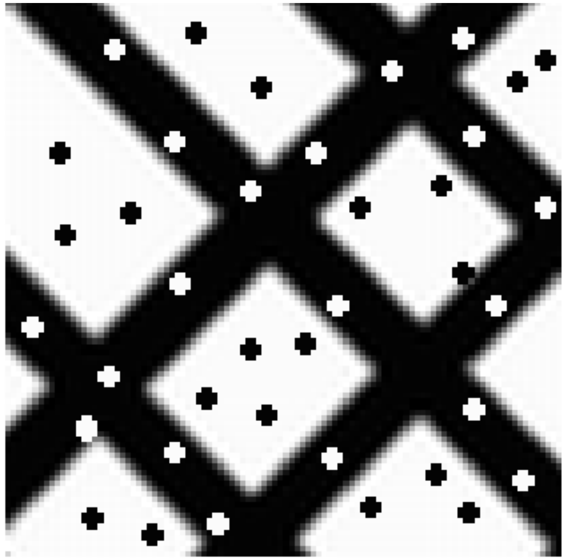
PRS L3

Hough Transform for line detection

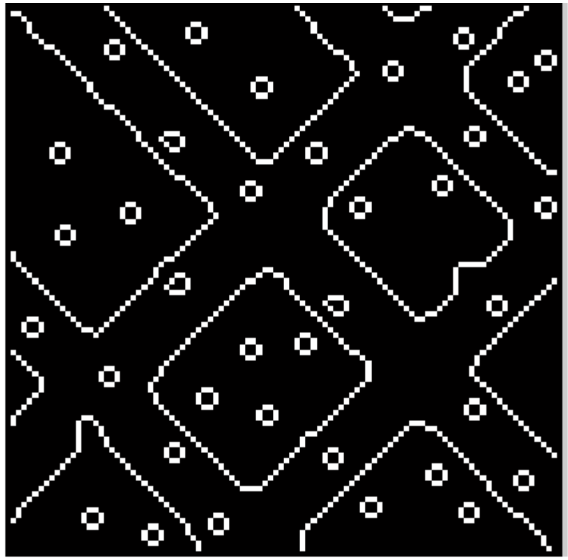
Hough

- Objective: finding lines in an image that contains a set of interest points. **Input:**
- Binary image
- **Output:** a set of lines



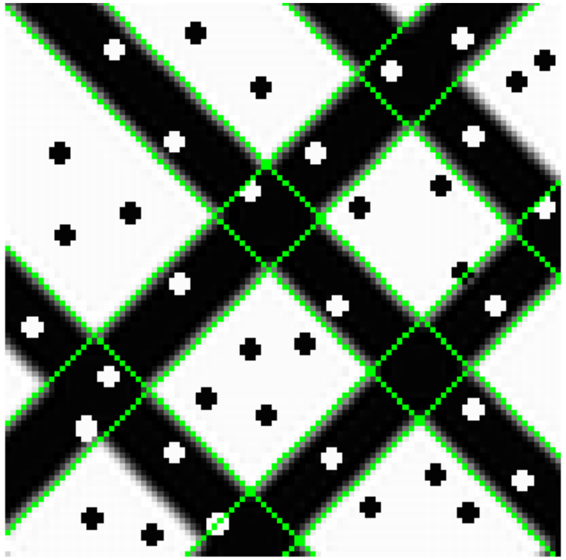
a.

A grayscale image containing a pattern with straight borders corrupted by salt-and-pepper like noise



b.

A binary image with the edges detected with the Canny edge detector



c.

