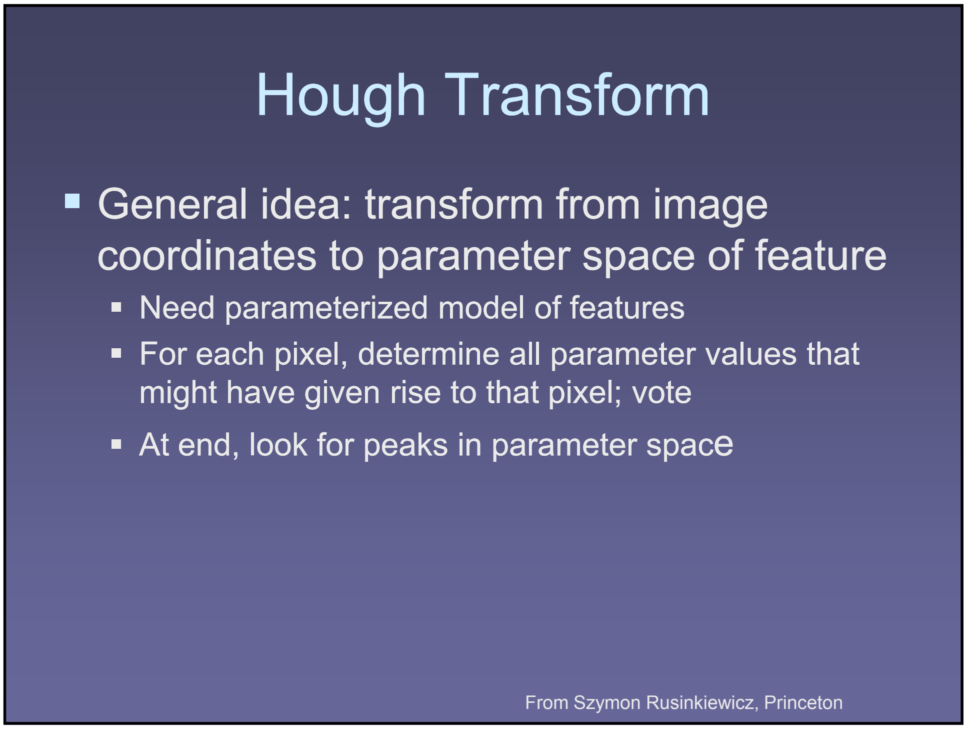
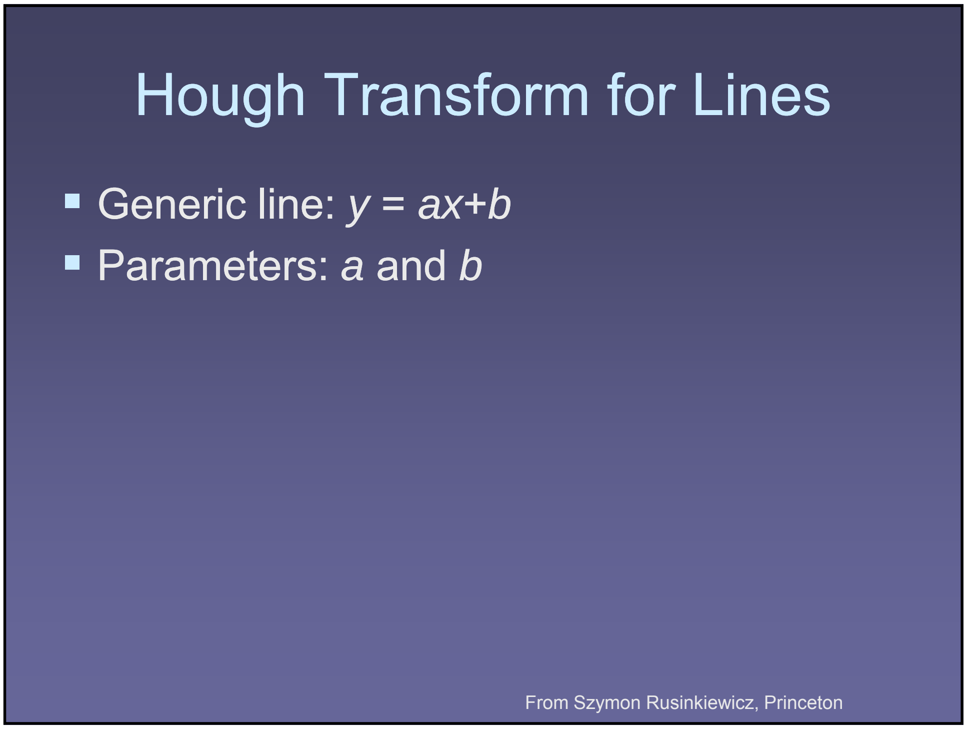


Hough Transform

- General idea: transform from image coordinates to parameter space of feature
  - Need parameterized model of features
  - For each pixel, determine all parameter values that might have given rise to that pixel; vote
  - At end, look for peaks in parameter space

