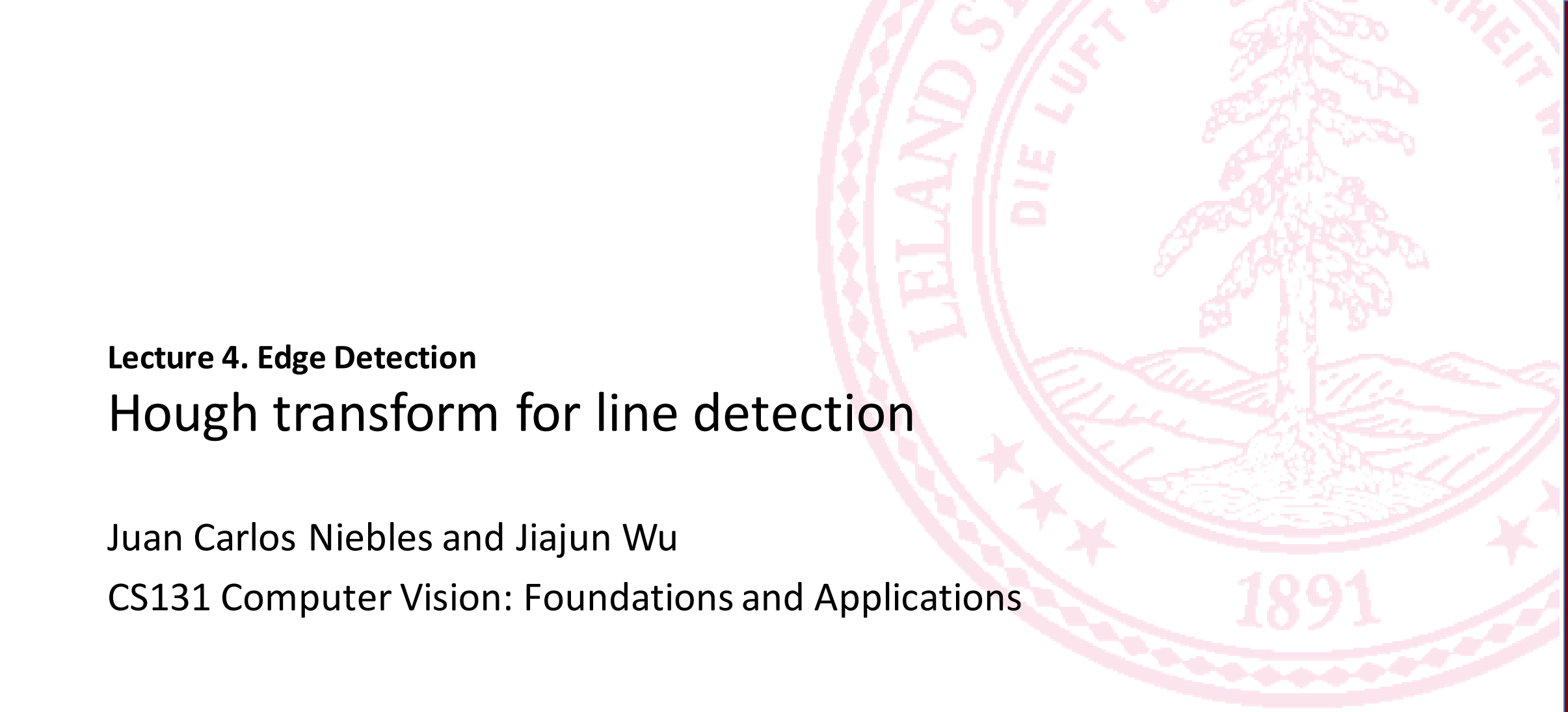


Lecture 4. Edge Detection

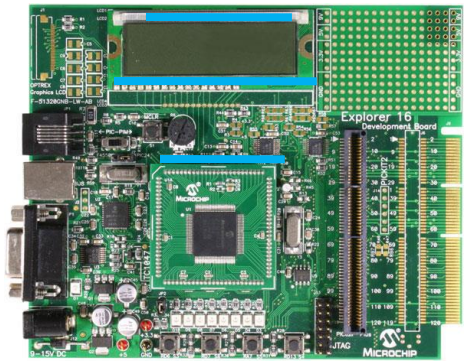
Hough transform for line detection

Juan Carlos Niebles and Jiajun Wu

CS131 Computer Vision: Foundations and Applications

Line Detection

- Why detect lines? Many objects characterized by presence of straight lines



- Wait, why aren't we done just by running edge detection?

