Imperial College London

Hough Transform

Dr Wenjia Bai

Department of Computing & Brain Sciences

Hough transform

- After edge detection, we get a binary edge map, which contains a lot of edge points.
- If these edge points belong to a line, how can we get a parametric representation of the line?

Line parameterisation

- For example, a line can be represented by $y = mx + b$, i.e. just two parameters m and b.
- This is a much more efficient representation than a lot of edge points.

Line parameterisation

- Slope intercept form

$$y = mx + b$$

- Double intercept form

$$\frac{x}{a} + \frac{y}{b} = 1$$

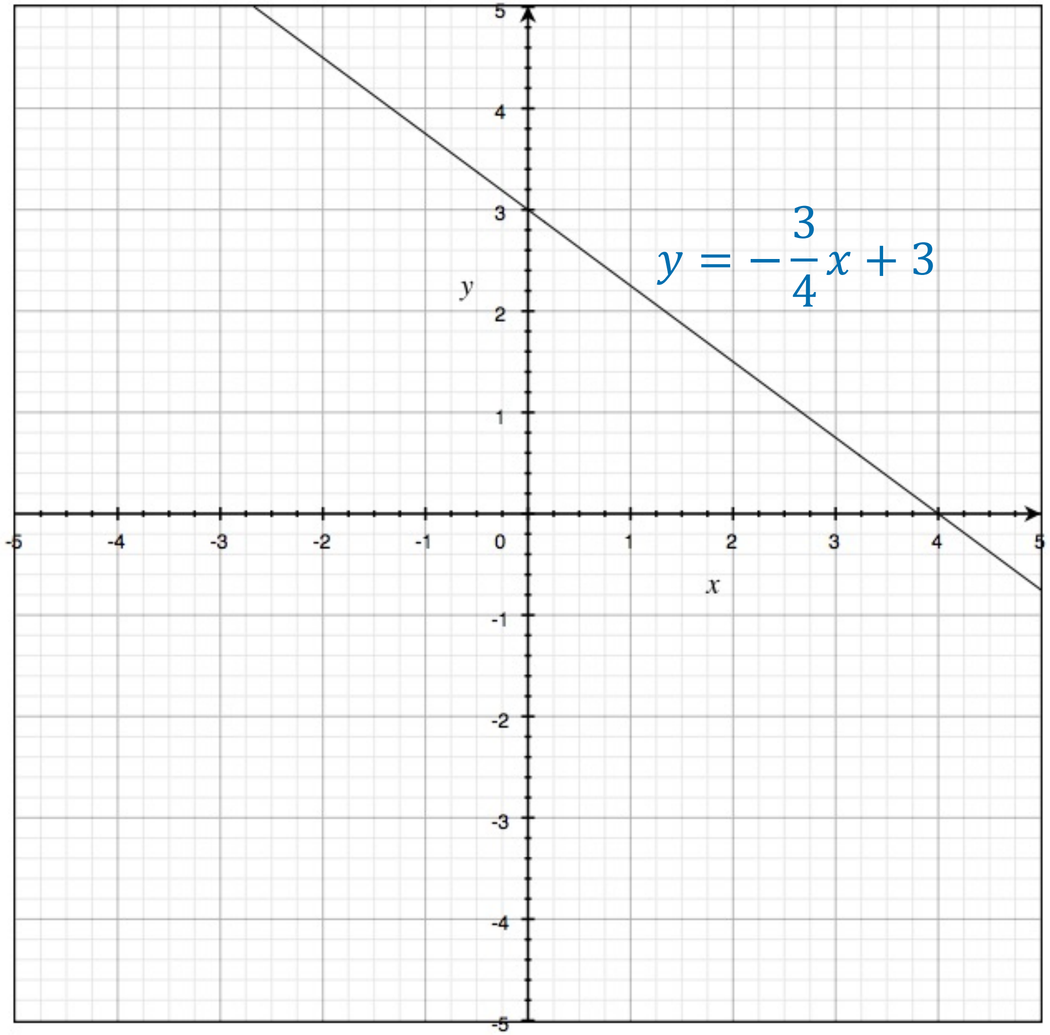
- Normal form

$$xcos(\theta) + ysin(\theta) = \rho$$

Slope intercept form

$$y = mx + b$$

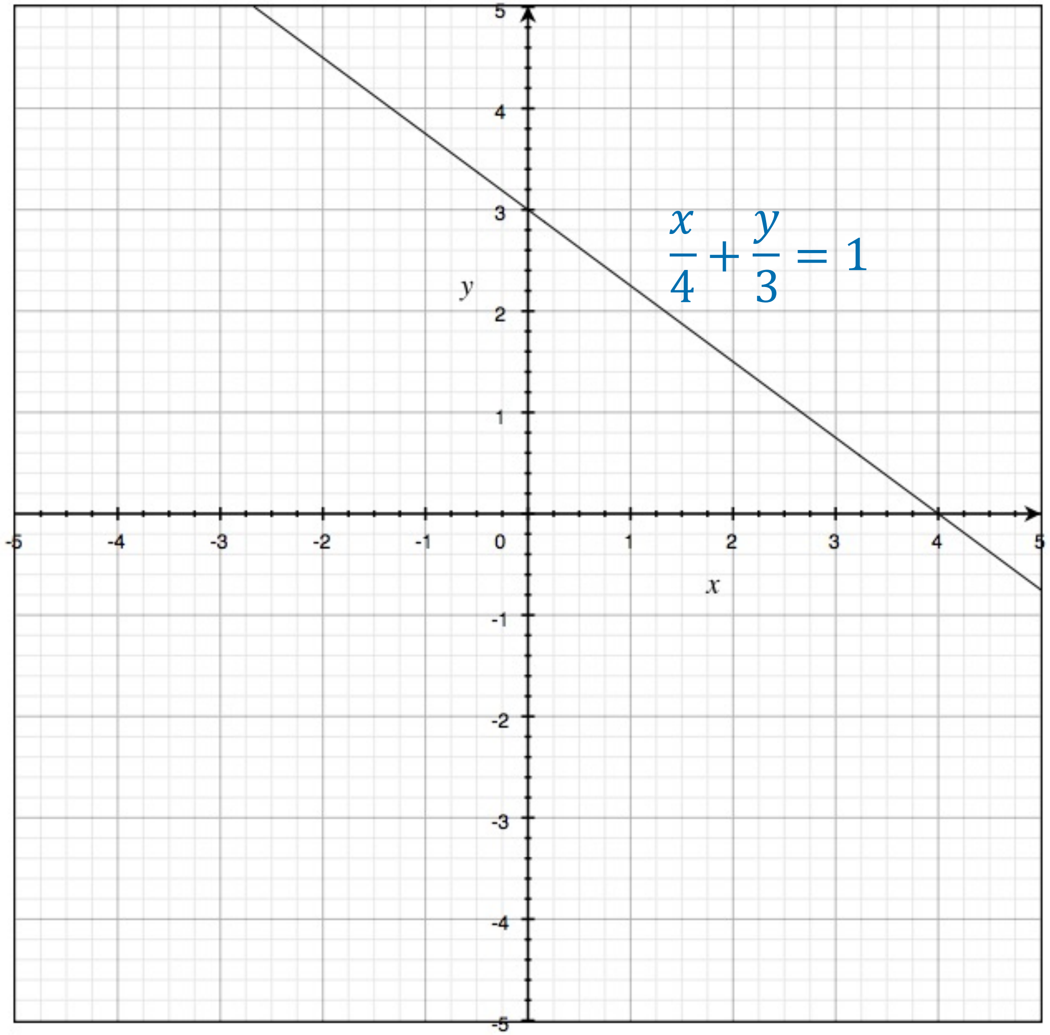
- m: slope
- b: y-intercept



Double intercept form

$$\frac{x}{a} + \frac{y}{b} = 1$$

- a: x-intercept
- b: y-intercept



Normal form

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

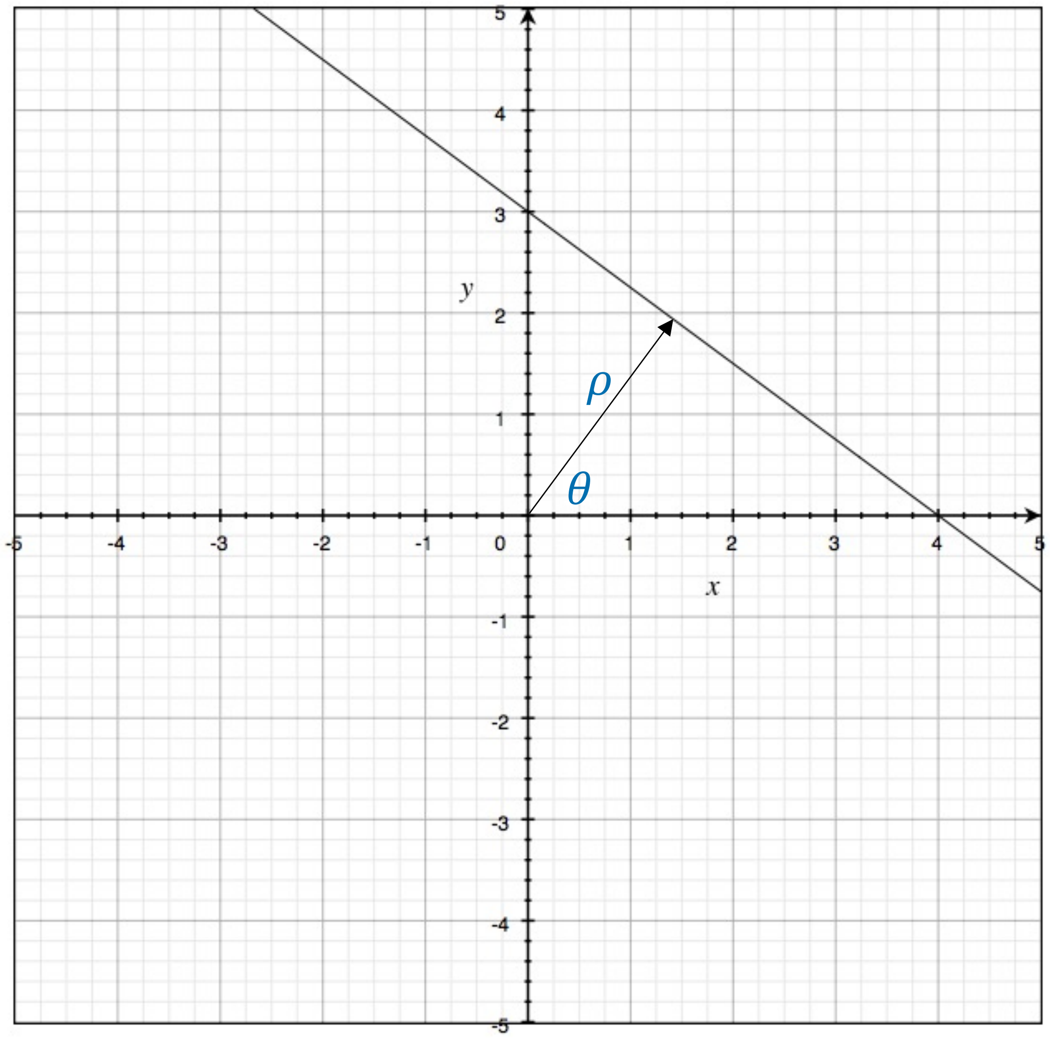
- θ: angle
- ρ: distance from origin

- Derivation

x-intercept $a = \frac{\rho}{\cos(\theta)}$, y-intercept $b = \frac{\rho}{\sin(\theta)}$

Plug into $\frac{x}{a} + \frac{y}{b} = 1$

We can get the normal form.



Hough transform

- Hough transform is a transform from image space to parameter space (e.g. from an edge map to the two parameters of a line).
- Its output is a parametric model, given the input edge points.
- The basic idea is that each edge point votes for possible models in the parameter space.

