

CSci 4968 and 6270  
Computational Vision  
Lecture 6  
Lines and Curves

Charles Stewart

Department of Computer Science  
Rensselaer Polytechnic Institute

2009/09/21

## Outline of Lecture 6

Since we are one lecture behind, there will be no labeled "Lecture 5."

- Review from Lecture 4
- Approximating edgel chains with lines and curves
- Line representation in 2d
- Least-squares estimation of line parameters
- Hough transformations and robust estimation
- Arcs of circles and ellipses

## Review from Lecture 4

- Non-linear smoothing helps preserve edges
- Smoothing, differentiation and gradient computation are applied at the start of edge detection
- Non-maximum suppression, sub-pixel localization, linking and adaptive thresholding produce the final edgel chains.
- These edge detection results do not always coincide with the boundaries that humans perceive.

## Approximating Edgel Chains with Lines

### Simple Technique to Split Chain

- 1 Form line between chain end points
- 2 Find point on contour that is furthest from the line
- 3 If this distance is below a threshold, stop
- 4 Otherwise, split the contour into two chains at this farthest point, and continue recursively.

Produces what is called a "poly-line".

## Line Representations

Many different representations, starting with the simple version most of us learned first.

- **Slope-intercept:** Parameters are  $m$  and  $b$ ,

$$y = mx + b.$$

This has two problems for our use, as we will discuss in class.

- **Point / normal:** Parameters are normal  $\boldsymbol{\eta}$  and point on line  $\mathbf{p}_0$ . Point  $\mathbf{p} = (x, y)^T$  is on the line iff

$$\boldsymbol{\eta} \cdot (\mathbf{p} - \mathbf{p}_0) = 0.$$

This appears to have four degrees of freedom, but really there are only two.

## Line Representations — Continued

Two more representations:

- **Implicit form:** Given parameter vector  $\mathbf{a} = (a, b, c)^T$ ,  $\mathbf{p} = (x, y)^T$  is on the line if

$$ax + by + c = 0.$$

By imposing the constraint  $a^2 + b^2 = 1$ , we end up with only two degrees of freedom.

- **Parametric form:** Given  $x_0, x_1, y_0, y_1$ , we can generate points on the line as

$$x(t) = x_1 t + x_0$$

$$y(t) = y_1 t + y_0$$

## Line Representation Issues

Issues when choosing a representation and an estimation technique

- Need to represent all possible lines and introduce no bias in the representation
- Some representations are better for generating points and others for determining when a point is on a line and determining the point-to-line distance.
- Some parameterization have more parameters than degrees of freedom. We need to handle this during any estimation process.

## Calculations Needed on Lines

To address these we will have to digress into Lagrange multipliers...

- Is a point on a line?
- What is the distance of a point to a line?
- Given a set of points, what is the best-fitting line?

## Lagrange Multipliers

Often needed for solving constrained optimization problems where there are more parameters than degrees of freedom.

- Main problem: given a function  $f(\mathbf{a})$  that we want to minimize (or maximize) with respect to  $\mathbf{a}$ .
- Our difficulty is that either not all values of  $\mathbf{a}$  are allowed, or some subsets of values of  $\mathbf{a}$  are equivalent.
- To capture these restrictions and equivalences we impose a constraint on  $\mathbf{a}$ ,  $g(\mathbf{a}) = 0$ .
- Then, form a new optimization problem:

$$F(\mathbf{a}, \lambda) = f(\mathbf{a}) + \lambda g(\mathbf{a}).$$

- Finally, solve the simultaneous equations

$$\frac{\partial F}{\partial \mathbf{a}} = 0, \quad \frac{\partial F}{\partial \lambda} = 0.$$

## Examples in Line Fitting

We will develop solution to both of these in class for the *implicit* representation:

- Given a point  $\mathbf{p}_0 = (x_0, y_0)^T$  and a line  $ax + by + c = 0$ , (a) find the closest point on the line to  $\mathbf{p}_0$  and (b) as a result find the distance of  $\mathbf{p}_0$  to the line.
- Given a set of  $N$  points,  $\{\mathbf{p}_i\} = \{(x_i, y_i)^T\}$ , find the best fitting line, in the sense of minimizing the summed square distances of the points to the line.

## Back to Using a Set of Lines to Approximate a Contour

Recall that we are working with an edgel chain

- We are now sure how to compute the point to line distances in finding the furthest point from the line:
  - Perhaps you already knew all of this, but now you are sure and you know why.
- We can apply least-squares estimation to obtain better line approximations.

## We Need To Do Better Than Approximating a Single Edgel Chain

- Looking at some simple images, we will see that just linking edgels into chains and then approximating the chains with lines does not produce the desired results.
  - Contours are broken and sometimes take surprising turns!
- Approaches to solving this:
  - Perceptual grouping, a big topic
  - Hough transformations
  - Robust estimation
- We will focus here on the Hough transform.

