Contours
From Edges…

- Need a higher-level representation of edge pixels.
- Here we will talk about contours.
- Contours correspond to region boundaries.
- Linked list of edge pixels.
  - Can be open or closed.
Criteria...

- **Efficiency**: Simple, compact representation.
- **Accuracy**: Fit image features accurately.
- **Effectiveness**: Suitable for the operations performed in the later stages of the application.
Representing a Contour

The simplest representation of a contour is an ordered list of edge pixels.

- Not compact.
- Not efficient.
- Accurate.
A more powerful representation is fitting a curve having some analytical description (polygonal curves, line-segments etc..).
- More compact.
- Efficient.
- Accurate.
Basic Definitions

- An **edge list** is an ordered set of edge points.
- A **contour** is an edge list or the curve that has been used to represent the edge list.
- A **boundary** is the closed contour that surrounds a region.
Basic Definitions (contd.)

- A curve *interpolates* a list of points if it passes through the points.
- A curve *approximates* a list of points if it passes close to the edge points, but not necessarily passing exactly through them.
Interpolation Vs Approximation

interpolation

approximation
Digital Contours

Let \( P_1 = (x_1, y_1), P_2 = (x_2, y_2), \ldots, P_n = (x_n, y_n) \) be the coordinates of an edge in the edge list.

The length \( S \) of the digital contour can be approximated by adding the lengths of the individual segments between the edge points.

\[
S = \sum_{i=2}^{n} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}
\]
Digital Contours (contd.)

The distance between the end points of a contour is given by:

\[ D = \sqrt{(y_n - y_1)^2 + (x_n - x_1)^2} \]
Digital Contours (contd.)

- Estimate the slope using edge points that are not adjacent:
  - Left k-slope: direction from $p_{i-k}$ to $p_i$
  - Right k-slope: direction from $p_i$ to $p_{i+k}$
  - K-curvature: difference between the left and right k-slopes.
Simple Methods

- Chain codes
- Crack codes
- Slope representation
- Slope density function
- Centroidal profile representation
Chain Codes

- Directions are quantized into 4 or 8 directions.
- Typically, the chain code contains the start pixel address followed by a string of code words.
Chain Codes (contd.)

- 4- connectivity
- 8-connectivity
Chain Codes (contd.)

Algorithm:
1. Start at any boundary pixel, A.
2. Find the nearest edge pixel and code its orientation. In case of a tie, choose the one with the largest (or smallest) code value.
3. Continue until there are no more boundary pixels.
Example 1

8-directional chain code
Chain code: 2 3 3 4 4 5 6 7 6 4 5 7 7 0 1 1 1 2
Example 2

 Boundary pixel orientation: (A),76010655432421

 Chain code: A 111 110 000 001 000 110 101 101
 100 011 010 100 010 001
Limitations of Chain Codes

- Chain code of a boundary depends on the starting point.
- Sensitive to noise.
- Limited set of directions.
Difference Code

- Derivative of chain code.
- Obtained by using the first difference of the chain code.
- Rotation–invariant boundary description.
Example 1

- 4-direction chain code: 1 0 1 0 3 3 2 2
- First difference code: 3 1 3 3 0 3 0
Example 2

- Chain code: 2 3 3 4 4 5 6 7 6 4 5 7 7 0 1 1 12
- Difference code: 1 0 1 0 1 1 1 7 6 1 2 0 1 1 0 0 1 0

![Diagram showing chain code and difference code with a grid and arrows representing directions.](image-url)
Crack Code

- An alternative to the chain code for contour encoding is to use neither the contour pixels associated with the object nor the contour pixels associated with background but rather the line, the "crack", in between them.

- Directions are quantized into 4 directions.
Example

- Chain code: 5 6 6 7 7 0 0
- Crack code: 3 2 3 3 0 3 0 0
Slope Representation (Ψ - s plot)

A continuous version of chain code.

Algorithm:

1. Start at the first edge point and go clockwise around the contour.
2. Estimate the slope and arc length.
3. Plot the slope versus the arc-length.
Example
Slope Representation (contd.)

- Horizontal line segments in the plot correspond to straight lines in the contour.
- Line segments at other orientations correspond to circular arcs.
- Portions of the plot that are not straight line segments correspond to other curve primitives.
Some Comments

- Different starting points cause a shift in the s-axis.
- Rotations cause a shift in the $\Psi$ axis.
- Not very tolerant to noise.
- For a closed contour this plot is periodic.
Slope Density Function

- Histogram of the slopes along the contour.
- Orientation can be determined using correlation of slope histograms.
- Can be very useful for object recognition.
Centroidal Profile Representation

Plot the distance from the centroid to the boundary as a function of angle.
Some Comments

- Invariant to rotation but depends on the starting point.
- Tolerant to noise.
From Contour Representation

- Next problem is to approximate a contour to get a better compact representation.
- Start with straight lines.
- Move on to circular arcs or more complicated primitives.
Representation by Curve Fitting

- Curve Fitting Models:
  - Line segments
  - Circular arcs
  - Conic sections
  - Cubic splines
Evaluating The Goodness of Fit

- **Maximum Absolute Error:**

  \[ MAE = \max_i |d_i| \]

- **Mean Squared Error:**

  \[ MSE = \frac{1}{n \sum \sum d_i^2} \]
Evaluating The Goodness of Fit

- **Normalized Maximum Error**: Ratio of the MAE to the length of the curve.

  \[ NME = \frac{\max_i |d_i|}{s} \]

- **Number of Sign Changes**: In the error is a good indicator of the appropriateness of the curve as a model for the edges in the contour.
Contd..