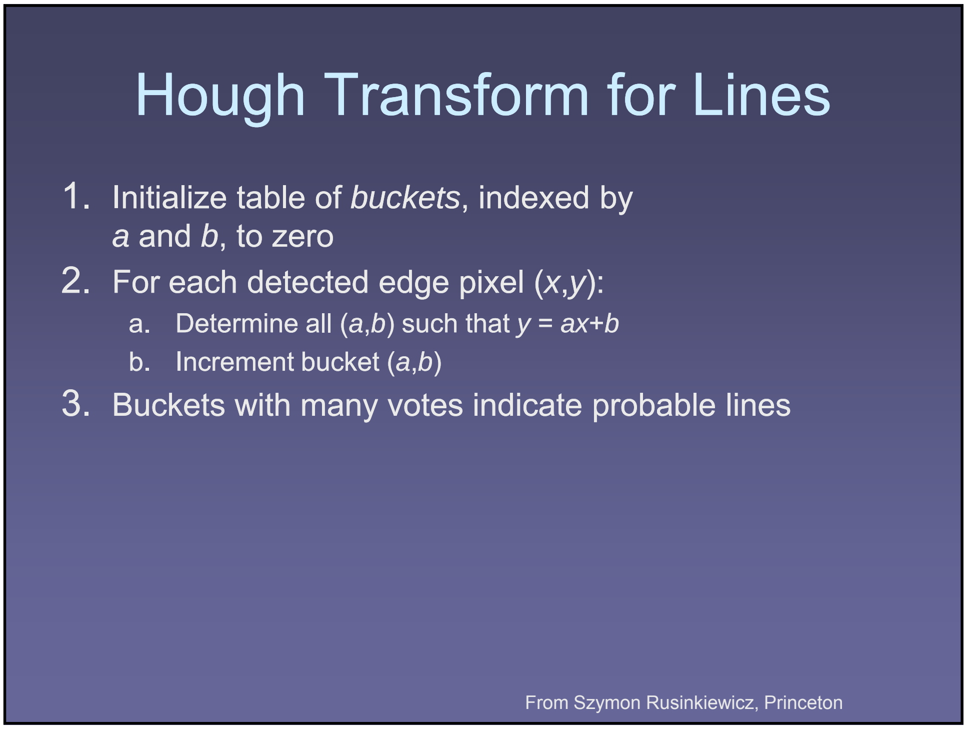
From Szymon Rusinkiewicz, Princeton



Hough Transform for Lines

- Generic line: $y = ax+b$
- Parameters: a and b

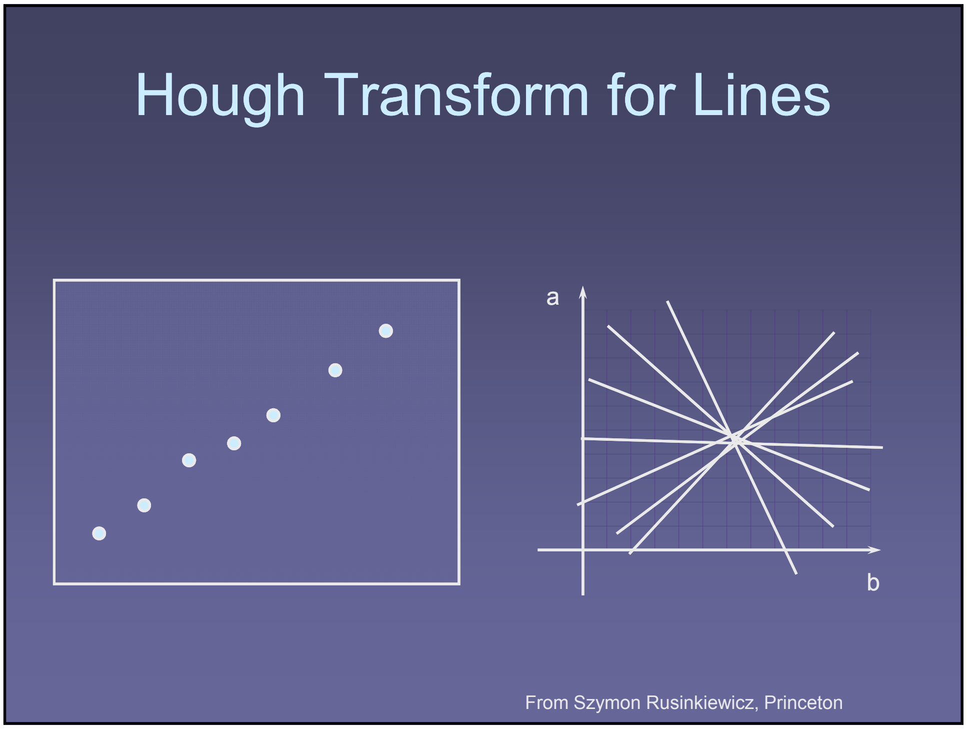
From Szymon Rusinkiewicz, Princeton



Hough Transform for Lines

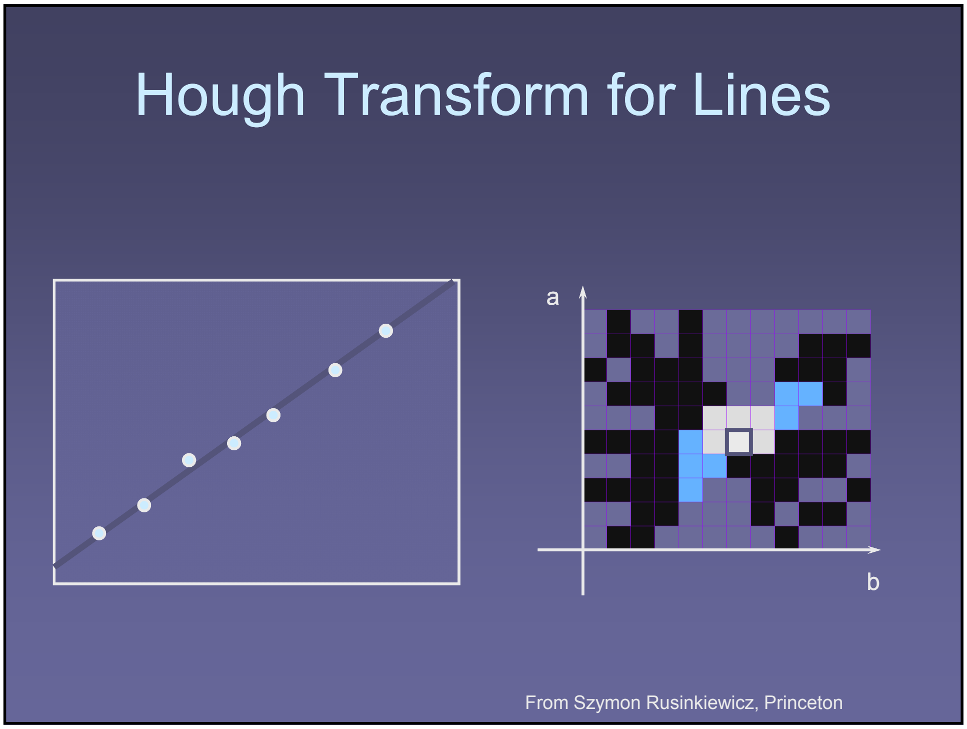
1. Initialize table of *buckets*, indexed by *a* and *b*, to zero
2. For each detected edge pixel (*x*,*y*):
   a. Determine all (*a*,*b*) such that *y* = *ax*+*b*
   b. Increment bucket (*a*,*b*)
3. Buckets with many votes indicate probable lines