Slide credit: Kristen Grauman

Intro to Hough transform

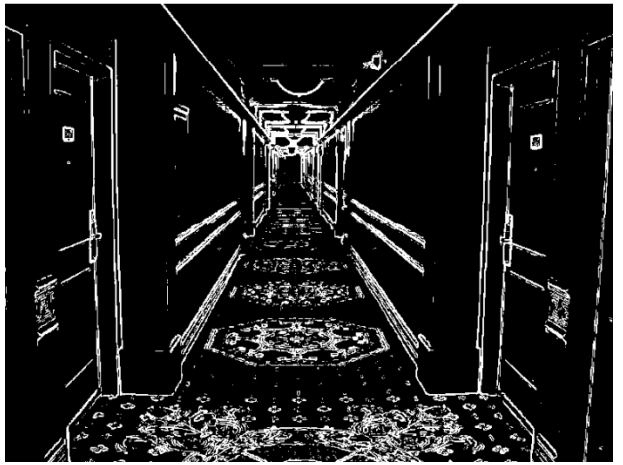
- The Hough transform (HT) can be used to detect lines.
- It was introduced in 1962 (Hough 1962) and first used to find lines in images a decade later (Duda 1972).
- Our goal with the Hough Transform is to find the location of lines in images.
- Hough transform can detect lines, circles and other structures ONLY if their parametric equation is known.
- It can give robust detection under noise and partial occlusion

Prior to Hough transform

- Assume that we have performed edge detection, for example, by thresholding the gradient magnitude image.
- Thus, we have some pixels that may partially describe the boundary of some objects.

Input Image

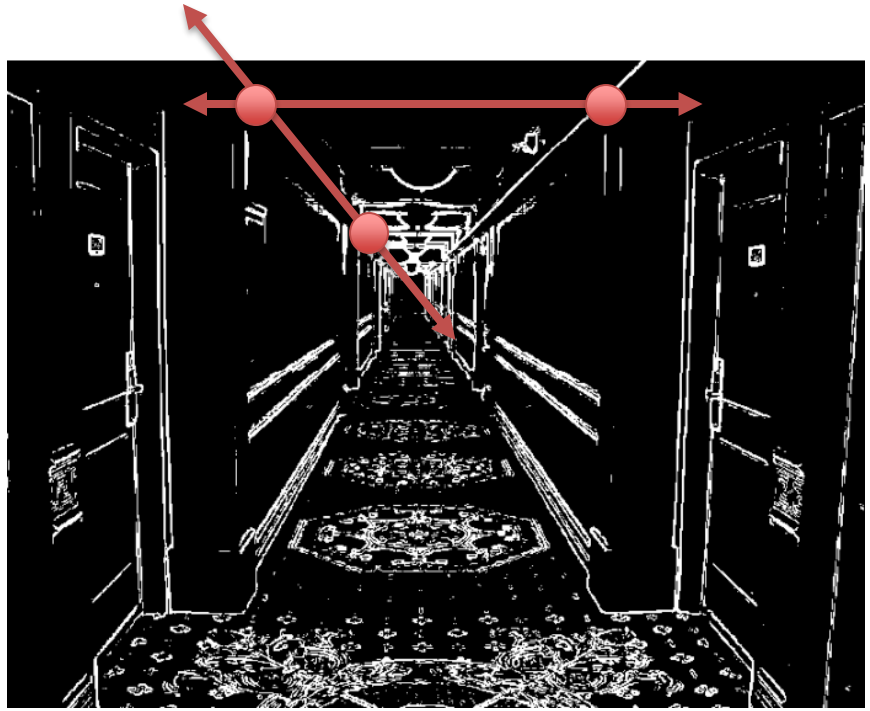
Image Gradients



Edge map
(binary image)

Naïve Line Detection

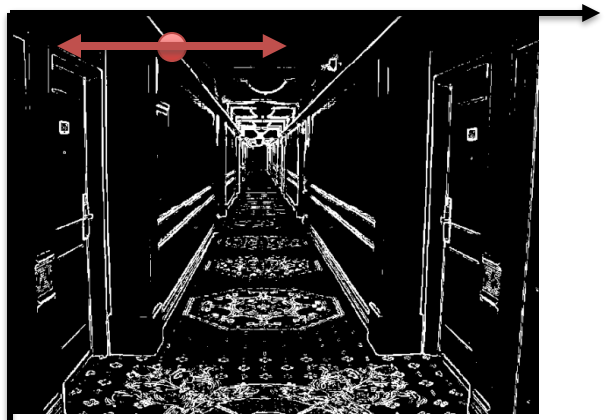
- For every pair of edge pixels
  - Compute equation of line
  - Check if other pixels satisfy equation
- Complexity?
  - $O(N^2)$ for an image with N edge pixels
- We can do better with the Hough Transform!