- Ratio of curve length to the end point distance: This is a good measure of the complexity of the curve.
Polyline Representation

- Splitting
- Merging
- Split and Merge Algorithm
- Hop-Along Algorithm
Splitting

**Algorithm:**

1. Approximate the curve by the line segment joining its end points (A,B).
2. Find the farthest edge point from the line segment.
3. If the normalized maximum error of the point from the line segment is above a threshold then split the line segment into two segments at that point.
4. Repeat the same procedure for each of the two sub-segments until the error is very small.
Example 1
Example 2
Merging

Algorithm:

1. Use the first two edge points to define a line segment.
2. Add a new edge point if it does not deviate too far from the current line segment.
3. Update the parameters of the line segment using least-squares.
4. Start a new line segment when the edge points deviate too far from the line segment.
Split and Merge

**Algorithm:**

1. After recursive subdivision (split), allow adjacent segments to be replaced by a single segment (merge).
2. Alternate applications of split and merge until no segments are merged or split.
Hop-Along Algorithm

1. Start with the first k edges from the list.
2. Fit a line segment between the first and the last edges in the sublist.
3. If the normalized maximum error is too large, shorten the sublist to the point of maximum error. Return to step 2.
4. If the line fit succeeds, compare the orientation of the current line segment with that of the previous line segment. If the lines have similar orientations, replace the two line segments with a single line segment.
5. Make the current line segment the previous line segment and advance the window of edges so that there are k edges in the sublist. Return to step 2.
Some Comments

Polyline representations are more economical than edge points.

We can make the errors as small as desired by splitting the contour into very small line segments.
Circular Arcs

Algorithm:
1. Initialize the window of vertices to the 3 end points of the first 2 line segments in the polyline.
2. Compute the ratio of the length of the part of the contour corresponding to the 2 line segments to the distance between the end points. If the ratio is large then leave the first line segment unchanged, advance the window of vertices by one vertex, and repeat this step.
3. Fit a circle through the 3 vertices.
4. Calculate the normalized error and number of sign changes.
Circular Arcs (Contd.)

5. If the normalized maximum error is too large or the number of sign changes is too small then leave the first line segment unchanged, advance the window of vertices and return to step 2.

6. If the circle fit succeeds, then try to include the next line segment in the circular arc. Repeat this step until no more line segments can be subsummed by this circular arc.

7. Advance the window to the next 3 polyline vertices after the end of the circular arc and return to step 2.
Conic Sections and Cubic Splines

- Conic sections correspond to hyperbolas, parabolas, and ellipses (2\textsuperscript{nd} degree polynomials).
- Cubic splines correspond to 3\textsuperscript{rd} degree polynomials.
- Conic sections and cubic splines allow more complex contours to be represented using fewer segments.
Fourier Descriptors

- Fourier coefficients are used to represent a contour.
  - Boundary of an object is expressed as a sequence of coordinates
    \( u(n) = [x(n), y(n)], n=0,1,2,\ldots N-1 \)
  - We can represent each coordinate pair as a complex number so that
    \( u(n) = x(n) + jy(n), n=0,1,2,\ldots N-1 \)
Fourier Descriptors (contd.)

- For a closed curves $u(n)$ is periodic:
- The DFT is given by:
  \[
u(n) = \sum_{k=0}^{N-1} a(k)e^{j2\pi kn/N}, \quad 0 \leq n \leq N - 1\]
- The inverse DFT is given by:
  \[a(k) = \frac{1}{N} \sum_{n=0}^{N-1} u(n)e^{-j2\pi kn/N}\]
- The complex coefficients $a(k)$ are called Fourier descriptors (FD) of the boundary.
Fourier Descriptors (contd.)

- Use lower order FD’s
  - If only the first $M$ coefficients are used
    $$\hat{u}(n) = \sum_{k=0}^{M-1} a(k) e^{j2\pi kn/N}, 0 \leq n \leq N - 1$$
  - $\hat{u}(n)$ is an approximation of $u(n)$
  - The approximation depends on $M$. 

30 June 2003
EE6358 Computer Vision
Ramadevi Majoju
Fourier Descriptors (contd.)

- Simple geometric transformations:
  - Translation: \( u(n) + t \rightarrow a(k) + t\delta(k) \)
  - Rotation: \( u(n)e^{j\theta} \rightarrow a(k)e^{j\theta} \)
  - Scaling: \( su(n) \rightarrow sa(k) \)
  - Starting point: \( u(n-t) \rightarrow a(k)e^{j2\pi kt/N} \)
Summary

- **Representation Using interpolation**
  - Contour representation using edge points.
  - Contour representation using curve fitting models.

- **Contour Representation using approximation**
  - Fourier Descriptors