Image Content Representation

• Representation for
  – curves and shapes
  – regions
  – relationships between regions
Reliable Representation

• **Uniqueness**: must uniquely specify an image
  – otherwise dissimilar images might have similar representations

• **Invariance**: must be invariant to viewing conditions: translation, rotation, scale invariant, viewing angle changes and symmetric transformations

• **Efficiency**: computationally efficient
More Criteria

- **Robustness**: resistance to moderate amounts of deformation and noise
  - distortions and noise should at least result in variations in the representations of similar magnitude

- **Scalability**: must contain information about the image at many levels of detail
  - images are deemed similar even if they appear at different view-scales (resolution)
Image Recognition

- Match the representation of an unknown image with representations computed from known images (model based recognition)
- Matching takes in two representations and computes their distance
  - the more similar the images are the lower the their distance is (and the reverse)
Shape Representation

• The shapes are represented by contours or curves
• *Contour*: a linked list of edge pixels
  – open or closed region boundaries
  – much space overhead
• *Curve*: the mathematical model of a contour
  – polygons
  – splines
Shape Matching

- The algorithm takes in two shapes and computes
  - their distance
  - the correspondences between similar parts in the shapes
Polylines

- Sequence of line segments approximating a contour joined end to end
- Polyline: \( P = \{(x_1,y_1),(x_2,y_2), \ldots,(x_n,y_n)\} \)
  – for closed contours \( x_1=x_n, y_n=y_1 \)
- Vertices: points where line segments are joined
Polyline Computation

- Start with a line segment between the end points
  - find the furthest point from the line
  - if its distance $> T$ a new vertex is inserted at the point
  - repeat the same for the two curve pieces
Hop along algorithm

- Works with lists of $k$ pixels at a time and adjusts their line approximation
  1. take the first $k$ pixels
  2. fit a line segment between them
  3. compute their distances from the line
  4. if a distance $> T$ take $k' < k$ pixels and go to step 2
  5. if the orientation of the current and previous line segments are similar, merge them to one
  6. advance $k$ pixels and go to step 2
B-Splines

• Piecewise polynomial curves
  – better aesthetic representation
  – analytic properties
  – continuity of first and second derivatives
Types of Splines

• The spline varies less than the guided polygon
• The spline lies between the convex hull of groups of \( n+1 \) consecutive points
  – \( n \) is the degree of the interpolative polynomial
  – cubic splines are frequently used
Spline Interpolation

- The interpolant though a set of points $x_i, i=1,2,...n$ is a vector piecewise polynomial function $x(t)$

$$x(t) = \sum_{i=0}^{n+1} v_i C_i(t)$$

- $v_i$’s are the vertices of the guided polygon
- there are $n+2$ coefficients
- the additional 2 coefficients are determined from boundary conditions
- by setting curvature = 0 at end points we get
  - $v_1 = (v_0 + v_2)/2$, $v_n = (v_{n-1} + v_{n+1})/2$
\[ x(t) = \sum_{i=0}^{n+1} v_i C_i(t) \]
Spline Base Functions

- \( C_i(s) \): piecewise cubic polynomials

\[
\begin{align*}
C_0(t) &= \frac{t^3}{6} \\
C_1(t) &= \frac{3t^3 + 3t^2 + 3t + 1}{6} \\
C_2(t) &= \frac{3t^3 - 6t^2 + 4}{6} \\
C_3(t) &= \frac{t^3 + 3t^2 - 3t + 1}{6}
\end{align*}
\]

- \( x(t) \) in \((i,i+1)\) is computed as

\[
x(t) = v_{i-1}C_3(t-1) + v_iC_2(t-i) + v_{i+1}C_1(t-i) + v_{i+2}C_0(t-i)
\]
Chain Codes

- Recording of the change of direction along a contour (4 or 8 directions)
  - start at the first edge and go clockwise
  - the derivative of the chain code is a rotation invariant

(a) Chain code: 11010103033032212322
(b) Derivative: 1003131331300133031130
Comments

• Independency of starting point is achieved by “rotating” the code until the sequence of codes forms the minimum possible integer