The most relevant lines given by Hough by processing image b. are displayed with green

The line in Hesse normal form

- the line is represented by the normal vector and the distance from origin to the line along its direction
- This representation is also called the normal parameterization or the ρ-θ representation
- This line parametrization is used by Hough

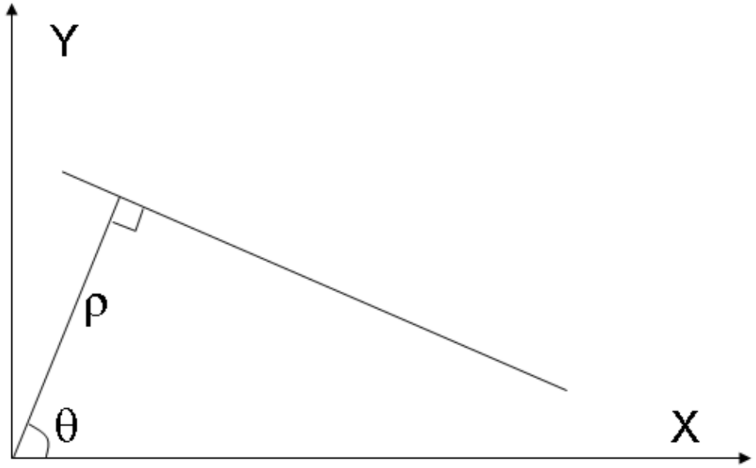


Fig. 1. Line represented by its normal vector at angle θ and the distance ρ along the normal vector from the origin to the line.

The line in Hesse normal form

- Line equation: $\rho = x \cos(\theta) + y \sin(\theta)$

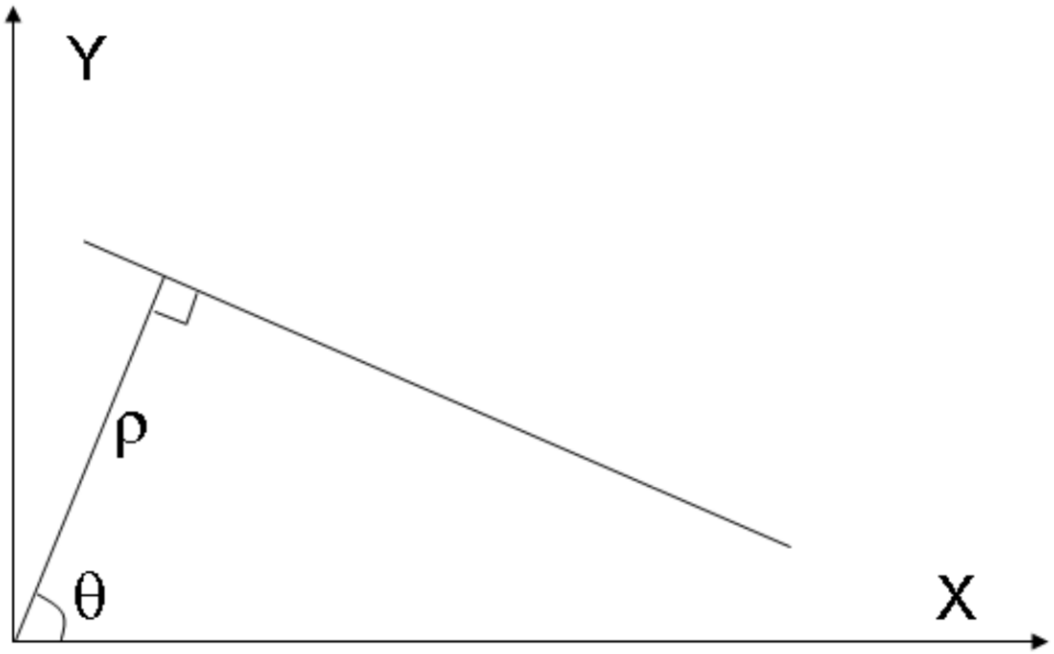
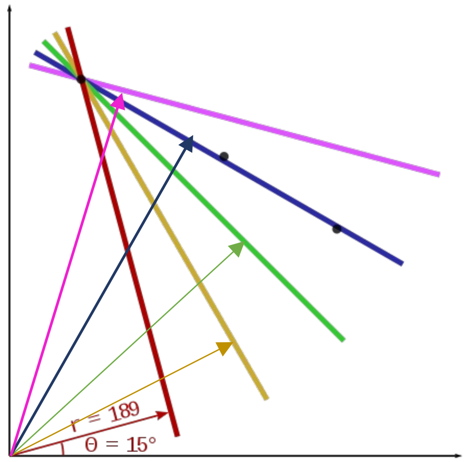


Fig. 1. Line represented by its normal vector at angle θ
and the distance ρ along the normal vector from the origin to the line.

