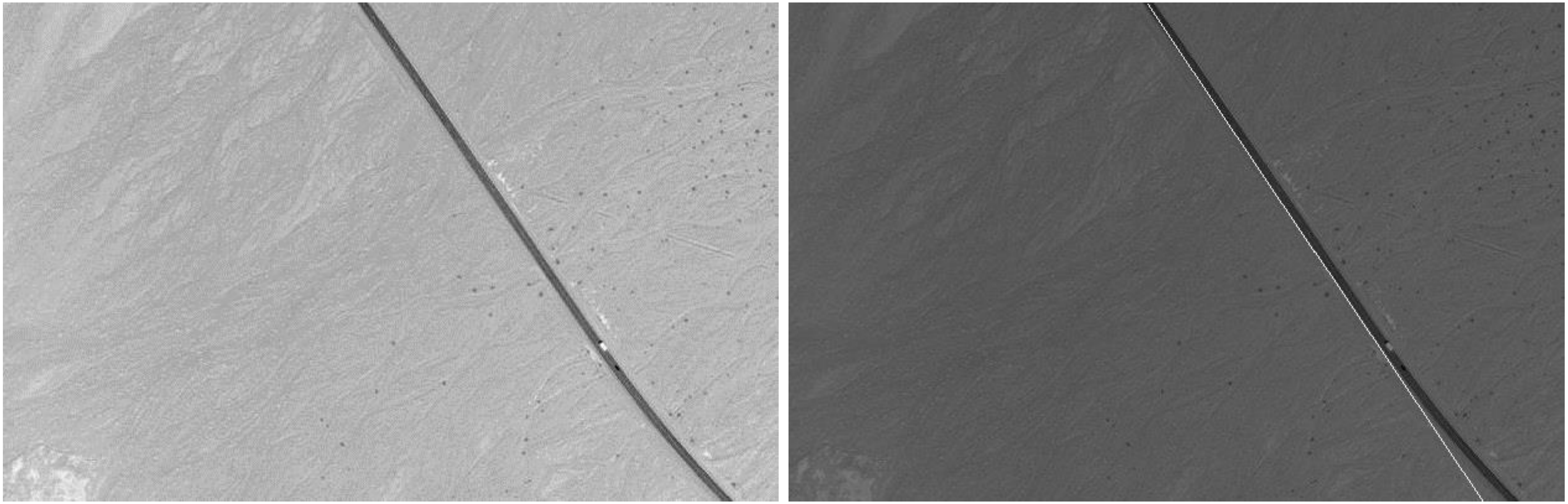
Hough Transforms

CSE 6367 – Computer Vision

Vassilis Athitsos
University of Texas at Arlington

Hough Transforms

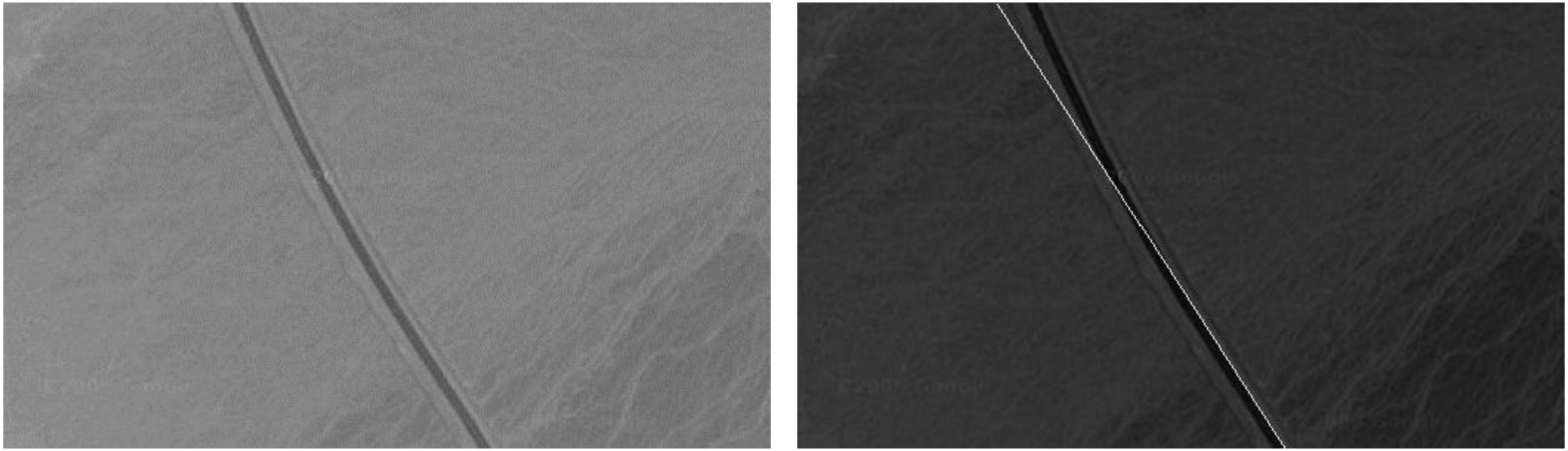
- Goal: identify simple geometric shapes in images, such as lines and circles.



