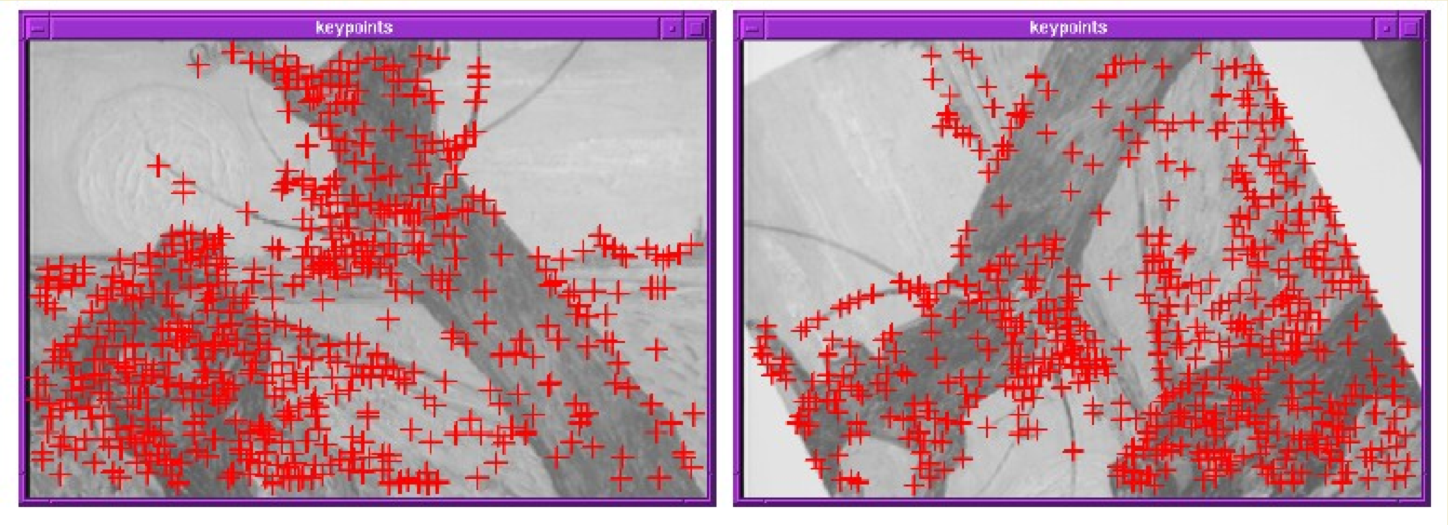
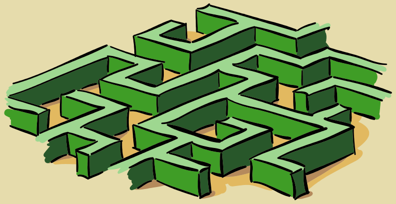
Finding Corners

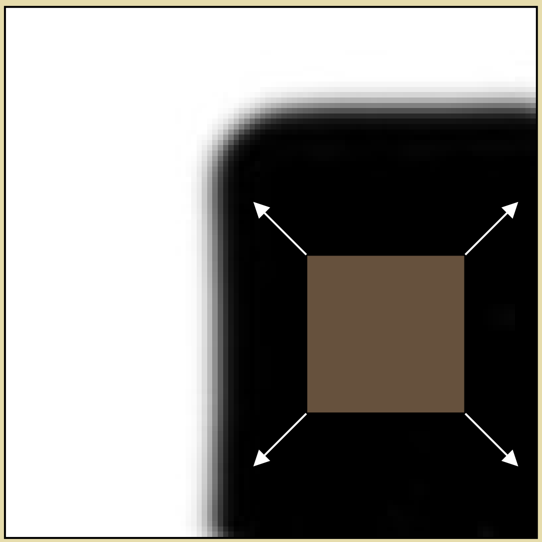


- Key property: in the region around a corner, image gradient has two or more dominant directions
- Corners are repeatable and distinctive

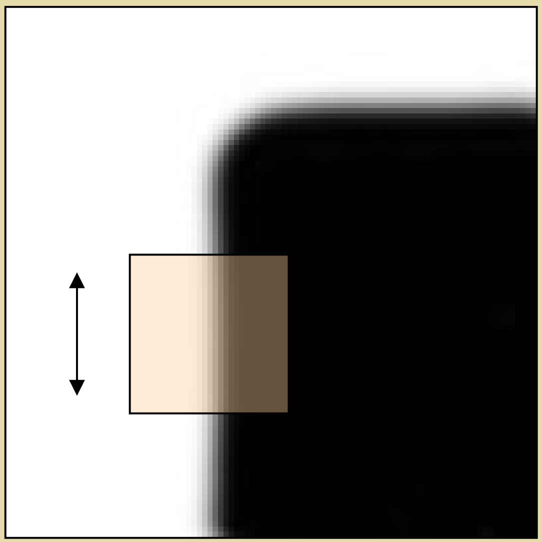


The Basic Idea

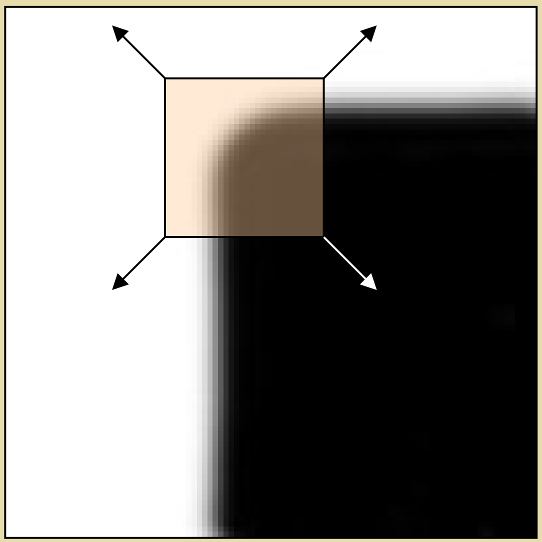
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in
all directions



"edge":
no change along
the edge
direction



"corner":
significant
change in all
directions

Source: A. Efros

Harris Corner Detector

Change in appearance for the shift [*u*,*v*]:

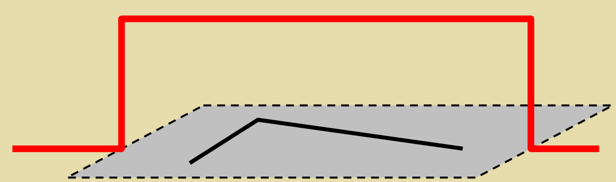
$$E(u,v)=\sum_{x,y} w(x,y)\left[I(x+u,y+v)-I(x,y)\right]^2$$

Window function

Shifted intensity

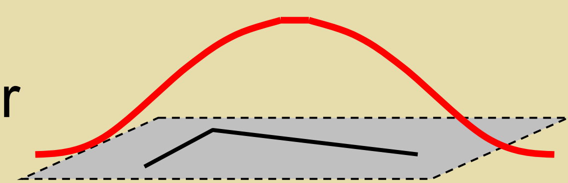
Intensity

Window function $w(x,y)$ =

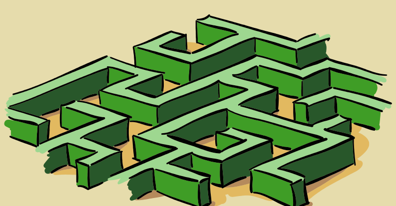


1 in window, 0 outside

or



Gaussian



Source: R. Szeliski

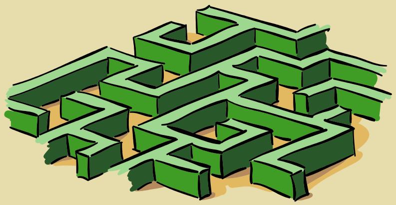
Harris Corner Detector

Change in appearance for the shift [*u*,*v*]:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Second-order Taylor expansion of *E*(*u*,*v*) about (0,0) (bilinear approximation for small shifts):

$$E(u,v) \approx E(0,0) + [u \quad v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2}[u \quad v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$



Harris Detector: Mathematics

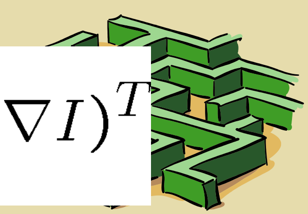
The bilinear approximation simplifies to

$$E(u,v) \approx [u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

where *M* is a 2×2 matrix computed from image derivatives:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \; I_y] = \sum \nabla I (\nabla I)^T$$

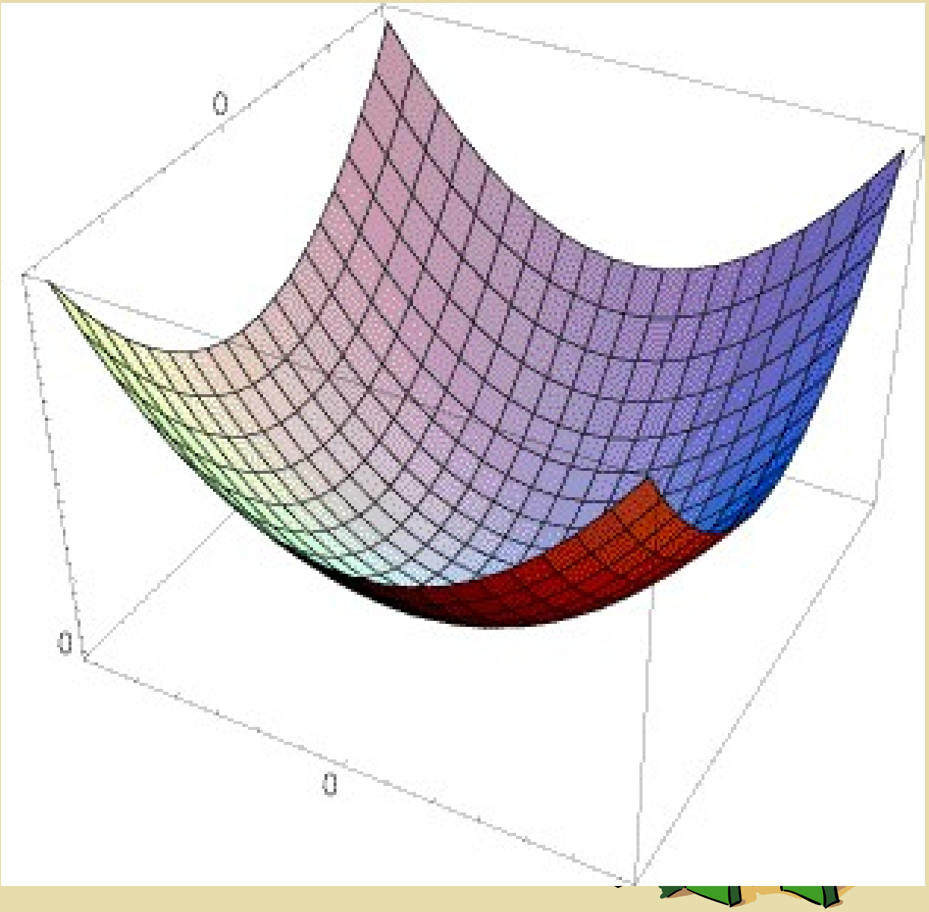


Interpreting the second moment matrix

The surface *E*(*u*,*v*) is locally approximated by a quadratic form. Let's try to understand its shape.

$$E(u,v) \approx [u \ \ v] \ M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



Interpreting the second moment matrix

First, consider the axis-aligned case (gradients are either horizontal or vertical)

$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

If either λ is close to 0, then this is **not** a corner, so look for locations where both are large.