Hough Transform

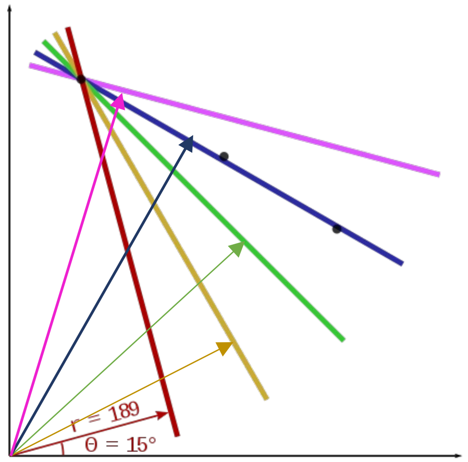
- In general, for each data point in the image plane (x_0, y_0) , we can define a set of lines that cross that points at all at different angles.
- To each line, a support line exists which is perpendicular to it (ρ) and which intersects the origin.
- We can compute ρ and θ for each line



Θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

Hough Transform

- In general, for each data point in the image plane (x_0, y_0) , we can define a set of lines that cross that points at all at different angles.
- To each line, a support line exists which is perpendicular to it (ρ) and which intersects the origin.
- We can compute ρ and θ for each line

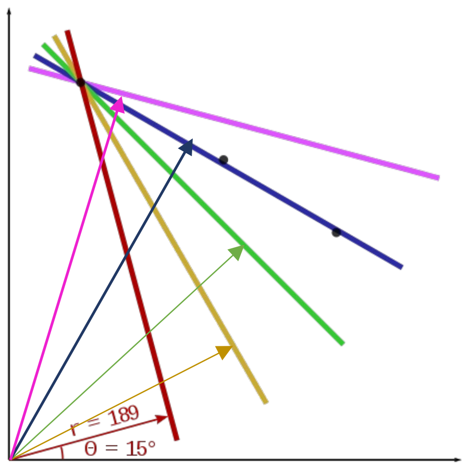


θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

- In this example, we plotted only the lines with θ = 15, 30, 45, 60, 75, but there is an infinity of lines that cross the point (x_0, y_0)
- Parameter quantization of (ρ , θ) (**Quantization**, in general, is the process of constraining an input from a continuous or otherwise large set of values (such as the real numbers) to a discrete set (such as the integers).)
- So we will detect lines with θ ∈ [0, 360) with a step of Δθ = 1 deg, so θ = 0, 1, 2, 3, 4 …., 359 (the step can be any other number 10 deg, 1 deg, 0.5 deg etc)
- In an image, ρ_{max} = the image diagonal = $\sqrt{height^2 + width^2}$, so $\rho \in [0, \rho_{max}]$ with a step of Δρ=1 pixel, so ρ = 0, 1, 2, 3, 4, …, ρ_{max}