Edge map
(binary image)

Detecting lines using Hough transform

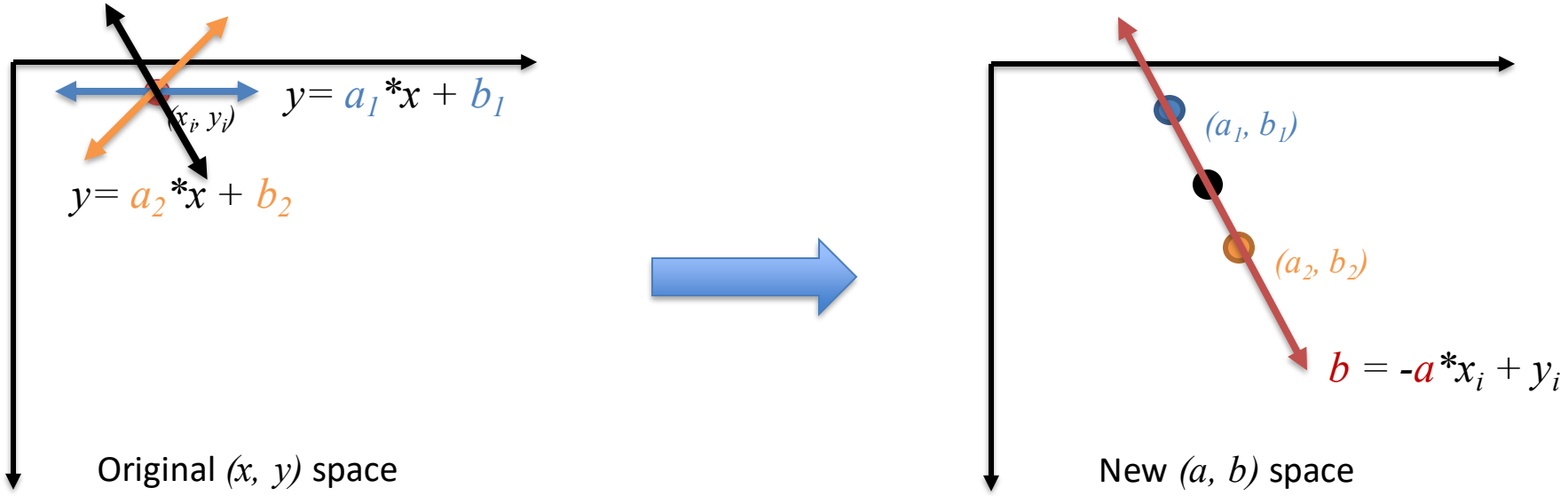
- We wish to find sets of pixels that make up straight lines.
- First step is to transform edge points into a new space.
- Consider an edge point of known coordinates (x_i, y_i) :
  - There are many potential lines passing through the point (x_i, y_i).
- This family of lines have the form $y_i = a*x_i + b$



Original (x, y) space

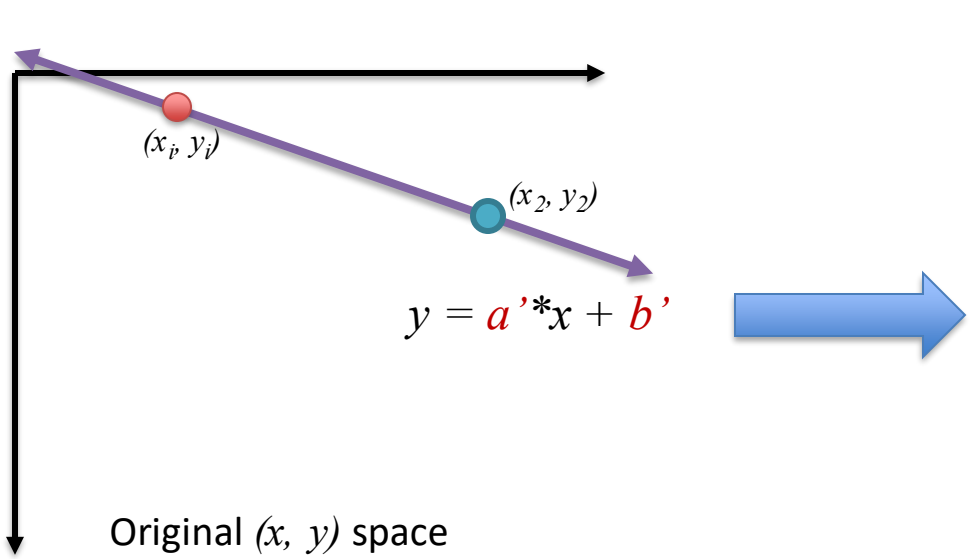
Detecting lines using Hough transform

- This family of lines have the form $y_i = a*x_i + b$.
- Note (x_i, y_i) are constants, while (a, b) can change. This gives rise to a new space where (a, b) are the variables.
- That means, a point (x_i, y_i) transforms into a line in the (a, b) space: $b = -a*x_i + y_i$.

