Lecture 12b
Parametric Transforms

sec. 8.5.2 & ch. 11 of *Machine Vision* by Wesley E. Snyder & Hairong Qi

Spring 2024

16-725 (CMU RI) : BioE 2630 (Pitt)

Dr. John Galeotti

Snyder ch. 11: Parametric Transforms

- Goal: Detect geometric features in an image
- Method: Exchange the role of variables and parameters
- References: Snyder 11 & ITK Software Guide book 2, 4.4

Geometric Features?

- For now, think of geometric features as shapes that can be graphed from an equation.
- Line: $\mathbf{y} = m\mathbf{x} + b$
- Circle: $R^2 = (\mathbf{x} - x_{center})^2 + (\mathbf{y} - y_{center})^2$

(variables are shown in **bold purple**, parameters are in black)

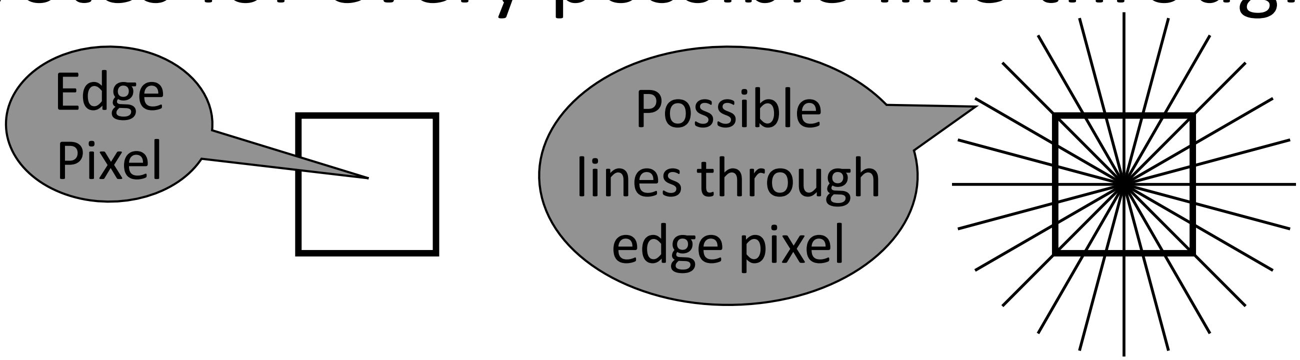
Why Detect Geometric Features?

- Guide segmentation methods
  - Automated initialization!
- Prepare data for registration methods
- Recognize anatomical structures

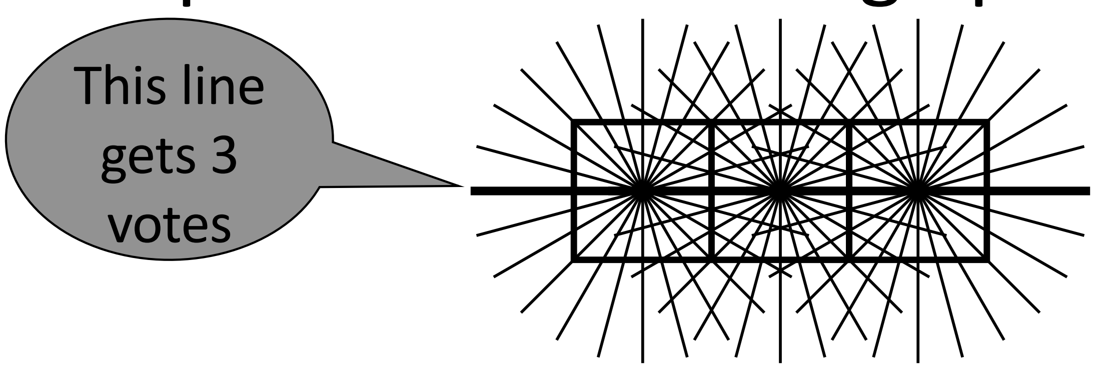
From the ITK Software Guide v 2.4, by Luis Ibáñez, et al., p. 596

How do we do this again?