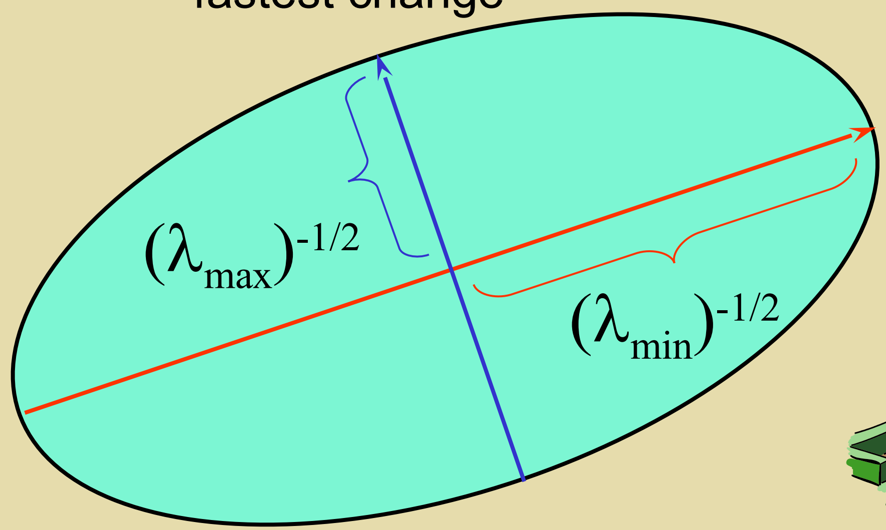
General Case

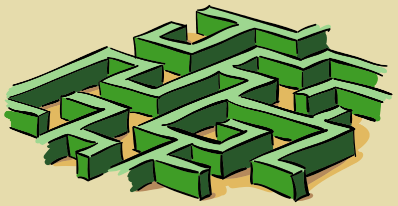
Since M is symmetric, we have $$M = R^{-1}\begin{bmatrix}\lambda_1 & 0 \\ 0 & \lambda_2\end{bmatrix}R$$

We can visualize *M* as an ellipse with axis lengths determined by the eigenvalues and orientation determined by *R*

Ellipse equation:

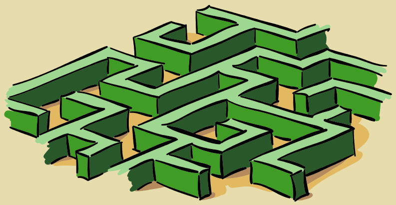
$$[u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$





Harris detector: Steps

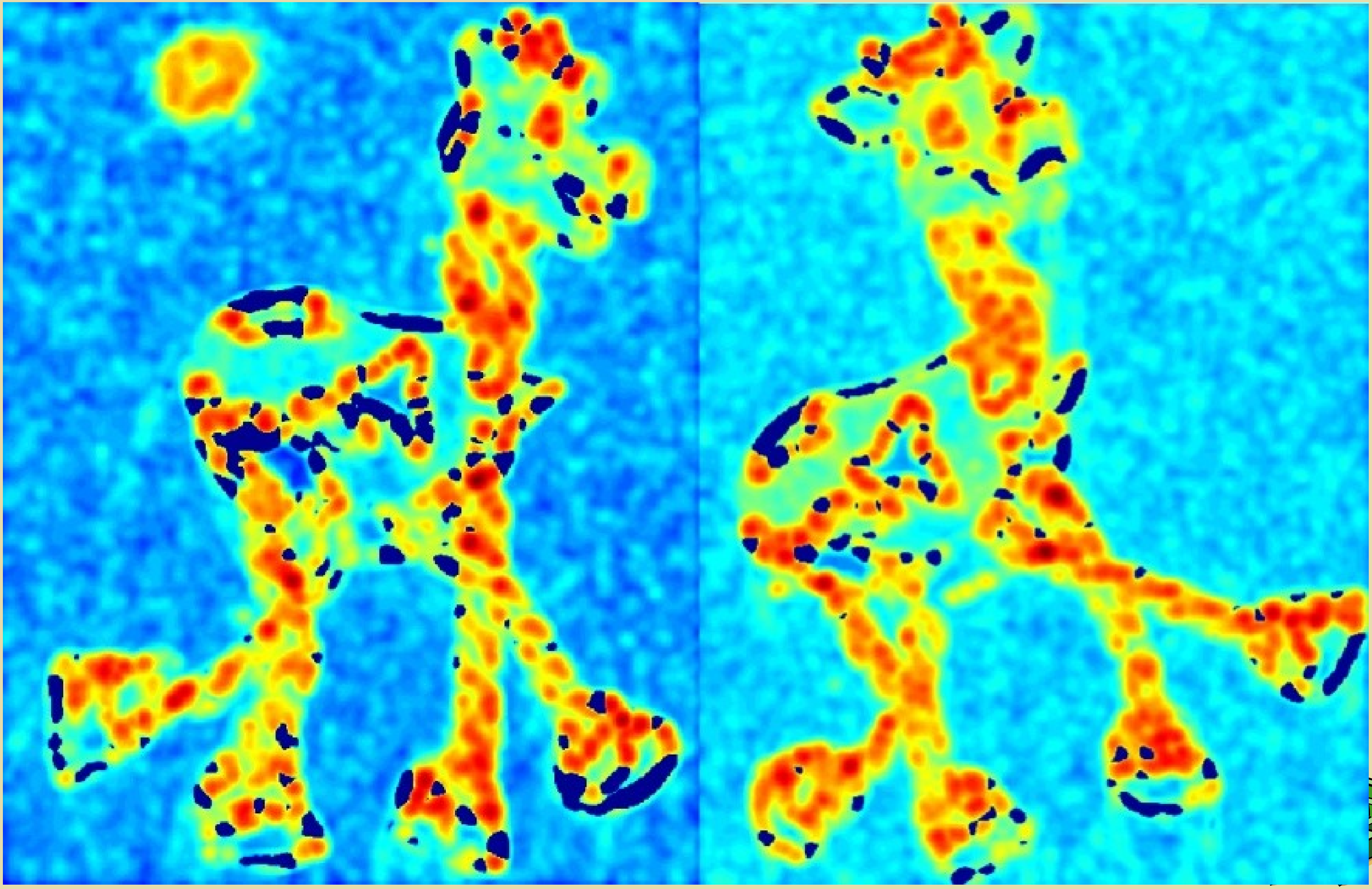
1. Compute Gaussian derivatives at each pixel
2. Compute second moment matrix *M* in a Gaussian window around each pixel
3. Compute corner response function *R*
4. Threshold *R*
5. Find local maxima of response function