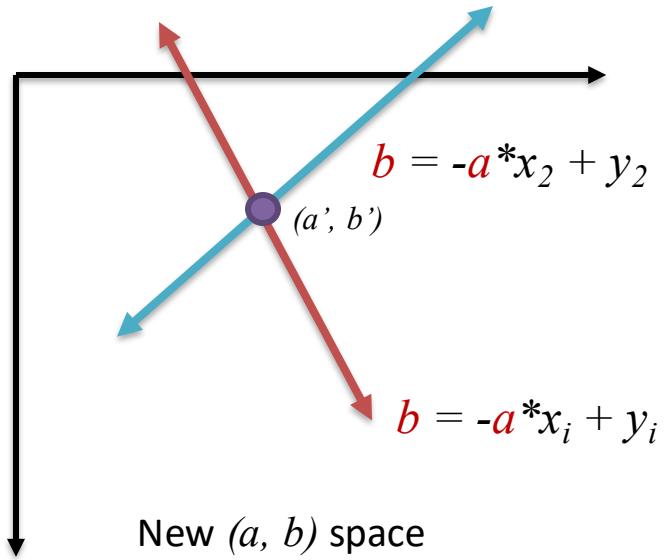


Detecting lines using Hough transform

- This family of lines have the form $y_i = a*x_i + b$.
- Note (x_i, y_i) are constants, while (a, b) can change. This gives rise to a new space where (a, b) are the variables.
- That means, a point (x_i, y_i) transforms into a line in the (a, b) space: $b = -a*x_i + y_i$.
- Another edge point (x_2, y_2) will give rise to another line in the (a, b) space.



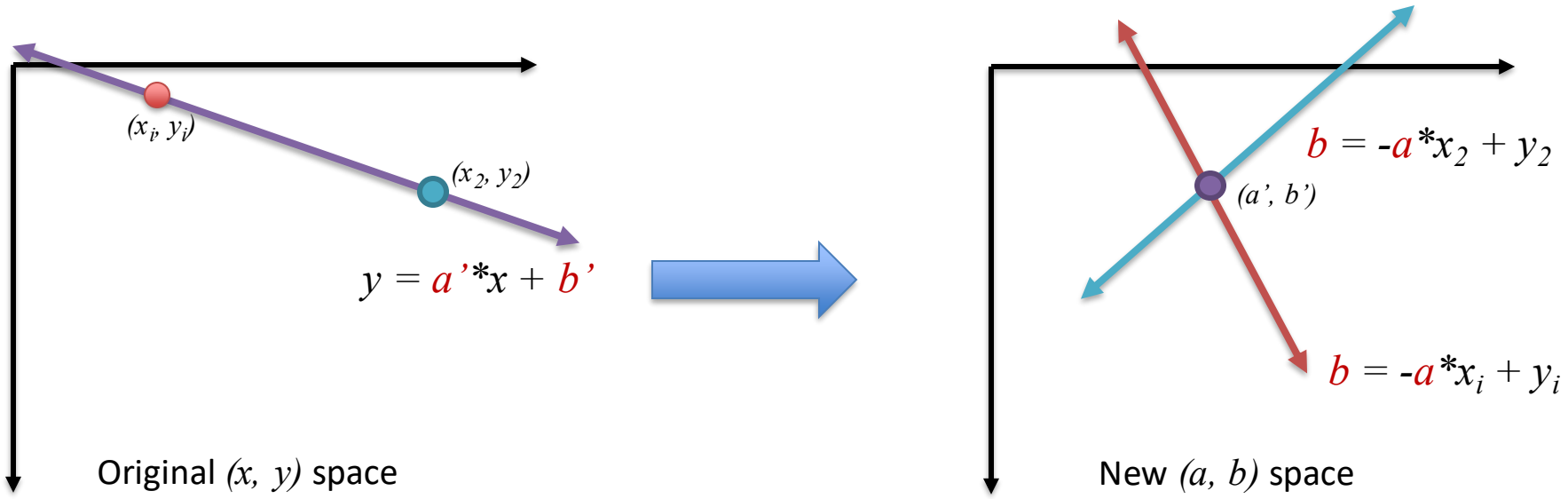
Original (x, y) space



New (a, b) space

Detecting lines using Hough transform

- Colinear points in the (x, y) space transform into lines in the (a, b) space that intersect at a single point (a', b').
- We can detect lines by finding such intersection points (a', b') in the (a, b) space.
- Our resulting line equation in the original space is $y = a'*x + b'$.