• Sensitive to noise and scale

• Extension: approximate the boundary using chain codes \((l_i, a_i)\), (Tsai & Yu, IEE PAMI 7(4):453-462, 1985)
  – \(l_i\): length
  – \(a_i\): angle difference between consecutive vectors
Matching Chain Codes

• Editing distance \( D(A,B) \)
  – \( D(A,B) \): minimum cost to transform \( A \) to \( B \)
  – \textit{operations}: “insert”, “delete”, “change”
  – \textit{costs}:
    • cost of changing \( \uparrow \) to \( \downarrow \) is 2
    • cost of changing \( \uparrow \) to \( \rightarrow \) is 1
    • cost of deletion is 2
    • cost of insertion is 2
}\[
A: \text{aabbccdd} \\
C: \text{a bbb dd} \\
D(A, C) = 4
\]

\[
B: \text{aaabccccdd} \\
C: \text{a bbb dd} \\
D(B, C) = 4
\]
Matching Algorithm

- Compute $D(A,B)$
  - $\#A$, $\#B$ lengths of $A$, $B$
  - 0: null symbol
  - $R$: cost of an edit operation
  - $D(0,0) = 0$
  - for $i = 0$ to $\#A$: $D(i,0) = D(i-1,0) + R(A[i] \rightarrow 0)$;
  - for $j = 0$ to $\#B$: $D(0,j) = D(0,j-1) + R(0 \rightarrow B[j])$;
  - for $i = 0$ to $\#A$
    - for $j = 0$ to $\#B$
      1. $m_1 = D(i,j-1) + R(0 \rightarrow B[j])$;
      2. $m_2 = D(i-1,j) + R(A[i] \rightarrow 0)$;
      3. $m_3 = D(i-1,j-1) + R(A[i] \rightarrow B[j])$;
      4. $D(i,j) = \min\{m_1, m_2, m_3\}$;
<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>j</td>
<td></td>
<td>a</td>
<td>b</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>a</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>c</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>c</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>c</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>d</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

*initialization cost*

*total cost*
**Ψ-s Slope Representation**

- Plot of tangent Ψ versus arc length s
  - Ψ-s is a representation of the shape of a contour
  - line segments → horizontal line segments in Ψ-s
  - circular arcs → other line segments in Ψ-s
  - use Ψ-s to segment a curve into lines and arcs
  - for a closed contour Ψ-s is periodic
  - Ψ-s is translation and scale invariant
  - the derivative of Ψ-s is also rotation invariant
$\Psi$-s Examples

From Ballard and Brown’84

a) triangular curve
b) regions of high curvature
c) resultant segmentation
Fourier Representation

- Fourier transform of contour representation
  - \( u(n) = x(n) + j y(n) \), \( n = 0, 1, 2, \ldots, N - 1 \) or
  - \( u(n) = \Psi(n) - 2\pi n/L \) (subtracts rising component)
  - for closed curves \( u(n) \) is periodic

\[
\begin{align*}
  u(n) &= \sum_{k=0}^{N-1} a(k) e^{\frac{j2\pi kn}{N}}, 0 \leq n \leq N - 1 \\
  a(k) &= \frac{1}{N} \sum_{n=0}^{N-1} u(n) e^{-\frac{j2\pi kn}{N}}
\end{align*}
\]
Fourier Descriptors (FDs)

- The complex coefficients $a(k)$ are called *Fourier Descriptors (FD)* of the boundary
  - use the lower order FD’s
  - if only the first $M$ coefficients are used
    \[
    u(n) = \sum_{k=0}^{M-1} a(k) e^{\frac{j2\pi kn}{N}}, 0 \leq n \leq M - 1
    \]
  - $u(n)$ is an approximation of $u(n)$
  - the approximation depends on $M$

- Shape distance: distance between vectors of FD’s
FDs and Invariance

• Simple geometric transformations:
  – translation: \( u(n) + t \rightarrow a(k) + t\delta(\kappa) \)
  – rotation: \( u(n)e^{j\theta} \rightarrow a(k)e^{j\theta} \)
  – scaling: \( su(n) \rightarrow sa(k) \)
  – starting point: \( u(n - t) \rightarrow a(k) e^{j2\pi tk/N} \)

