Edge Detection

Slides from Cornelia Fermüller and Marc Pollefeys
Edge detection

- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels
Origin of Edges

- Edges are caused by a variety of factors:
  - depth discontinuity
  - surface normal discontinuity
  - surface color discontinuity
  - illumination discontinuity

- Edges are caused by a variety of factors
Edge detection

1. Detection of short linear edge segments (edgels)
2. Aggregation of edgels into extended edges
3. Maybe parametric description
Edge is Where Change Occurs

- Change is measured by derivative in 1D
- Biggest change, derivative has maximum magnitude
- Or 2\textsuperscript{nd} derivative is zero.
Image gradient

- The gradient of an image:
  \[ \nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right] \]

- The gradient points in the direction of most rapid change in intensity.

- The gradient direction is given by:
  \[ \nabla f = [0, \frac{\partial f}{\partial y}] \]

- The gradient direction is given by:
  \[ \theta = \tan^{-1}\left( \frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}} \right) \]
  - Perpendicular to the edge

- The edge strength is given by the magnitude:
  \[ ||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \]
How discrete gradient?

- By finite differences
  - \( f(x+1,y) - f(x,y) \)
  - \( f(x, y+1) - f(x,y) \)
The Sobel operator

- Better approximations of the derivatives exist
  - The *Sobel* operators below are very commonly used

\[
\begin{array}{ccc}
\frac{1}{8} & -1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{ccc}
\frac{1}{8} & 1 & 2 & 1 \\
0 & 0 & 0 \\
-1 & -2 & -1 \\
\end{array}
\]

- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn’t make a difference for edge detection
  - the 1/8 term *is* needed to get the right gradient value, however
Gradient operators

(a): Roberts’ cross operator  
(b): 3x3 Prewitt operator  
(c): Sobel operator  
(d) 4x4 Prewitt operator

\[
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
0 & 1 \\
-1 & 0 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
1 & 0 \\
0 & -1 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
-1 & 0 \\
-1 & 0 \\
-1 & 0 \\
-1 & 0 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
-1 & 0 \\
-1 & 0 \\
-1 & 0 \\
-1 & 0 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
-1 & 0 \\
-2 & 0 \\
-1 & 0 \\
-1 & 0 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
1 & 2 \\
0 & 0 \\
-1 & -2 \\
-3 & -3 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
-1 & 1 \\
-1 & 1 \\
-1 & 1 \\
-1 & 1 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
-1 & 3 \\
-1 & 3 \\
-1 & 3 \\
-1 & 3 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
1 & 3 \\
1 & 3 \\
1 & 3 \\
1 & 3 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
3 & 3 \\
3 & 3 \\
3 & 3 \\
3 & 3 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
3 & 3 \\
3 & 3 \\
3 & 3 \\
3 & 3 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
\end{array}
\quad
\begin{array}{cc}
\Delta_1 & \Delta_2 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
-3 & -3 \\
\end{array}
\end{array}
Finite differences responding to noise

Increasing noise ->
(this is zero mean additive gaussian noise)
Smoothing reduces noise

- Generally expect pixels to “be like” their neighbours
  - surfaces turn slowly
  - relatively few reflectance changes
- Generally expect noise processes to be independent from pixel to pixel
- Implies that smoothing suppresses noise, for appropriate noise models
- Scale
  - the parameter in the symmetric Gaussian
  - as this parameter goes up, more pixels are involved in the average
  - and the image gets more blurred
  - and noise is more effectively suppressed
Solution: smooth first

- Look for peaks in \( \frac{\partial}{\partial x}(h \ast f) \)
Derivative theorem

\[
\frac{\partial}{\partial x} (h \ast f) = (\frac{\partial}{\partial x} h) \ast f
\]

- This saves us one operation:
Second derivative zero

- How to find second derivative?
- \( f(x+1, y) - 2f(x,y) + f(x-1,y) \)
- In 2D
- What is an edge?
  - Look for zero crossings
  - With high contrast
Laplacian of Gaussian: Marr-Heldrith