From Szymon Rusinkiewicz, Princeton



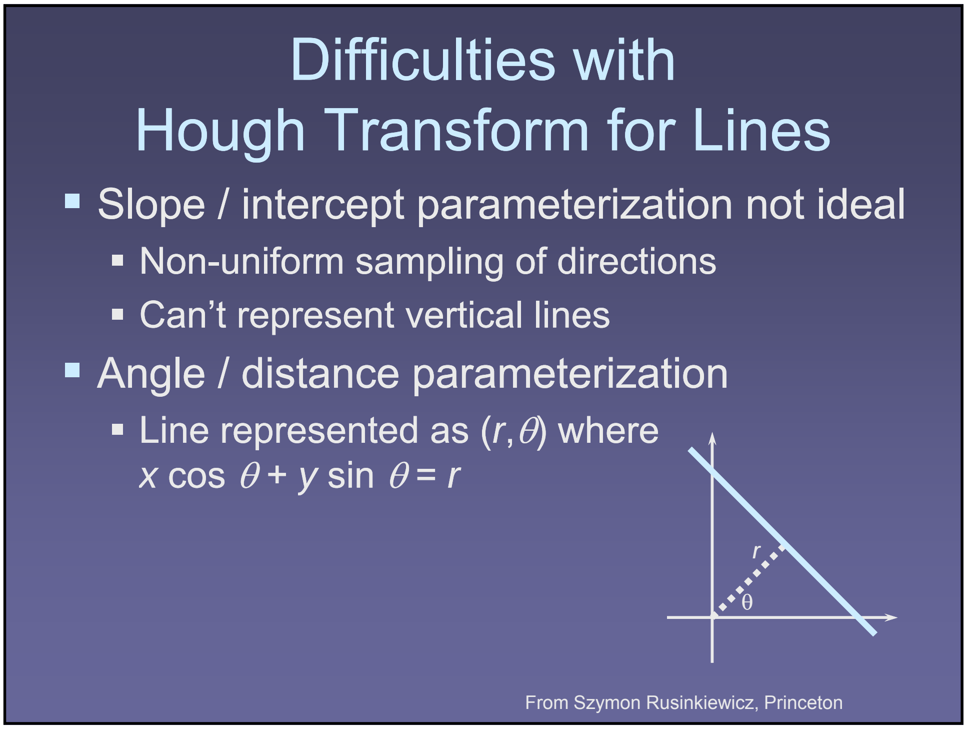
Hough Transform for Lines

From Szymon Rusinkiewicz, Princeton



Hough Transform for Lines

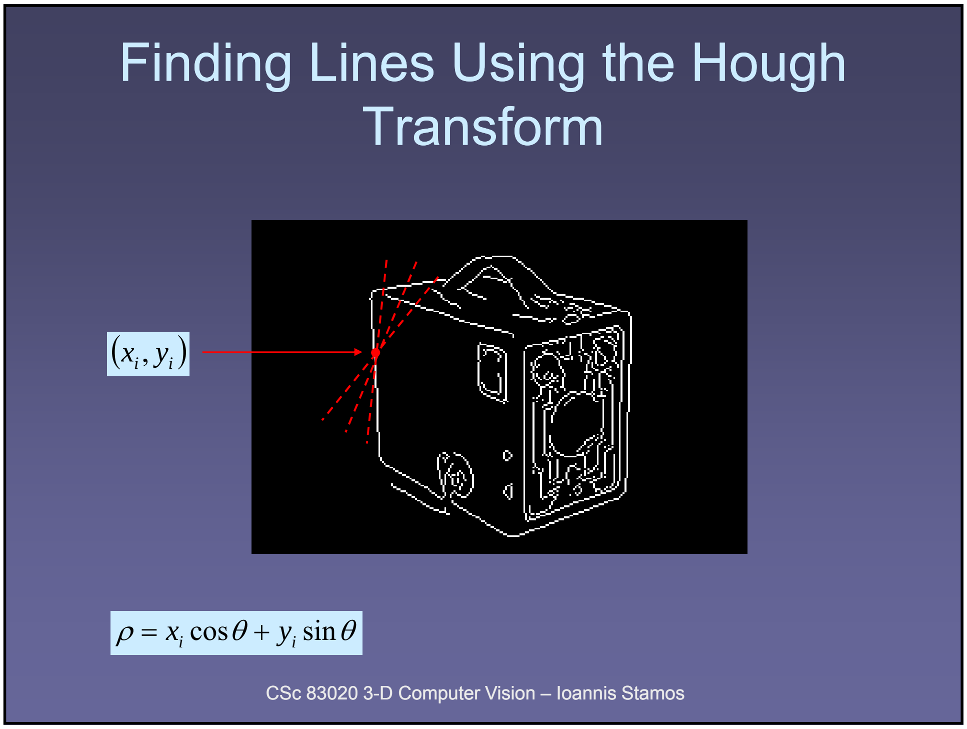
From Szymon Rusinkiewicz, Princeton



Difficulties with Hough Transform for Lines

- Slope / intercept parameterization not ideal
  - Non-uniform sampling of directions
  - Can’t represent vertical lines
- Angle / distance parameterization
  - Line represented as (r, θ) where $x \cos \theta + y \sin \theta = r$