Harris Detector: Steps

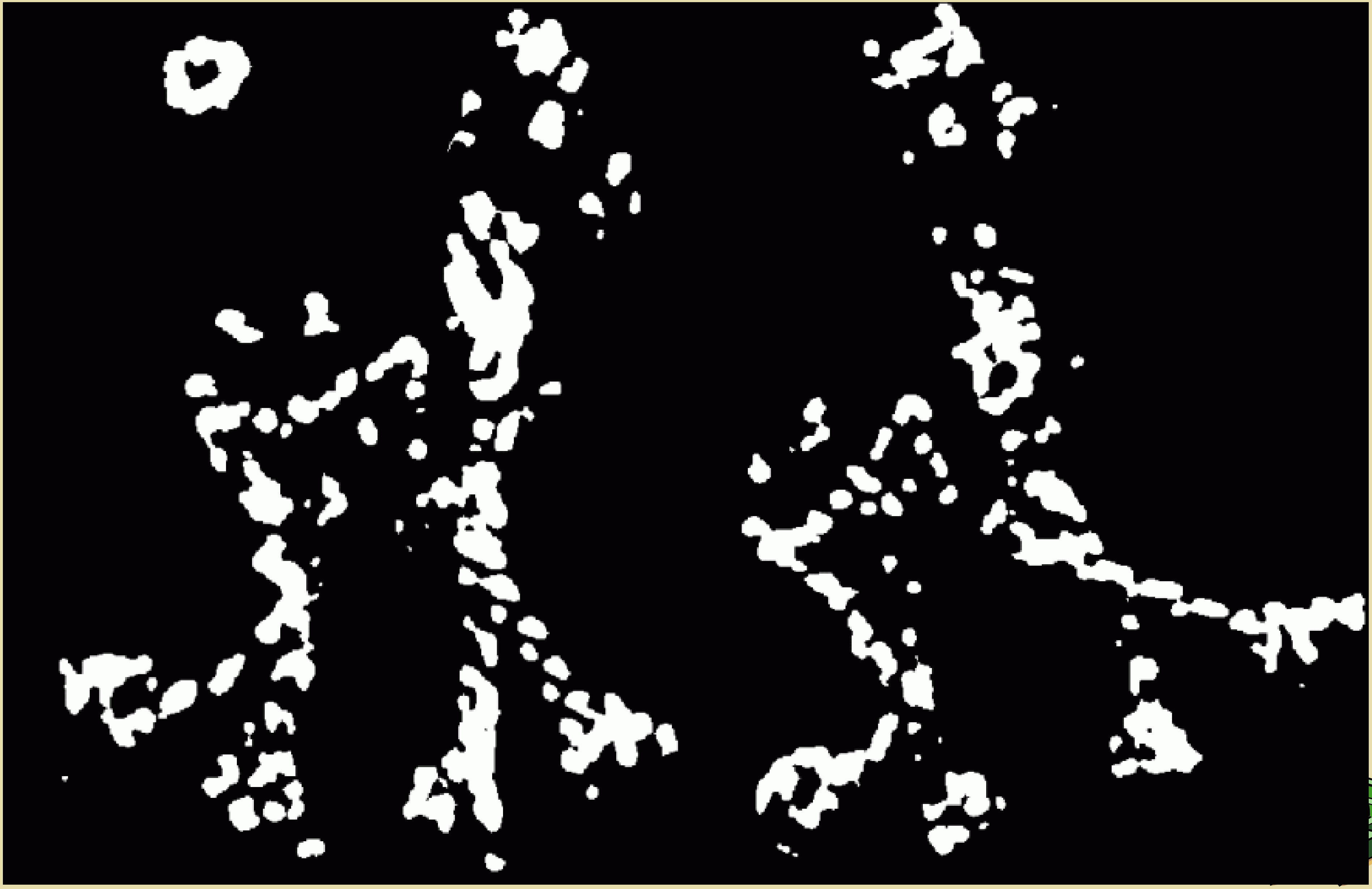
Harris Detector: Steps

Compute corner response R



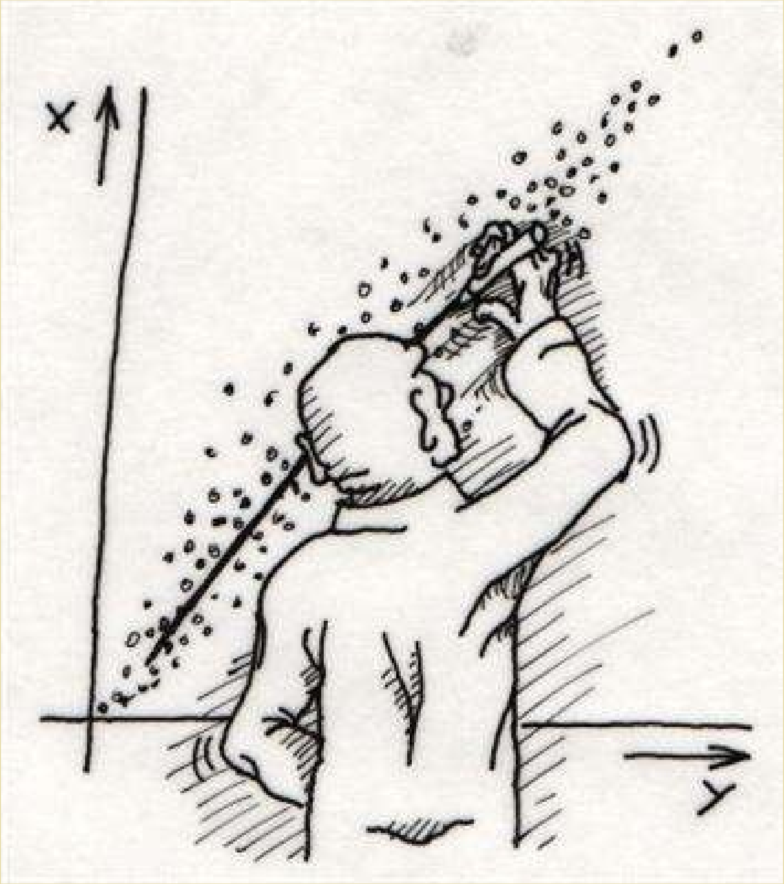
Harris Detector: Steps

Find points with large corner response: *R*>threshold



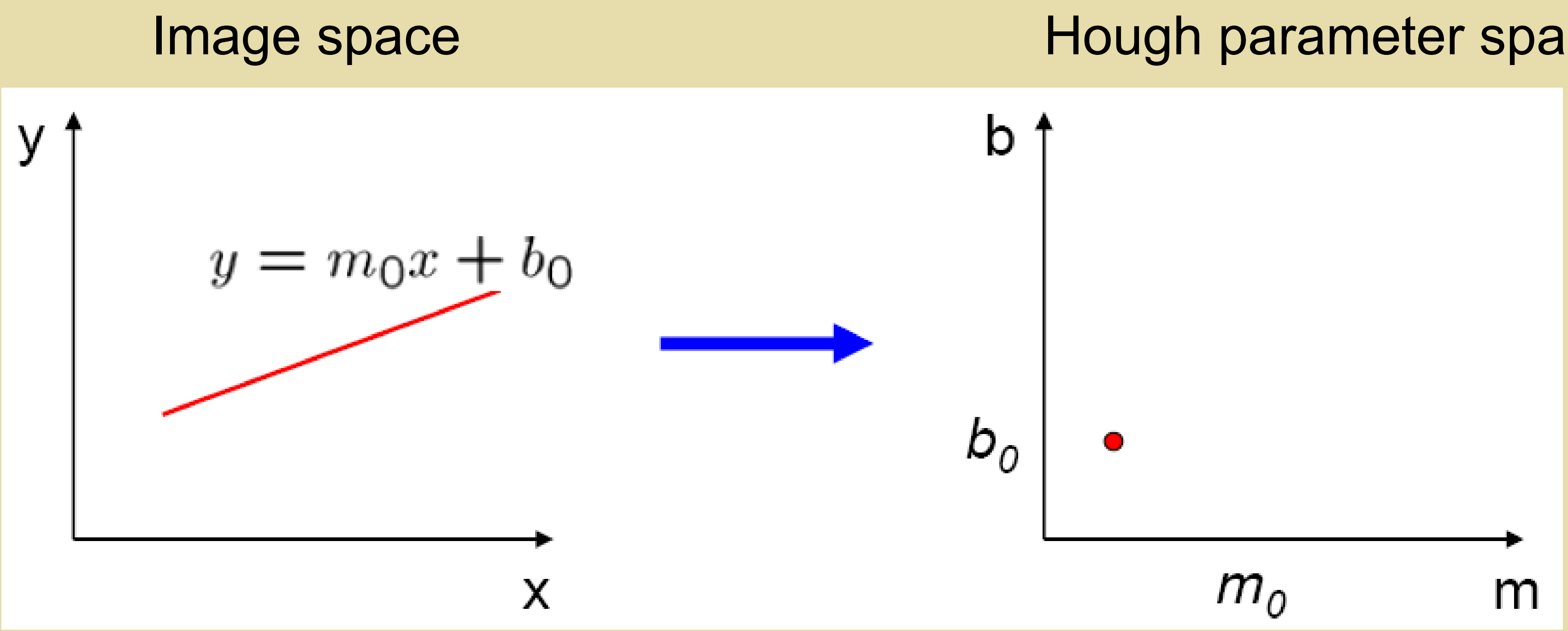
Harris Detector: Steps

The Hough transform



Parameter space representation

- A line in the image corresponds to a point in Hough space



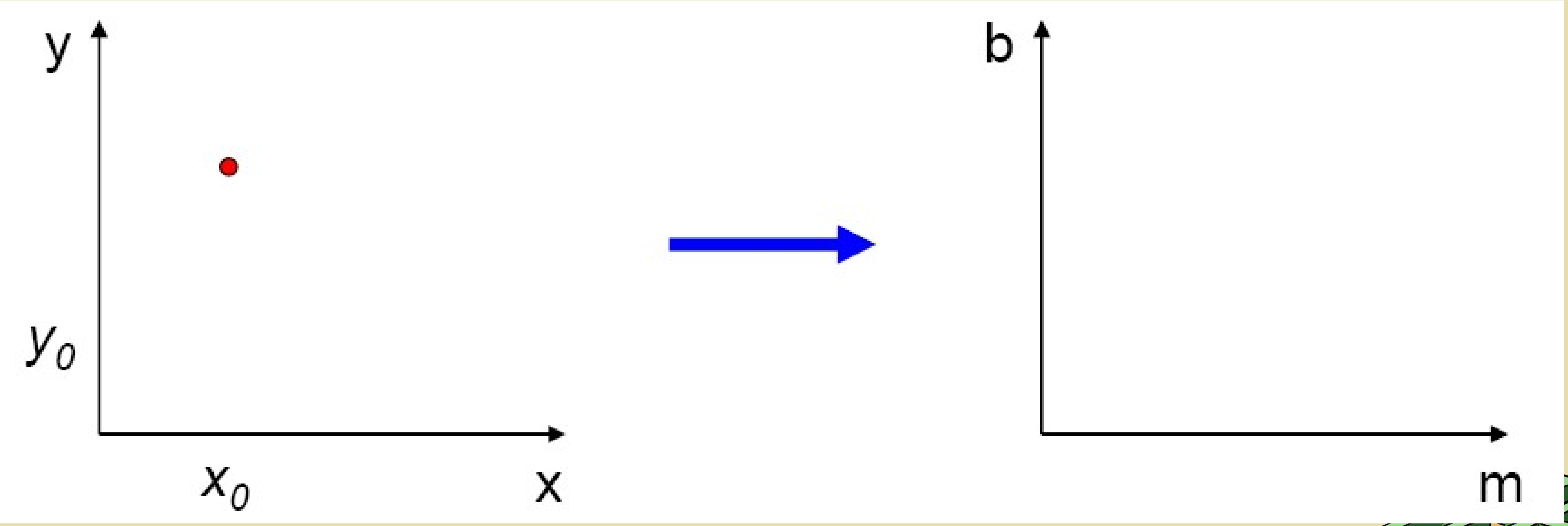
Source: S. Seitz

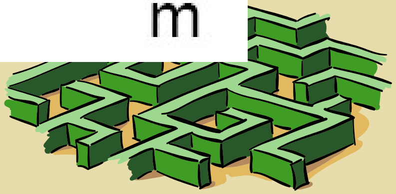
Parameter space representation

- What does a point (x_0, y_0) in the image space map to in the Hough space?

Image space

Hough parameter space



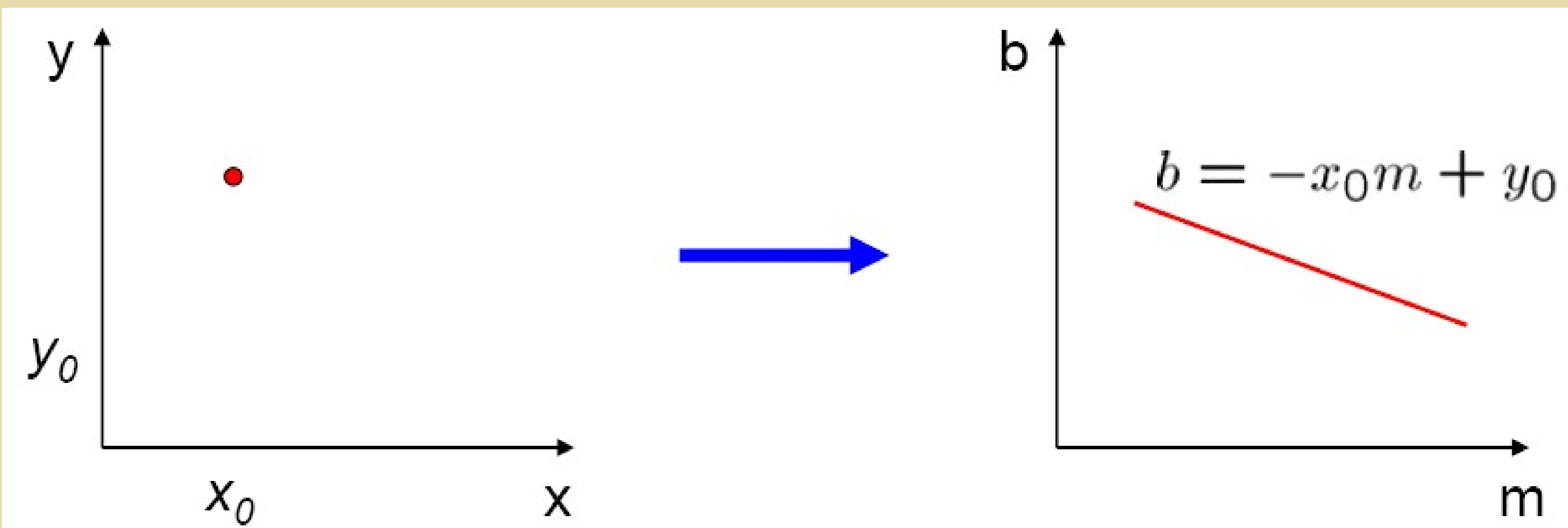


Parameter space representation

- What does a point (x_0, y_0) in the image space map to in the Hough space?
  - Answer: the solutions of $b = -x_0 m + y_0$
  - This is a line in Hough space

