Edge Detection, Lines

Introduction to Computer Vision
CSE 152
Lecture 10

Announcements

- Assignment 2 due Tuesday, May 4.

Last Lecture

Edges

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

Numerical Derivatives

Take Taylor series expansion of $f(x)$ about $x_0$

$$f(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{1}{2} f''(x_0)(x-x_0)^2 + \ldots$$

Consider samples taken at increments of $h$ and first two terms, we have

$$f(x_0+h) = f(x_0) + f'(x_0)h + \frac{1}{2} f''(x_0)h^2$$

$$f(x_0-h) = f(x_0) - f'(x_0)h + \frac{1}{2} f''(x_0)h^2$$

Subtracting and adding $f(x_0+h)$ and $f(x_0-h)$ respectively yields

$$f'(x_0) = \frac{f(x_0+h) - f(x_0-h)}{2h}$$

$$f''(x_0) = \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{2h^2}$$

Gradients:

$$\nabla I = \begin{bmatrix} \frac{dI}{dx} \\ \frac{dI}{dy} \end{bmatrix}$$

Is this $dI/dx$ or $dI/dy$?
Corner and Edge Detection
Smoothing and Differentiation
• Need two derivatives, in \( x \) and \( y \) direction.
• Filter with Gaussian and then compute gradient, or
• Use a derivative of Gaussian filter
  • because differentiation is convolution, and
  convolution is associative

Corner Detection Algorithm
• 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 \( e(x,y) = \min(b_1(x,y), b_2(x,y)) \)
    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

Back to Edges

Edge is Where Change Occurs: 1-D

• Change is measured by derivative in 1D

Ideal Edge

Smoothed Edge

First Derivative

Second Derivative

• Biggest change, derivative has maximum magnitude
• Or 2nd derivative is zero.

2D Edge Detection: Canny

1. Filter out noise
   – Use a 2D Gaussian Filter. \( J = I \ast G \)
2. Take a derivative
   – Compute the magnitude of the gradient:

\[
\nabla J = (J_x, J_y) = \left( \frac{\partial J}{\partial x}, \frac{\partial J}{\partial y} \right)
\]

is the Gradient

\[
\| \nabla J \|^2 = J_x^2 + J_y^2
\]

There is ALWAYS a tradeoff between smoothing and good edge localization!

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?
We wish to mark points along the curve where the magnitude is biggest. We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve. There are then two algorithmic issues: which point is the maximum, and where is the next point on the curve?

**Non-maximum suppression**

For every pixel in the image (e.g., q) we have an estimate of edge direction and edge normal (shown at q).

Using normal at q, find two points p and r on adjacent rows (or columns).

We have a maximum if the value at q is larger than those at both p and at r.

Interpolate to get values.

Assume the marked point 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 (here either r or s).

**HysteresisThresholding**

- Start tracking an edge chain at pixel location that is local maximum of gradient magnitude where gradient magnitude > $\tau_{high}$.
- Follow edge in direction orthogonal to gradient.
- Stop when gradient magnitude < $\tau_{low}$.
  - i.e., use a high threshold to start edge curves and a low threshold to continue them.
### Why is Canny so Dominant

- Still widely used after 25 years.
  1. Theory is nice.
  2. Details good (magnitude of gradient, non-max suppression).
  3. Hysteresis thresholding an important heuristic.
  4. Code was distributed.
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.

Finding lines in an image

Option 1:
- Search for the line at every possible position/orientation
- What is the cost of this operation?

Option 2:
- Use a voting scheme: Hough transform

Finding lines in an image

Option 1:
- Search for the line at every possible position/orientation
  - What is the cost of this operation?

Option 2:
- Use a voting scheme: Hough transform

Connection between image (x,y) and Hough (m,b) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points (x,y), find all (m,b) such that y = mx + b

Finding lines in an image

Connection between image (x,y) and Hough (m,b) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points (x,y), find all (m,b) such that y = mx + b

Hough transform algorithm

Because vertical lines can’t be represented,

Typically use a different parameterization

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

- d is the perpendicular distance from the line to the origin
- \( \theta \) is the angle this perpendicular makes with the x axis
- Why?
Hough transform algorithm

Basic Hough transform algorithm
1. Initialize \( H(d, \theta) = 0 \) \( H \) is called accumulator array
2. for each edge point \((x,y)\) in the image
   for \(\theta = 0\) to 180
   \( d = x \cos \theta + y \sin \theta \)
   \( H(d, \theta) += 1 \)
3. Find the value(s) of \((d, \theta)\) where \(H(d, \theta)\) is maximum
4. The detected line in the image is given by \( l = x \cos \theta + y \sin \theta \)

What’s the running time (measured in # votes)?

Extensions

Extension 1: Use the image gradient
1. same
2. for each edge point \((x,y)\) in the image
   compute unique \((d, \theta)\) based on image gradient at \((x,y)\)
   \( H(d, \theta) += 1 \)
3. same
4. same

What’s the running time measured in votes?

Hough Transform: 20 colinear points

- \( R, \theta \) representation of line
- Maximum accumulator value is 20

Hough Transform: “Noisy line”

- \( R, \theta \) representation of line
- Maximum accumulator value is 6

Hough Transform: Random points

- \( R, \theta \) representation of line
- Maximum accumulator value is 4

Mechanics of the Hough transform

- Difficulties
  - how big should the cells be? (too big, and we cannot distinguish between quite different lines; too small, and noise causes lines to be missed)
- How many lines?
  - count the peaks in the Hough array
- Which edgels belongs to which line?
  - tag the votes
- Complications, problems with noise and cell size
Number of votes that the real line of 20 points gets with increasing noise

As the noise increases in a picture without a line, the number of points in the max cell goes up