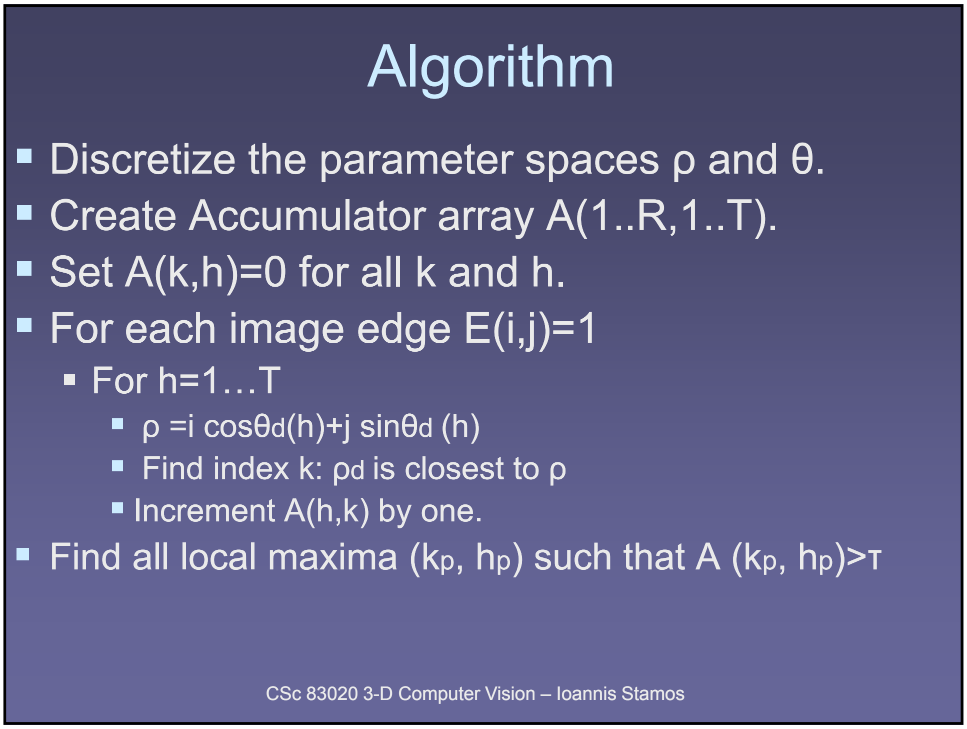
From Szymon Rusinkiewicz, Princeton



Finding Lines Using the Hough Transform

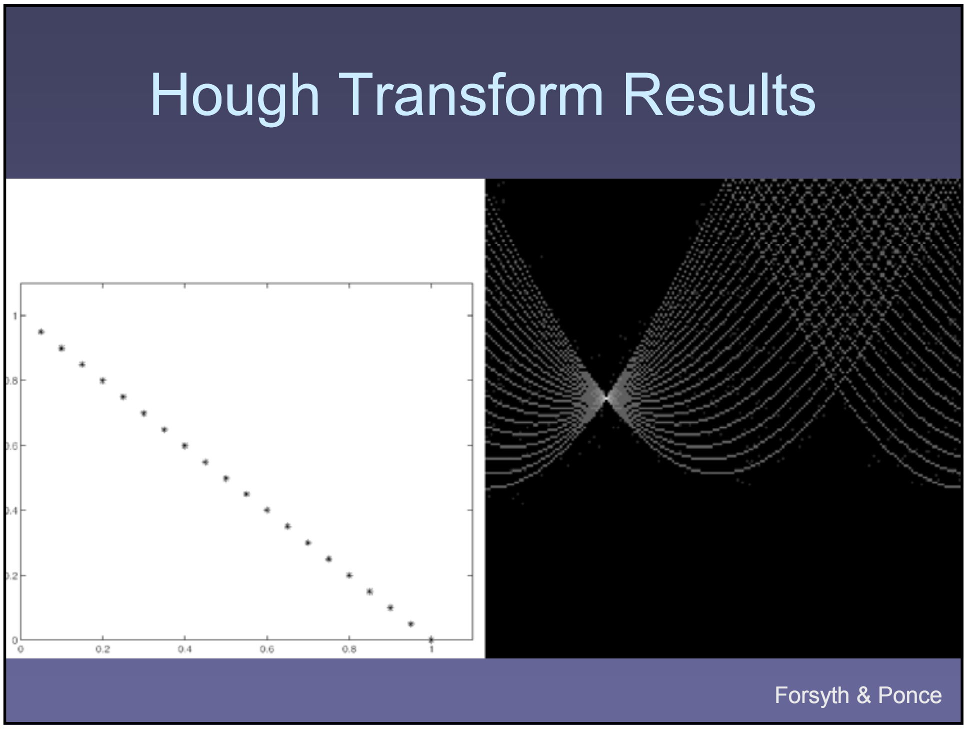
(x_i, y_i)

$\rho = x_i \cos\theta + y_i \sin\theta$



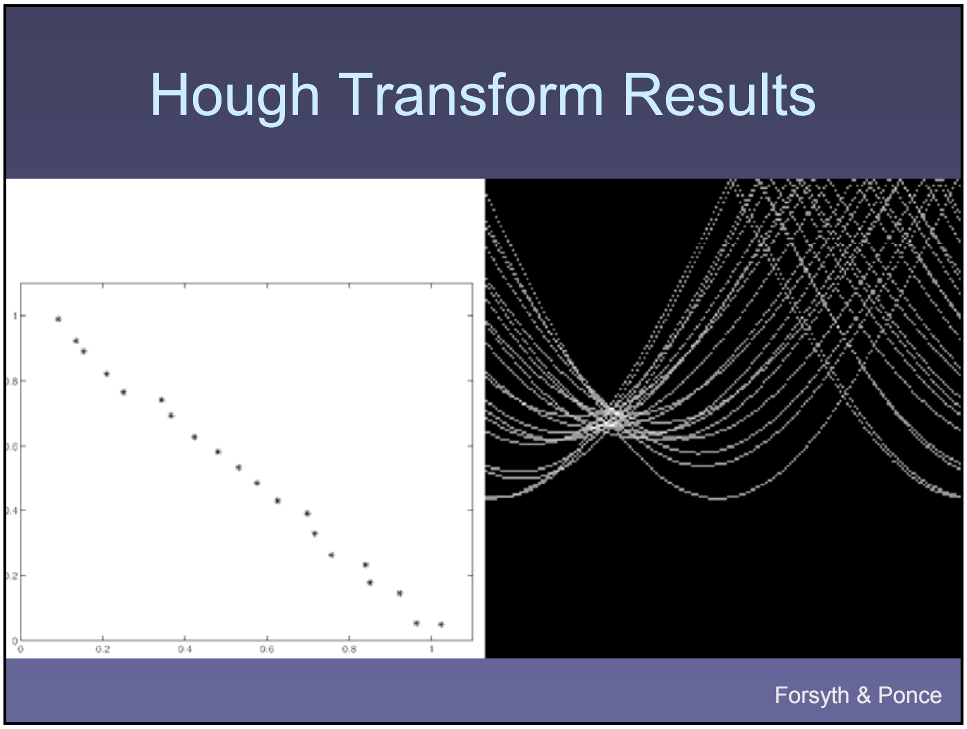
Algorithm

- Discretize the parameter spaces ρ and θ.
- Create Accumulator array A(1..R,1..T).
- Set A(k,h)=0 for all k and h.
- For each image edge E(i,j)=1
    - For h=1…T
        - $\rho = i \cos\theta_d(h) + j \sin\theta_d(h)$
        - Find index k: ρ_d is closest to ρ
        - Increment A(h,k) by one.
- Find all local maxima (k_p, h_p) such that A (k_p, h_p)>τ