Image space

Hough parameter space

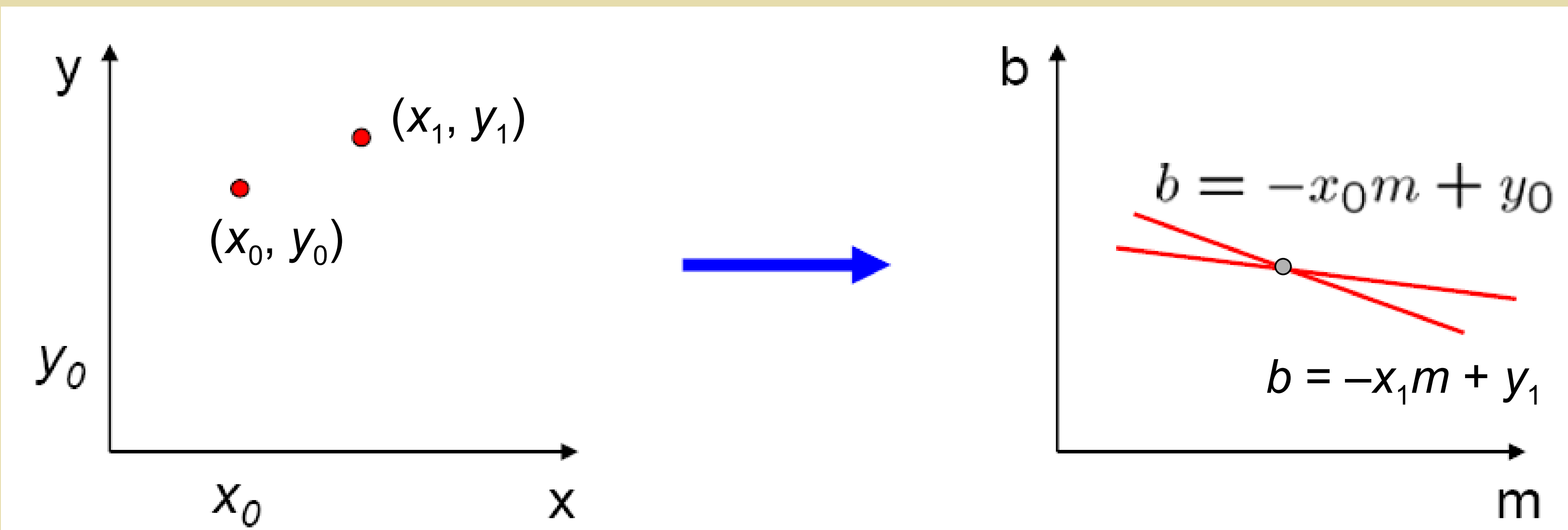


Parameter space representation

- Where is the line that contains both (x_0, y_0) and (x_1, y_1)?
  - It is the intersection of the lines $b = -x_0m + y_0$ and $b = -x_1m + y_1$

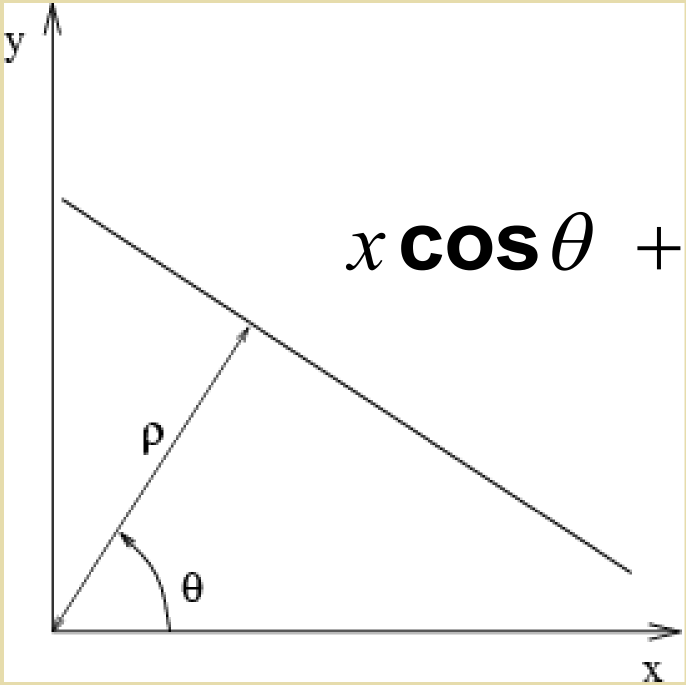
Image space

Hough parameter space



Parameter space representation

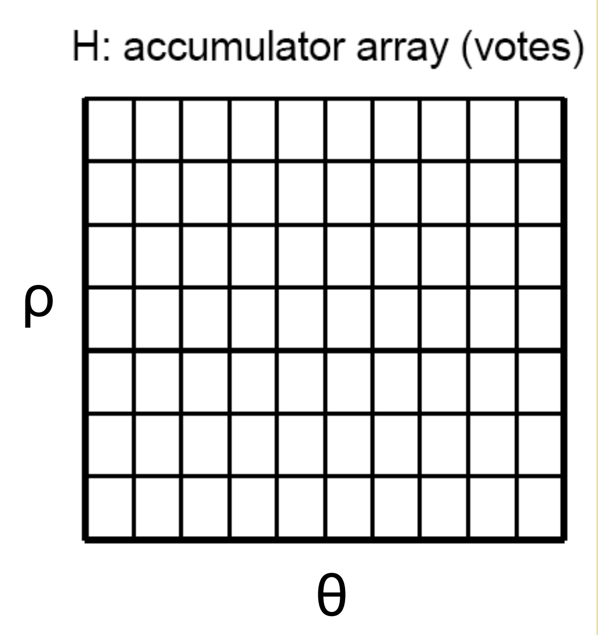
- Problems with the (m,b) space:
  - Unbounded parameter domain
  - Vertical lines require infinite m
- Alternative: polar representation



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

Each point will add a sinusoid in the (θ,ρ) parameter space

Algorithm outline



- Initialize accumulator H to all zeros
- For each edge point (x,y) in the image

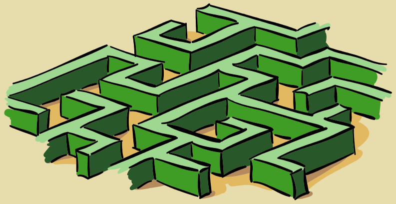
    For θ = 0 to 180

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

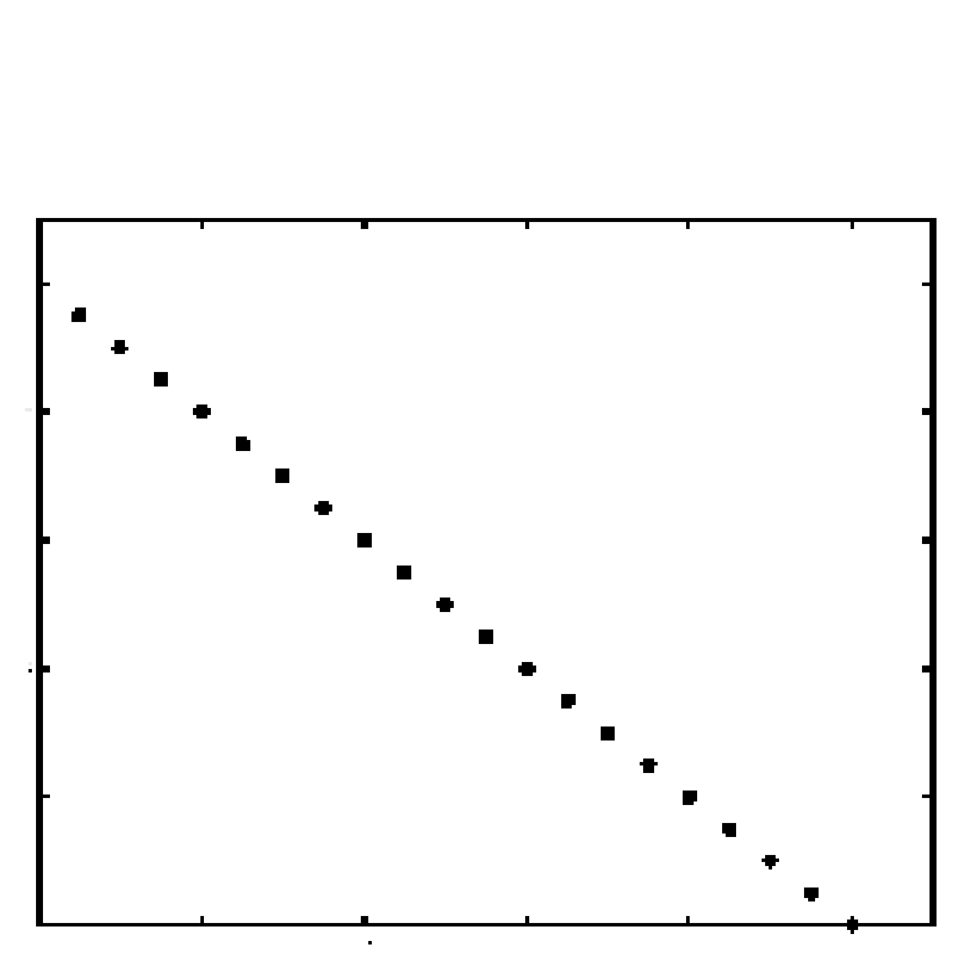
        $H(\theta, \rho) = H(\theta, \rho) + 1$