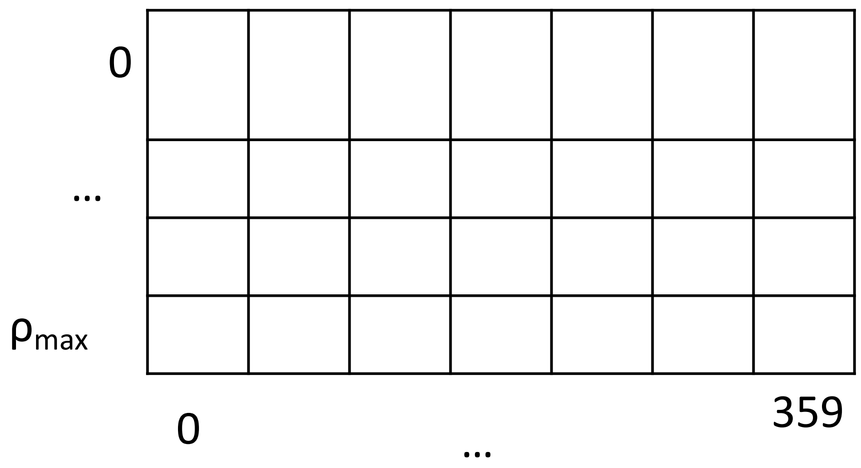
Hough Transform - Accumulator

- $\theta \in [0, 360)$ with a step of $\Delta\theta = 1$ deg, so $\theta = 0, 1, 2, 3, 4 \ldots, 359$
- $\rho \in [0, \rho_{max}]$ with a step of $\Delta\rho=1$ pixel, so $\rho = 0, 1, 2, 3, 4, \ldots, \rho_{max}$



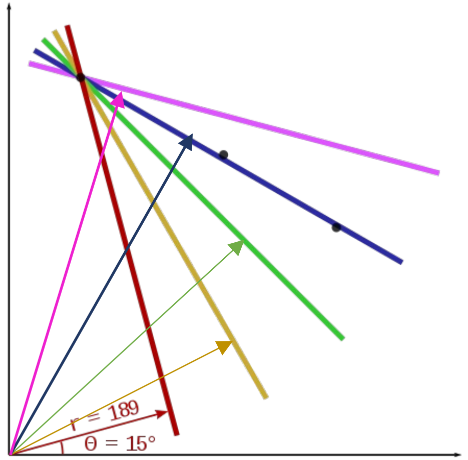
Θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

- Hough uses a 2D vector (matrix), named accumulator (H), for detecting the lines in the image
- The accumulator size is $\rho_{max}/\Delta\rho$ rows and $\theta_{max}/\Delta\theta$ columns
- For an image with height *h* and width *w*, the Hough accumulator has $\sqrt{h^2 + w^2} + 1$ rows and 360 columns



Hough Transform - Algorithm

- $\theta \in [0, 360)$ with a step of $\Delta\theta = 1$ deg, so $\theta = 0, 1, 2, 3, 4 \ldots., 359$
- $\rho \in [0, \rho_{max}]$ with a step of $\Delta\rho=1$ pixel, so $\rho = 0, 1, 2, 3, 4, \ldots, \rho_{max}$



Θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

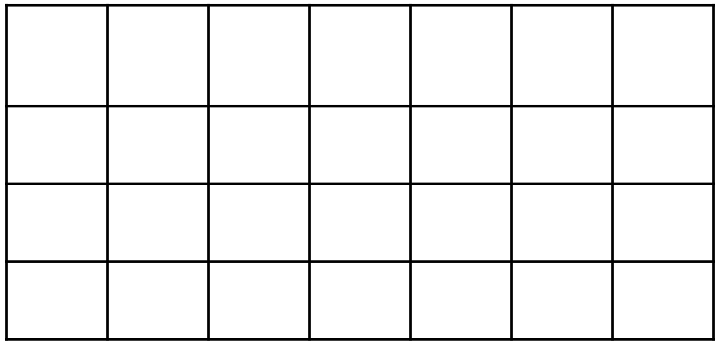
1. Define the accumulator and initialize the accumulator with zeros

```
H = np.zeros((ro_max, theta_max), dtype=np.uint)
```

0

…

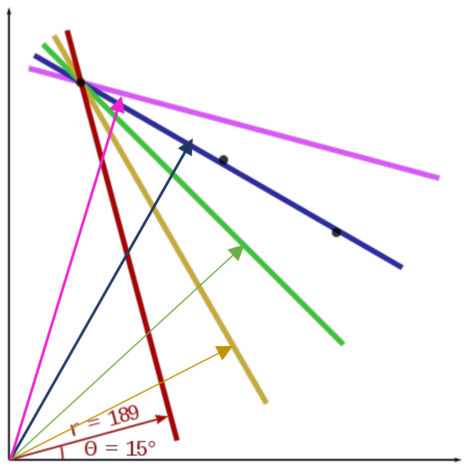
ρ_{max}



0 … 359

Hough Transform - Algorithm

- $\theta \in [0, 360)$ with a step of $\Delta\theta = 1$ deg, so $\theta = 0, 1, 2, 3, 4 \ldots., 359$
- $\rho \in [0, \rho_{max}]$ with a step of $\Delta\rho=1$ pixel, so $\rho = 0, 1, 2, 3, 4, \ldots, \rho_{max}$