Model fitting

- Some of you may have a different idea here, especially if you know optimisation and model fitting.
- One way to solve this problem is to fit a line model onto the edge points.
  - Suppose we have a set of points $(x_1, y_1), (x_2, y_2), \cdots$ and we would like to fit a line model $y = mx + b$ to these points.
  - (m, b) can be estimated by minimising the fitting error

$$\min_{m,b} \sum_i [\underbrace{y_i}_{\text{real } y \text{ of a point}} - (\underbrace{mx_i + b}_{\hat{y} \text{ estimated by our line model}})]^2$$

- How will Hough transform solve the problem differently?

Hough transform

- Let us use the slope intercept form for a line model

$$y = mx + b$$
$$\downarrow$$
$$b = y - mx$$

- We have edge points in the image space $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ …
- Each point votes for a line model in the parameter space.
- For example, the first point will vote for $b = y_1 - mx_1$.

Hough transform

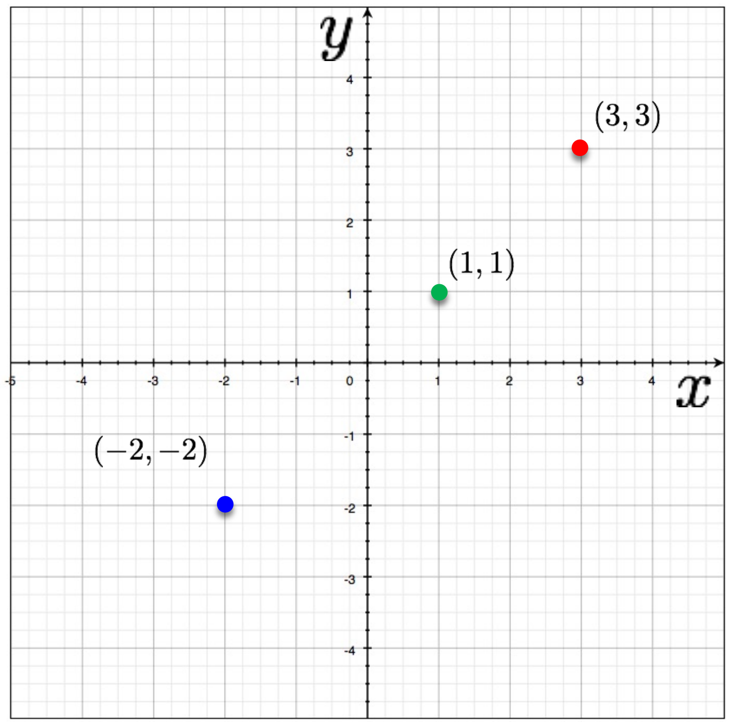
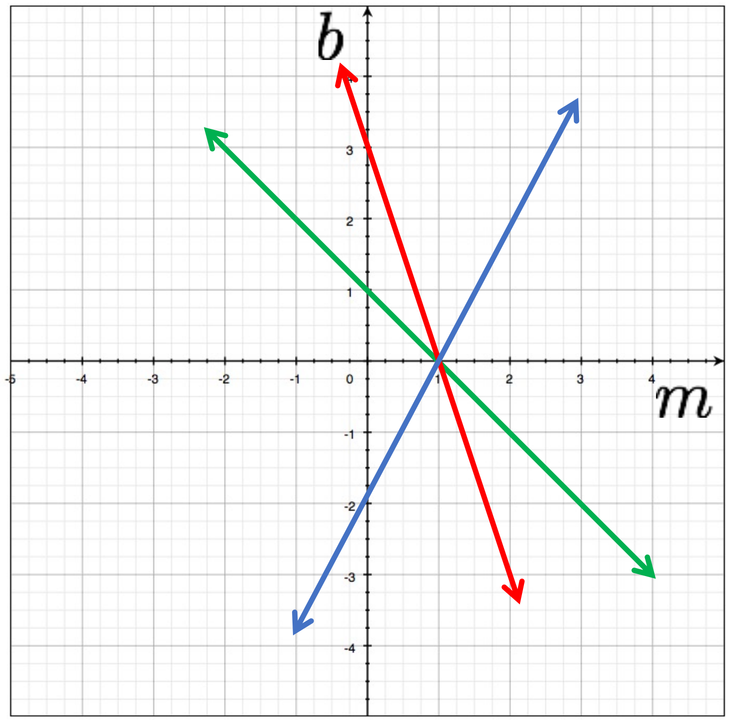


Image space

$$b = y - mx$$



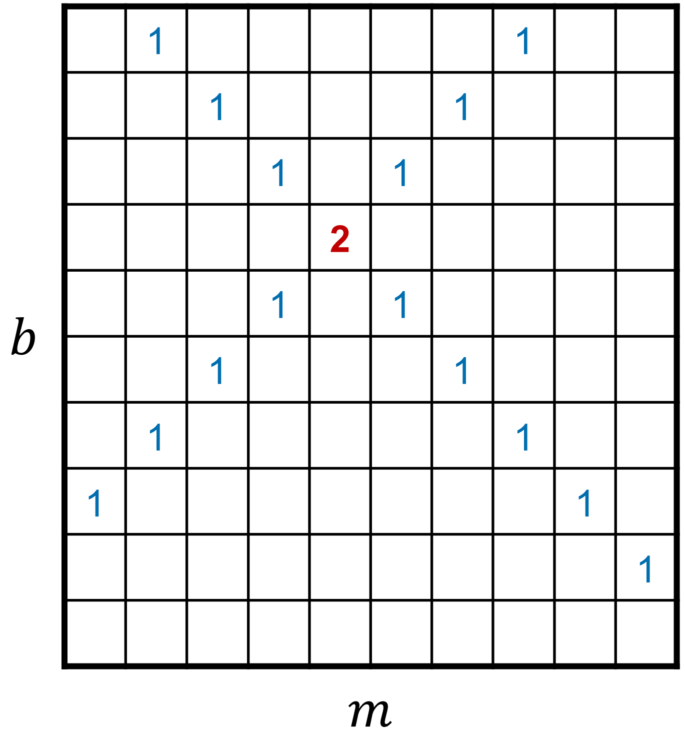
Parameter space

Vote result:
$m = 1, b = 0$
$y = x$

Slide credit: Ioannis Gkioulekas

Hough transform

- In practice, the parameter space is divided into 2D bins.
- Each point increments the vote by 1 in one of the bins.
- One problem with the slope intercept form:
  - The parameter space is too large.
  - $m \in [-\infty, +\infty],\ b \in [-\infty, +\infty]$
  - We need a lot of bins.