Example: find the most prominent line in an image.

Hough Transforms

- Goal: identify simple geometric shapes in images, such as lines and circles.



Example: find the most prominent line in an image.

Code for Previous Results

```
function result = hough_demo(image, threshold, result_number)

% detect edges
edges = canny(image, threshold);

% find lines
[h theta rho] = hough(edges);

% draw the most prominent line on the image.
result = image * 0.7;
for i = 1:result_number
    max_value = max(max(h));
    [rho_indices, theta_indices] = find(h == max_value);
    rho_index = rho_indices(1);
    theta_index = theta_indices(1);
    distance = rho(rho_index);
    angle = theta(theta_index);
    result = draw_line2(result, distance, angle);
    h(rho_index, theta_index) = 0;
end
```

What is a Straight Line?

What is a Straight Line?

- It is infinite.
- It is defined in multiple ways:

Defining Straight Lines

- How many parameters do we need to define a line?

Defining Straight Lines

- How many parameters define a line?
- One option:
  - – a point (x0, y0) and a theta.
- Another option:
  - – two points (x0, y0) and (x1, y1).
- NOTE: for representing 2D points, two conventions are common:
    - (x, y), where x is horizontal coord., y is vertical coord.
    - (i, j), where i is vertical coord, j is horizontal coord.
  - – In any piece of code, ALWAYS VERIFY THE CONVENTION.

Defining Straight Lines

- How many parameters define a line?
- One option:
  - a point (x0, y0) and a theta.
- Another option:
  - two points (x0, y0) and (x1, y1).
- Any problem with the above two parametrizations?

Defining Straight Lines

- How many parameters define a line?
- One option:
  - – a point (x0, y0) and a theta.
- Another option:
  - – two points (x0, y0) and (x1, y1).
- Any problem with the above two parametrizations?
  - – It is redundant.
    - A line has infinite parametrizations/representations.

Defining Straight Lines

- $y = c1 * x + c2$
  - Problems?

Defining Straight Lines

- y = c1 * x + c2
  - The above parametrization cannot represent vertical lines.

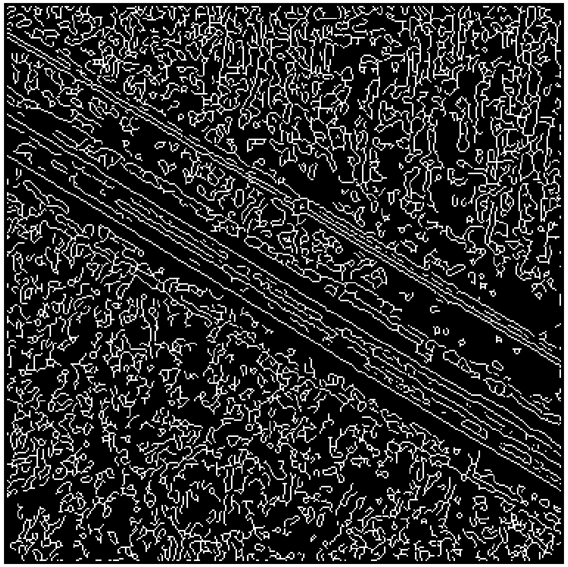
Defining Straight Lines

- Can we define a line in a non-redundant way?