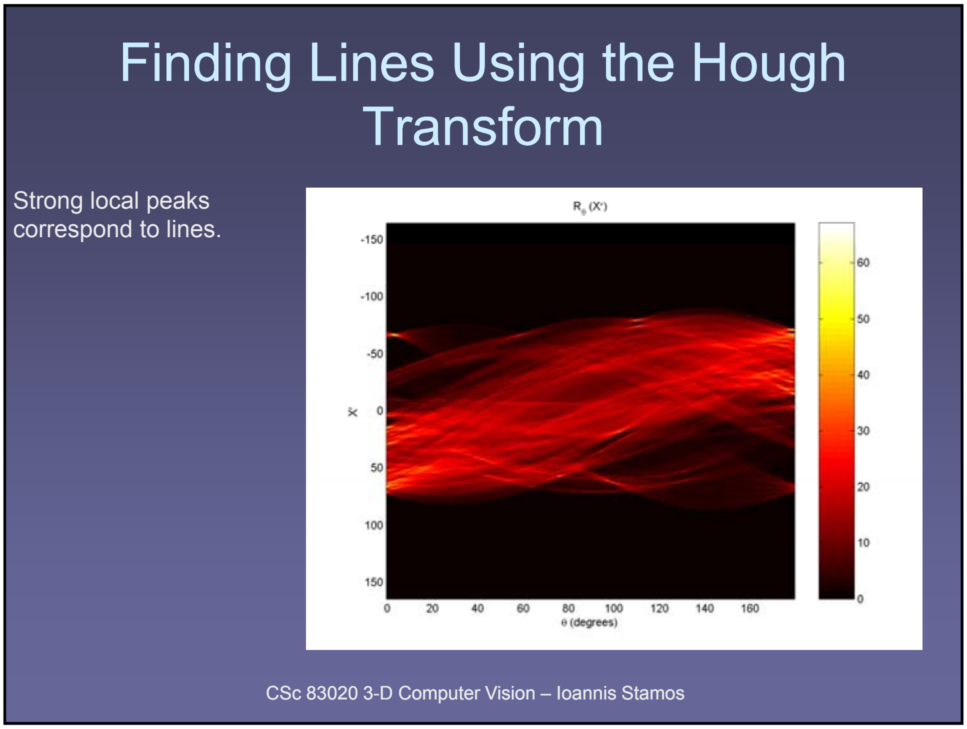
Hough Transform Results

Forsyth & Ponce



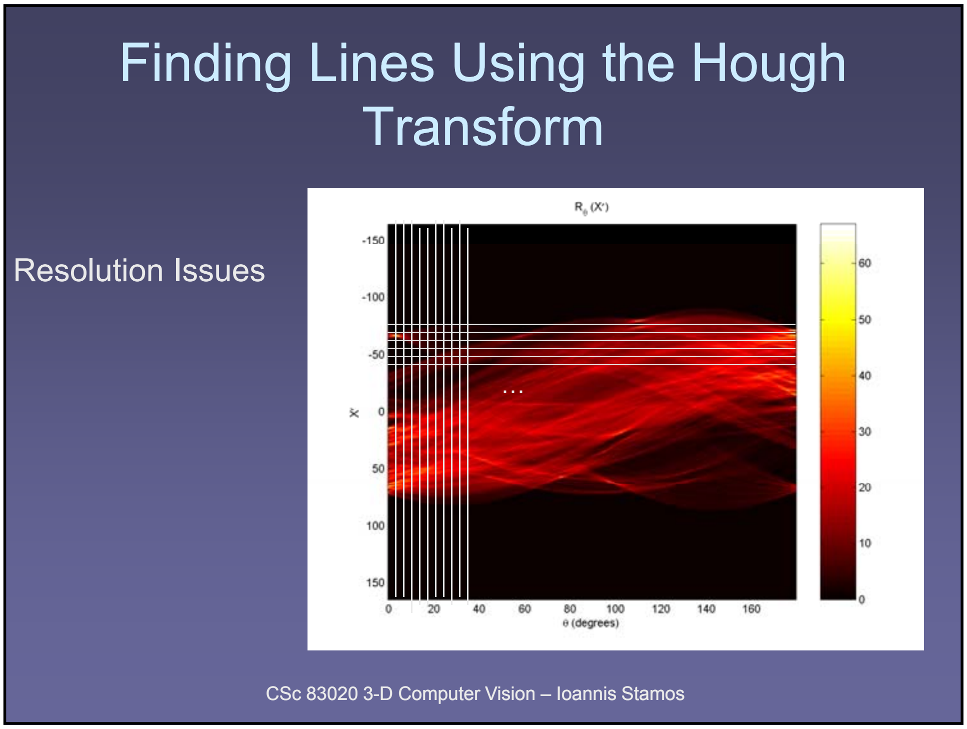
Hough Transform Results

Forsyth & Ponce



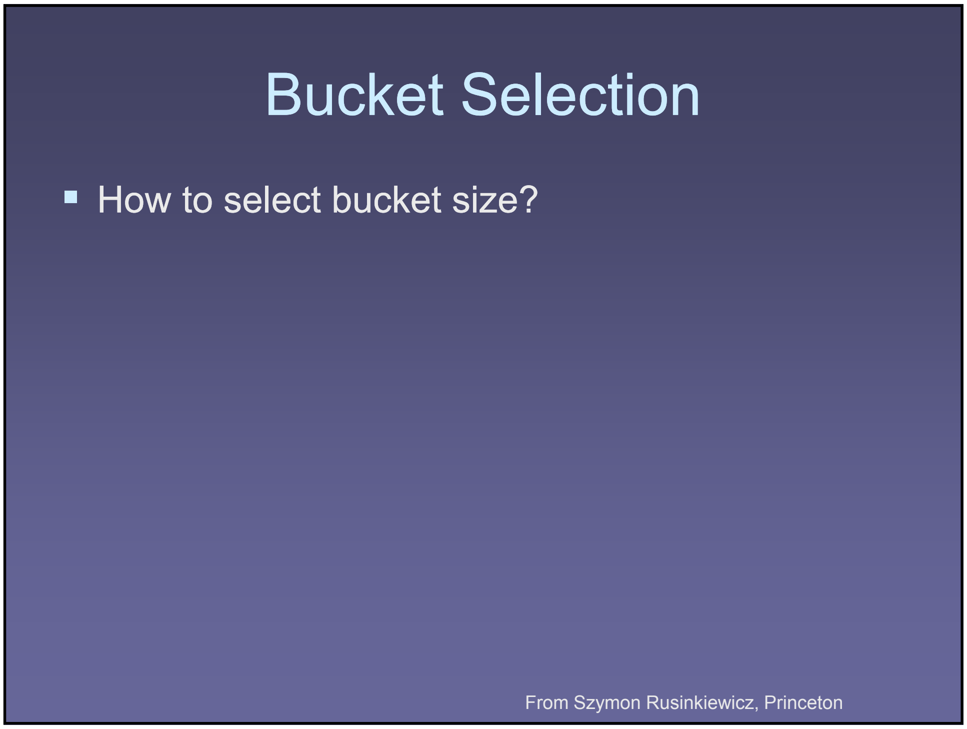
Finding Lines Using the Hough Transform

Strong local peaks correspond to lines.