- Actually, each edge pixel “votes”
- If we are looking for lines, each edge pixel votes for every possible line through itself:



- Example: 3 collinear edge pixels:



How to Find All Possible Shapes for each Edge Pixel

- Exchange the role of variables and parameters:
- Example for a line: $y = \mathbf{m}x + \mathbf{b}$

  (variables are shown in **bold purple**)
- Each edge pixel in the image:
  - Has its own (x, y) coordinates
  - Establishes its own equation of $(\mathbf{m},\mathbf{b})$

This is the set of all possible shapes through that edge point

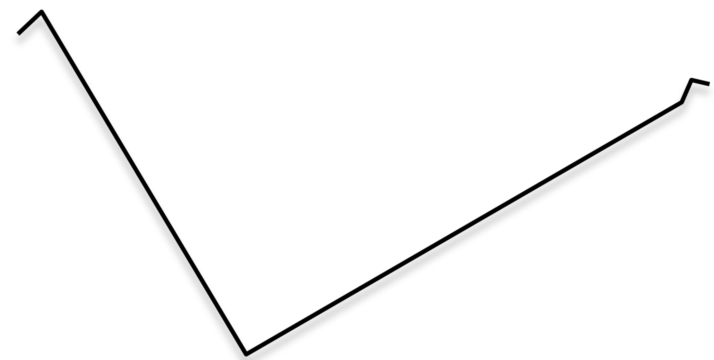
How to Implement Voting

- With an accumulator
  - Think of it as an image in parameter space
  - Its axes are the new variables (which were formally parameters)
  - But, writing to a pixel increments (rather than overwriting) that pixel’s value.
- Graph each edge pixel’s equation on the accumulator (in parameter space)
- Maxima in the accumulator are located at the parameters that fit the shape to the image.

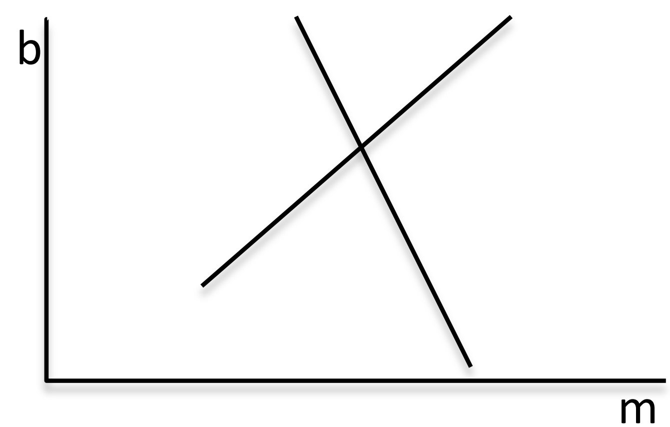
Example 1: Finding Lines

- If we use $\mathbf{y} = \mathrm{m}\mathbf{x} + \mathrm{b}$
- Then each edge pixel results in a line in parameter space: $\mathbf{b} = \mathbf{-m}\mathrm{x} + \mathrm{y}$

Edge Detection Results
(contains 2 dominant line segments)