Hough transform

- Solution
  - Use the normal form instead
    $$x\cos(\theta) + y\sin(\theta) = \rho$$
  - Although $\rho \in [-\infty, +\infty]$, at least $\theta \in [0, \pi)$.
  - We can use much fewer bins.
  - By the way, in practice ρ is not infinite either. It is limited by the image size.
- The transform from image space to parameter space will look different.

Hough transform

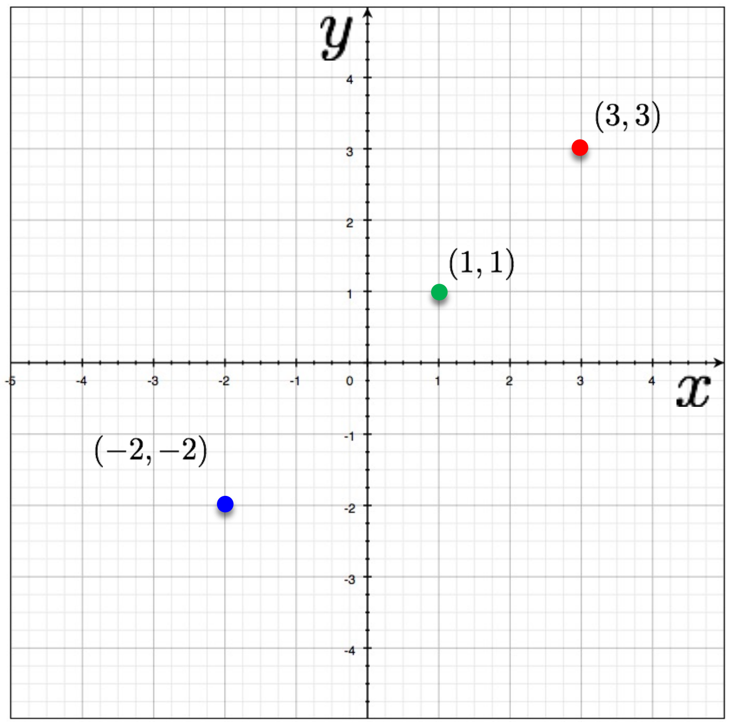
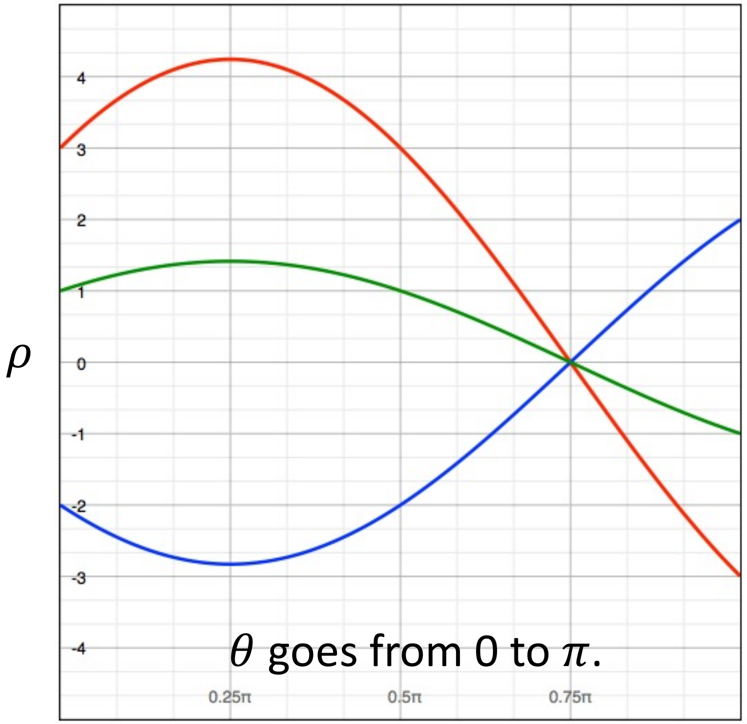


Image space

$xcos(\theta) + ysin(\theta) = \rho$



Parameter space

Vote result:
$\theta = \frac{3}{4}\pi, \rho = 0$

Line detection by Hough transform

Algorithm

Initialise the bins $H(\rho, \theta)$ to all zeros.

For each edge point (x, y)

    For θ from 0 to π

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

        Accumulate $H(\rho, \theta) = H(\rho, \theta) + 1$

Find (ρ, θ) where $H(\rho, \theta)$ is a local maximum and larger than a threshold.

The detected lines are given by $\rho = x\cos\theta + y\sin\theta$.

Line detection by Hough transform

Algorithm

Initialise the bins $H(\rho, \theta)$ to all zeros.

For each edge point (x, y)

For θ from 0 to π

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

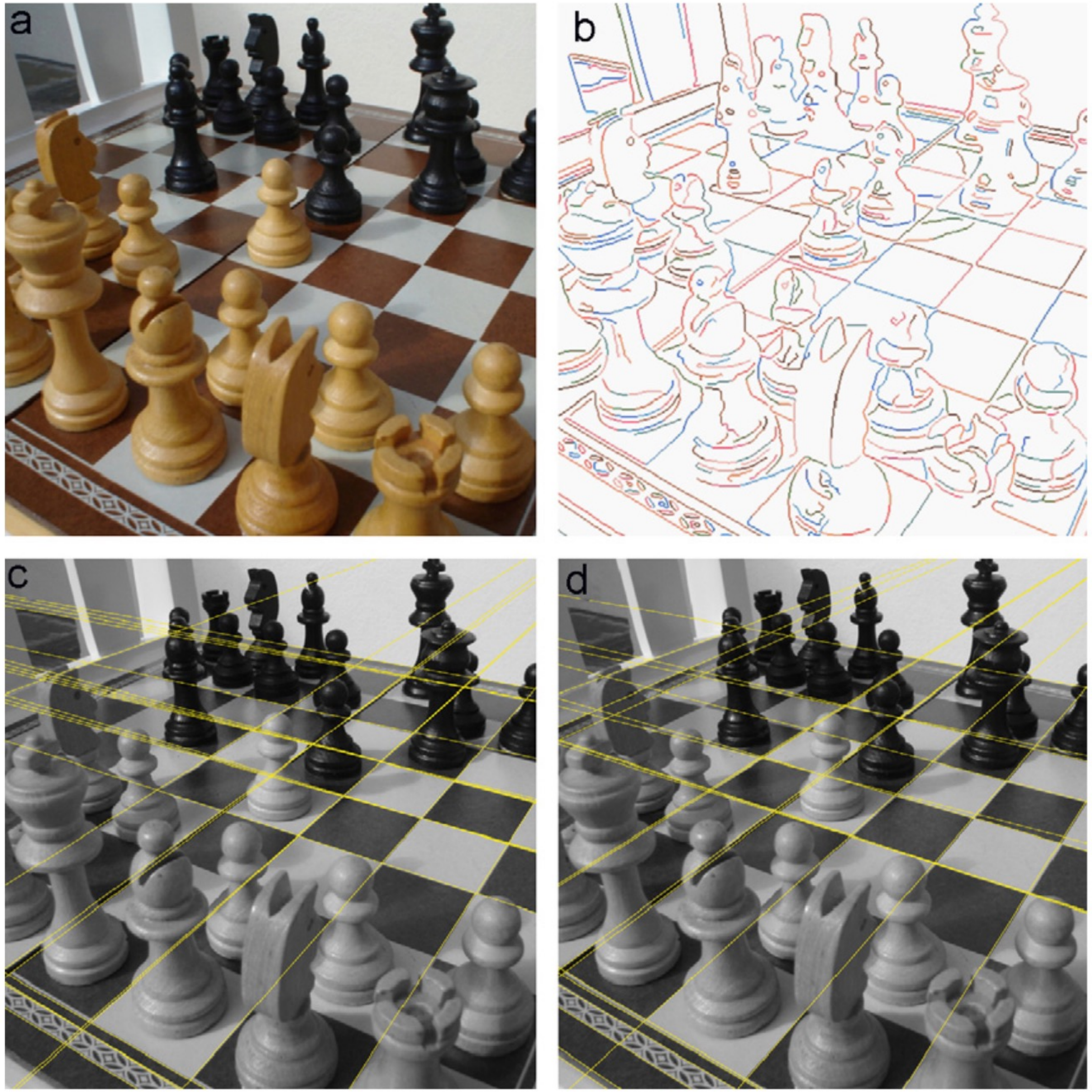
Accumulate $H(\rho, \theta) = H(\rho, \theta) + 1$

Find (ρ, θ) where $H(\rho, \theta)$ is a local maximum and larger than a threshold.