• FDs of \( \Psi\)-s: invariant

• FDs of \( (x(n), y(n)) \): see Wallace & Winz “Efficient 3-D Aircraft Recognition using FDs”, CGIP, 13:99-126, 1980
FDs and Occluded Shapes

- FDs of derivative of $\Psi$-s: rotation invariance
- Ignoring $a(0)$: translation invariance
- Normalizing all FD’s by $|a(1)|$: scale invariance
- Vector of $M$ coefficients starting from $|a(2)|$
Moment Invariants [Hu 62]

• An object is represented by its binary image

• A set of 7 features can be defined based on central moments

\[ m_{pq} = \sum_{(x,y)\in R} x^p y^q, \quad \bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}} \]

\[ \mu_{pq} = \sum_{(x,y)\in R} (x - \bar{x})^p (y - \bar{y})^q, \quad p, q = 0, 1, 2... \]
Central Moments [Hu 62]

\[
\begin{align*}
\phi_1 &= \mu_{20} + \mu_{02} \\
\phi_2 &= (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2 \\
\phi_3 &= (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2 \\
\phi_4 &= (\mu_{30} + \mu_{12})^2 + (3\mu_{21} + \mu_{03})^2 \\
\phi_5 &= (\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12})[2(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] + \\
&\quad + (3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03})[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2] \\
\phi_6 &= (\mu_{20} - \mu_{02})[(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2] + 4\mu_{11}(\mu_{30} + \mu_{12})(\mu_{21} + \mu_{03}) \\
\phi_7 &= (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})[3(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] - \\
&\quad - (\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03})[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2]
\end{align*}
\]

- Invariant to translation and rotation
- Use \( \eta_{pq} = \mu_{pq}/\mu_{00}^{\gamma} \) where \( \gamma = (p+q)/2 + 1 \) for \( p+q = 2, 3 \ldots \) instead of \( \mu \)'s in the above formulas to achieve scale invariance
Scale-Space Descriptions (SSD)

- Shape matching using representations at various level of detail (resolution)
- **Method**: “Scale-Space Descriptions and Recognition of Planar Curves”, F. Mokhtarian and A. Mackworth, IEEE PAMI 1986
- **SSD**s were originally proposed by Witkin
Zero Crossings

- **SSD**: representation of “zero-crossings” of the curvature $k$ for all possible values of $\sigma$
- Curve: $\{x(t), y(t)\}$, $t$ in $[0, 1]$
- Curvature:
  - $k = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{(\dot{x}^2 + \dot{y}^2)^{3/2}}$

\[ y' = \frac{dy}{dx}, \quad y'' = \frac{d^2 y}{dx^2} \]
\[ \dot{x} = \frac{dx}{dt}, \quad \dot{y} = \frac{dy}{dt} \]
\[ \ddot{x} = \frac{d^2 x}{dt^2}, \quad \ddot{y} = \frac{d^2 y}{dt^2} \]
Curvature

• Compute $k$ on $x(t), y(t)$ at various levels of detail $\Rightarrow$ convolve with

$$g(t, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{t^2}{2\sigma^2}}$$

$$x(t, \sigma) = x(t) \ast g(t, \sigma) = \int_{-\infty}^{\infty} x(u) \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(t-u)^2}{2\sigma^2}} \, du$$

$$y(t, \sigma) = y(t) \ast g(t, \sigma) = \ldots.$$
Smoothing a Curve

\[ \kappa(t, \sigma) \]

Fig. 1. Shoreline of Africa

\[ \kappa = \frac{x\dot{y} - y\dot{x}}{(x^2 + y^2)^{3/2}}. \]
SSD of Africa
Matching SSDs

• Variant of A* algorithm
  1. compare the two higher curves first
  2. compare the curves included starting from the two higher curves etc. until all included curves are matched
  3. compare the curves next to the two higher

• Some curves may be missing
  – assign a cost for missing curves
Matching Scale-Space Curves