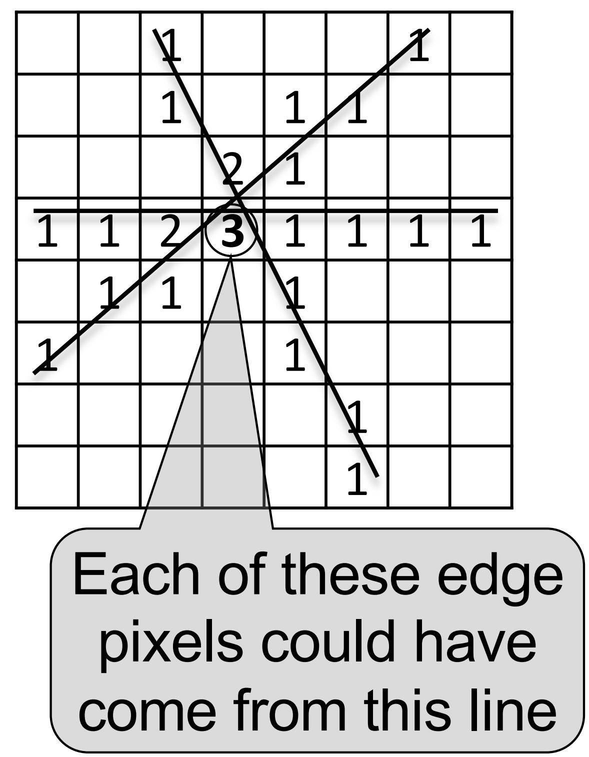


Accumulator Intermediate Result
(after processing 2 edge pixels)



Example 1: Finding Lines

- A closer look at the accumulator after processing 2 and then 3 edge pixels
- The votes from each edge pixel are graphed as a line in parameter space
- Each accumulator cell is incremented each time an edge pixel votes for it
    - I.e., each time a line in parameter space passes through it



Example 2: Finding Lines… A Better Way

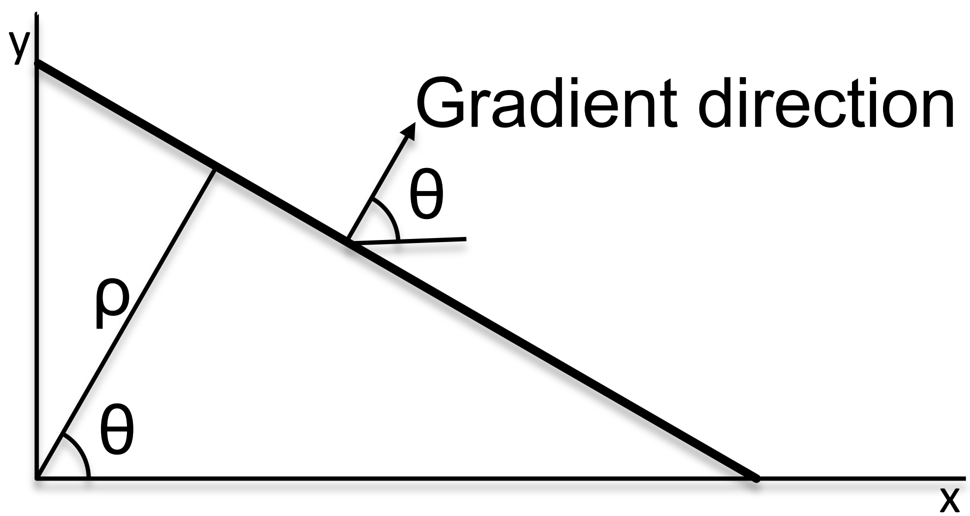
- What's wrong with the previous example?
  - Consider vertical lines: $m = \infty$
  - My computer doesn't like infinite-width accumulator images. Does yours?
- For parametric transforms, we need a different line equation, one with a bounded parameter space.

Example 2: Finding Lines… A Better Way

- A better line equation for parameter voting:

$$\rho = x \cos \theta + y \sin \theta$$

- $\rho \leq$ the input image diagonal size
  - But, to make math easy, ρ can be - too.
- θ is bounded within $[0,2\pi]$