Finding Lines Using the Hough Transform

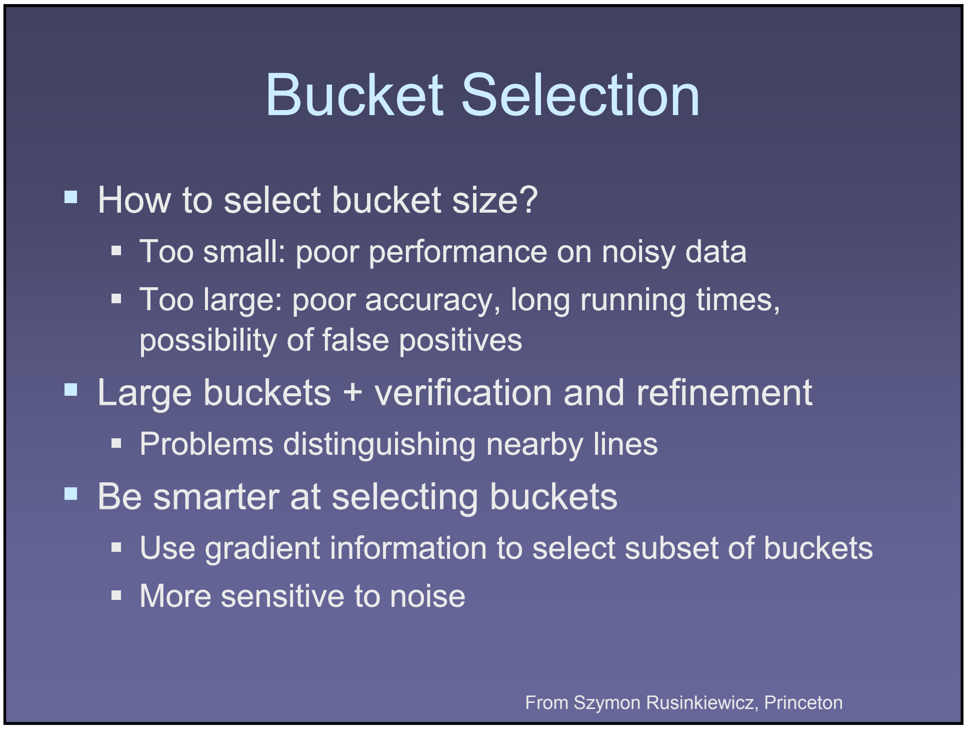
Resolution Issues



Bucket Selection

- How to select bucket size?

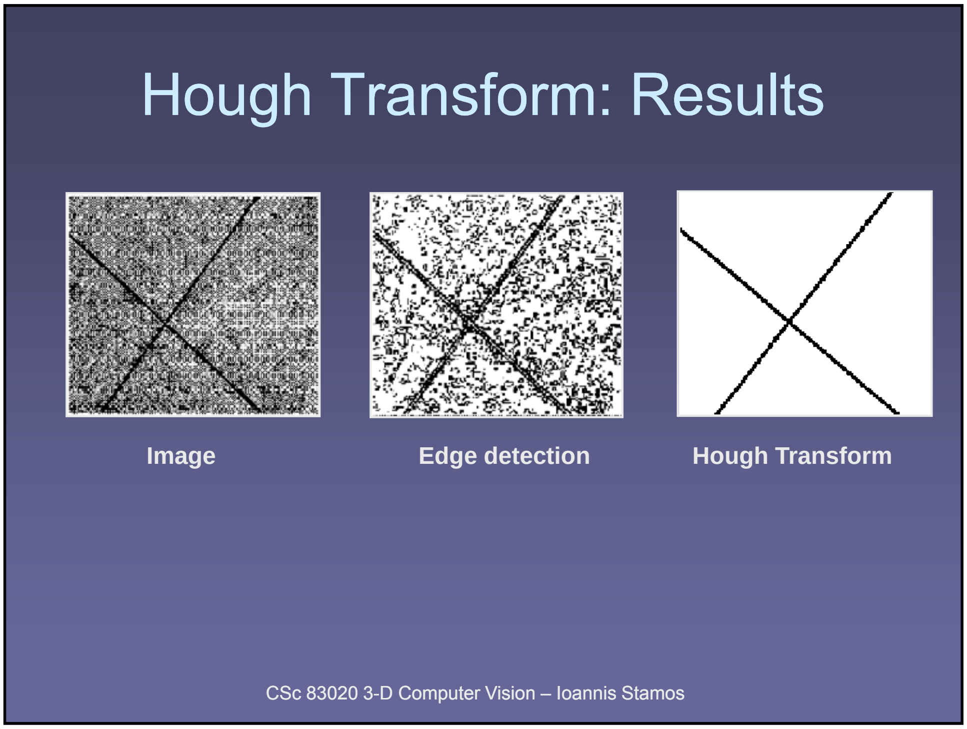
From Szymon Rusinkiewicz, Princeton



Bucket Selection

- How to select bucket size?
  - Too small: poor performance on noisy data
  - Too large: poor accuracy, long running times, possibility of false positives
- Large buckets + verification and refinement
  - Problems distinguishing nearby lines
- Be smarter at selecting buckets
  - Use gradient information to select subset of buckets
  - More sensitive to noise

