
<td>$V_L$, $W_C$, $V_6$</td>
<td>&gt;0</td>
</tr>
</tbody>
</table>

© 2002 R. C. Gonzalez & R. E. Woods
Minimum Perimeter Polygon

- 8-connected boundary
- MPP resampled 2x2, 206 vertices
- MPP resampled 3x3, 160 vertices
- MPP resampled 4x4, 127 vertices
- MPP resampled 8x8, 66 vertices
- MPP resampled 16x16, 32 vertices
- MPP resampled 32x32, 13 vertices

566x566 binary image

FIGURE 11.8
(a) 566 × 566 binary image.
(b) 8-connected boundary.
(c) through (i), MMPs obtained using square cells of sizes 2, 3, 4, 6, 8, 16, and 32, respectively (the
numbers of vertices in (c) through (i) are 206, 160, 127, 92, 66, 32, and 13, respectively.

© 2002 R. C. Gonzalez & R. E. Woods
Subdivide a segment successively until a specified criterion is satisfied

1. Identify the two most distant points joined by ab

2. Identify the most distant point up from ab

3. Identify the most distant point down from ab
A signature is a 1-D representation of a boundary.

Signatures can be made invariant to translation but may be sensitive to rotation and scale.

Methods of selecting starting point can make signature independent of rotation:
1. Select point farthest from centroid
2. Select point farthest from centroid along eigenaxis
3. Use a chain code

These signatures are r-θ but there are others such as ψ(tangent)-s.

**FIGURE 11.10**
Distance-versus-angle signatures. In (a) $r(\theta)$ is constant. In (b), the signature consists of repetitions of the pattern:
$$r(\theta) = A \sec \theta \text{ for } 0 \leq \theta \leq \pi/4 \text{ and } r(\theta) = A \csc \theta \text{ for } \pi/4 < \theta \leq \pi/2.$$
FIGURE 11.11
Two binary regions, their external boundaries, and their corresponding $r(\theta)$ signatures. The horizontal axes in (e) and (f) correspond to angles from $0^\circ$ to $360^\circ$, in increments of $1^\circ$. 
The convex hull $H$

The convex deficiency $H-S$

The boundary of the object can be coded by the points where the boundary passes in and out of a convex deficiency

**FIGURE 11.12**
(a) A region, $S$, and its convex deficiency (shaded).
(b) Partitioned boundary.
Median Axis Transformation (MAT) of a region R with border B (guarantees connectivity of the skeleton)
1. For each point p in R find its closest* neighbor on B
2. If p has more than one “closest”* neighbor it belongs to the medial axis (skeleton) of B
* Closest is defined using Euclidian distance

Reduce the structure of a shape to a skeleton. However, as seen in our previous examples a morphological skeleton will not necessarily be connected.

FIGURE 11.13
Medial axes (dashed) of three simple regions.
Thinning

A suitable thinning algorithm is computationally MUCH faster than the MAT.

Definitions
A border point is any pixel with value 1 and at least one 8-connected neighbor with value 0. $N(p_1)$ is the number of non-zero neighbors of $p_1$, i.e., $N(p_1) = p_2 + p_3 + ... + p_8 + p_9$. $T(p_1)$ is the number of 0->1 transitions in the ordered sequence $p_1p_2p_3...p_9p_2$.

**Figure 11.14**
Neighborhood arrangement used by the thinning algorithm.

<table>
<thead>
<tr>
<th></th>
<th>$p_9$</th>
<th>$p_2$</th>
<th>$p_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_8$</td>
<td>$p_1$</td>
<td>$p_4$</td>
<td></td>
</tr>
<tr>
<td>$p_7$</td>
<td>$p_6$</td>
<td>$p_5$</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 11.15**
Illustration of conditions (a) and (b) in Eq. (11.1-4). In this case $N(p_1) = 4$ and $T(p_1) = 3$.

$p_1$ is a boundary point

$N(p_1) = 4$

$T(p_1) = 3$
Algorithm for thinning binary images

Step 1. Mark for deletion any border point which has all of the following:
   (a) $2 \leq N(p_1) \leq 6$ \hspace{1cm} \% don't delete if $p_1$ is an end point or inside region
   (b) $T(p_1) = 1$ \hspace{1cm} \% prevents breaking lines
   (c) $p_2 p_4 p_6 = 0$ \hspace{1cm} \% (c) and (d) say that either $p_2$ AND $p_8$ are 0
   (d) $p_4 p_6 p_8 = 0$ \hspace{1cm} \% or $p_4$ or $p_6$ are 0, i.e., not part of the skeleton

Step 2. Mark for deletion any border point which has all of the following:
   (a) $2 \leq N(p_1) \leq 6$ \hspace{1cm} \% don't delete if $p_1$ is an end point or inside region
   (b) $T(p_1) = 1$ \hspace{1cm} \% prevents breaking lines
   (c) $p_2 p_4 p_8 = 0$ \hspace{1cm} \% (c) and (d) say that either $p_4$ AND $p_6$ are 0
   (d) $p_4 p_6 p_8 = 0$ \hspace{1cm} \% or $p_2$ OR $p_8$ are 0, i.e., not part of the skeleton

Iterate by applying Step 1 to all border points and deleting marked points. Then apply Step 2 to all remaining border points and delete marked points. Continue until no further points are deleted.
Skeletonized ("thinned") image.