The detected lines are given by $\rho = x\cos\theta + y\sin\theta$.

- Why local maximum?
  - Similar as non-maximum suppression in edge detection.
- Why thresholding?
  - A few random points would not lead to a line being detected.

Input image



Canny edge detection

Lines detected by Hough transform

L.A.F. Fernandes et al. Pattern Recognition, 2008.

Hough transform

- In model fitting, (m, b) are estimated by minimising the fitting error

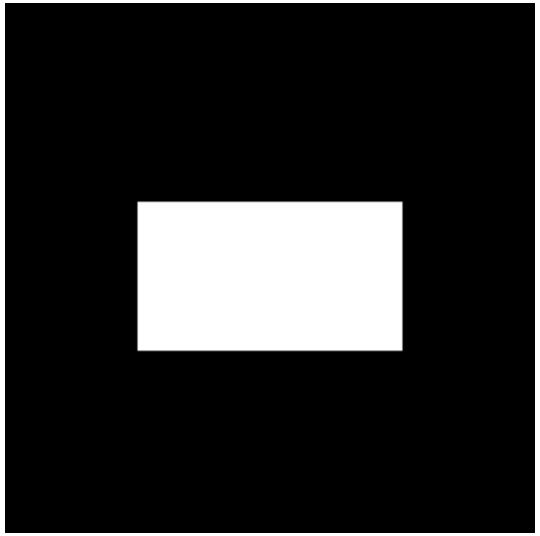
$$\min_{m,b} \sum_i [y_i - (mx_i + b)]^2$$

- Only one line will be detected.

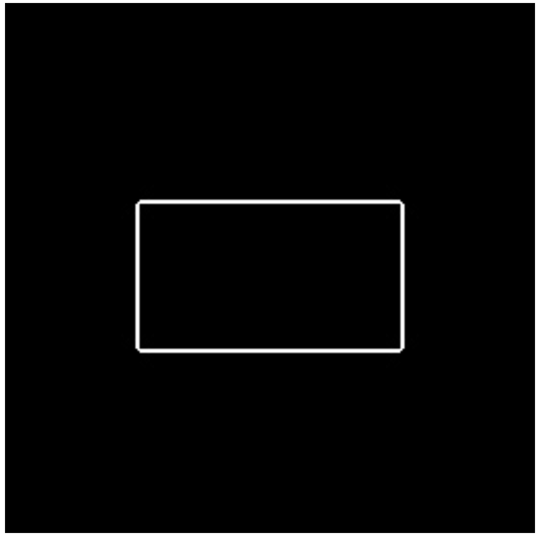
- On the contrary, Hough transform can simultaneously detect multiple lines, as long as they are local maxima above a threshold.

Hough transform

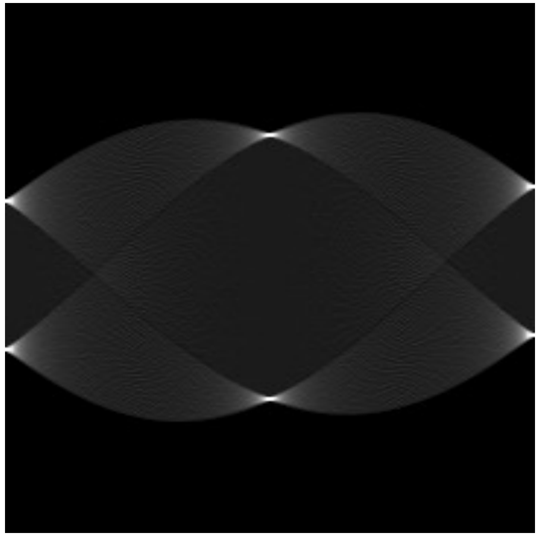
- It can detect multiple lines simultaneously.



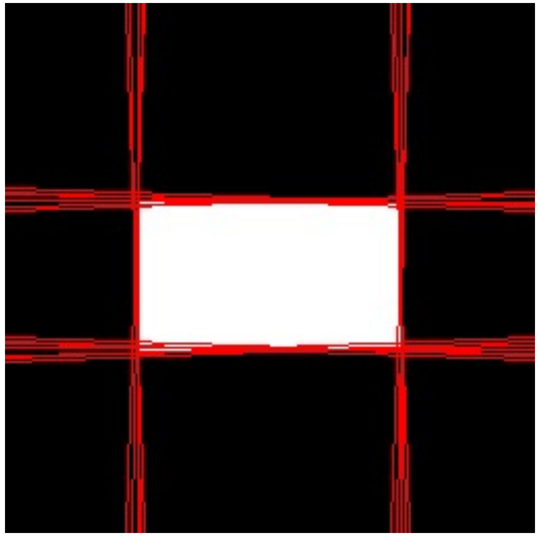
Input image



Edge map



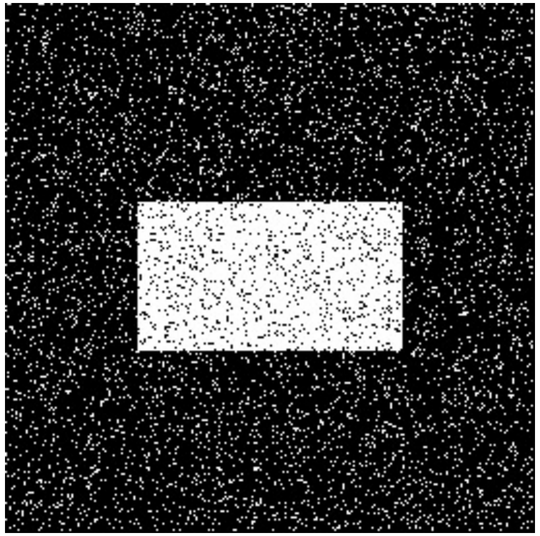
Parameter space



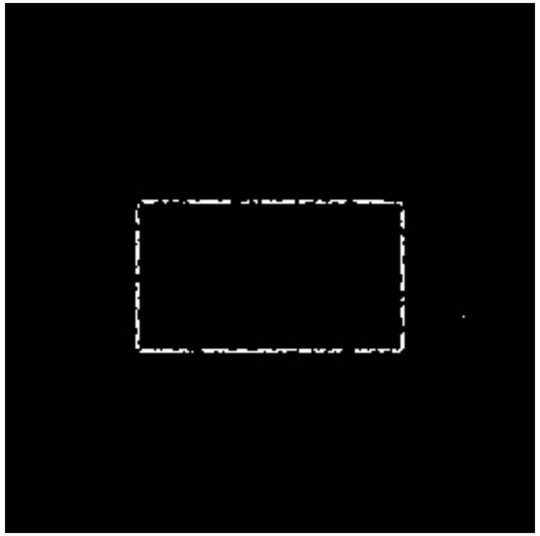
Detected lines

Hough transform

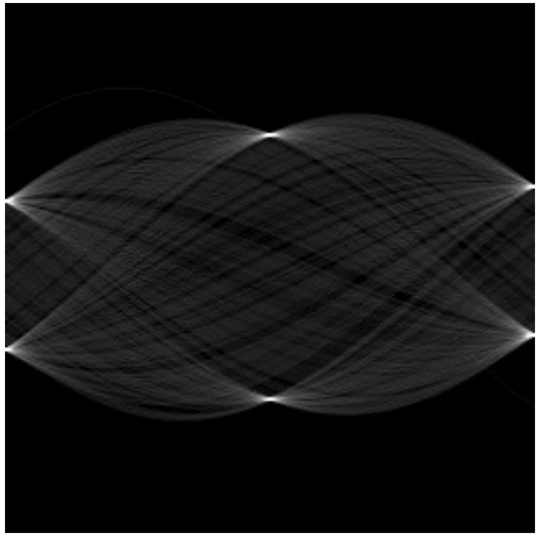
- It is robust to noise.
  - Edge map is often generated after image smoothing.
  - Broken edge points can still vote and contribute to line detection.



