The Canny Edge Detector and the Hough Transform

CS 510
Lecture #17
March 2, 2018
Recall - Sobel Edges

Source  Dx Image  Dy Image
Sobel Edges: A Local Decision

• Magnitude = \((dx^2 + dy^2)^{1/2}\)

• Orientation = \(\tan^{-1}\ dy/dx\)

• \(dy/dx\) responses are signed

• Edge Masks: sum of weights is zero

• Edges tend to be “thick”
Symbolic Edge Detection

• Although Sobel edges are optimal estimators for the slope of a planar facet, as symbols they:
  – Are continuous; edge yes/no based on threshold
  – May be “thick”; need to be localized
  – Are isolated; need to be grouped into longer lines

• If they correspond to scene structure (e.g. discontinuities), we want a model of how scene structures map to images.
Seminal Work – Canny Edges
Canny Edge Detection (Step 1)

• In order to maximize the likelihood of finding step-edges,
  1. Smooth image with a Gaussian filter
     • Size is determined by noise model
  2. Compute image gradients over the same size mask

• The bigger the mask, the better detection is but the worse localization is...
Canny Edge Detection (step 2)

• Non-maximal suppression
  – So far, edges are still “thick”
  – For every edge pixel:
    • 1) Calculate direction of edge (gradient)
    • 2) Check neighbors in edge direction
    • If either neighbor is “stronger”, set edge to zero.

Drexel Tutorial –
http://dasl.mem.drexel.edu/alumni/bGreen/www.pages.drexel.edu/_weg22/can_tut.html
Canny Edge Detection (Step 3): Hysteresis Thresholding

• Continuous values still need thresholding
• Algorithm takes two thresholds: high & low
  – Any pixel with edge strength above the high threshold is an edge
  – Any pixel above the low threshold and next to an edge is an edge
• Iteratively label edges
  – they “grow out” from high points.
  – This is called hysteresis.
Canny Example (cont.)

\[ \text{Sigma} = 3.0 \]
\[ \text{low} = 0.4, \text{high} = 0.9 \]

\[ \text{Sigma} = 1.0 \]
\[ \text{low} = 0.4, \text{high} = 0.9 \]
Canny Example (III)

Sigma = 2.0
low = 0.4, high = 0.6

Sigma = 2.0
low = 0.4, high = 0.99
Canny Example (IV)

Sigma = 2.0
low = 0.2, high = 0.9

Sigma = 2.0
low = 0.6, high = 0.9
Motivation – Apprx. Invariance

Some Photoshop liberties have been taken to illustrate the larger point here 😊
Canny in OpenCV

Canny Edge Detector

Goal

In this tutorial you will learn how to:

- Use the OpenCV function `Canny` to implement the Canny Edge Detector.

Theory

1. The *Canny Edge detector* was developed by John F. Canny in 1986. Also known to many as the *optimal detector*, Canny algorithm aims to satisfy three main criteria:
   - **Low error rate**: Meaning a good detection of only existent edges.
   - **Good localization**: The distance between edge pixels detected and real edge pixels have to be minimized.
   - **Minimal response**: Only one detector response per edge.
Hough Transform: Grouping

• The idea of the Hough transform is that a change in representation converts a point grouping problem into a peak detection problem.

• Standard line representations:
  – $y = mx + b$ -- *compact, but problems with vertical lines*
  – $(x_0, y_0) + t(x_1, y_1)$ -- your raytracer used this form, but it is highly redundant (4 free parameters)
  – $ax + by + c = 0$ -- Bresenham’s uses this form. Still redundant (3 free parameters)

• How else might you represent a line?
Hough Grouping (cont.)

• Represent infinite lines as \((\phi, \rho)\):
Hough Grouping (III)

• Why? This representation is:
  – Small: only two free parameters (like $y=mx+b$)
  – Finite in all parameters: $0 \leq \rho < \sqrt{\text{row}^2+\text{col}^2}$, $0 \leq \phi < 2\pi$
  – Unique: only one representation per line

• General Idea:
  – Hough space $(\phi, \rho)$ represents all possible lines
  – Next step - use discrete Hough space
  – Let every point “vote for” any line it might belong to.
Hough Grouping: Directed Edges

- Every edge has a location and position, so it can be part of only one (infinitely extended) line.

- Co-linear edges map to one bucket in Hough space.
Hough Grouping: Edges

• Reduces line grouping to peak detection
  – Each edge votes for a bucket (line)
  – # of votes equates to support
    • The # of participating edges.
  – Position of bucket provides the $\phi$, $\rho$ parameters

• Problem: if “true” line parameters are on the boundary of a bucket, supporting data may be split

• Solution: smooth the histogram (Hough image) before selecting peaks.
Basic Hough – Infinite Lines

• The Hough Transform in pure form …
• Does not return end-points
• Instead, it returns a rho and theta pairs.

```c
for (size_t i = 0; i < lines.size(); i++) {
    float rho = lines[i][0], theta = lines[i][1];
    Point pt1, pt2;
    double a = cos(theta), b = sin(theta);
    double x0 = a * rho, y0 = b * rho;
    pt1.x = cvRound(x0 + 1000 * (-b));
    pt1.y = cvRound(y0 + 1000 * (a));
    pt2.x = cvRound(x0 - 1000 * (-b));
    pt2.y = cvRound(y0 - 1000 * (a));
```
Min Edge Threshold: 71, Min Votes: 194
Min Edge Threshold: 72, Min Votes: 194
Min Edge Threshold: 71, Min Votes: 194
Hough Fitting

• After finding the peaks in the Hough Transform - still two potential problems:
  – Resolution limited by bucket size.
  – Infinite lines, not line segments

• Both of these problems can be fixed,
  – If you kept a linked list of edges (not just #)
  – Of course, this is more expensive...
Hough Fitting (II)

• Sort your edges
  – rotate edge points according to \( \rho \)
  – sort them by (rotated) x coordinate

• Look for gaps
  – have the user provide a “max gap” threshold
  – if two edges (in the sorted list) are more than max gap apart, break the line into segments
  – if there are enough edges in a given segment, fit a straight line to the points
Open CV Hough Version 2

• Second Hough algorithm in OpenCV
• Returns segments – based on work below
Min Edge Threshold: 70, Min Votes: 100
Min Edge Threshold: 70, Min Votes: 100
Min Edge Threshold: 70, Min Votes: 100
Building Example

http://docs.opencv.org/modules/imgproc/doc/feature_detection.html