Edge Detection
Edge Detection

- Edge = a point in the image where intensities are changing rapidly
- We have looked at the Sobel edge operator
  - Does digital approximation to first derivative
  - Then take magnitude of the gradient

\[
\begin{array}{ccc}
\frac{\partial}{\partial x} & -1 & 0 & +1 \\
-2 & 0 & +2 \\
-1 & 0 & +1 \\
\end{array}
\quad
\begin{array}{ccc}
\frac{\partial}{\partial y} & -1 & -2 & -1 \\
0 & 0 & 0 \\
+1 & +2 & +1 \\
\end{array}
\]

Sobel operators

\[|\nabla f| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}\]

- But this does not identify edge points ... it simply gives a gradient magnitude at each pixel
Sobel Edge Detector Example

Original image “pout.tif”

Gradient magnitude, using Sobel masks
Threshold Edge Operator Results

- We can threshold edge magnitudes, to produce binary image

Low threshold  
High threshold  
Non-maxima suppression

- Problem – only want one response to an edge
- Solution – We take the point that is the local maximum of the gradient magnitude (in the direction of the gradient)
Derivatives amplify noise

First and second derivatives of noisy ramp edge

- Even a small amount of noise greatly affects the output
- Need to smooth the image before taking derivatives
Scale Space Edge Operators

• Intensity changes occur at different scales in an image – we need operators of different sizes to detect them
Scale-Space

- The space of images created by applying a series of operators of different scales
- Gaussians of different sizes can be used to filter the image at different scales

\[ G_\sigma(x, y) = \frac{1}{2\pi \sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \]

- Convolving with a Gaussian blurs the image, wiping out all structure much smaller than \( \sigma \)
Derivative of Gaussian

• The gradient of the smoothed image is

\[ \nabla [G_\sigma * I] = [\nabla G_\sigma] * I \]

• The gradient operators are

\[ \nabla G_\sigma = \begin{pmatrix} \frac{\partial G_\sigma}{\partial x} \\ \frac{\partial G_\sigma}{\partial y} \end{pmatrix}^T \]

\[ = (-x \quad -y) \frac{1}{\sigma^3} \exp \left( -\frac{x^2 + y^2}{2\sigma^2} \right) \]

• An edge point is a peak in the gradient magnitude, in the direction of the gradient
Canny Edge Operator

- We can derive the optimal edge operator to find step edges in the presence of white noise, where “optimal” means
  - Good detection (minimize the probability of detecting false edges and missing real edges)
  - Good localization (detected edges must be close to the true edges)
  - Single response (return only one point for each true edge point)

- Canny found that a very good approximation to the optimal operator is the first derivative of a Gaussian, in the direction of the gradient
  - Then suppress nonmaxima along this direction

- Algorithm:
  - Convolve image with derivative of Gaussian operators (dG/dx, dG/dy)
  - Find the gradient direction at each pixel; quantize into one of four directions (north-south, east-west, northeast-southwest, northwest-southeast)
  - If magnitude of gradient is larger than the two neighbors along this direction, it is a candidate edge point
Edge Linking

- We want to join edge points into connected curves or lines
  - This facilitates object recognition

- Problem:
  - Some edge points along the curve may be weak, causing us to miss them
  - This would result in a broken curve

- Solution:
  - We use a high threshold to make sure we capture true edge points
  - Given these detected points, link additional edge points into contours using a lower threshold (“hysteresis”)

- Algorithm
  - Find all edge points greater than $t_{\text{high}}$
  - From each strong edge point, follow the chains of connected edge points in both directions perpendicular to the edge normal
  - Mark all points greater than $t_{\text{low}}$
Matlab

- \([E, \text{thresh}] = \text{edge}(I, \text{‘canny’}, \text{thresh}, \text{sigma});\)
  - \(\text{thresh}\) is \([t\text{Low} \ t\text{High}]\)
  - \(\text{sigma}\) is std deviation of Gaussian

- Try
  - Varying sigma
    ```matlab
    for s = 0.5:0.5:5
      s
      E = \text{edge}(I, \text{‘canny’}, [], s);
      \text{imshow}(E);
      pause;
    end
    ```
  - Varying thresholds
    ```matlab
    for tHigh = 0.05:0.05:0.4
      tHigh
      E = \text{edge}(I, \text{‘canny’}, [0.4*tHigh tHigh], 1.5);
      \text{imshow}(E);
      pause;
    end
    ```
  - See effect of using only one threshold
    - Let \(\text{thresh} = [t\text{Low} \ t\text{Low}]\) – picks up too many edges
    - Let \(\text{thresh} = [t\text{High} \ t\text{High}]\) – misses too many edges

Image “house.jpg”