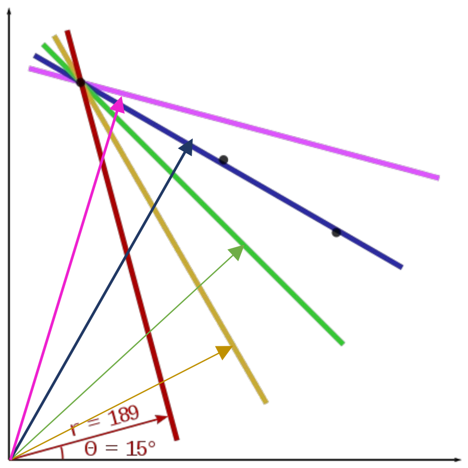
Θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

3. Iterate on every pixel in the image
    - If the pixel is on the edge P(x, y) (==255?)
        - compute all possible lines that cross the point P(x, y)
        - Each line (ρ, θ) will vote in the Hough cell corresponding to that line

	0		...				359
0	0	0	1	0	2	1	0
	0	10	4	6	1	2	0
...	0	50	5	0	1	0	0
ρ_{max}	0	0	2	0	0	0	0

Hough Transform - Algorithm

- $\theta \in [0, 360)$ with a step of $\Delta\theta = 1$ deg, so $\theta = 0, 1, 2, 3, 4 \ldots, 359$
- $\rho \in [0, \rho_{max}]$ with a step of $\Delta\rho=1$ pixel, so $\rho = 0, 1, 2, 3, 4, \ldots, \rho_{max}$



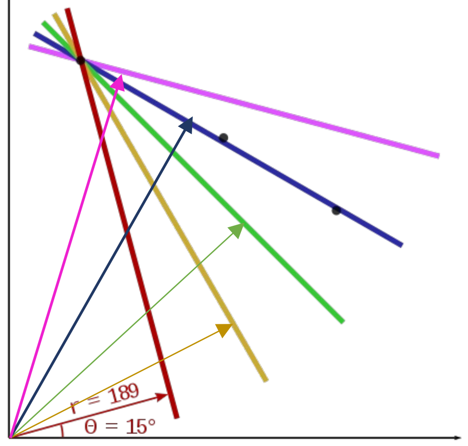
Θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

3. Iterate on every pixel in the image
   - If the pixel is on the edge P(x, y) (==255?)
     - For each θ from a 0 to θ_{max} (with a step of $\Delta\theta$)
       - Θ from degree to radians:
         `θrad = θ * CV_PI/180;`
       - Compute ρ :
         $\rho = x \cos(\theta_{rad}) + y \sin(\theta_{rad})$
       - If $\rho \in [0, \rho_{max}]$ increment the cell in the Hough accumulator : `H[ro][theta] += 1`
4. [Visualize] Display the Hough accumulator

```
plt.imshow(H, cmap='gray’)
plt.axis('off’)
plt.show()
```

Hough Transform - Algorithm

- $\theta \in [0, 360)$ with a step of $\Delta\theta = 1$ deg, so $\theta = 0, 1, 2, 3, 4 \ldots, 359$
- $\rho \in [0, \rho_{max}]$ with a step of $\Delta\rho=1$ pixel, so $\rho = 0, 1, 2, 3, 4, \ldots, \rho_{max}$



Θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

5. Find the local maximum (peaks) in the accumulator (the cells with the highest number of votes in a window)

- **Check if a Hough element is a local maximum in a squared window (n x n) centered on the element.**
- **Store local maxima that have their values larger than a threshold**

