CAP 4453
Robot Vision
Dr. Gonzalo Vaca-Castaño
gonzalo.vacacastano@ucf.edu
Administrative details

• Homework due Friday 11:59pm via webcourses
• Point 5 – refers to filter center in second row
• Any other question?
Robot Vision

6. Edge detection II
Credits

• Some slides comes directly from:
  • Yogesh S Rawat (UCF)
  • Noah Snavely (Cornell)
  • Ioannis (Yannis) Gkioulekas (CMU)
  • Mubarak Shah (UCF)
  • S. Seitz
  • James Tompkin
  • Ulas Bagci
  • L. Lazebnik
Short Review from last class
Edge detectors

• Gradient operators
  • Prewit
  • Sobel
• Marr-Hildreth (Laplacian of Gaussian)
• Canny (Gradient of Gaussian)
### Gradient operators edge detector algorithm

1. **Compute derivatives**
   - In x and y directions
   - Use Sobel or Prewitt filters

2. **Find gradient magnitude**

3. **Threshold gradient magnitude**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sobel</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>-2</td>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prewitt</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
Marr-Hildreth edge detector algorithm

1. Smooth image by Gaussian filtering
2. Apply Laplacian to smoothed image
   • Used in mechanics, electromagnetics, wave theory, quantum mechanics
3. Find Zero crossings
   • Scan along each row, record an edge point at the location of zero-crossing.
   • Repeat above step along each column
Laplacian of Gaussian vs Derivative of Gaussian filtering

Zero crossings are more accurate at localizing edges.

Laplacian of Gaussian filtering

Derivative of Gaussian filtering

zero-crossing

peak
45 years of boundary detection

[Pre deep learning]
Precision Recall

Precision = \frac{\text{How many relevant items are selected?}}{\text{How many selected items are relevant?}}

Recall = \frac{\text{How many selected items are relevant?}}{\text{How many relevant items are selected?}}
Precision Recall

Edges according to method

Groundtruth

Non-Edges

relevant elements

false negatives

true negatives

true positives

false positives

selected elements

How many selected items are relevant?

How many relevant items are selected?

Precision =

Recall =
Precision Recall

Edges according to method

relevant elements

false negatives
true positives
false positives
true negatives

selected elements

How many selected items are relevant? How many relevant items are selected?

Precision =

Recall =
Edge Detection with Deep Learning
Canny edge detector
Design Criteria for Edge Detection

- Good detection: find all real edges, ignoring noise or other artifacts
- Good localization
  - as close as possible to the true edges
  - one point only for each true edge point

![Diagram showing true edges, poor robustness to noise, poor localization, and too many responses.]
Problems

• We get thick edges
• Redundant, especially if we are going to be searching in places where edges are found
Solution

- Identify the local maximums
- Called “non-maximal suppression”
Canny Edge detector algorithm

1. Smooth image with Gaussian filter
2. Compute derivative of filtered image
3. Find magnitude and orientation of gradient
4. Apply “Non-maximum Suppression”
5. Apply “Hysteresis Threshold”
Canny Edge detector algorithm

1. Smooth image with Gaussian filter

\[ S = I \ast g(x, y) = g(x, y) \ast I \]

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

2. Compute derivative of filtered

\[ \nabla S = \nabla (g \ast I) = (\nabla g) \ast I \]

\[ \nabla g = \begin{bmatrix} \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial y} \end{bmatrix} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} \]

\[ \nabla S = \begin{bmatrix} g_x \\ g_y \end{bmatrix} \ast I = \begin{bmatrix} g_x \ast I \\ g_y \ast I \end{bmatrix} \]
Canny Edge detector algorithm

1. Smooth image with Gaussian filter

\[ S = I \ast g(x, y) = g(x, y) \ast I \]

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

2. Compute derivative of filtered image

\[ \nabla S = \begin{bmatrix} g_x \\ g_y \end{bmatrix} \ast I = \begin{bmatrix} g_x \ast I \\ g_y \ast I \end{bmatrix} \]

\[ \nabla g = \begin{bmatrix} \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial y} \end{bmatrix} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} \]
Canny Edge detector algorithm

1. **Smooth image with Gaussian filter**
2. **Compute derivative of filtered image**
Canny Edge detector algorithm

1. Smooth image with Gaussian filter
2. Compute derivative of filtered image
3. **Find magnitude and orientation of gradient**
4. Apply “Non-maximum Suppression”
5. Apply “Hysteresis Threshold”
Canny Edge detector algorithm

3. Find magnitude and orientation of gradient

original image (Lena)
Canny Edge detector algorithm

3. Find magnitude and orientation of gradient

original image (Lena)
Canny Edge detector algorithm

3. Find magnitude and orientation of gradient

\[ \Theta = \text{atan2}(S_y, S_x) \]
Canny Edge detector algorithm

1. Smooth image with Gaussian filter
2. Compute derivative of filtered image
3. Find magnitude and orientation of gradient
4. **Apply “Non-maximum Suppression”**
5. **Apply “Hysteresis Threshold”**
Canny Edge detector algorithm

4. Apply “Non-maximum Suppression”

Goal: keep pixels along the curve where magnitude is largest

How to: looking for a maximum along a slice normal to the curve
Canny Edge detector algorithm

4. Apply “Non-maximum Suppression”

Goal: keep pixels along the curve where magnitude is largest

How to: looking for a maximum along a slice normal to the curve

That is the direction of the gradient!
Canny Edge detector algorithm

4. Apply “Non-maximum Suppression”

- Suppress the pixels in $|\nabla S|$ which are not local maximum

\[
M(x,y) = \begin{cases} 
|\nabla S|(x,y) & \text{if } |\nabla S|(x,y) > |\Delta S|(x',y') \\
& \text{& } |\Delta S|(x,y) > |\Delta S|(x'',y'') \\
0 & \text{otherwise}
\end{cases}
\]

$x'$ and $x''$ are the neighbors of $x$ along normal direction to an edge.
Canny Edge detector algorithm

4. Apply “Non-maximum Suppression”

Consider a 3x3 set of pixels.