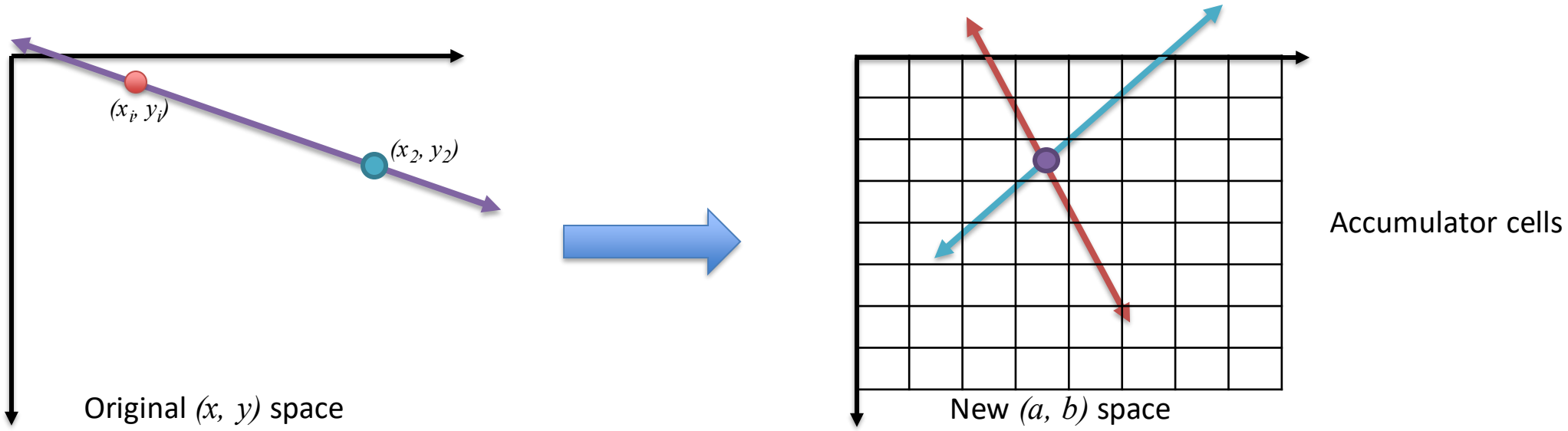


Original (x, y) space

New (a, b) space

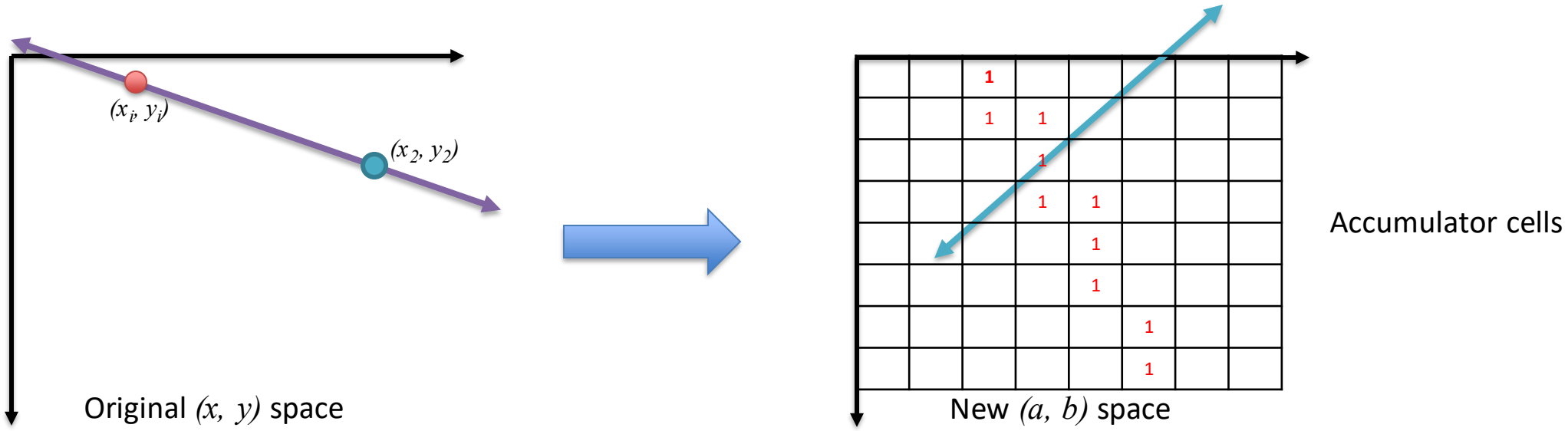
Detecting lines using Hough transform

- We efficiently find the intersection points in the (a, b) space by quantizing it into cells.
- Instead of transforming a point to an explicit line, we vote on the discrete cells that are 'activated' by the transformed line in (a, b).