    end

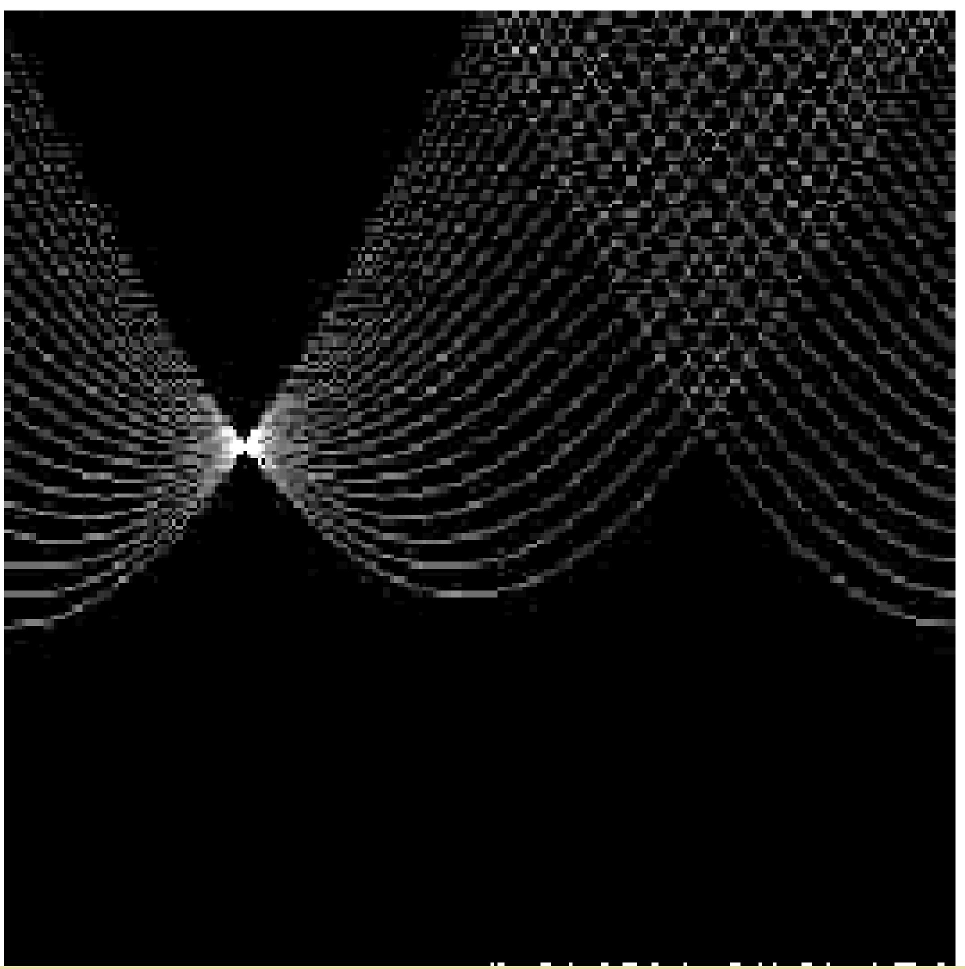
  end
- Find the value(s) of (θ, ρ) where $H(\theta, \rho)$ is a local maximum
  - The detected line in the image is given by $\rho = x \cos\theta + y \sin\theta$



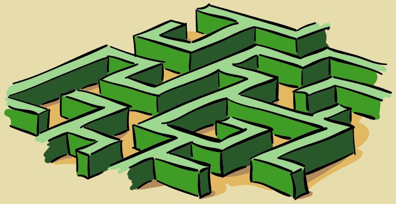
Basic illustration



features

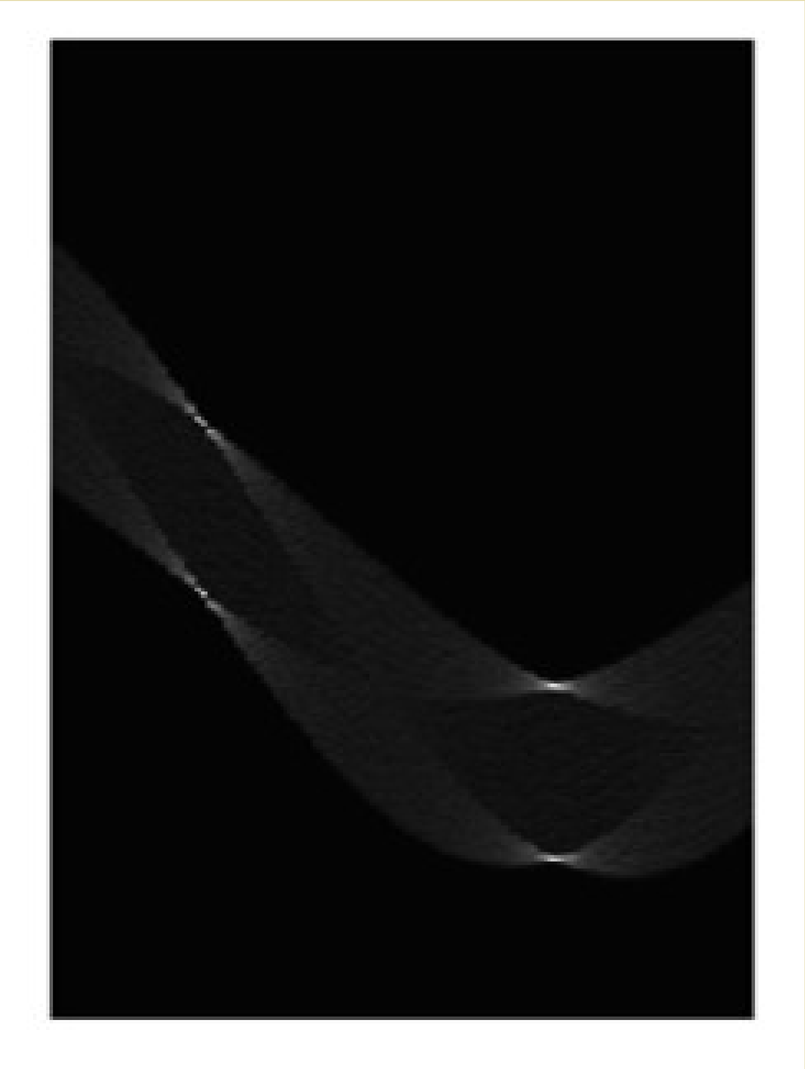


votes

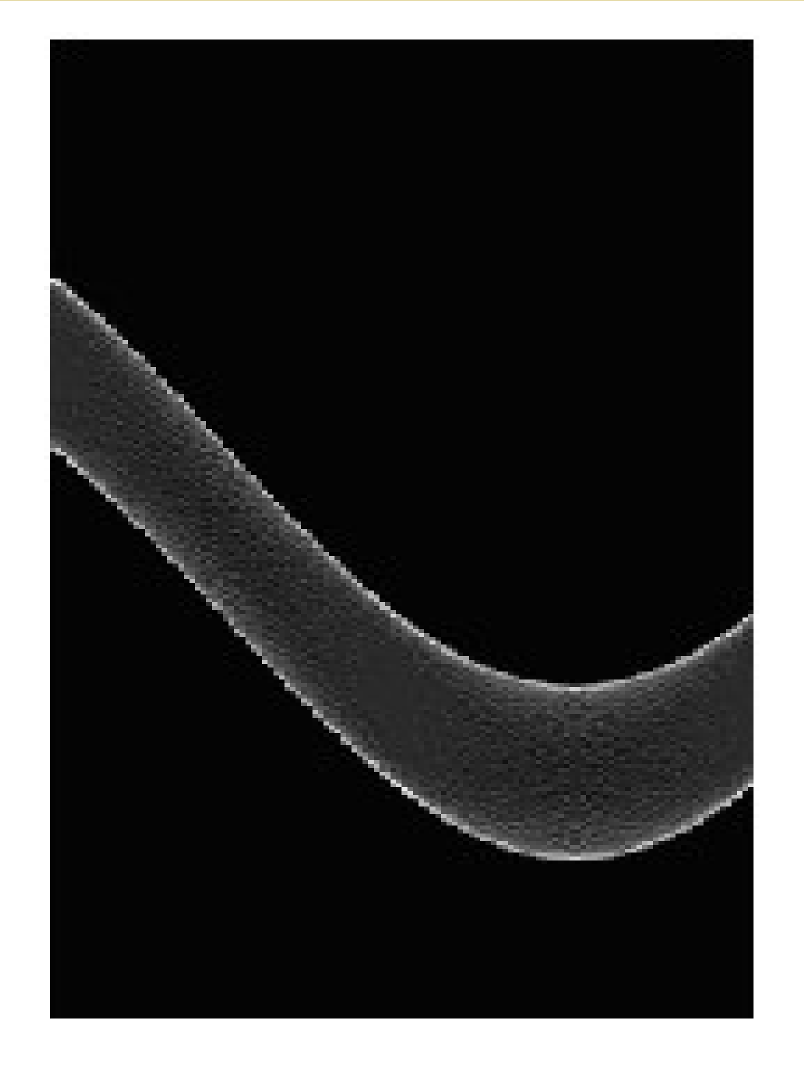


Other shapes

?

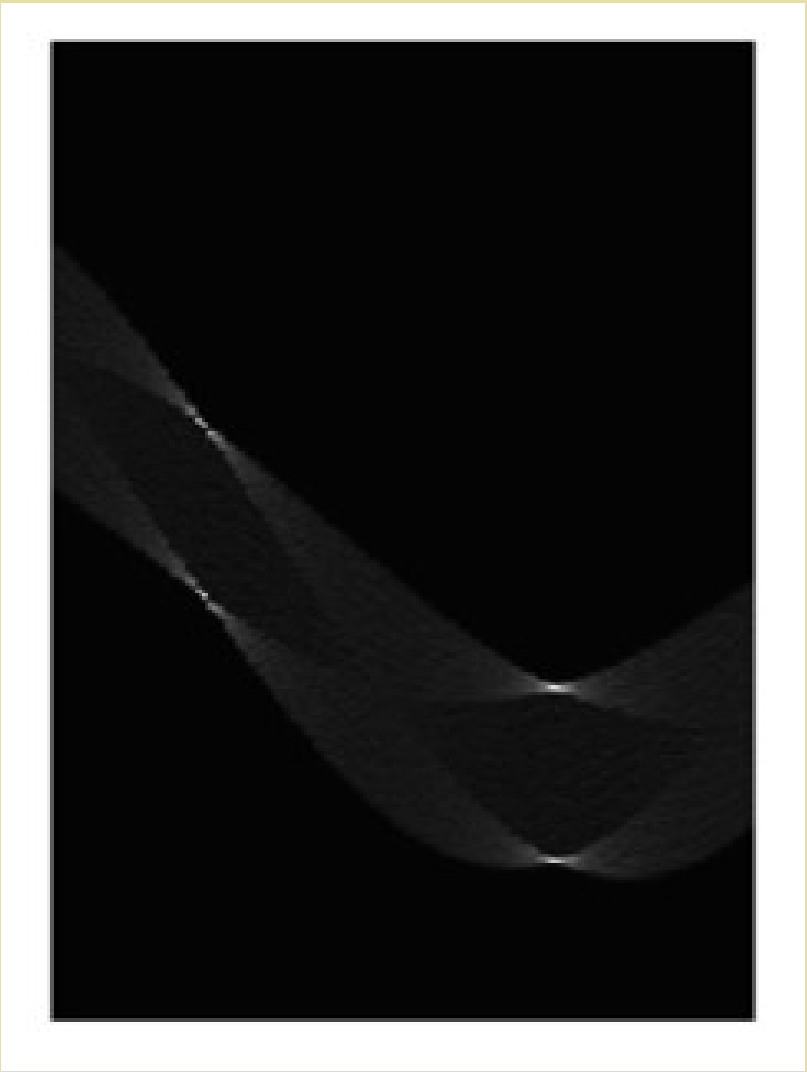


?