Defining Straight Lines

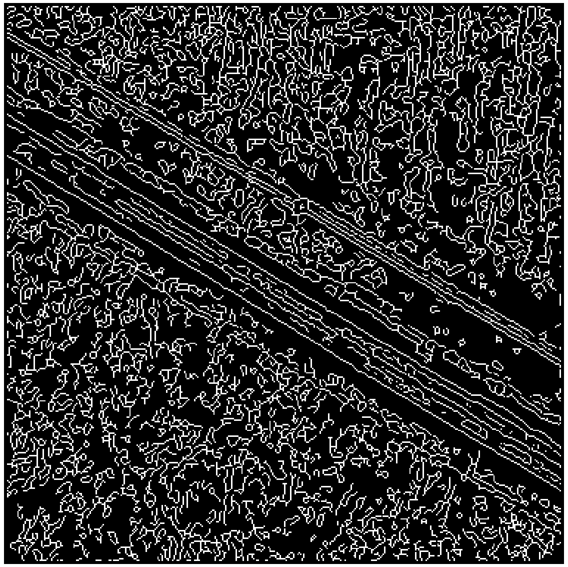
- Defining a line using rho and theta:
- rho = x*cos(theta) + y*sin(theta)
  - rho: distance of line from origin.
  - theta: direction **PERPENDICULAR** to line.
  - The line is the set of (x, y) values satisfying the equation.

Voting for Lines



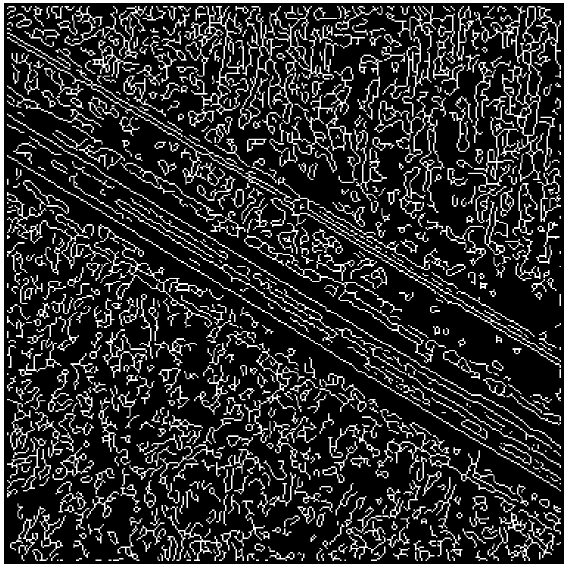
- Every edge pixel votes for all the lines it is a part of.
- Vote array: # of rhos x # of thetas.
  - We choose how much we want to discretize.
  - Votes collected in a single for loop.

Voting for Lines - Pseudocode



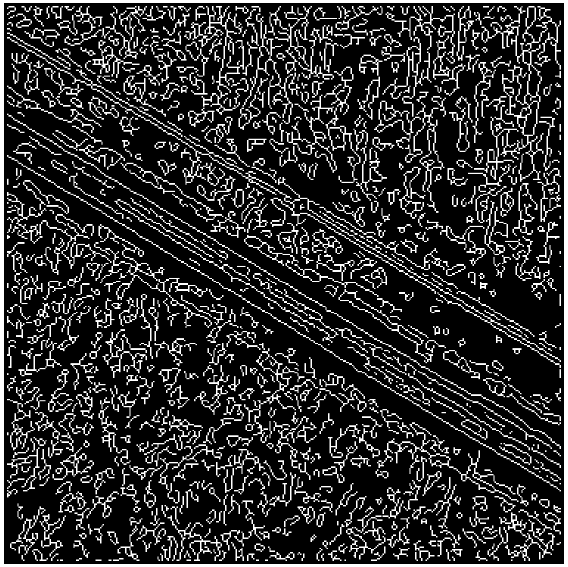
```
• counters = zeros(# of rhos, # of thetas)
• For every pixel (i, j):
    – if (i, j) not an edge pixel, then continue
    – for every theta in thetas:
        • rho = find_rho(i, j, theta)
        • counters(rho, theta)++;
```

Voting for Lines - Pseudocode



- counters = zeros(# of rhos, # of thetas)
- For every pixel (i, j)*:*
  - – if (i, j) not an edge pixel, then continue
  - – for every theta in thetas:
    - rho = find_rho(i, j, theta)
    - counters(rho, theta)++;
- How long does it take?

