Announcements

• Assignment 1 will be returned on Tuesday
• Assignment 2 due Thursday, February 14 (But do it before).
• Midterm: Thursday, February 14
• Oscar will answer questions about the midterm on Tuesday.

Edges

1. Object boundaries
2. Surface normal discontinuities
3. Reflectance (albedo) discontinuities
4. Lighting discontinuities (shadow boundaries)

Effects of Noise

• Consider a single row or column of the image
  – Plotting intensity as a function of position gives a signal

Where is the edge??

(from Srinivasa Narasimhan)

Edge is Where Change Occurs: 1-D

• Change is measured by derivative in 1D

Take Taylor series expansion of f(x) about x₀
\[ f(x) = f(x₀) + f'(x₀)(x-x₀) + \frac{1}{2} f''(x₀)(x-x₀)^2 + \cdots \]

Consider Samples taken at increments of h and first two terms, we have
\[ f(x₀+h) = f(x₀) + f'(x₀)h + \frac{1}{2} f''(x₀)h^2 \]
\[ f(x₀-h) = f(x₀) - f'(x₀)h + \frac{1}{2} f''(x₀)h^2 \]

Subtracting and adding f(x₀+h) and f(x₀-h) respectively yields

Numerical Derivatives

Convolve with

First Derivative: \[-1/2h  0 1/2h\]
Second Derivative: \[-1/h^2 2/h^2 -1/h^2\]

Can often drop h or h² in denominator

Yielding [1 0 1] and [-1 2 -1] kernels.
Implementing 1-D Edge Detection

1. Filter out noise: convolve with Gaussian
2. Take a derivative: convolve with [-1 0 1]
   - We can combine 1 and 2.
3. Find the peak of the magnitude of the convolved image: Two issues:
   - Should be a local maximum.
   - Should be sufficiently high.

Canny Edge Detector

1. Smooth image by filtering with a Gaussian
2. Compute gradient at each point in the image.
3. At each point in the image, compute the direction of the gradient and the magnitude of the gradient.
4. Perform non-maximal suppression to identify candidate edgels.
5. Trace edge chains using hysteresis thresholding.

2D Edge Detection: Canny

1. Filter out noise
   - Use a 2D Gaussian Filter. $J = G \ast I$
2. Take a derivative -- gradient
   - Compute the magnitude of the gradient
   - Compute the direction of the gradient

Gradient

- Given a function $f(x,y)$ -- e.g., intensity is $f$
- Gradient equation: $\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$.
- Represents direction of most rapid change in intensity

<table>
<thead>
<tr>
<th>$\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$</td>
</tr>
</tbody>
</table>

- Gradient direction: $\theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$
- The edge strength is given by the gradient magnitude

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

There are three major issues:
1. The gradient magnitude at different scales is different; which scale should we choose?
2. The gradient magnitude is large along thick trail; how do we identify the significant points?
3. How do we link the relevant points up into curves?
Non-maximum suppression

Loop over every point $q$ in the image, decide whether $q$ is a candidate edge point using gradient direction at $q$, find two points $p$ and $r$ on adjacent rows (or columns).

If $|\nabla I(q)| > |\nabla I(p)|$ and $|\nabla I(q)| > |\nabla I(r)|$
then $q$ is a candidate edge.

The Canny Edge Detector

original image (Lena)

The Canny Edge Detector

magnitude of the gradient

The Canny Edge Detector

After non-maximum suppression

An Idea: Single Threshold

1. Smooth Image
2. Compute gradients & Magnitude
3. Non-maximal suppression
4. Compare to a threshold: $T$
An OK Idea: Single Threshold

1. Smooth Image
2. Compute gradients & Magnitude
3. Non-maximal supression
4. Compare to a threshold: T

A Better Idea: Linking + Two Tresholds

Linking: Assume the marked point $q$ is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (either $r$ or $s$).

A Better Idea: Hysteresis Thresholding

- Define two thresholds $\tau_{\text{low}}$ and $\tau_{\text{high}}$
- Starting with output of nonmaximal supression, find a point $q_0$ where $\nabla I(q_0)$ is a local maximum.
- Start tracking an edge chain at pixel location $q_0$ in one of the two directions
- Stop when gradient magnitude < $\tau_{\text{low}}$ - i.e., use a high threshold to start edge curves and a low threshold to continue them.

Hysteresis thresholding

Single Threshold

$T=15$  $T=5$

Hysteresis $T_h=15$ $T_l=5$

Fine scale high threshold
Why is Canny so Dominant

- Still widely used after 20 years.
  1. Theory is nice
  2. Details good (magnitude of gradient, non-max suppression).
  3. Hysteresis thresholding an important heuristic.
  4. Code was distributed.

Corner Detection

Feature extraction: Corners and blobs

Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?
Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?

Step 1: extract features
Step 2: match features
Step 3: align images

Corners contain more info than lines.

- A point on a line is hard to match.

Image 1  Image 2

Corners contain more info than lines.

- A corner is easier to match

Image 1  Image 2

The Basic Idea

- We should easily recognize the point by looking through a small window
- Shifting a window in any direction should give a large change in intensity

“flat” region: no change in all directions
“edge”: no change along the edge direction
“corner”: significant change in all directions

Finding Corners

Intuition:

- Right at corner, gradient is ill-defined.
- Near corner, gradient has two different values.
Finding Corners

For each image location \((x,y)\), we create a matrix \(C(x,y)\):

\[
C(x,y) = \begin{bmatrix}
\sum I^2_x & \sum I_x I_y \\
\sum I_x I_y & \sum I^2_y
\end{bmatrix}
\]

Gradient with respect to \(x\), times gradient with respect to \(y\)

Matrix is symmetric

**WHY THIS?**

Because \(C\) is a symmetric positive definite matrix, it can be factored as:

\[
C = R^{-1} \begin{bmatrix}
\lambda_1 & 0 \\
0 & \lambda_2
\end{bmatrix} R
\]

where \(R\) is a 2x2 rotation matrix and \(\lambda_1\) and \(\lambda_2\) are non-negative.

1. \(\lambda_1\) and \(\lambda_2\) are the Eigenvalues of \(C\).
2. The columns of \(R\) are the Eigenvectors of \(C\).
3. Eigenvalues can be found by solving the characteristic equation \(\det(C - \lambda I) = 0\) for \(\lambda\).

**What is region like if:**

1. \(\lambda_1 = 0\)?
2. \(\lambda_2 = 0\)?
3. \(\lambda_1 = 0\) and \(\lambda_2 = 0\)?
4. \(\lambda_1 > 0\) and \(\lambda_2 > 0\)?

**So, to detect corners**

- Filter image with a Gaussian.
- Compute the gradient everywhere.
- Move window over image and construct \(C\) over the window.
- Use linear algebra to find \(\lambda_1\) and \(\lambda_2\).
- If they are both big, we have a corner.
  1. Let \(e(x,y) = \min(\lambda_1(x,y), \lambda_2(x,y))\)
  2. \((x,y)\) is a corner if it’s local maximum of \(e(x,y)\) and \(e(x,y) > \tau\)

Parameters: Gaussian std. dev, window size, threshold

Corner Detection Sample Results

Threshold=25,000  
Threshold=10,000  
Threshold=5,000
What to do with edges?

- Segment linked edge chains into curve features (e.g., line segments).
- Group unlinked or unrelated edges into lines (or curves in general).
- Accurately fitting parametric curves (e.g., lines) to grouped edge points.