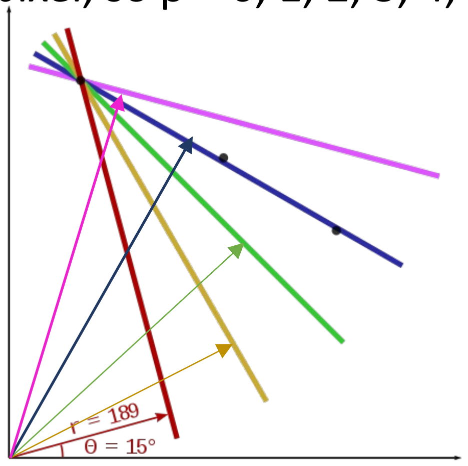
ρ							
0	0	0	1	0	2	1	0
…	0	10	4	6	1	2	0
	0	50	5	0	1	0	0
ρ_{max}	0	0	2	0	0	0	0
	0			…			359

θ

Slide a window of n x n on the accumulator and check if the element in the center of the window is greater than its neighbours in the window and greater than a threshold (for example th = 20)

Hough Transform - Algorithm

- $\theta \in [0, 360)$ with a step of $\Delta\theta = 1$ deg, so $\theta = 0, 1, 2, 3, 4 \ldots., 359$
- $\rho \in [0, \rho_{max}]$ with a step of $\Delta\rho=1$ pixel, so $\rho = 0, 1, 2, 3, 4, \ldots, \rho_{max}$



Θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

5. Find the local maximum (peaks) in the accumulator (the cells with the highest number of votes in a window)

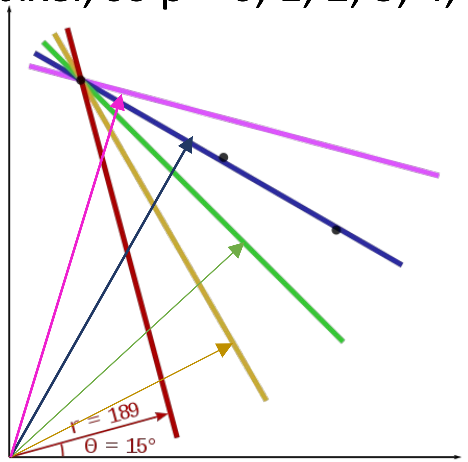
- **Check if a Hough element is a local maximum in a squared window (n x n) centered on the element.**
- **Store local maxima that have their values larger than a threshold**

ρ \ θ	0			…			359
0	0	0	1	0	2	1	0
…	0	10	4	6	1	2	0
	0	50	5	0	1	0	0
ρ_{max}	0	0	2	0	0	0	0

Slide a window of n x n on the accumulator and check if the element in the center of the window is greater than its neighbours in the window and greater than a threshold (for example th = 20)

Hough Transform - Algorithm

- $\theta \in [0, 360)$ with a step of $\Delta\theta = 1$ deg, so $\theta = 0, 1, 2, 3, 4 \ldots, 359$
- $\rho \in [0, \rho_{max}]$ with a step of $\Delta\rho=1$ pixel, so $\rho = 0, 1, 2, 3, 4, \ldots, \rho_{max}$



Θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

5. Find the local maximum (peaks) in the accumulator (the cells with the highest number of votes in a window)

- **Check if a Hough element is a local maximum in a squared window (n x n) centered on the element.**
- **Store local maxima that have their values larger than a threshold**

ρ

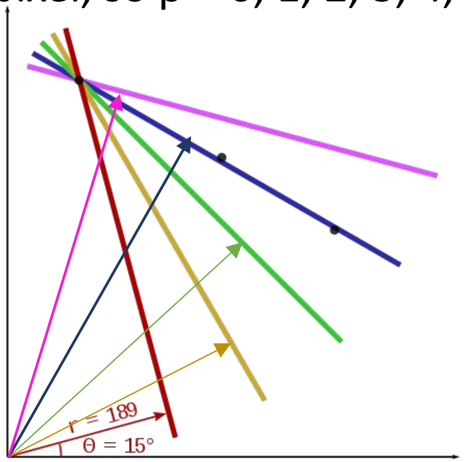
0	0	0	1	0	2	1	0
…	0	10	4	6	1	2	0
	0	50	5	0	1	0	0
ρ_{max}	0	0	2	0	0	0	0
	0			…			359