Voting for Lines - Pseudocode



- counters = zeros(# of rhos, # of thetas)
- For every pixel (i, j)*:*
  - – if (i, j) not an edge pixel, then continue
  - – for every theta in thetas:
    - rho = find_rho(i, j, theta)
    - counters(rho, theta)++;
- How long does it take?
  - – # pixels * # thetas.
  - – # edge pixels * # thetas.

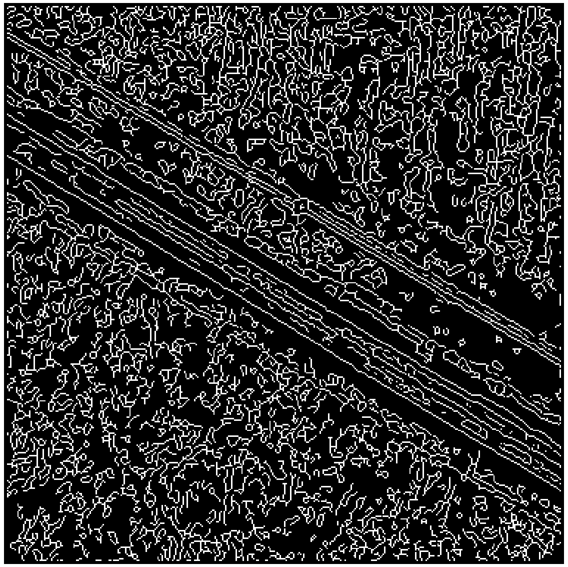
Hough Transform for Lines

- The Hough transform for lines simply computes the votes for all lines.

Using the Matlab Hough Function

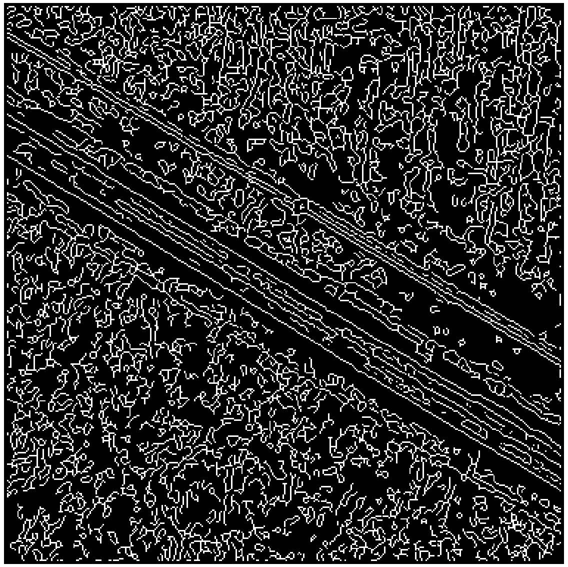
```
function result = hough_demo(image, threshold, result_number)

% calls canny(image, threshold), does hough transform
% on the resulting edge
% image, and draws the top "result_number" results.

edges = canny(image, threshold);
[h theta rho] = hough(edges);

result = image * 0.5;
for i = 1:result_number
    max_value = max(max(h));
    [rho_indices, theta_indices] = find(h == max_value);
    rho_index = rho_indices(1);
    theta_index = theta_indices(1);
    distance = rho(rho_index);
    angle = theta(theta_index);
    result = draw_line2(result, distance, angle);
    h(rho_index, theta_index) = 0;
end
```