- Consider \( \frac{\partial^2}{\partial x^2} (h \ast f) \)

\[ f \]

\[ \frac{\partial^2}{\partial x^2} h \]

\[ (\frac{\partial^2}{\partial x^2} h) \ast f \]
2D edge detection filters

\[ h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}} \]

Gaussian

\[ \frac{\partial}{\partial x} h_\sigma(u, v) \]

derivative of Gaussian

\[ \nabla^2 h_\sigma(u, v) \]

Laplacian of Gaussian

\[ \nabla^2 \]

is the **Laplacian** operator:

\[ \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \]
Edge detection by subtraction

original
Edge detection by subtraction

smoothed (5x5 Gaussian)
Edge detection by subtraction

Why does this work?

smoothed – original
(scaled by 4, offset +128)
Gaussian - image filter

Gaussian

delta function

Laplacian of Gaussian
Optimal Edge Detection: Canny

• Assume:
  – Linear filtering
  – Additive Gaussian noise

• Edge detector should have:
  – Good Detection. Filter responds to edge, not noise.
  – Good Localization: detected edge near true edge.
  – Single Response: one per edge

• Detection/Localization trade-off
  – More smoothing improves detection
  – And hurts localization.
Optimal Edge Detection: Canny

- Smoothing: Noise Removal
- Gradient Computation
- Non-Maximal Suppression: Thinning
- Double Thresholding
- Hysteresis
The Canny edge detector

- original image (Lena)
The Canny edge detector

- norm of the gradient
The Canny edge detector

- Thresholding (thick edges)
The Canny edge detector

- thinning
Effect of $\sigma$ (Gaussian kernel size)

- The choice depends on what is desired
  - large $\sigma$ detects large scale edges
  - small $\sigma$ detects fine features
Multi-Scale Edge Detection
Multi-Scale Edge Detection

- Edges in coarser level do not disappear in finer levels
- New edges are added
- Coarser level edges are most important
- Advances like a hierarchy
Scale Integration

- Different resolution images in different levels
- How do we know where the coarser level edges are in the finer edge detected image
- Seems very complex yet eye does it easily
Witkin’s Explanation

- If we do a continuous subsampling
  - Not possible in digital domain
- Edges are retained, new edges are added with refinement
Identifying parametric edges

- Can we identify lines?
- Can we identify curves?
- More general
  - Can we identify circles/ellipses?
- Voting scheme called Hough Transform
Hough Transform

- Only a few lines can pass through \((x,y)\)
  \[-mx + b\]
- Consider \((m,b)\) space
- Red lines are given by a line in that space
  \[-b = y - mx\]
- Each point defines a line in the Hough space
- Each line defines a point (since same \(m,b\))
How to identify lines?

- For each edge point
  - Add intensity to the corresponding line in Hough space
- Each edge point votes on the possible lines through them
- If a line exists in the image space, that point in Hough space will get many votes and hence high intensity
- Find maxima in Hough space
- Find lines by equations $y = mx + b$
Example

Input Image

Rendering of Transform Results

Distance from Centre

Angle
Problem with \((m,b)\) space

- Vertical lines have infinite \(m\)
- Polar notation of \((d, \theta)\)
- \(d = x\cos\theta + y\sin\theta\)
Basic Hough Transform

1. Initialize \( H[d, \theta] = 0 \)

2. for each edge point \( I[x, y] \) in the image
   for \( \theta = 0 \) to 180
       \( H[d, \theta] += 1 \)

3. Find the value(s) of \( (d, \theta) \) where \( H[d, \theta] \) is maximum

4. The detected line in the image is given by
Extensions

- Use the image gradient
  
  1. same
  
  2. for each edge point $I[x,y]$ in the image, compute unique $(d, \theta)$ based on image gradient at $(x,y)$
      
      $H[d, \theta] += 1$

  3. same

  4. same

- Give more votes for stronger edges
- Change the sampling of $(d, \theta)$ to give more/less resolution
- The same procedure can be used with circles, squares, or any other shape