Double branch since wider than left side.
The shape number \( n \) is the smallest magnitude first difference chain code. \( n \) is even for closed boundaries.
Definitions:

**Diameter** = \( \max[D(p_i), D(p_j)] \) where \( p_i \) and \( p_j \) are points on the boundary.

**Major axis** is the line segment of length equal to the diameter and connecting two points on the boundary.

**Minor axis** is the line perpendicular to the major axis and of such length that a box passing through the outer four points of intersection of the boundary and the major/minor axes completely enclose the boundary.

This box enclosing the boundary is called the **basic rectangle**.

**Eccentricity** is the ratio of major to minor axis.
1. The major and minor axes

2. The basic rectangle

3. Use the rectangle with n=18 (given) which best approximates the shape of basic rectangle, i.e., 6x3=18.

4. Resample boundary or use polygonal approximation

5. Computer chain code, its first difference, and the corresponding shape number.
1. Represent each point on a digital boundary as \( s(k) = x(k) + jy(x) \)

2. Compute the DFT of the set of boundary points

\[
a(u) = \frac{1}{K} \sum_{k=0}^{K-1} s(k) e^{-j \frac{2\pi uk}{K}}, \quad u = 0, 1, 2, ..., K - 1
\]

3. The coefficients \( a(u) \) are the Fourier descriptors of the boundary

\[
s(k) = \sum_{u=0}^{K-1} a(u) e^{+j \frac{2\pi uk}{K}}
\]

Since \( K \) can be large we usually approximate the boundary by a smaller set of points, i.e., \( P \), so that

\[
s(k) \approx \hat{s}(k) = \sum_{u=0}^{P-1} a(u) e^{+j \frac{2\pi uk}{K}}
\]
There are the same number of points in each reconstructed boundary but only the first $P$ terms of the Fourier boundary descriptor were used to reconstruct the boundary. Basically, this is low-pass filtering of the shape.
Fourier shape reconstruction using the first $P=1434, 286, 144, 72, 36, 18$ and $8$ terms respectively.

**FIGURE 11.20** (a) Boundary of human chromosome (2868 points). (b)–(h) Boundaries reconstructed using 1434, 286, 144, 72, 36, 18, and 8 Fourier descriptors, respectively. These numbers are approximately $50\%, 10\%, 5\%, 2.5\%, 1.25\%, 0.63\%$, and $0.28\%$ of 2868, respectively.
Rotation, scale, and translation of a boundary have simple effects on the Fourier description of that boundary.

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Boundary</th>
<th>Fourier Descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td>( s(k) )</td>
<td>( a(u) )</td>
</tr>
<tr>
<td>Rotation</td>
<td>( s_c(k) = s(k)e^{i\theta} )</td>
<td>( a_c(u) = a(u)e^{i\theta} )</td>
</tr>
<tr>
<td>Translation</td>
<td>( s_t(k) = s(k) + \Delta_{xy} )</td>
<td>( a_t(u) = a(u) + \Delta_{xy} \delta(u) )</td>
</tr>
<tr>
<td>Scaling</td>
<td>( s_s(k) = \alpha s(k) )</td>
<td>( a_s(u) = \alpha a(u) )</td>
</tr>
<tr>
<td>Starting point</td>
<td>( s_p(k) = s(k - k_0) )</td>
<td>( a_p(u) = a(u)e^{-j2\pi k_0 u/K} )</td>
</tr>
</tbody>
</table>

**Table 11.1**

Some basic properties of Fourier descriptors.

- Shifts the DC \((k=0)\) term
- Multiplies each term in a known way
Connect start and stop points and compute perpendicular displacement from this line.

Can compute the mean displacement \( m \) and higher order moments \( \mu_n \)

\[
m = \sum_{i=0}^{K-1} r_i g(r_i)
\]

\[
\mu_n(r) = \sum_{i=0}^{K-1} (r_i - m)^n g(r_i)
\]
Use ratio of white pixels to total area to estimate electrical energy consumption.
Topology - properties that are unaffected by "rubber" sheet deformations

The set of pixels that are connected to any pixel in $S$ is called a connected component of $S$. 

**FIGURE 11.23** A region with two holes.

**FIGURE 11.24** A region with three connected components.
Euler number $E = C - H$

$C$ is the number of connected components; $H$ is the number of holes

$E = C - H = 1 - 1 = 0$

$E = C - H = 1 - 2 = -1$

**FIGURE 11.25**
Regions with Euler numbers equal to 0 and $-1$, respectively.
This polygonal network has:
7 vertices
11 edges
2 faces
3 holes
1 connected region

Euler number $E = V - Q + F$

$E = V - Q + F = 7 - 11 + 2 = -2$

$V$ is the number of vertices
$Q$ is the number of edges
$F$ is the number of faces
Single infrared image

Result of thresholding (largest T before river becomes disconnected) gives 1591 connected components

Single component with largest number of pixels (8479)

Computed skeleton of this component useful for computing length of river branches, etc.