- \(D(A,B) = |h_1 - h_2| + |ll_1 - ll_2| + |lr_1 - lr_2|\)
- Treat translation and scaling: compute \((d,k)\)
  - \(t' = kt + d, k = h_1/h_2, d = |d_1 - d_2|, \sigma' = \kappa\sigma\)
  - Normalize \(A, B\) before matching
- Cost of matching: least cost matching
Relational Structures

• Representations of the relationships between objects
  – Attributed Relational Graphs (ARGs)
  – Semantic Nets (SNs)
  – Propositional Logic

• May include or combined with representations of objects
Attributed Relational Graphs (ARGs)

- Objects correspond to nodes, relationships between objects correspond to arcs between nodes
  - both nodes and arcs may be labeled
  - label types depend on application and designer
  - usually feature vectors
  - recognition is based on graph matching which is $NP$-hard
**ARG for Cup**

- **Node feature:** compactness = area/perimeter$^2$
- **Arc features:** bigger ($\text{area}_1/\text{area}_2$), adjacency (percentage of common boundary), distance between centers of gravity
**ARG for Face**

- **Node feature**: perimeter $l$
- **Arc features**: relative distance $r$, angle with the horizontal $a$
ARG for Doll

- Connection graph representing the connections between the pieces of an object
Semantic Nets (SNs)

- Generalization of ARG
  - SNs represent information at the high – semantic level
  - e.g. a representation of chairs around a table
Hierarchical SNs (HSN)

- Information organized in
  - ISA and PART-OF hierarchies
  - ISA: generalization hierarchy, generalization of classes and relationships between instances and classes
  - PART-OF: relationships between parts and whole
  - Inheritance: lower level classes inherit properties of the higher level classes
    - House → Building
    - Camel → Mammal
Example *HSN*

*top of a transistor*

*SN of a silicon chip*

*transistor → silicon chip*
Long / Short Term Memory

• Long Term Memory (LTM): schema or model representation of an image at high-semantic level
  – virtual classes in C++

• Short Term Memory (STM): representation of instances to LTM objects
  – instances to virtual classes
Propositional Representations

• Collection of facts and rules in information base
  – new facts are deduced from existing facts
  • \textit{transistor}(\textit{region}_1)
  • \textit{transistor}(\textit{region}_2)
  • \textit{greater}(\textit{area}(\textit{region}_1), 100.0) \&
    \textit{less}(\textit{area}(\textit{region}_1), 4.0) \&
    \textit{is-connected}(\textit{region}_1,\textit{region}_2) \& \textit{base}(\textit{region}_2)
  \rightarrow \textit{transistor}(\textit{region}_2)
Comments

• **Pros**: 
  – clear and compact
  – expandable representations of image knowledge

• **Cons**: 
  – non-hierarchical
  – not easy to treat uncertainty and incompatibilities
  – complexity of matching
Matching Relational Structures

- Matching between $A$, $B$ is transformed to a graph or sub-graph isomorphism problem
  - *graph isomorphism*: one to one mapping of nodes and arcs between the structures of $A$, $B$ (NP-hard!)
  - *sub-graph isomorphism*: isomorphism of a sub-graph of $A$ and $B$ (harder!!)
  - *double sub-graph isomorphism*: isomorphism of a sub-graph of $A$ and a sub-graph of $B$ (even harder!!!)
(a), (b) are isomorphic

(a), (c) have many sub-graph isomorphisms

(a), (d) have many double sub-graph isomorphisms
Matching Algorithm

• Find all sub-graph isomorphisms
• Branch and bound search with backtracking
  – at each step expand a partial solution at all possible directions
  – when search fails (a partial solution can’t be expanded) backtrack to an earlier partial solution
  – the cost of complete solution is an upper bound to prune the expansion of non-promising solutions with greater cost
  – keep the least cost complete solution
the graph of (a) has to be matched with the graph of (b)

arcs are unlabeled

different shapes denote different shape properties: different shapes cannot be matched

partial matches

complete matches
Matching Cost

• **Cost of matching:** the matching with the *least cost*
  
  – *cost*: distance between their feature vectors
  
  – cost of matching nodes plus to the cost of matching arcs
  
  – plus the cost of missing nodes and arcs
    (sometimes extra or missing nodes or arcs in model graph are ignored)
query

model
Least Cost Matching