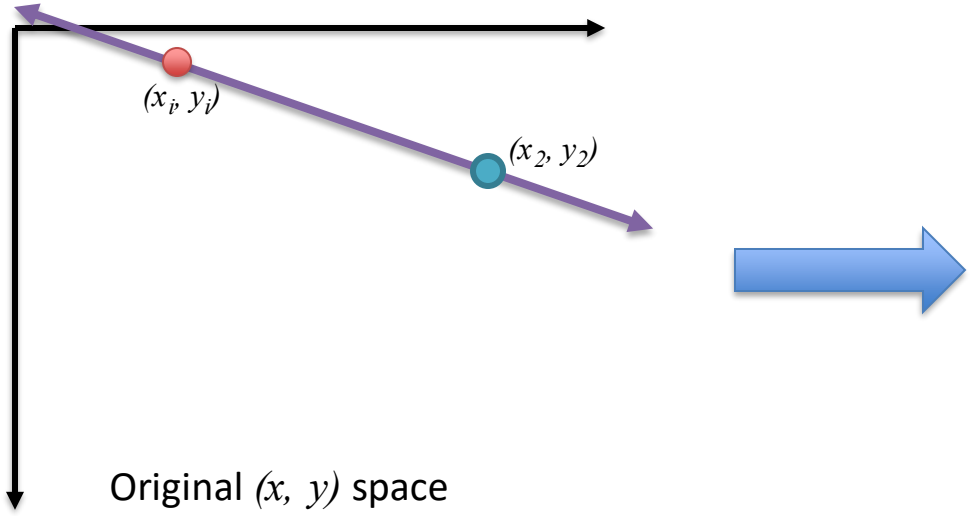
Detecting lines using Hough transform

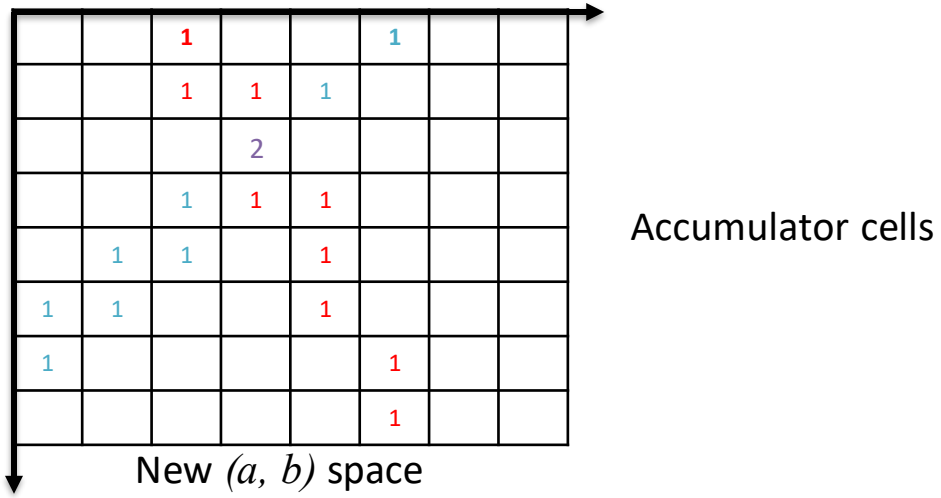
- We efficiently find the intersection points in the (a, b) space by quantizing it into cells.
- Instead of transforming a point to an explicit line, we vote on the discrete cells that are 'activated' by the transformed line in (a, b).



Detecting lines using Hough transform

- We efficiently find the intersection points in the (a, b) space by quantizing it into cells.
- Instead of transforming a point to an explicit line, we vote on the discrete cells that are 'activated' by the transformed line in (a, b).
- Cells that receive more than a certain number of votes are assumed to correspond to lines in (x, y) space