From Szymon Rusinkiewicz, Princeton

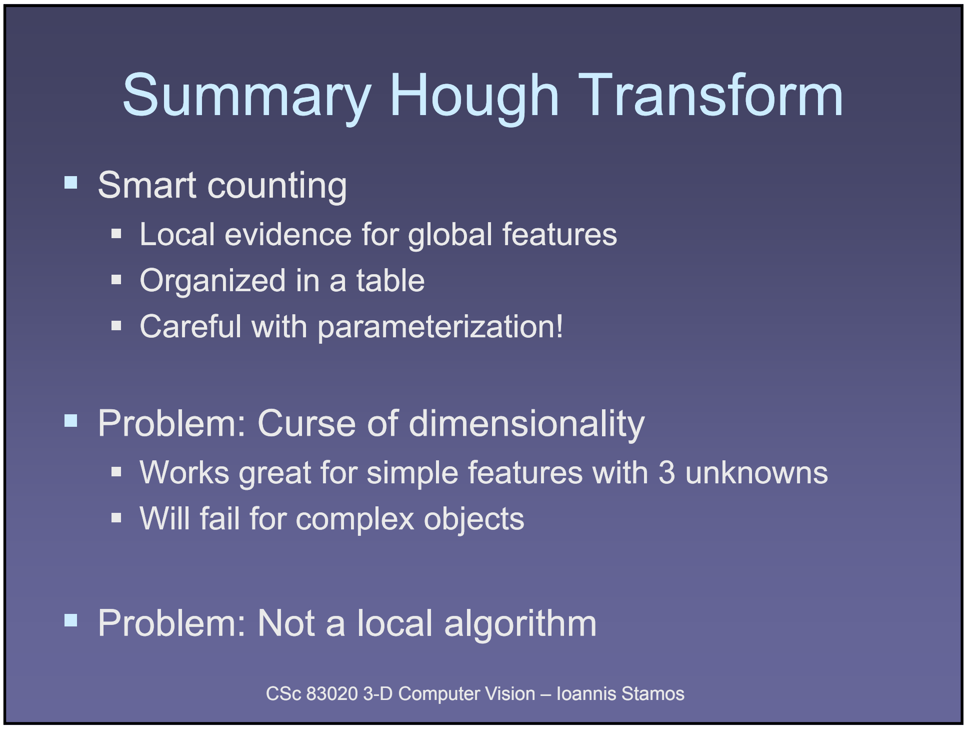


Hough Transform: Results

Image

Edge detection

Hough Transform



Summary Hough Transform

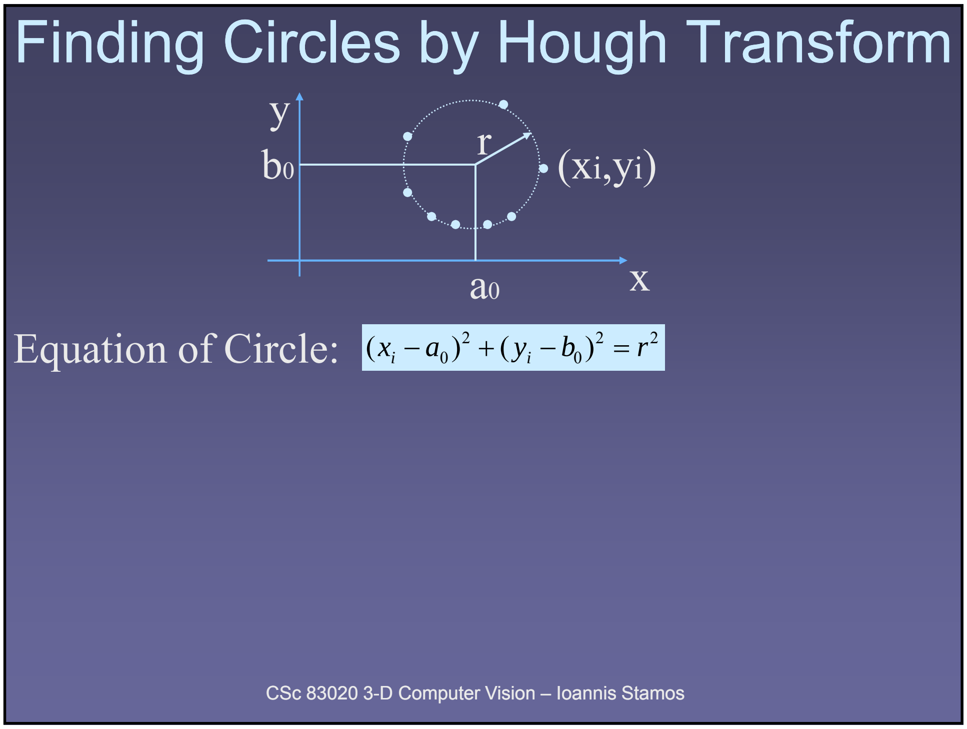
- Smart counting
  - Local evidence for global features
  - Organized in a table
  - Careful with parameterization!
- Problem: Curse of dimensionality
  - Works great for simple features with 3 unknowns
  - Will fail for complex objects
- Problem: Not a local algorithm

Hough Transform

- What else can be detected using Hough transform?

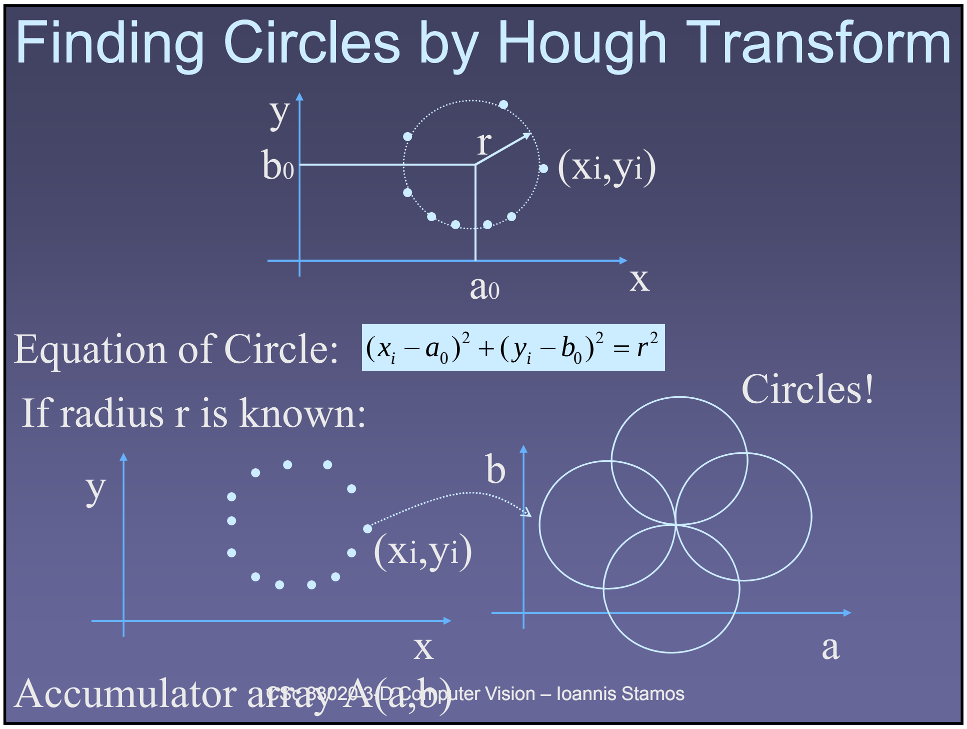
Hough Transform

- What else can be detected using Hough transform?
- Anything, but *dimensionality* is key