## Introduction to Hough Transforms

### Our first example of the concept of "voting" in computer vision

- Each point location  $(x, y)$  (each edge location) will vote for the set of lines that it could reasonably belong to.
- Votes are accumulated from all edge points:
  - This could be across an entire image or it could be in a region of the image.
- Parameters of lines with a large number of votes are likely to correspond to true lines in the image.
- To make this work, we need to re-examine our line representations.

## Line Parameterization for the Hough Transform

Consider a point  $(x, y)^T$ , and think about what lines it might belong to?

- For the implicit form this is all lines with parameters  $(a, b, c)$  such that

$$ax + by + c = 0.$$

- Remember, though, this is over parameterized.
- Impose the constraint  $a^2 + b^2 = 1$ , and parameterize the normal vector  $(a, b)$  in terms of the angle  $\theta$ , so that the line becomes

$$x \cos \theta + y \sin \theta + c = 0.$$

- Using this representation,  $c$  is the (signed) distance of the origin of our coordinate system to the line.
- Now, for each possible value of  $\theta$ , there is one possible value of  $c$ :

$$c = -(x \cos \theta + y \sin \theta).$$

## Hough Transformation Voting

### Form a 2d accumulator array

- The column (horizontal) axis will correspond to  $\theta$  in the interval  $[0, \pi)$ :
  - We stop at  $\pi$  instead of  $2\pi$  because there is a sign ambiguity in the value of  $\theta$  in our line equation.
  - If we allow increments of  $\Delta\theta$  (e.g.  $2\pi/180$  radians), then we have

$$k_\theta = \lceil \pi / \Delta\theta \rceil$$

columns.

- The row (vertical) axis will correspond to  $c$ :
  - Move the coordinate system to the center of the image and normalize the coordinate system so that the domain is  $[-1, 1] \times [-1, 1]$ .
  - Then, the values of  $c$  must cover the interval  $[-\sqrt{2}, \sqrt{2}]$ .
  - Using increments of  $\Delta c$ , we end up with

$$k_c = \lceil 2\sqrt{2} / \Delta c \rceil$$

- Overall, the accumulator array has  $k_\theta k_c$  entries.

## Hough Transformation Voting Procedure

### Let $A$ be the accumulator array

- For each point  $(x_i, y_i)$ 
  - For each  $j$  in  $[0, \dots, k_\theta)$ ,
    - $\theta_j = j\Delta\theta$
    - $c = -(x_i \cos \theta_j + y_i \sin \theta_j)$ .
    - $m = c\Delta c$
    - $A(m, j) + 1$ .
- Analyze the accumulator array to find peaks. These determine the lines in the image.

## Hough Examples

We will study the behavior of the Hough transform using the following Matlab functions

- checkerboard
- edge
- hough
- houghpeaks
- houghlines

## Details and Variations of the Hough Transformation

- Handle and discretization effects by smoothing the Hough array prior to peak extraction or by having each point vote for multiple  $c$  values for each  $\theta_j$ .
- Compute a final line estimate using a least-squares fit to the points that contribute to a peak.
- Use edgel gradient direction to narrow the voting domain for each point.
- Hierarchical voting, starting with large "bins" (large values of  $\Delta\theta$  and  $\Delta c$ ) and then "focusing" the search around peaks.
- Implicit representation of the Hough array.

## Robust Line Estimation

This is an alternative to the Hough transform, although there are strong mathematical connections between the techniques.

- Looking at the least-squares objective function

$$\sum_i (ax_i + by_i + c)^2$$

it is pretty clear that a point far from the "true" line can skew the estimate fairly heavily!

- Two solutions are typically used,
  - Replace the square cost by a more slowly growing function.
  - Replace the summation by a different function usually based on the *order statistics* of the errors. The median is the simplest version of this.

## Robust Line Estimation

### Estimation techniques

- Iterative minimization for the non-square (non-quadratic) cost function
- Random sampling for both types of cost functions.

We will return to these and consider them in more detail when we discuss estimating inter-image transformations.

## Summary and Review

- Use lines to approximate contours
- Among several line representation techniques, we use the implicit form.
- Computing the distance of a point to a line and estimating the parameters of the line that best approximates a set of points can both be achieved using Lagrange multipliers.
- When we restrict our attention to single edgel chains we usually do not get long enough or complete enough lines.
- We examined Hough transformations and, briefly, robust estimation as ways to extract longer lines.
- These are more global estimation methods, but are they enough?

## Project Ideas

The Szeliski text includes a discussion of finding vanishing points and rectangles in images. We will soon be discussing vanishing points in the context of camera modeling and camera calibration. The project idea is to study vanishing point and rectangle estimate methods, choose one, implement it, and evaluate it carefully.