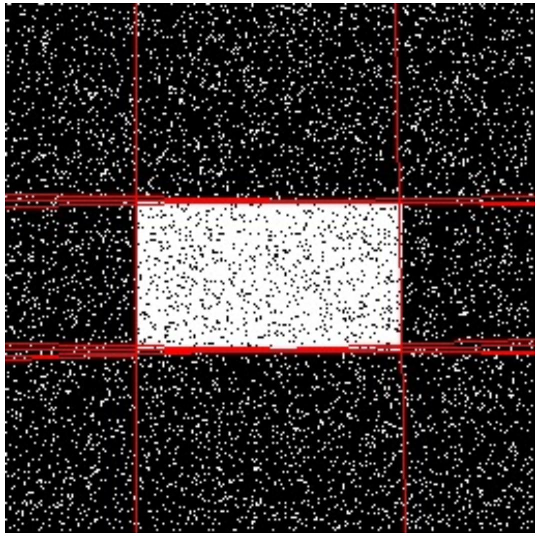
Input image



Edge map



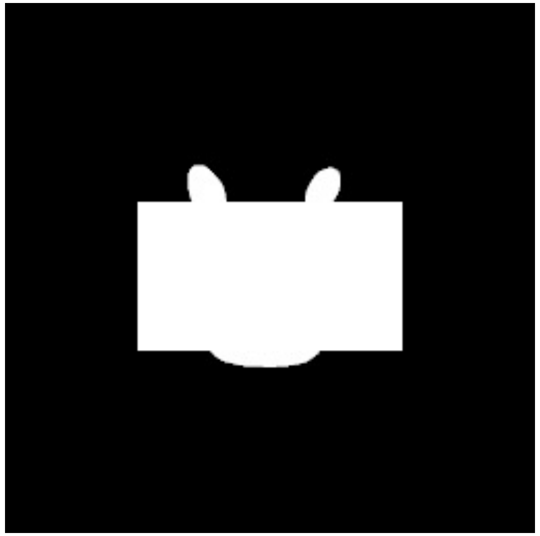
Parameter space

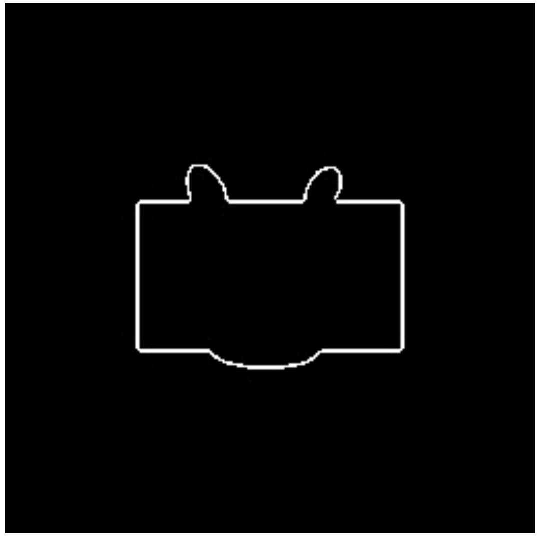


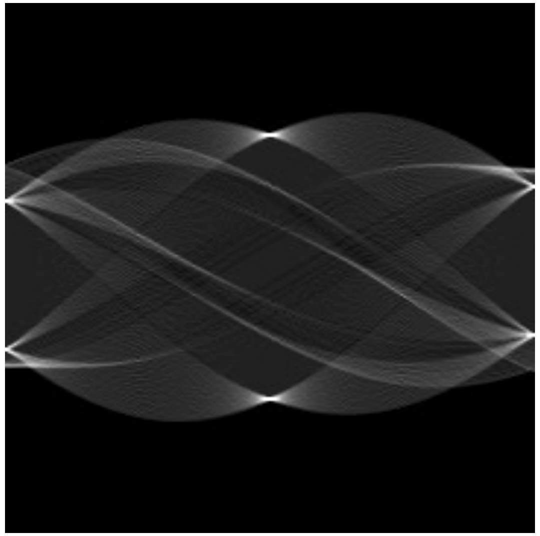
Detected lines

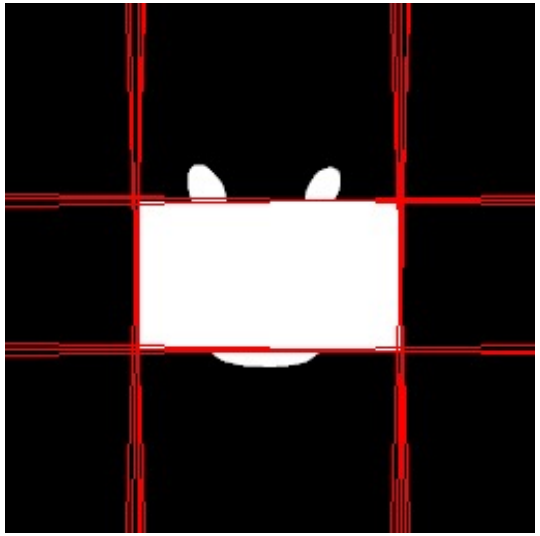
Hough transform

- It is robust to object occlusion (e.g. the rectangle covered by a bunny).
  - The remaining edge points vote and contribute to line detection.


Input image


Edge map


Parameter space


Detected lines

Hough transform

- It is used not just for detecting lines, but also for other shapes, such as circles.
- We can parameterise a circle as,

$$(x-a)^2 + (y-b)^2 = r^2$$

  - The image space (x, y) is transformed to the parameter space (a, b, r).
  - This is a very large search space (a lot of bins).
  - However, if we know the radius r, it would be easier.

H.K. Yuen et al. Comparative study of Hough transform methods for circle finding. Image Vision Computing, 1990.

Circle detection

$$(x-a)^2+(y-b)^2=r^2$$

- If the radius r is known, then for each edge point (x, y), we only need to vote for possible values of (a, b).
- It is still a circle in the parameter space $H(a, b)$.

$$(a-x)^2+(b-y)^2=r^2$$

Circle detection

$(a - x)^2 + (b - y)^2 = 1$, assume $r = 1$

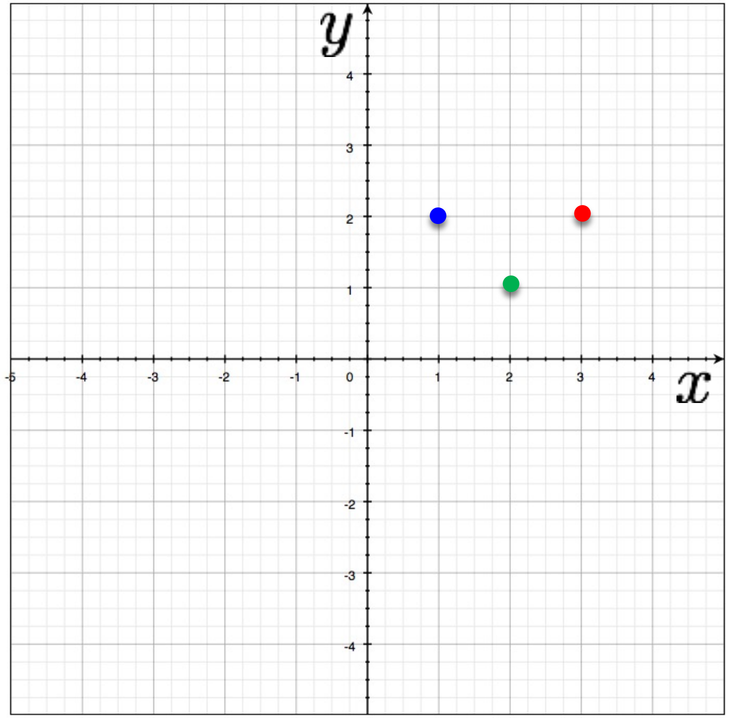
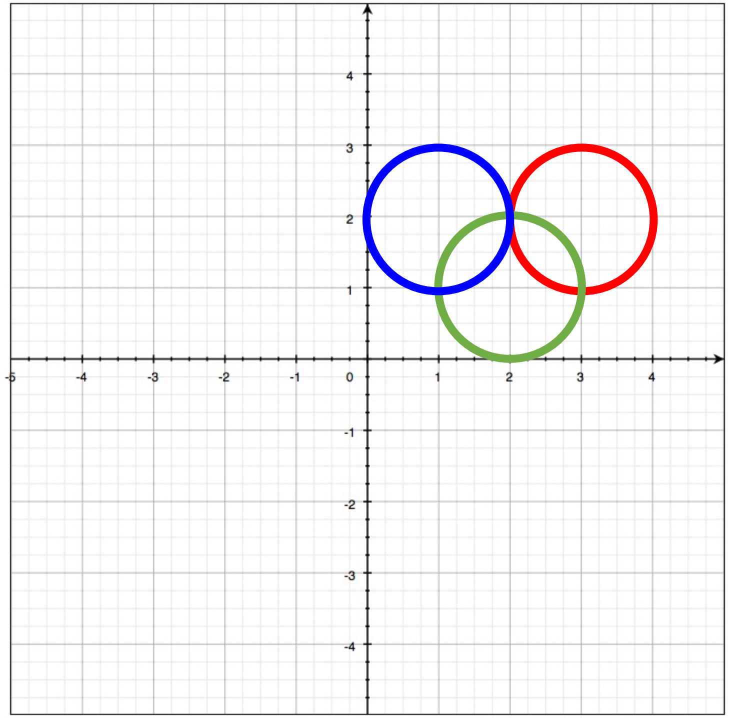


Image space



Parameter space

Vote result:
$a = 2, b = 2$

Circle detection

$$(x-a)^2 + (y-b)^2 = r^2$$

- If the radius r is unknown, then it is a 3D parameter space $H(a, b, r)$.
- We set a range for the radius r.

