Edge linking

• Edge detection rarely finds the entire set of edges in an image. Normally there are breaks due to noise, non-uniform illumination, etc.

• If we want to obtain region boundaries (for segmentation) we need to link edge pixels into longer edge curves
  – This process needs to be able to bridge gaps in detected edges due to the reason mentioned above

• Two types of approaches
  – Local/Regional
  – Global
Edge Linking
Local and regional approaches

• Local edge linking
  – Compute the gradient magnitude and gradient angle images
  – If a pixel in N has similar gradient orientation and magnitude to any of its neighbors link them.
    • Linking can be implemented by assigning an unique integer identifier to linked edge segment.
  – What to do about gaps?
    • Consider a larger region than immediate neighbors for potential links.
Regional edge linking

• Regional edge linking
  – Find curves in moderate sized regions (not local, not global) of the image

• Example
  – Given an ordered set of points known to be on the same boundary, fit a polynomial model \((x(s), y(s))\) to the edge curve.
  – The model can be evaluated at any \(s\) which allows for interpolation between the original detected edge points (hence closing gaps)
To get a binary edge map we can threshold the gradient magnitude of the image.

Choosing a threshold is a trade-off between losing weak edges and keeping edges due to noise.

- Smoothing can help with this problem to some extent.
FIGURE 10.26
(a) Original head CT image of size $512 \times 512$ pixels, with intensity values scaled to the range $[0, 1]$. (b) Thresholded gradient of smoothed image. (c) Image obtained using the Marr-Hildreth algorithm. (d) Image obtained using the Canny algorithm. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)
Finding global lines in images

• We can search for lines in images globally
  – Given n points detected as edge pixels
  – Consider every pair of points as a line candidate
    • There are $n(n-1) \sim n^2$ candidate lines
  – Find all points that lie close to each line candidates. If a candidate line has enough points located close then it is marked as a valid line
    • Have to check n points for each candidate line; therefore, the total complexity is on the order $n^3$
  – There is a better way: Hough transform
Parameter space

- Consider an edge point \((x_i, y_i)\)
  - Infinitely many lines can pass through \((x_i, y_i)\)
  - Any slope, intercept pair \((a, b)\) satisfying \(y_i = a \cdot x_i + b\) defines a line that passes through \((x_i, y_i)\)

- Rewrite as \(b = y_i - a \cdot x_i\).
  - Fix \((x_i, y_i)\), vary \(a\) and solve for \(b\)
  - This is the equation of a line in the ab-plane.
  - The *ab-plane* is called the **parameter space**.
Lines in parameter space

- Now consider a second point \((x_j, y_j)\). This point yields a second line in the parameter space.
- The two lines in the parameter space intersect at a point \((a', b')\). These are the slope and intercept parameters of the line (in the **xy-space**) passing through both \((x_i, y_i)\) and \((x_j, y_j)\).
- In fact all points on the line passing through both \((x_i, y_i)\) and \((x_j, y_j)\) will have lines in the parameter space that pass through \((a', b')\).
Hough transform

• Draw lines in the parameter space for all detected edge points \((x_i, y_i)\)

• Find points in the parameter space where a large number of parameter-space lines intersect
  – These locations are the parameters for the lines in the xy-space which have a lot of evidence supporting them.

• Linear complexity in \(n\) (number of edges)

• One difficulty is that the slope parameter \((a)\) approaches infinity as the line in xy-space becomes close to vertical.
  – This would create a problem with the implementation of the Hough transform
We can use the normal representation of a line instead.

\[ x \cos \theta + y \sin \theta = \rho \]

Then the parameter space is the \( \theta \rho \)-plane.

Points in the \( xy \)-space correspond to sinusoidal curves in the \( \theta \rho \)-plane.
Hough transform implementation

• Divide the \( \theta \rho \)-plane into discrete accumulator cells.
• \( A_{ij} \) is the accumulator value for cell corresponding to \((\theta_i, \rho_j)\).
• Range \( \theta \): -90° to 90° and \( \rho \): -D to D (D is length of image diagonal)
• How many cells should we use?
Hough transform implementation

- Initialize all $A_{ij}$ to 0
- For each edge point $(x_k, y_k)$
  - Let $\theta$ equal all of the allowed subdivision values and solve for corresponding $\rho$.
  - Round resulting to nearest integer
  - If a choice of $\theta$ results in $\rho$ then $A(p,q) = A(p,q) + 1$
Hough transform implementation

• Threshold $A_{ij}$ to find strong lines in the image
• Number of subdivisions determines the accuracy of the parameters of the detected lines.
  – More subdivisions = higher accuracy
  – However, the trade-off is that the detection can become less robust with more subdivisions.
  – As the number of subdivisions increases even the strongest line will produce a smaller $A_{ij}$.
  – Therefore a smaller threshold will be needed, possibly resulting in the detection of some false lines.
Hough transform example
Generalized Hough transform

• The Hough transform can be extended to find shapes in an image that can be expressed as a mathematical equation with parameters.
  – Examples:
    • circle \((x-c_1)^2 + (y-c_2)^2 = c_3^2\).
      This will require a 3D parameter-space
    • Ellipse centered at origin \((x/a)^2+(y/b)^2 = 1\).
      This will require a 2D parameter-space

• The complexity of the Hough transform increases with the number of parameters (dimension of the parameter-space). Usually more than 3-4 parameters is not feasible.