Finding Circles by Hough Transform

Equation of Circle: $(x_i - a_0)^2 + (y_i - b_0)^2 = r^2$

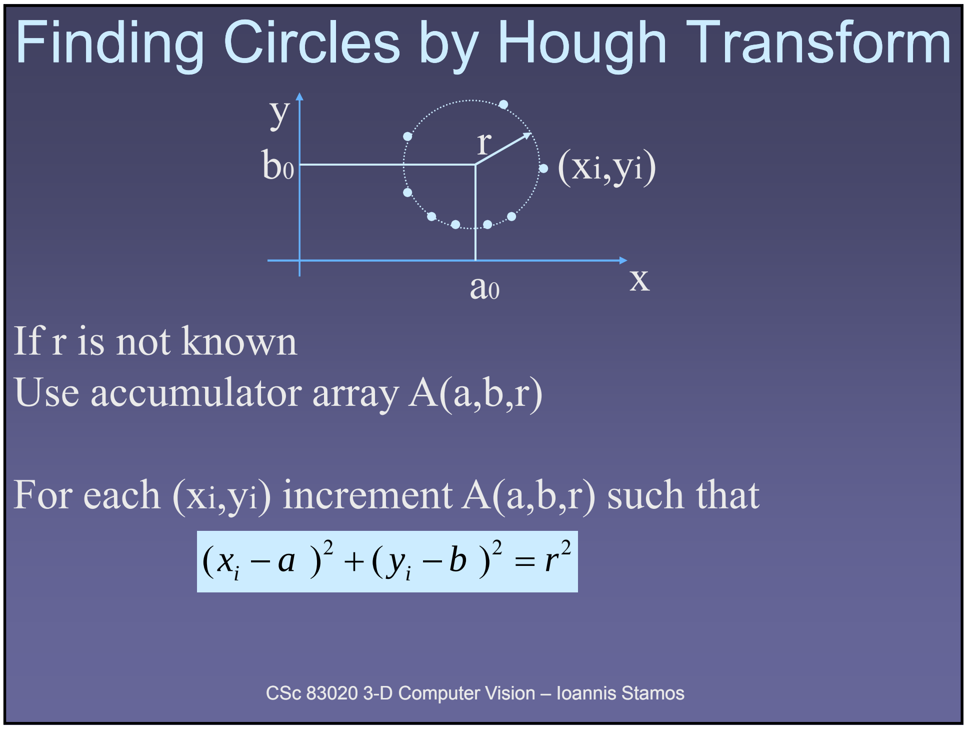


Finding Circles by Hough Transform

Equation of Circle: $(x_i - a_0)^2 + (y_i - b_0)^2 = r^2$

If radius r is known:

Accumulator array A(a,b)

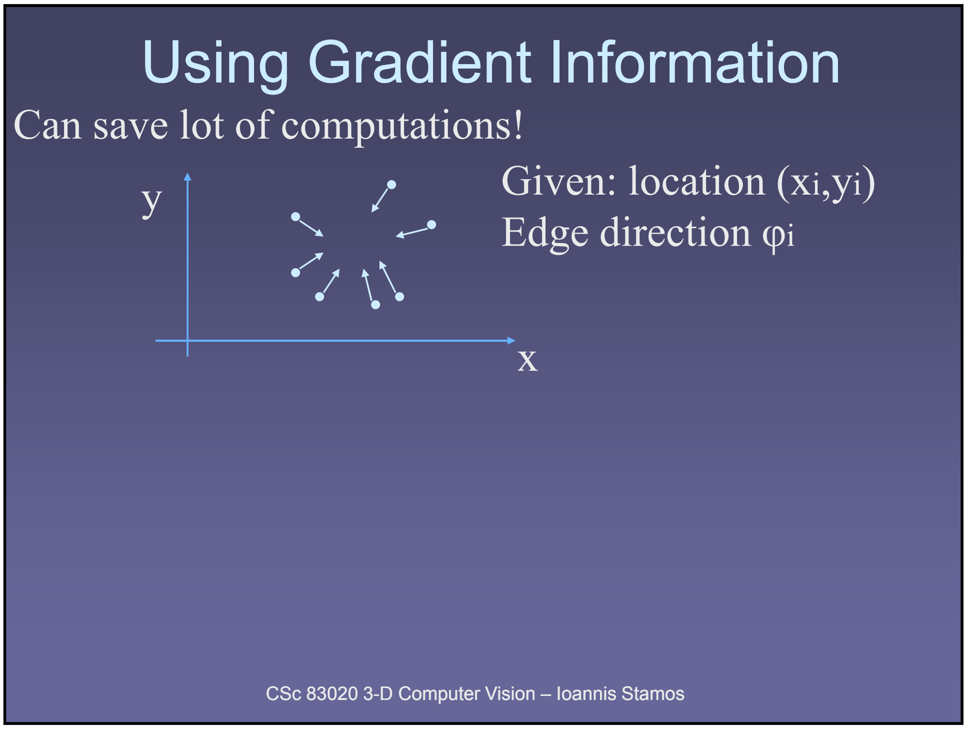


Finding Circles by Hough Transform

If r is not known
Use accumulator array A(a,b,r)

For each (xi,yi) increment A(a,b,r) such that

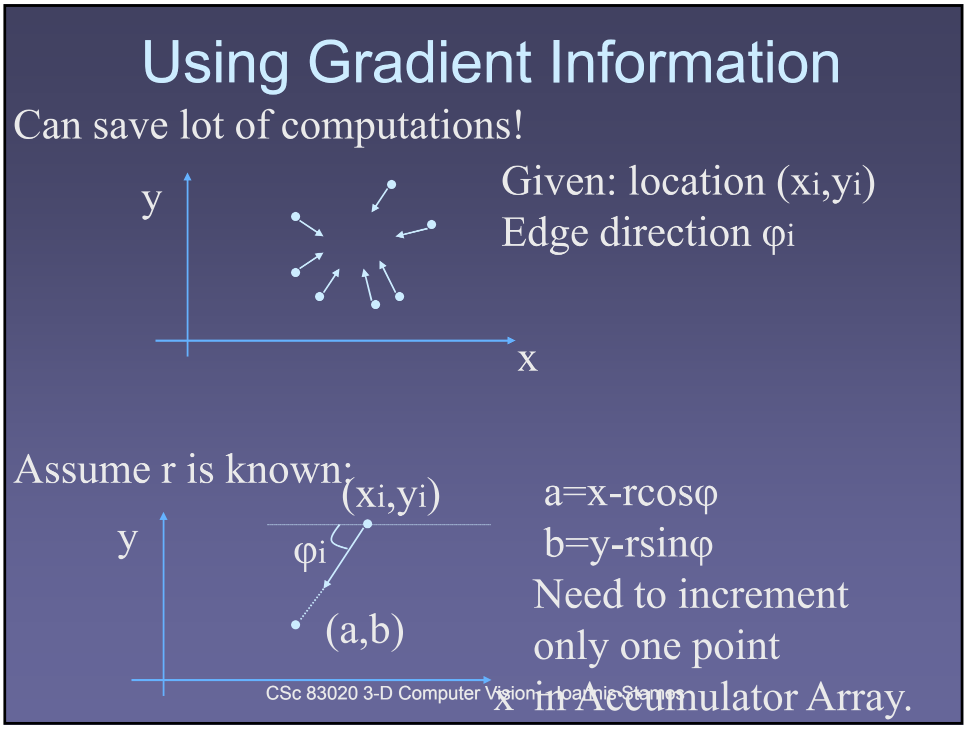
$$(x_i - a)^2 + (y_i - b)^2 = r^2$$



Using Gradient Information

Can save lot of computations!

Given: location (xi,yi)
Edge direction φi



Using Gradient Information

Can save lot of computations!

Given: location (x_i,y_i)

Edge direction φ_i

Assume r is known:

$a=x-r\cos\varphi$

$b=y-r\sin\varphi$

Need to increment only one point in Accumulator Array.