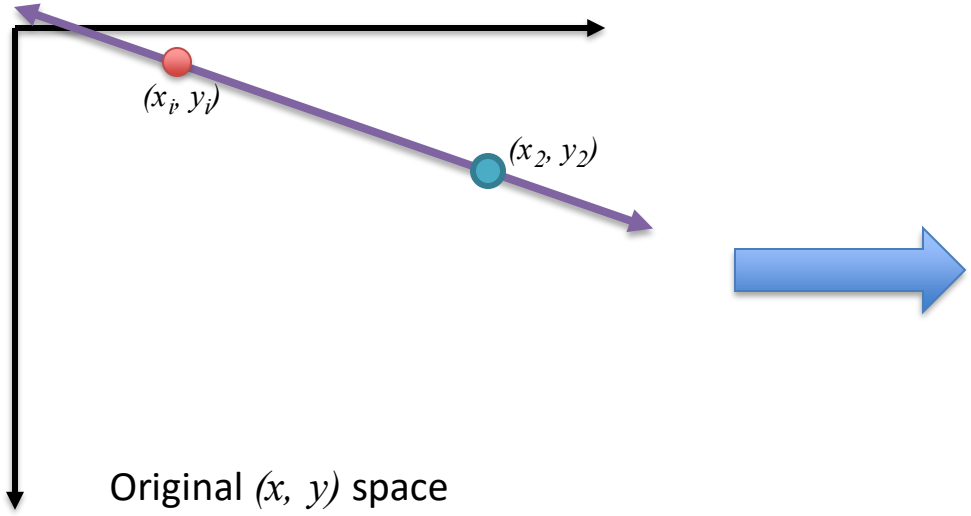




Accumulator cells

Hough Transform Algorithm

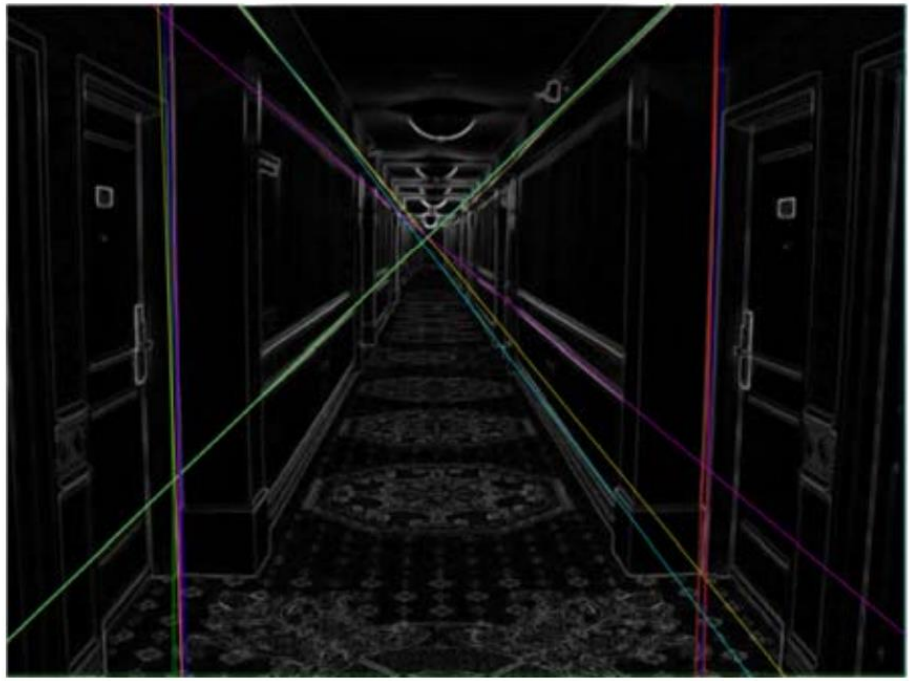
- For each (x, y) edge point:
  - Vote on cells that satisfy the corresponding (a, b) line equation
- Find cells with more votes than threshold.
- Complexity?
  - Linear on number of edge points
  - Linear on number of accumulator cells



Accumulator cells

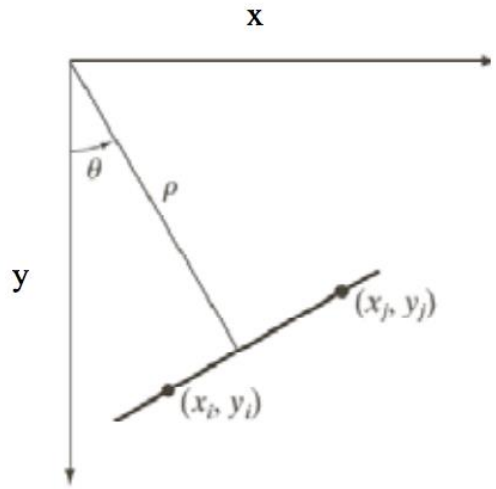
Output of Hough transform

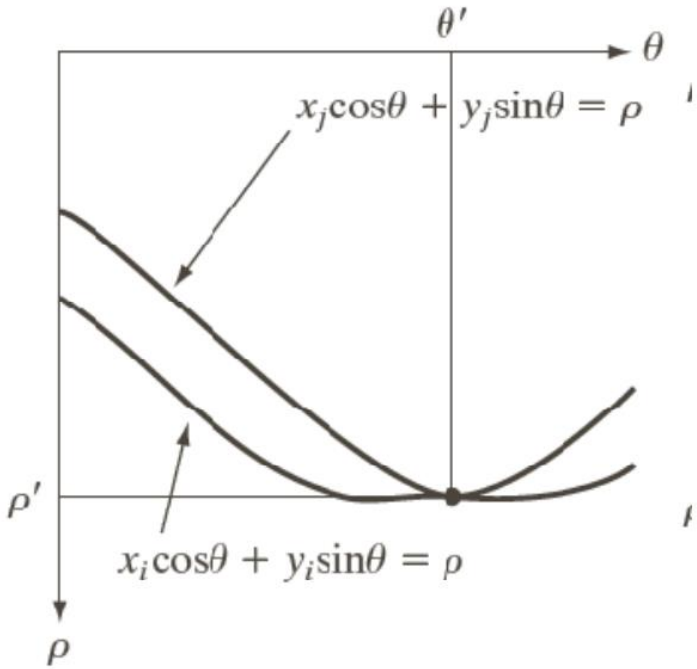
- Here are the top 20 most voted lines in the image:



Other Hough transformations

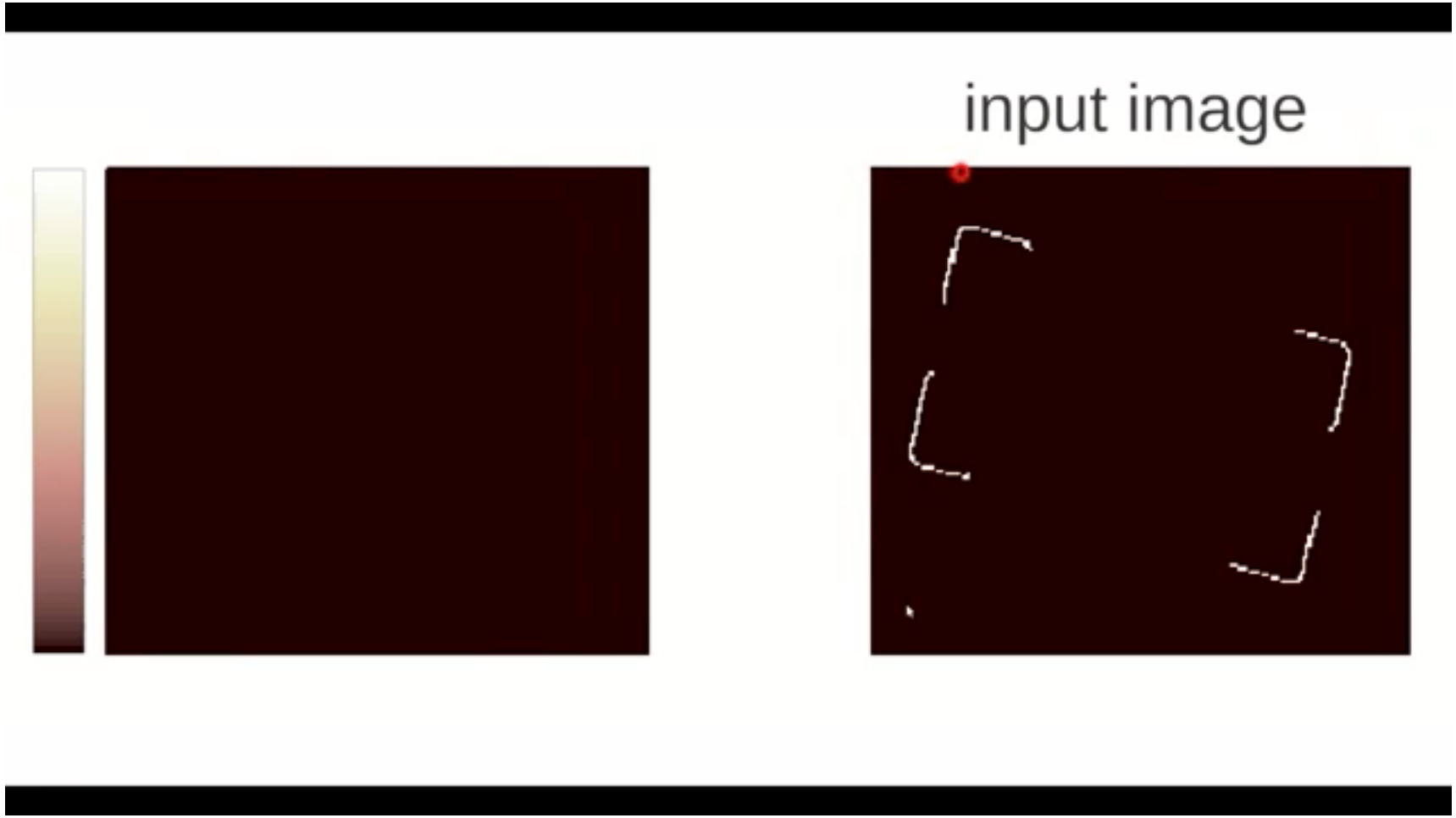
- We can represent lines as polar coordinates instead of $y = a*x + b$
- Polar coordinate representation:
  - $x*cos\theta + y*sin\theta = \rho$
- A vertical line will have $\theta = 90$ and ρ equal to the intercept with the x-axis.
- A horizontal line will have $\theta = 0$ and ρ equal to the intercept with the y-axis.
- Note that lines in (x, y) space are not lines in (ρ, θ) space, unlike (a, b) space.





Example video

- https://youtu.be/4zHbI-fFIlI?t=3m35s



Concluding remarks

- Advantages:
  - Conceptually simple.
  - Easy implementation.
  - Handles missing and occluded data very gracefully.
  - Can be adapted to many types of forms, not just lines.
- Disadvantages:
  - Computationally complex for objects with many parameters.
  - Looks for only one single type of object.
  - Can be “fooled” by “apparent lines”.
  - The length and the position of a line segment cannot be determined.
  - Co-linear line segments cannot be separated.