  For each $r \in [r_{min}, r_{max}]$

  For each edge point (x, y)

  We vote for possible values of (a, b) and accumulate $H(a, b, r)$.
- For example, we can start from r = 1 pixel to 10 pixels, each time increasing by 1 pixel.

Circle equations

- Standard form

$$(x-a)^2 + (y-b)^2 = r^2$$

- Parametric form using trigonometric functions

$$x = a + r \cdot \cos\theta$$
$$y = b + r \cdot \sin\theta$$

  - This form gives us some ideas for acceleration.
  - If we know the angle θ (direction) from the edge point (x, y) to the circle centre (a, b), we can more accurately vote in the parameter space.

Circle detection

$$x = a + r \cdot \cos\theta$$
$$y = b + r \cdot \sin\theta$$

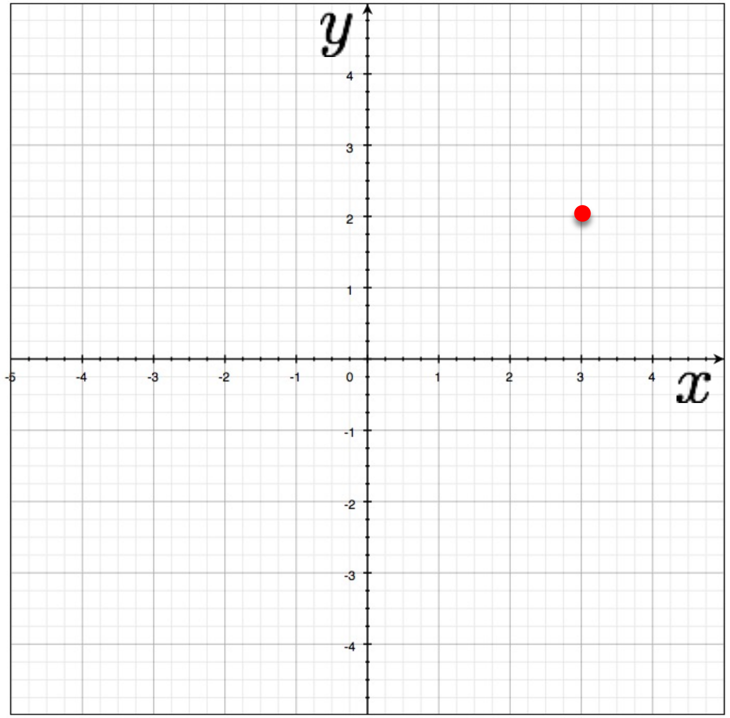
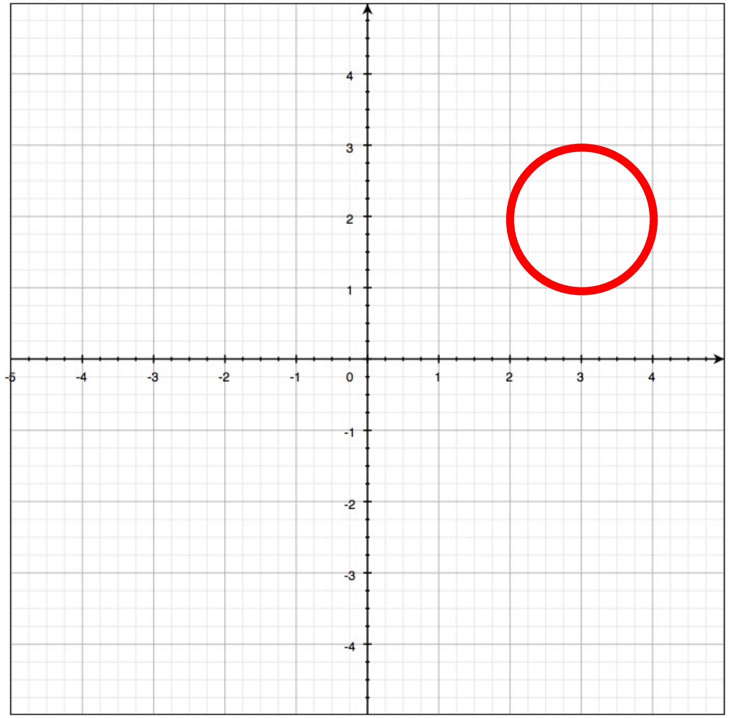


Image space



Parameter space

If we do not know θ, we vote to a whole circle.

Circle detection

$$x = a + r \cdot \cos\theta$$
$$y = b + r \cdot \sin\theta$$

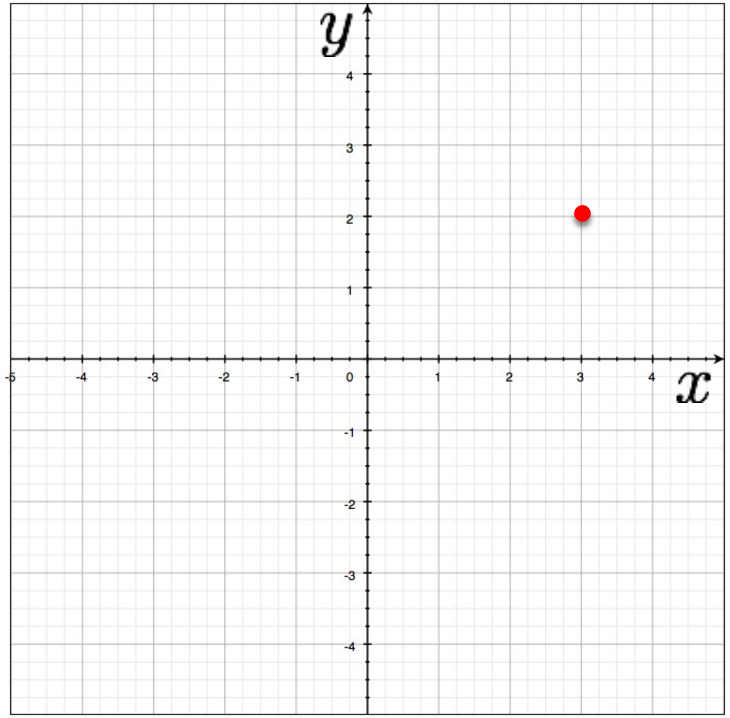
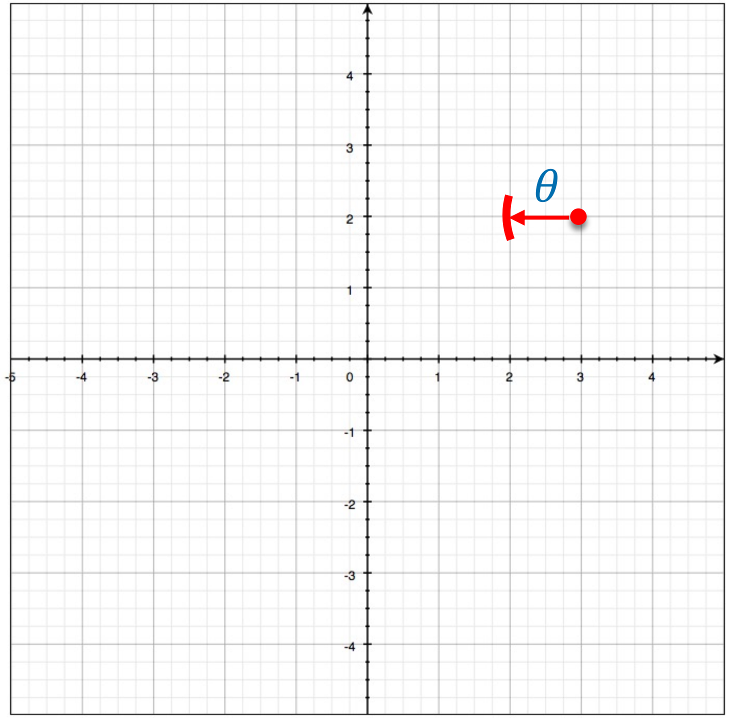


Image space

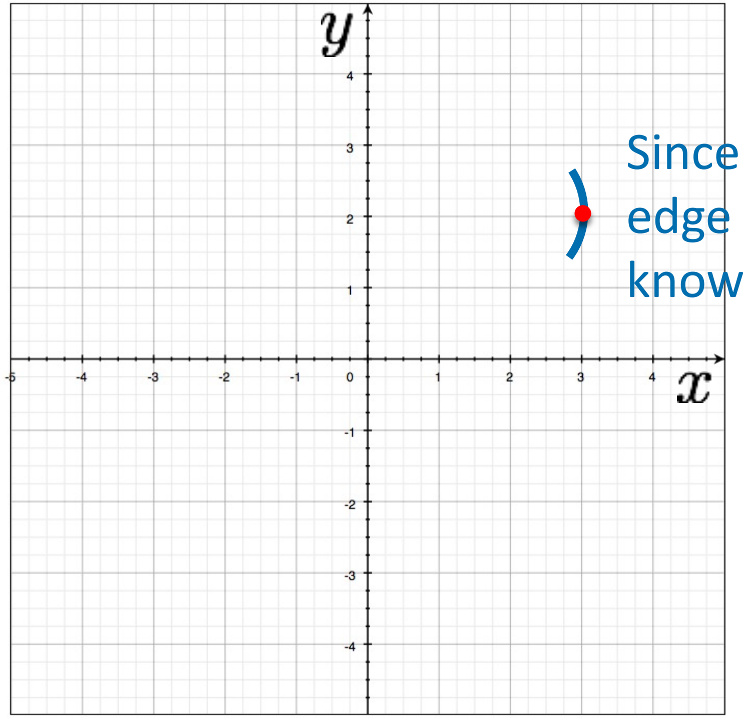


Parameter space

If we know θ, we will only vote along this direction.