Voting for Lines



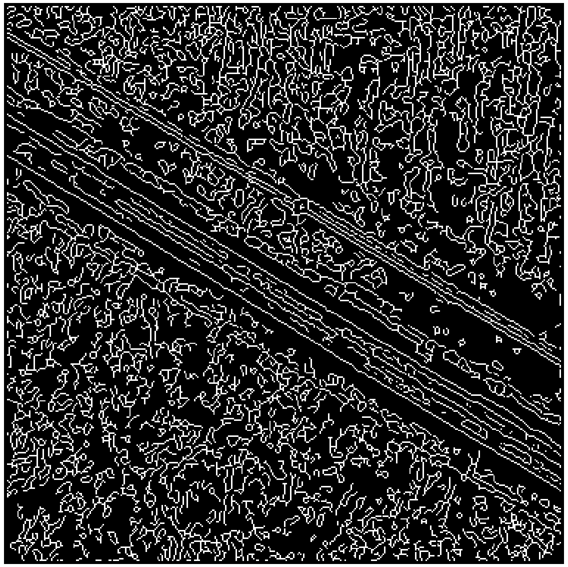
- Using edge orientations to make it faster:

Voting for Lines



- Using edge orientations to make it faster:
- Use the orientation of an edge pixel to limit the thetas that it votes for.

Voting for Lines – Pseudocode 2



```
• counters = zeros(# of rhos, # of thetas)
• For every pixel (i, j):
    – if (i, j) not an edge pixel, then continue
    – o = gradient_orientations(i, j)
    – pixel_thetas = thetas such that abs(angle(o, theta)) <= thr.
    – for every theta in pixel_thetas:
        • rho = find_rho(i, j, theta)
        • counters(rho, theta)++;
```

Defining Circles

Defining Circles

- Parameters: center_x, center_y, radius.
- $(x - center\_x)^2 + (y - center\_y)^2 = radius^2$
- The circle is the set of all (x, y) values satisfying the above equation.

Hough Transform on Circles

- What is the voting space?
- Who votes?
- What does each voter vote for?

Hough Transform on Circles

- What is the voting space?
  - – the set of all circles that can be defined.
  - – size of array:
- Who votes?
- What does each voter vote for?

Hough Transform on Circles

- What is the voting space?
  - the set of all circles that can be defined.
  - size of array: # centers * # radii.
- Who votes?
- What does each voter vote for?

Hough Transform on Circles

- What is the voting space?
  - the set of all circles that can be defined.
  - size of array: # centers * # radii.
  - Coarser discretizations can be used.
    - combine 3x3 neighborhoods for center locations.
    - choose step at which radii are sampled.
- Who votes?
- What does each voter vote for?

Hough Transform on Circles

- What is the voting space?
  - the set of all circles that can be defined.
- Who votes?
- What does each voter vote for?

Hough Transform on Circles

- What is the voting space?
  - – the set of all circles that can be defined.
- Who votes?
  - – Every edge pixel.
- What does each voter vote for?

Hough Transform on Circles

- What is the voting space?
  - the set of all circles that can be defined.
- Who votes?
  - Every edge pixel.
- What does each voter vote for?
  - Every edge pixel votes for all the circles that it belongs to.

Hough Transform on Circles

- What is the voting space?
  - – the set of all circles that can be defined.
- Who votes?
  - – Every edge pixel.
- What does each voter vote for?
  - – Every edge pixel votes for all the circles that it can belong to.
  - – Faster version:

Hough Transform on Circles

- What is the voting space?
  - – the set of all circles that can be defined.
- Who votes?
  - – Every edge pixel.
- What does each voter vote for?
  - – Every edge pixel votes for all the circles that it can belong to.
  - – Faster version: every edge pixel (i, j) votes for all the circles that it can belong to, such that the line from (i, j) to the circle center makes a relatively small angle with the gradient orientation at (i, j).

Hough Transform on Circles

- What does each voter vote for?
  - – Every edge pixel votes for all the circles that it can belong to.
  - – Faster version: every edge pixel (i, j) votes for all the circles that it can belong to, such that the line from (i, j) to the circle center makes a relatively small angle with the gradient orientation at (i, j).
- How long does it take?

Hough Transform on Circles

- What does each voter vote for?
  - – Every edge pixel votes for all the circles that it can belong to.
  - – Faster version: every edge pixel (i, j) votes for all the circles that it can belong to, such that the line from (i, j) to the circle center makes a relatively small angle with the gradient orientation at (i, j).
- How long does it take?
  - – # edge pixels * # pixels * # radii.

General Hough Transforms

- In theory, we can use Hough transforms to detect more complicated shapes.
- In practice, this requires memory and time *exponential* to the number of parameters, and is usually not practical.