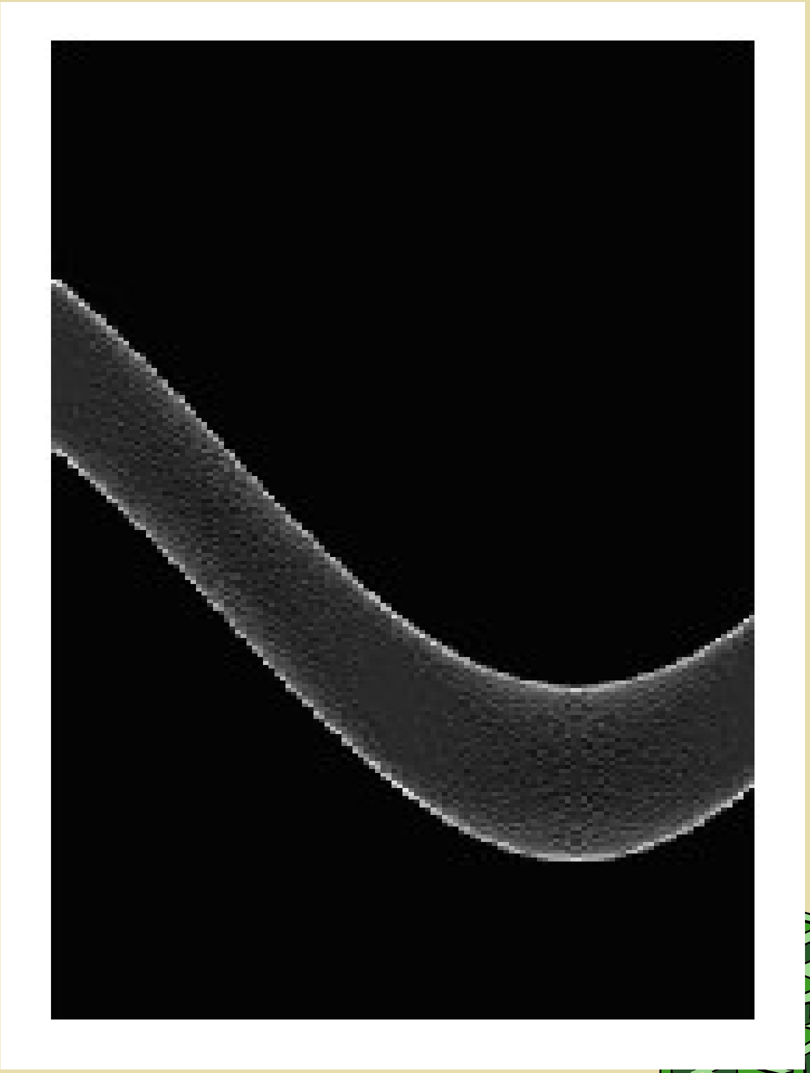


Other shapes

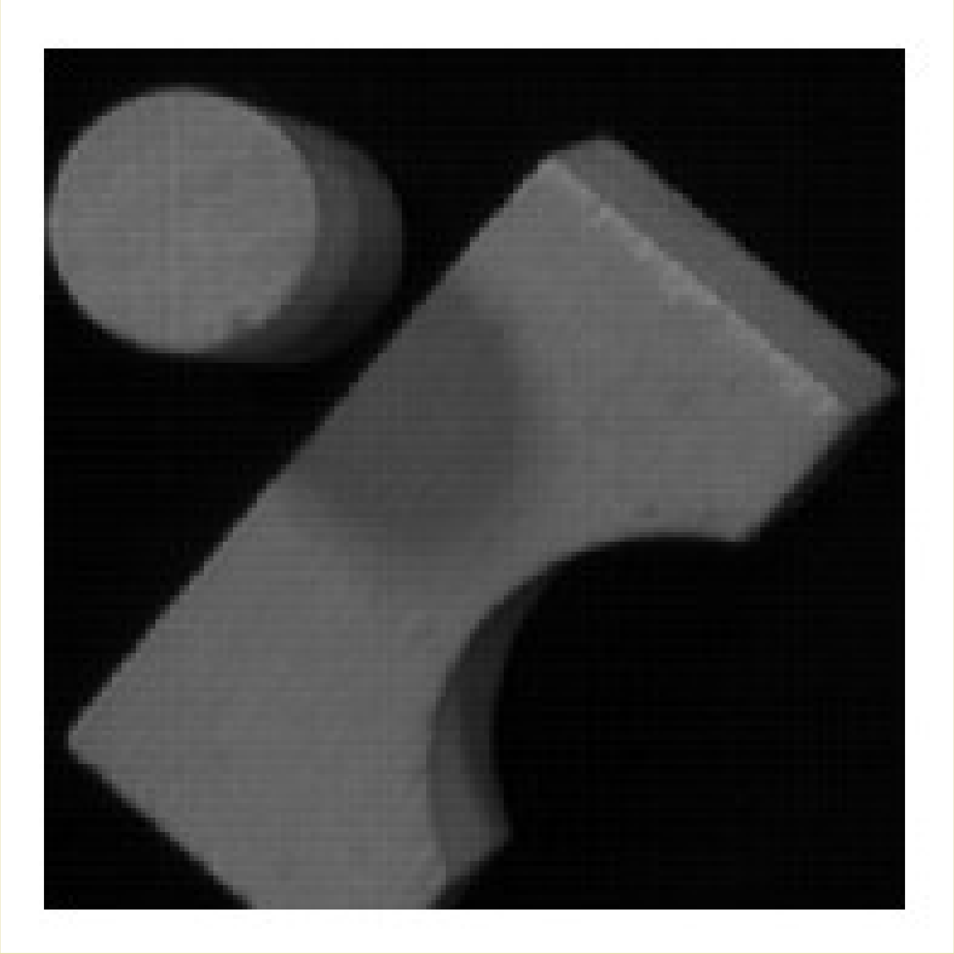
Square

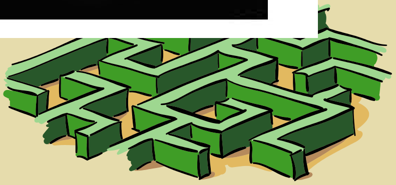


Circle

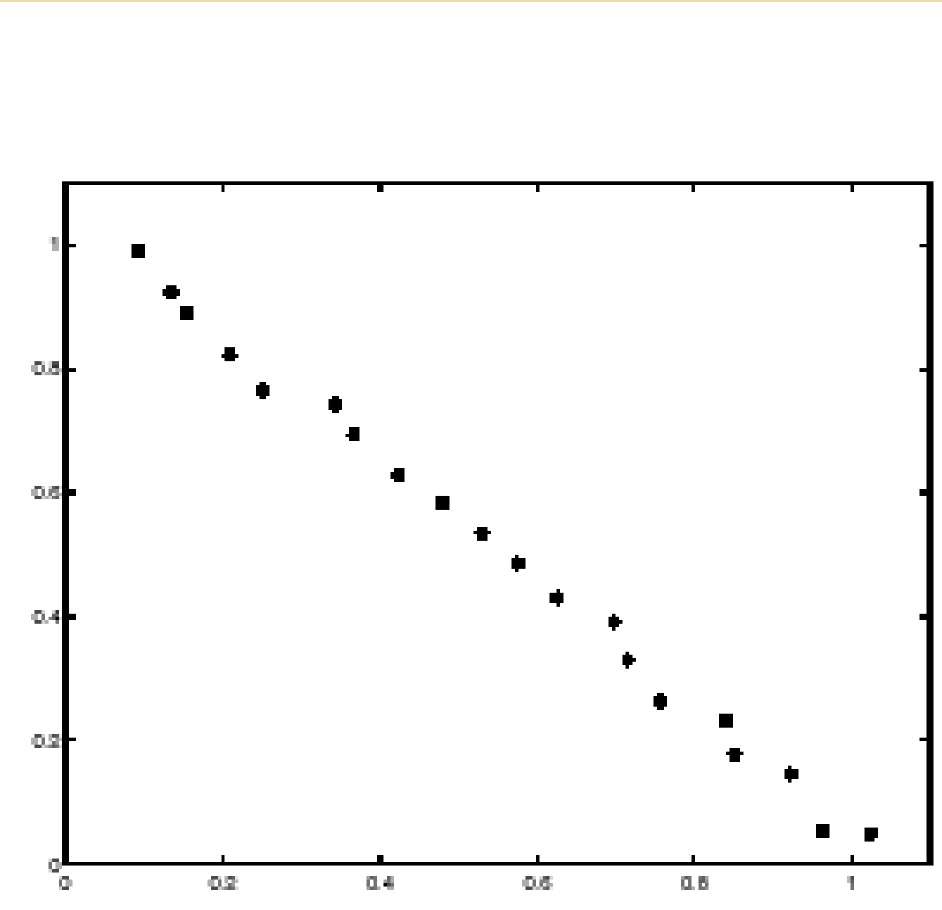


Several lines

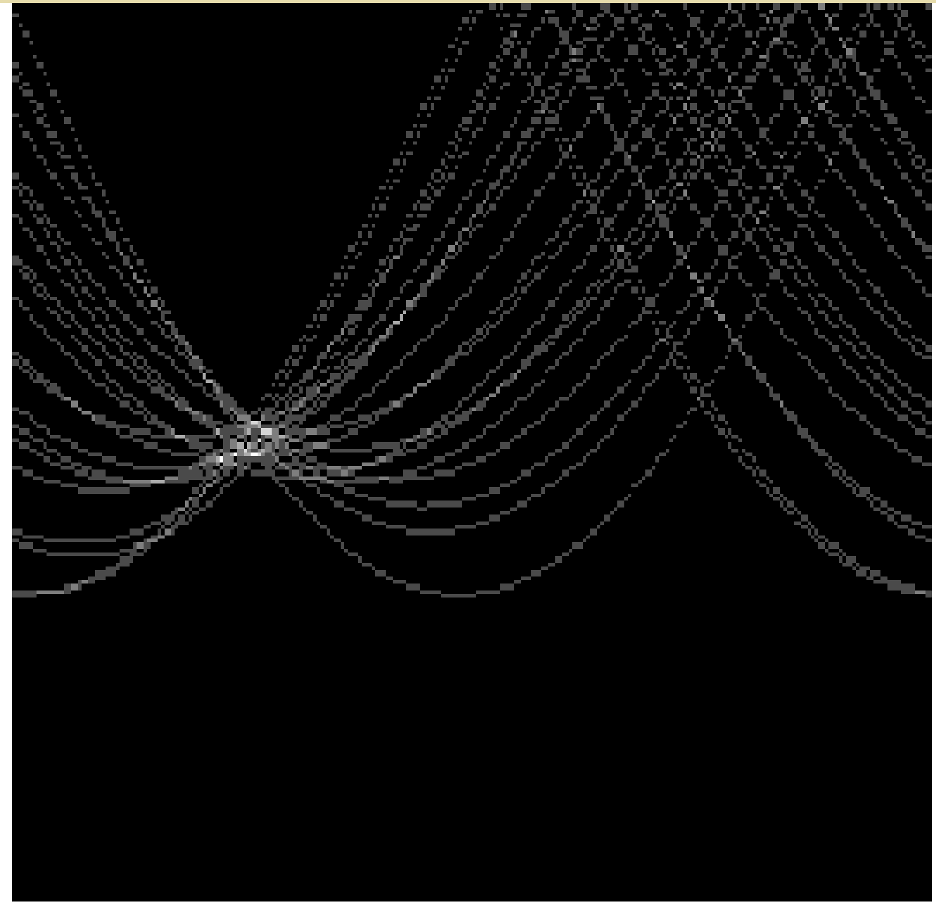




Effect of noise



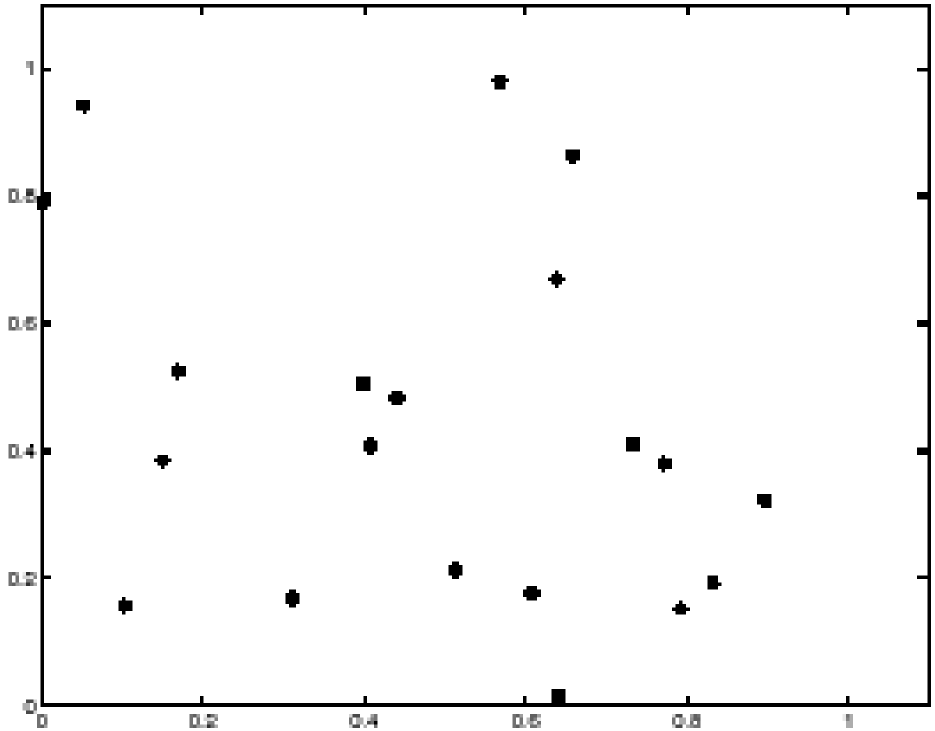
features



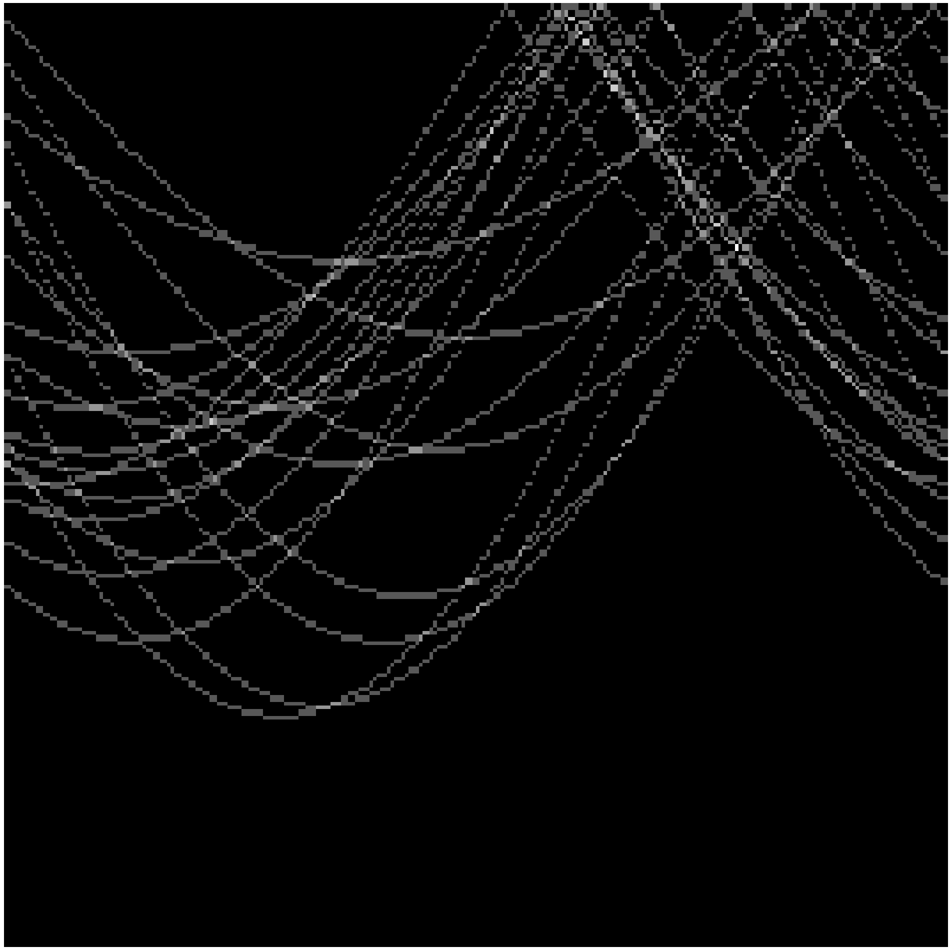
votes

- Peak gets fuzzy and hard to locate

Random points



features

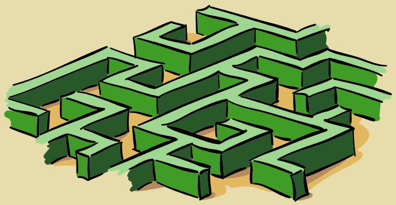


votes

- Uniform noise can lead to spurious peaks in the array

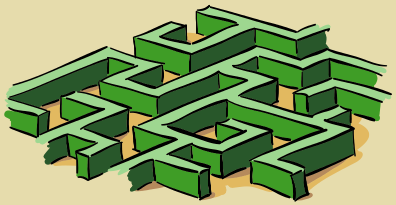
Practical details

- Try to get rid of irrelevant features
  - Take only edge points with significant gradient magnitude
- Choose a good grid / discretization
  - Too coarse: large votes obtained when too many different lines correspond to a single bucket
  - Too fine: miss lines because some points that are not exactly collinear cast votes for different buckets