We need to examine triples along the directions shown to see if the center pixel is a peak (in magnitude) compared to its two neighbors.
Canny Edge detector algorithm

4. Apply “Non-maximum Suppression”

So, we need to isolate a triple, and ask about the three adjacent values.

There are 4 such directions, one triple per direction.
Canny Edge detector algorithm

4. Apply “Non-maximum Suppression”

So, consider the 4 directions

For each direction, we will ask whether the center mag value exceeds the mag value of neighbor on one side, and neighbor on other side; if yes, mark as peak.
Canny Edge detector algorithm

4. Apply “Non-maximum Suppression”

For a center pixel, do we ask the maxing question for all four directions???
Canny Edge detector algorithm

4. **Apply “Non-maximum Suppression”**

No, we should only do the maxing test for the direction that is dictated by the direction of the edge.

We always want to max-test in the direction perpendicular to the edge, i.e., across the edge. This means in the direction of the gradient vector.
Canny Edge detector algorithm

4. Apply “Non-maximum Suppression”

So, given these 4 possible directions, each direction will be called up, if the gradient vector coincides with the max-test direction.
Canny Edge detector algorithm

4. Apply “Non-maximum Suppression”

Hence, we ascertain the gradient vector’s direction, and then depending on it, we proceed to the max-test
Canny Edge detector algorithm

4. Apply “Non-maximum Suppression”

We know the angles of each direction.
Canny Edge detector algorithm

4. Apply “Non-maximum Suppression”

We break up the angle space into 4 cones of directions
Canny Edge detector algorithm

4. **Apply “Non-maximum Suppression”**

For example, if it is found that the gradient at the center pixel is between -22.5 and +22.5, which max-test should we employ?
Canny Edge detector algorithm

4. Apply “Non-maximum Suppression”

So, put $s = \text{sign in}$

For each pixel (i.e., double-for loop)

Get pixel's gradient direction, $\text{Dir}$

If $-22.5 < \text{Dir} \leq 22.5$

Employ Horizontal Max-Test

else if $+22.5 < \text{Dir} \leq +67.5$

Employ Test involving SW and NE pixels

else if $-67.5 < \text{Dir} \leq -22.5$

Employ Test involving SE and NW pixels

else

Employ Vertical test
Canny Edge detector algorithm

4. **Apply “Non-maximum Suppression”**

- Suppress the pixels in $|\nabla S|$ which are not local maximum

$$M(x, y) = \begin{cases} 
|\nabla S|(x, y) & \text{if } |\nabla S|(x, y) > |\Delta S|(x', y') \\
& \text{and } |\Delta S|(x, y) > |\Delta S|(x'', y'') \\
0 & \text{otherwise}
\end{cases}$$

$x'$ and $x''$ are the neighbors of $x$ along normal direction to an edge.
Canny Edge detector algorithm

4. Apply “Non-maximum Suppression”

\[ |\Delta S| = \sqrt{S_x^2 + S_y^2} \]

For visualization
\[ M \geq \text{Threshold} = 25 \]
Comparison

Gradient Thresholding

With non-maximal suppression
Canny Edge detector algorithm

1. Smooth image with Gaussian filter
2. Compute derivative of filtered image
3. Find magnitude and orientation of gradient
4. Apply “Non-maximum Suppression”
5. Apply “Hysteresis Threshold”
Hysteresis Thresholding

• Edges tend to be continuous
• Still threshold the gradient
• Use a lower threshold if a neighboring point is an edge
• The “Canny Edge Detector” uses all of these heuristics
Canny Edge detector algorithm

5. Apply “Hysteresis Threshold”

- If the gradient at a pixel is
  - above “High”, declare it as an ‘edge pixel’
  - below “Low”, declare it as a “non-edge-pixel”
  - between “low” and “high”

  - Consider its neighbors iteratively then declare it an “edge pixel” if it is connected to an ‘edge pixel’ directly or via pixels between “low” and “high”.
Canny Edge detector algorithm

5. Apply “Hysteresis Threshold”

- If the gradient at a pixel is
  - above “High”, declare it as an ‘edge pixel’
  - below “Low”, declare it as a “non-edge-pixel”
  - between “low” and “high”

  - Consider its neighbors iteratively then declare it an “edge pixel” if it is connected to an ‘edge pixel’ directly or via pixels between “low” and “high”.
Canny Edge detector algorithm

5. Apply “Hysteresis Threshold”

- Scan the image from left to right, top-bottom.
  - The gradient magnitude at a pixel is above a high threshold declare that as an edge point
  - Then recursively consider the neighbors of this pixel.
    - If the gradient magnitude is above the low threshold declare that as an edge pixel.
Canny Edge detector algorithm

5. Apply “Hysteresis Threshold”

\[ M \]

regular
\[ M \geq 25 \]

Hysteresis
\[ \text{High} = 35 \]
\[ \text{Low} = 15 \]
Canny Edge detector algorithm

Before non-max suppression

After non-max suppression
Canny Edge detector algorithm

Threshold at low/high levels to get weak/strong edge pixels
• Do connected components, starting from strong edge pixels

Final Canny Edge
Canny Edge detector algorithm

Final Canny Edge
Effect of $\sigma$ (Gaussian kernel spread/size)

The choice of $\sigma$ depends on desired behavior

- large $\sigma$ detects large scale edges
- small $\sigma$ detects fine features

Source: S. Seitz
Questions?