θ

Slide a window of n x n on the accumulator and check if the element in the center of the window is greater than its neighbours in the window and greater than a threshold (for example th = 20)

Hough Transform - Algorithm

- $\theta \in [0, 360)$ with a step of $\Delta\theta = 1$ deg, so $\theta = 0, 1, 2, 3, 4 \ldots, 359$
- $\rho \in [0, \rho_{max}]$ with a step of $\Delta\rho=1$ pixel, so $\rho = 0, 1, 2, 3, 4, \ldots, \rho_{max}$



Θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

5. Find the local maximum (peaks) in the accumulator (the cells with the highest number of votes in a window)

- **Check if a Hough element is a local maximum in a squared window (n x n) centered on the element.**
- **Store local maxima that have their values larger than a threshold**

ρ

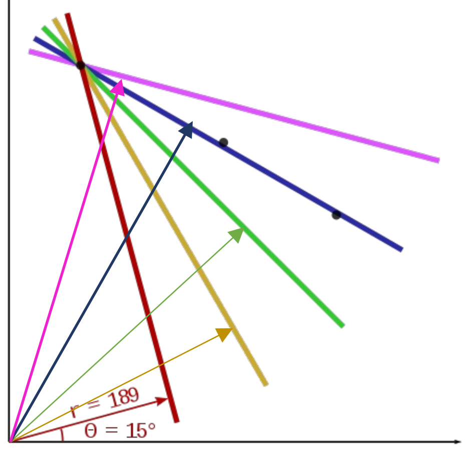
0	0	0	1	0	2	1	0
…	0	10	4	6	1	2	0
	0	50	5	0	1	0	0
ρ_{max}	0	0	2	0	0	0	0
	0			…			359

θ

Slide a window of n x n on the accumulator and check if the element in the center of the window is greater than its neighbours in the window and greater than a threshold (for example th = 20)

Hough Transform - Algorithm

- $\theta \in [0, 360)$ with a step of $\Delta\theta = 1$ deg, so $\theta = 0, 1, 2, 3, 4 \ldots, 359$
- $\rho \in [0, \rho_{max}]$ with a step of $\Delta\rho=1$ pixel, so $\rho = 0, 1, 2, 3, 4, \ldots, \rho_{max}$



Θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

5. Find the local maximum (peaks) in the accumulator (the cells with the highest number of votes in a window)

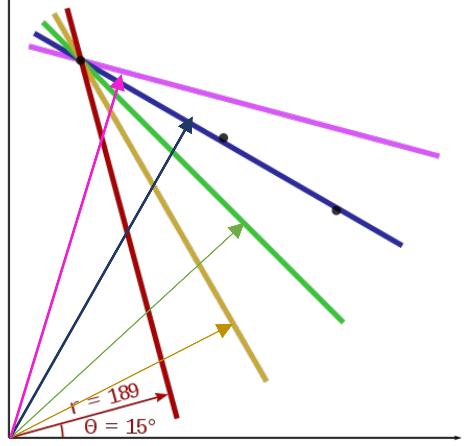
- **Check if a Hough element is a local maximum in a squared window (n x n) centered on the element.**
- **Store local maxima that have their values larger than a threshold**

ρ \ θ	0			…			359
0	0	0	1	0	2	1	0
…	0	10	4	6	1	2	0
	0	50	5	0	1	0	0
ρ_{max}	0	0	2	0	0	0	0

Slide a window of n x n on the accumulator and check if the element in the center of the window is greater than its neighbours in the window and greater than a threshold (for example th = 20)

Hough Transform - Algorithm

- $\theta \in [0, 360)$ with a step of $\Delta\theta = 1$ deg, so $\theta = 0, 1, 2, 3, 4 \ldots, 359$
- $\rho \in [0, \rho_{max}]$ with a step of $\Delta\rho=1$ pixel, so $\rho = 0, 1, 2, 3, 4, \ldots, \rho_{max}$



Θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

5. Find the local maximum (peaks) in the accumulator (the cells with the highest number of votes in a window)

- **Check if a Hough element is a local maximum in a squared window (n x n) centered on the element.**
- **Store local maxima that have their values larger than a threshold**

ρ

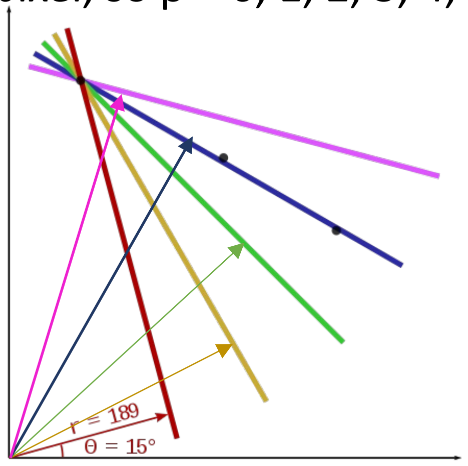
0	0	0	1	0	2	1	0
	0	10	4	6	1	2	0
…	0	50	5	0	1	0	0
ρ_{max}	0	0	2	0	0	0	0
	0			…			359

θ

Slide a window of n x n on the accumulator and check if the element in the center of the window is greater than its neighbours in the window and greater than a threshold (for example th = 20)

Hough Transform - Algorithm

- $\theta \in [0, 360)$ with a step of $\Delta\theta = 1$ deg, so $\theta = 0, 1, 2, 3, 4 \dots, 359$
- $\rho \in [0, \rho_{max}]$ with a step of $\Delta\rho=1$ pixel, so $\rho = 0, 1, 2, 3, 4, \dots, \rho_{max}$



Θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

5. Find the local maximum (peaks) in the accumulator (the cells with the highest number of votes in a window)

- **Check if a Hough element is a local maximum in a squared window (n x n) centered on the element.**
- **Store local maxima that have their values larger than a threshold**

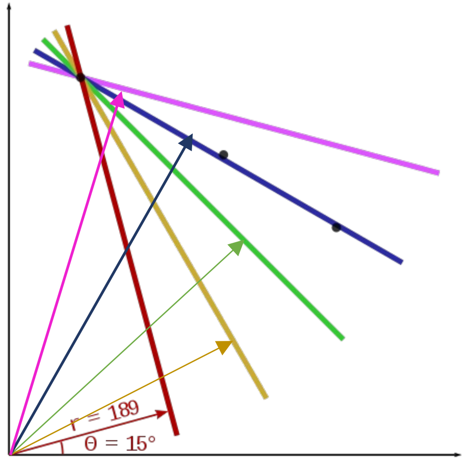
ρ (rows), θ (columns)

ρ \ θ	0		...				359
0	0	0	1	0	2	1	0
...	0	10	4	6	1	2	0
	0	50	5	0	1	0	0
ρ_{max}	0	0	2	0	0	0	0

Slide a window of n x n on the accumulator and check if the element in the center of the window is greater than its neighbours in the window and greater than a threshold (for example th = 20)

Hough Transform - Algorithm

- $\theta \in [0, 360)$ with a step of $\Delta\theta = 1$ deg, so $\theta = 0, 1, 2, 3, 4 \ldots, 359$
- $\rho \in [0, \rho_{max}]$ with a step of $\Delta\rho=1$ pixel, so $\rho = 0, 1, 2, 3, 4, \ldots, \rho_{max}$



Θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

5. Find the local maximum (peaks) in the accumulator (the cells with the highest number of votes in a window)

- Check if a Hough element is a local maximum in a squared window (n x n) centered on the element. (try different values eg. 3x3, 7x7, 11x11)
- Store local maxima that have their values larger than a threshold (eg. 20)
- Sort the stored values and keep the top k values (eg. k = 10)
- Draw the lines corresponding to the top k lines that you detected:
    - Each detected line has ρ-θ
    - $\rho = x \cos(\theta) + y \sin(\theta)$