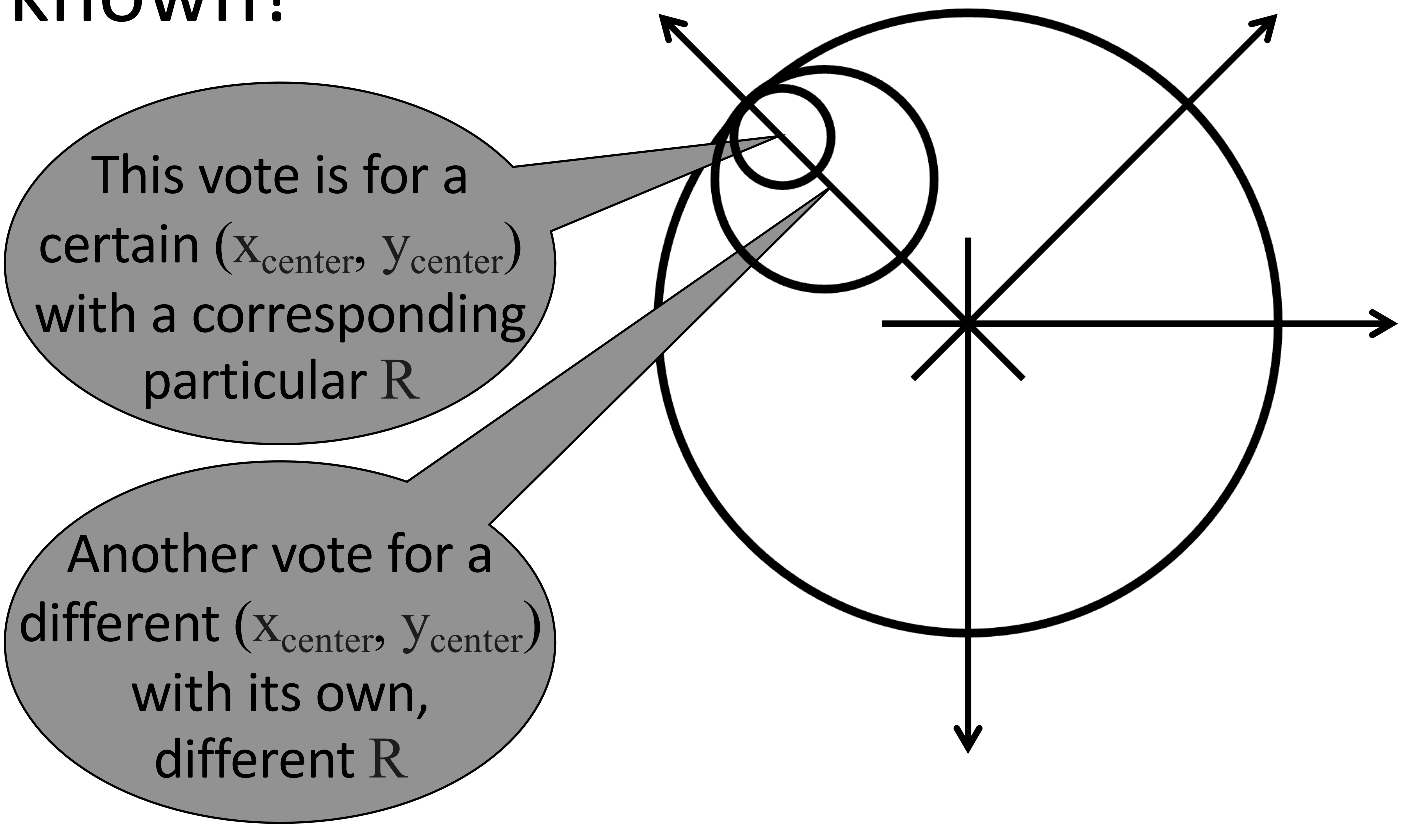
See *Machine Vision* Fig. 11.5 for example of final accumulator for 2 noisy lines

Computational Complexity

- This can be really slow
  - Each edge pixel yields a lot of computation
  - The parameter space can be huge
- Speed things up:
  - Only consider parameter combinations that make sense…
  - Each edge pixel has an apx. direction attached to its gradient, after all.

Example 3: Finding Circles

- Equation: $R^2 = (x\text{-}x_{center})^2 + (y\text{-}y_{center})^2$
- Must vote for 3 parameters if R is not known!



Example 4: General Shapes

- What if our shape is weird, but we can draw it?
  - Being able to draw it implies we know how big it will be
- See Snyder 11.4 for details
- Main idea:
  - For each boundary point, record its coordinates in a local reference frame (e.g., at the shape's center-of-gravity).
  - Itemize the list of boundary points (on our drawing) by the direction of their gradient