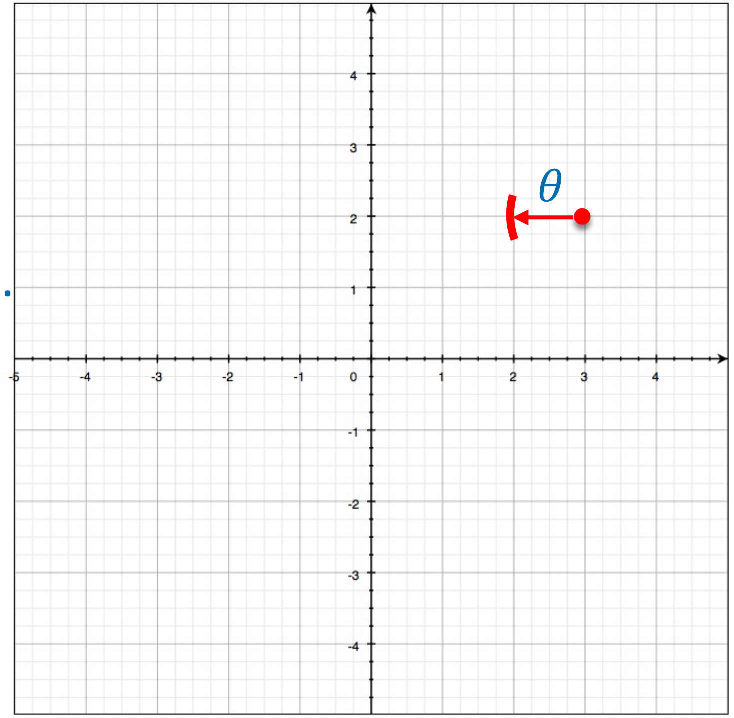
Circle detection

$$x = a + r \cdot \cos\theta$$
$$y = b + r \cdot \sin\theta$$



Since this is an edge point, we know its direction.

Image space



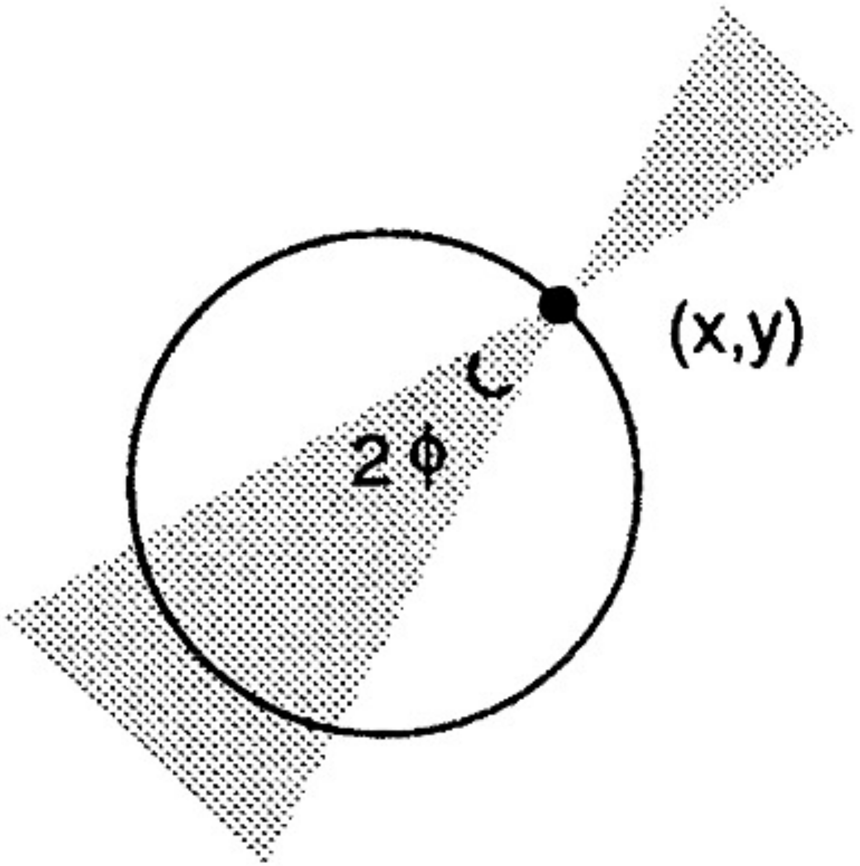
Parameter space

If we know θ, we will only vote along this direction.

Circle detection

- Parametric form using trigonometric functions
$$x = a + r \cdot \cos\theta$$
$$y = b + r \cdot \sin\theta$$
- The edge point (x, y) comes from an edge detection algorithm, so of course we know its direction θ.
- This narrows down our voting area in the parameter space $H(a, b, r)$. We simply move long θ (or opposite θ) for a distance of r.

Circle detection



We can assume that the edge direction θ is measured to accuracy of $\pm\phi$ and vote within this shaded area.

H.K. Yuen et al. Comparative study of Hough transform methods for circle finding. Image Vision Computing, 1990.

Circle detection by Hough transform

Algorithm

Initialise the bins $H(a, b, r)$ to all zeros.

For each possible radius $r \in [r_{min}, r_{max}]$

    For each edge point (x, y)

        Let θ to be gradient direction, or opposite gradient direction

        Calculate $a = x - r \cdot \cos\theta$, $b = y - r \cdot \sin\theta$

        Accumulate $H(a, b, r) = H(a, b, r) + 1$

Find (a, b, r) where $H(a, b, r)$ is a local maximum and larger than a threshold.

The detected circles are given by $x = a + r \cdot \cos\theta$, $y = b + r \cdot \sin\theta$.