- Increment neighboring bins (smoothing in accumulator array)
- Who belongs to which line?
  - Tag the votes



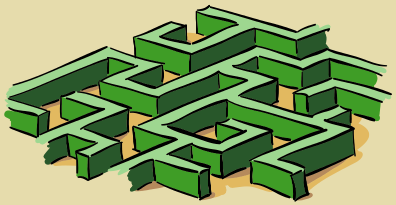
Hough transform: Pros

- Can deal with non-locality and occlusion
- Can detect multiple instances of a model in a single pass
- Some robustness to noise: noise points unlikely to contribute consistently to any single bin



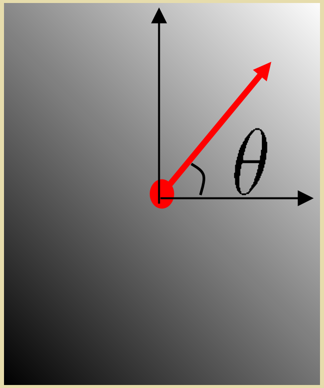
Hough transform: Cons

- Complexity of search time increases exponentially with the number of model parameters
- Non-target shapes can produce spurious peaks in parameter space
- It’s hard to pick a good grid size



Hough Extension: Using image gradients

- When an edge point is detected, the gradient direction is known
- But this means that the line is uniquely determined!

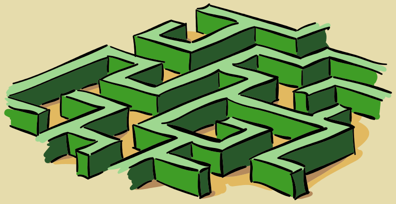


$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- Modified Hough transform:

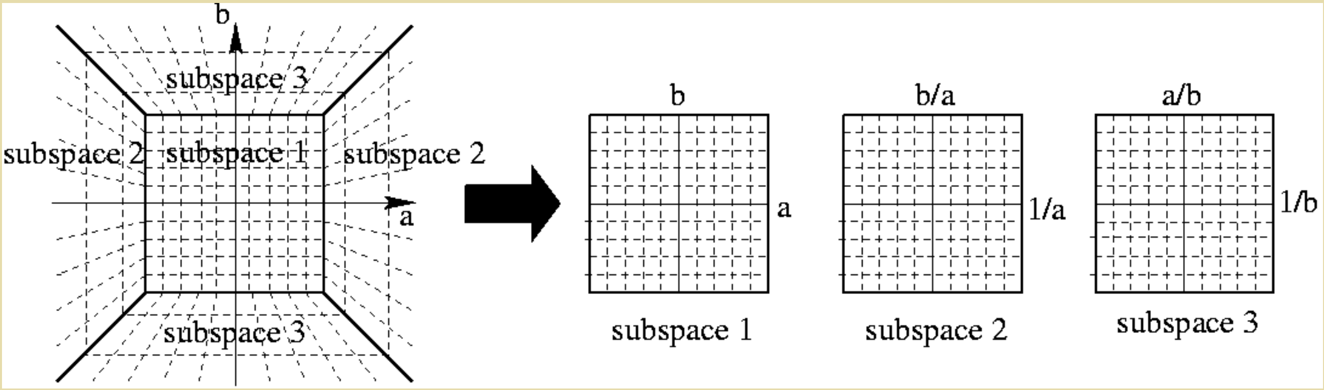
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

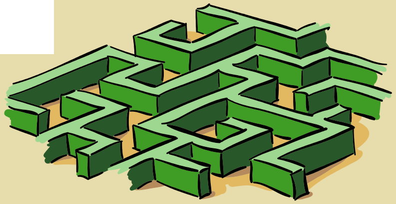
For each edge point (x,y)
  θ = gradient orientation at (x,y)
  ρ = x cos θ + y sin θ
  H(θ, ρ) = H(θ, ρ) + 1
end



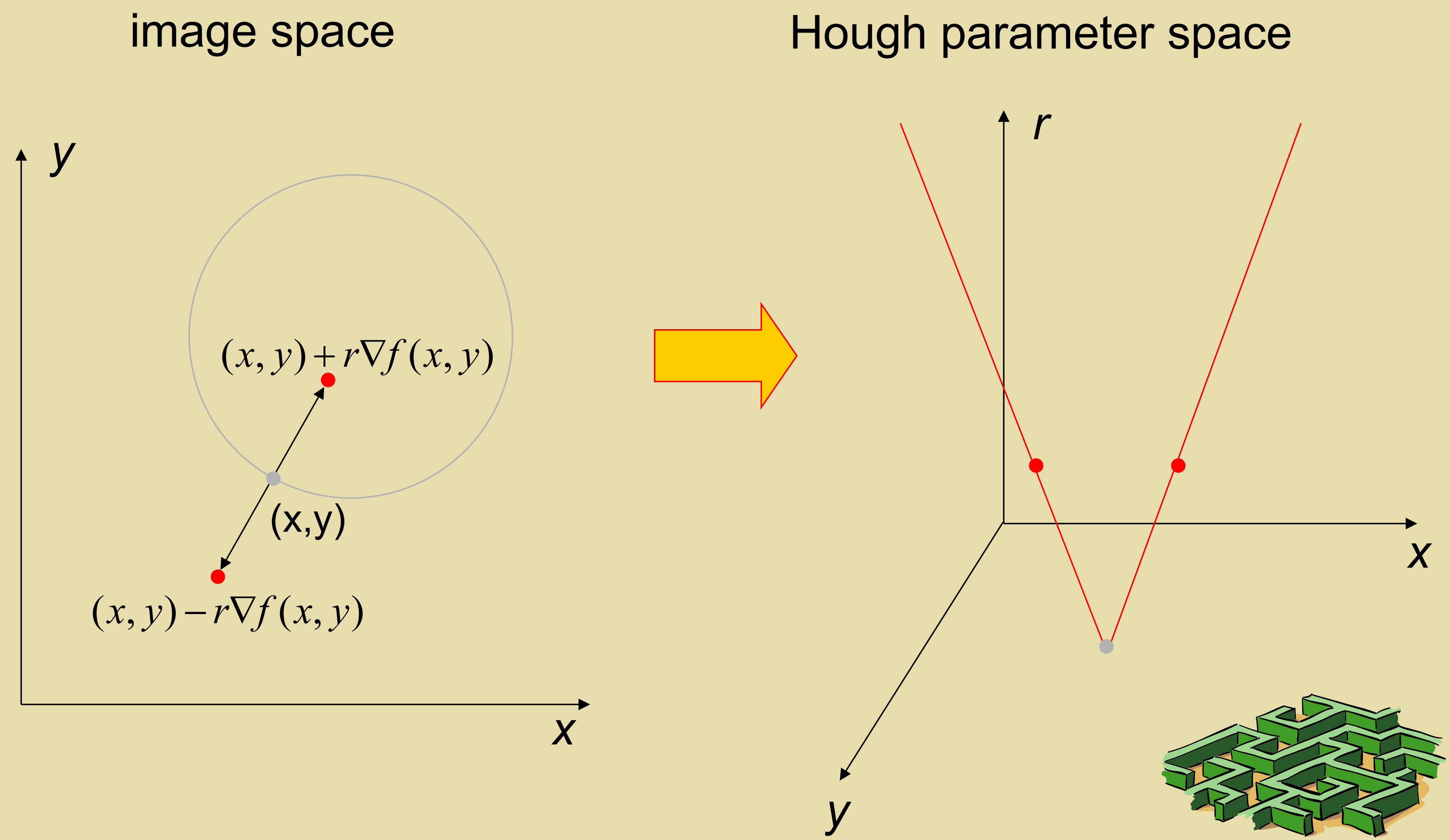
Extension: Cascaded Hough transform

- Let's go back to the original (m,b) parametrization
- A line in the image maps to a pencil of lines in the Hough space
- What do we get with parallel lines or a pencil of lines?
  - Collinear peaks in the Hough space!
- So we can apply a Hough transform to the output of the first Hough transform to find vanishing points
- Issue: dealing with unbounded parameter space



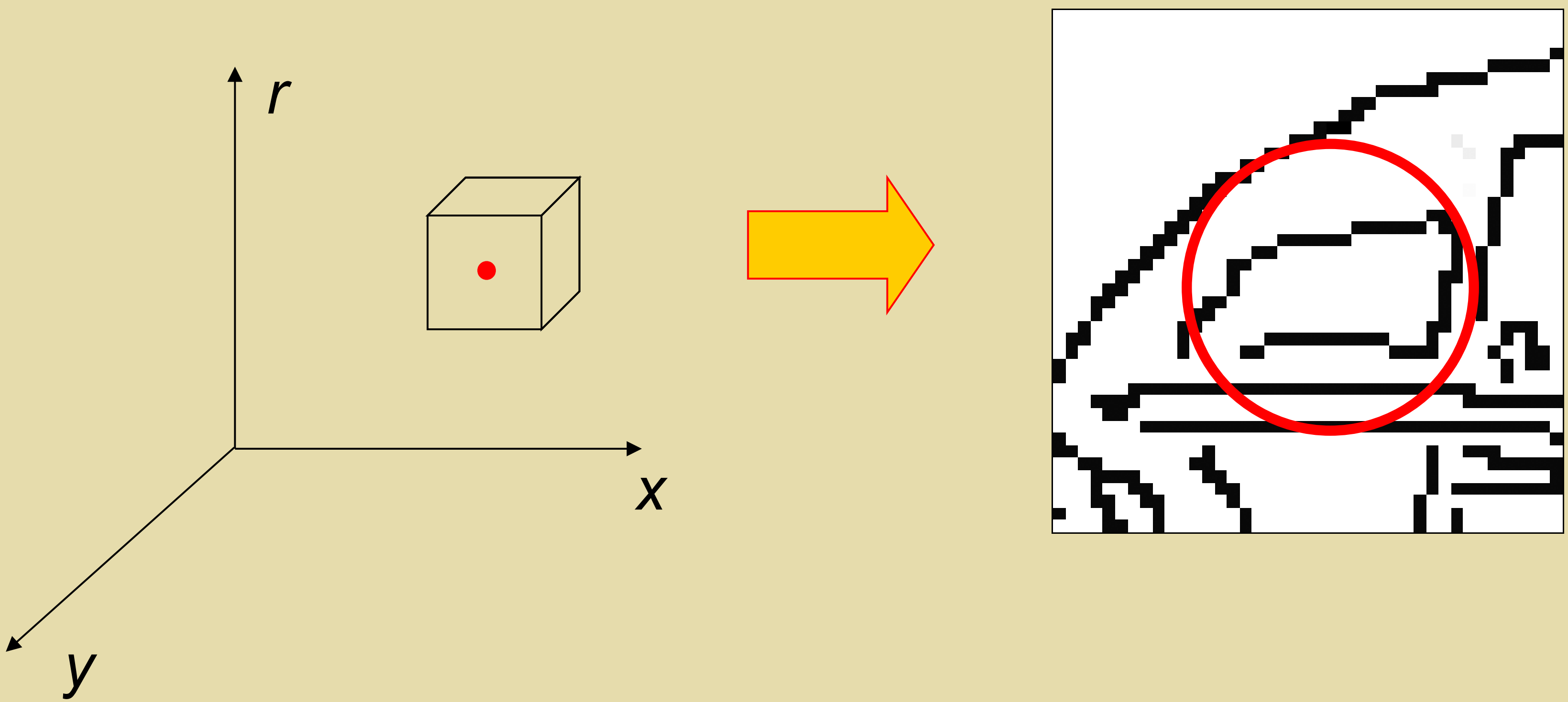


Hough transform for circles



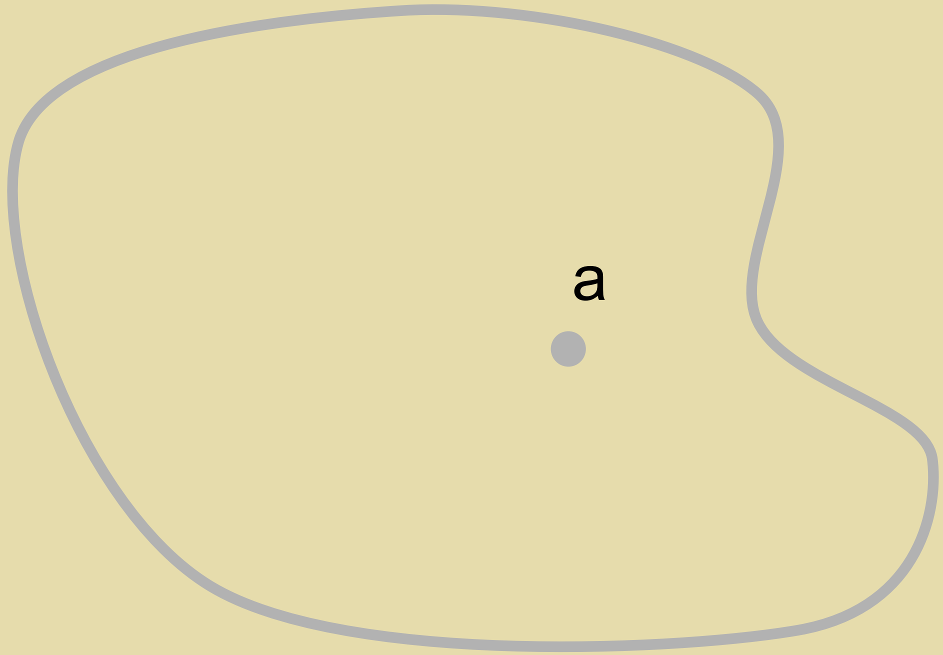
Hough transform for circles

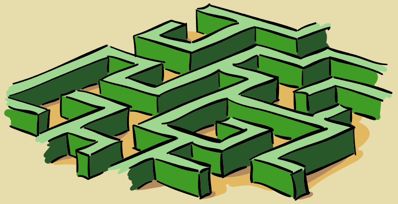
- Conceptually equivalent procedure: for each (x,y,r), draw the corresponding circle in the image and compute its "support"



Generalized Hough transform

- We want to find a shape defined by its boundary points and a reference point





Generalized Hough transform

- We want to find a shape defined by its boundary points and a reference point
- For every boundary point p, we can compute the displacement vector r = a – p as a function of gradient orientation θ