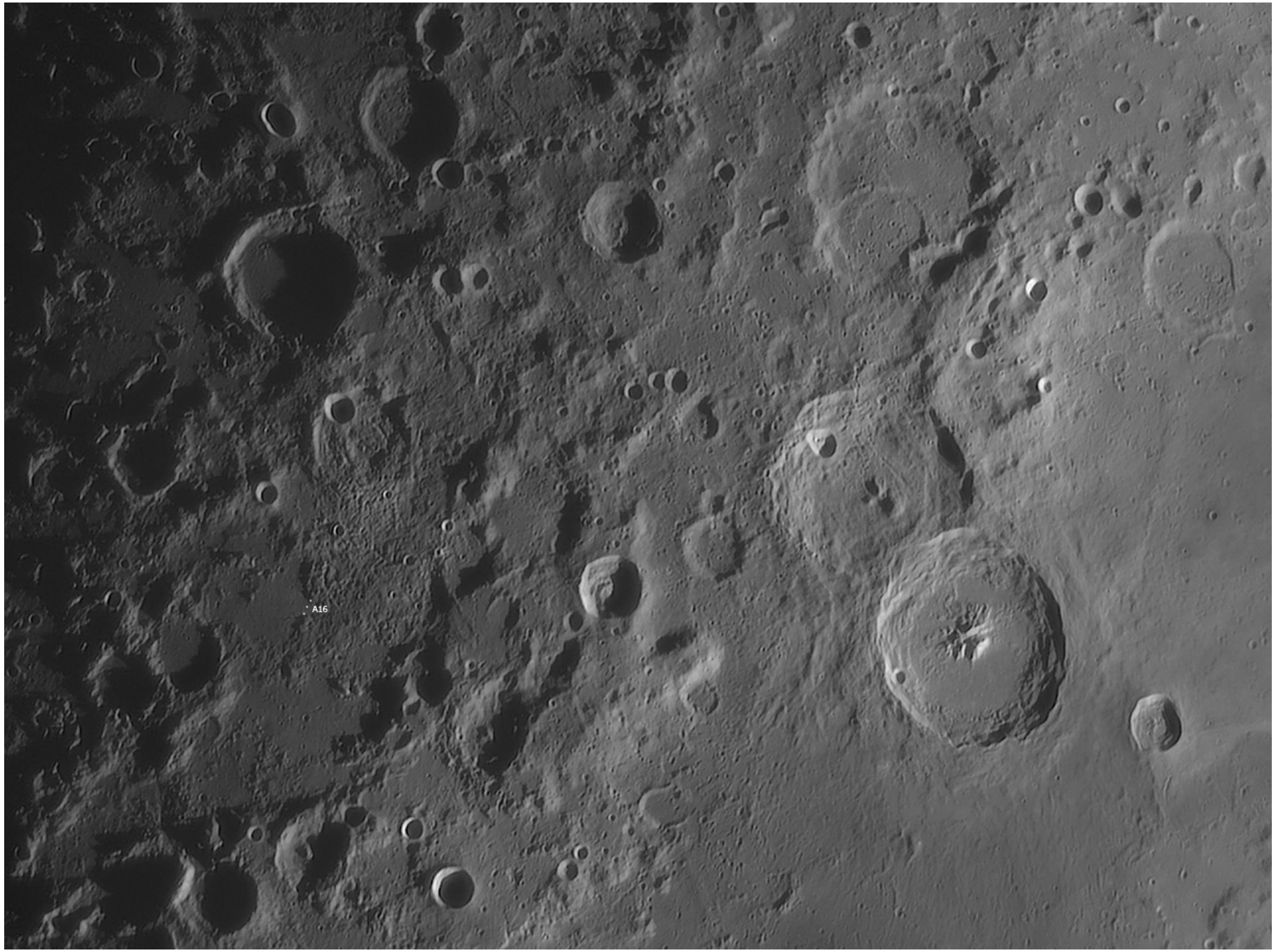
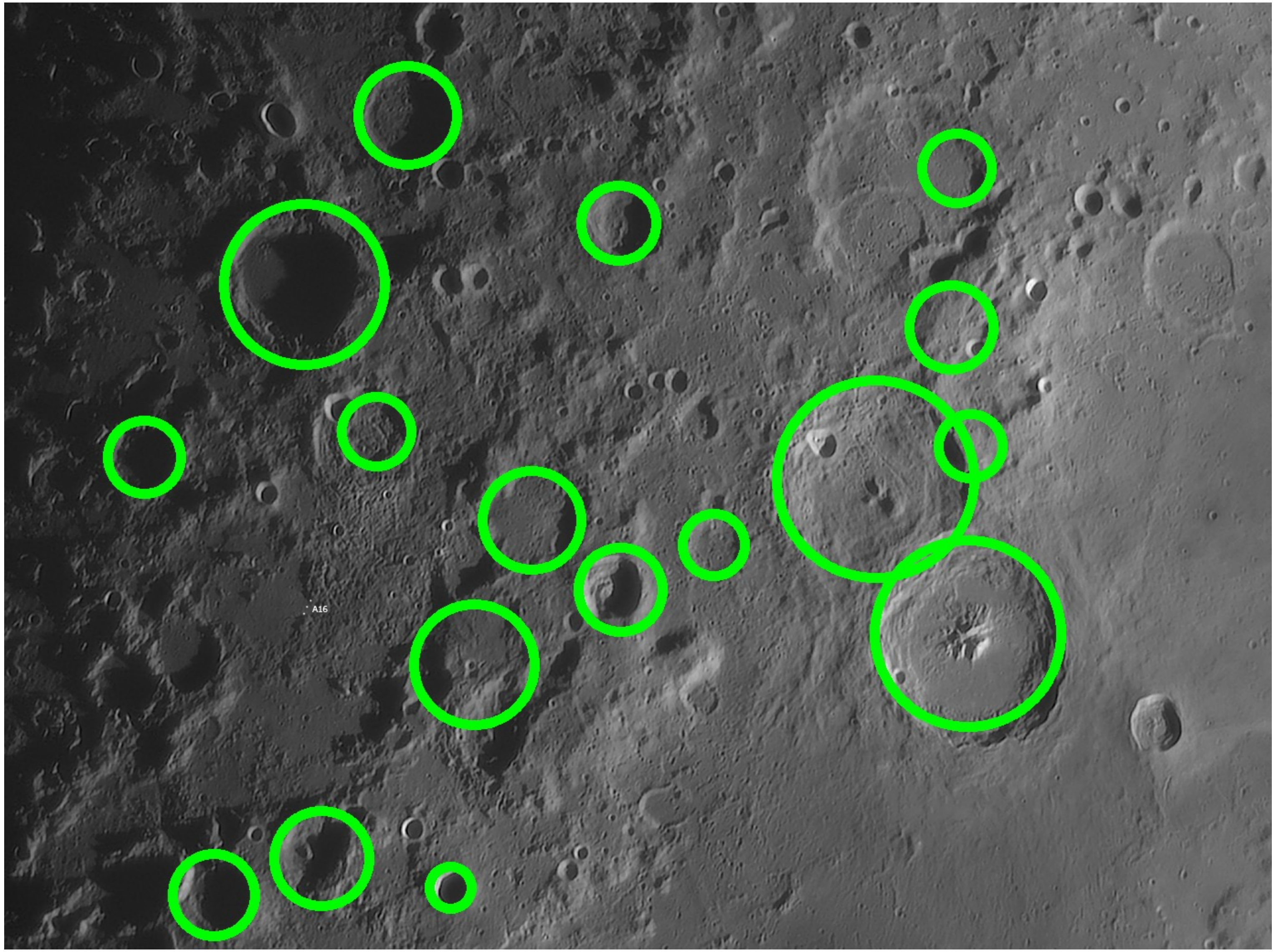


Image of the moon surface, where Apollo 16 landed.



Circle detection by Hough transform

Hough transform

- Advantages
  - It detects multiple instances.
  - Robust to image noise.
  - Robust to occlusion.
- Limitations
  - The computational complexity can be high. For each edge point, we need to vote to a 2D or even 3D parameter space.
  - We need to carefully set some parameters, such as the parameters for the edge detector, the threshold for the accumulator or the range of circle radius.

Hough transform

- Apart from lines and circles, we can also use Hough transform for detecting other shapes.
  - Ellipses

$$\frac{(x-c)^2}{a^2} + \frac{(y-d)^2}{b^2} = 1$$

  - Planes in a 3D space

$$\frac{x}{a} + \frac{y}{b} + \frac{z}{c} = 1$$

  - Other shapes that can be analytically represented

Hough transform

- When we vote in the parameter space, we can add some weights.
- Instead of using equal vote for each edge point, the vote can be weighted by the gradient magnitude so stronger edge points get higher weights.

Generalise the Hough transform idea

- In Hough transform, our aim is to detect some shapes or objects.
- There are two spaces, the image space and the parameter space (Hough space).
  - If the shape can be described by some parameters using an analytical equation, we use this equation to perform voting to the parameter space.
  - If it is not a simple shape without an analytical equation, as long as we have a model to describe it, we can still vote.

Hough forests for pedestrian detection

- The vote is performed by a machine learning model (random forest).
- The model predicts a displacement vector from the patch centre, given the image feature of the patch.

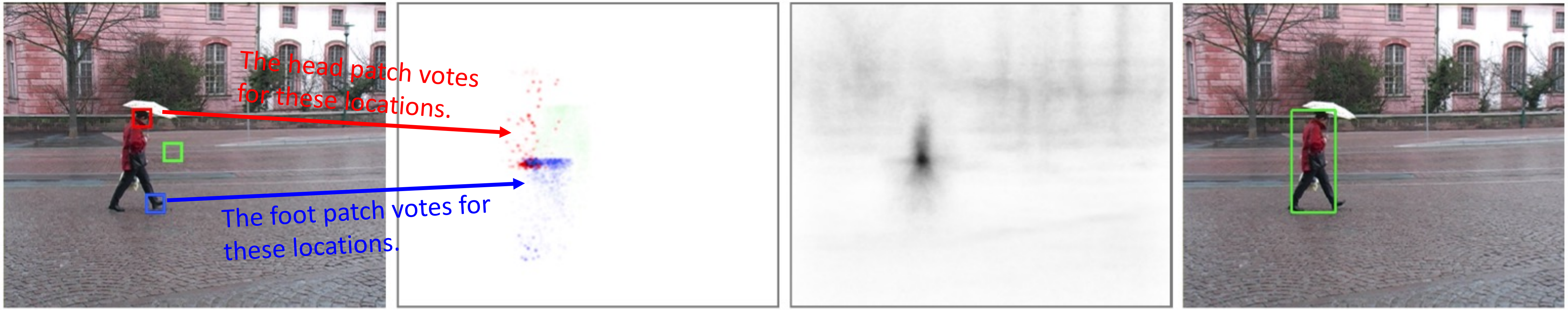


Image and patches | Votes by patches | Hough space of pedestrian location | Detection

J Ball et al. Hough forests for object detection, tracking, and action recognition. TPAMI, 2011.

Hough transform

- Hough transform is an image analysis technique to detect shapes by a voting procedure.
  - Line detection
  - Circle detection
  - General shapes or patterns

References

- Sec. 4.3.2 Hough transforms. Richard Szeliski, Computer Vision: Algorithms and Applications (http://szeliski.org/Book).