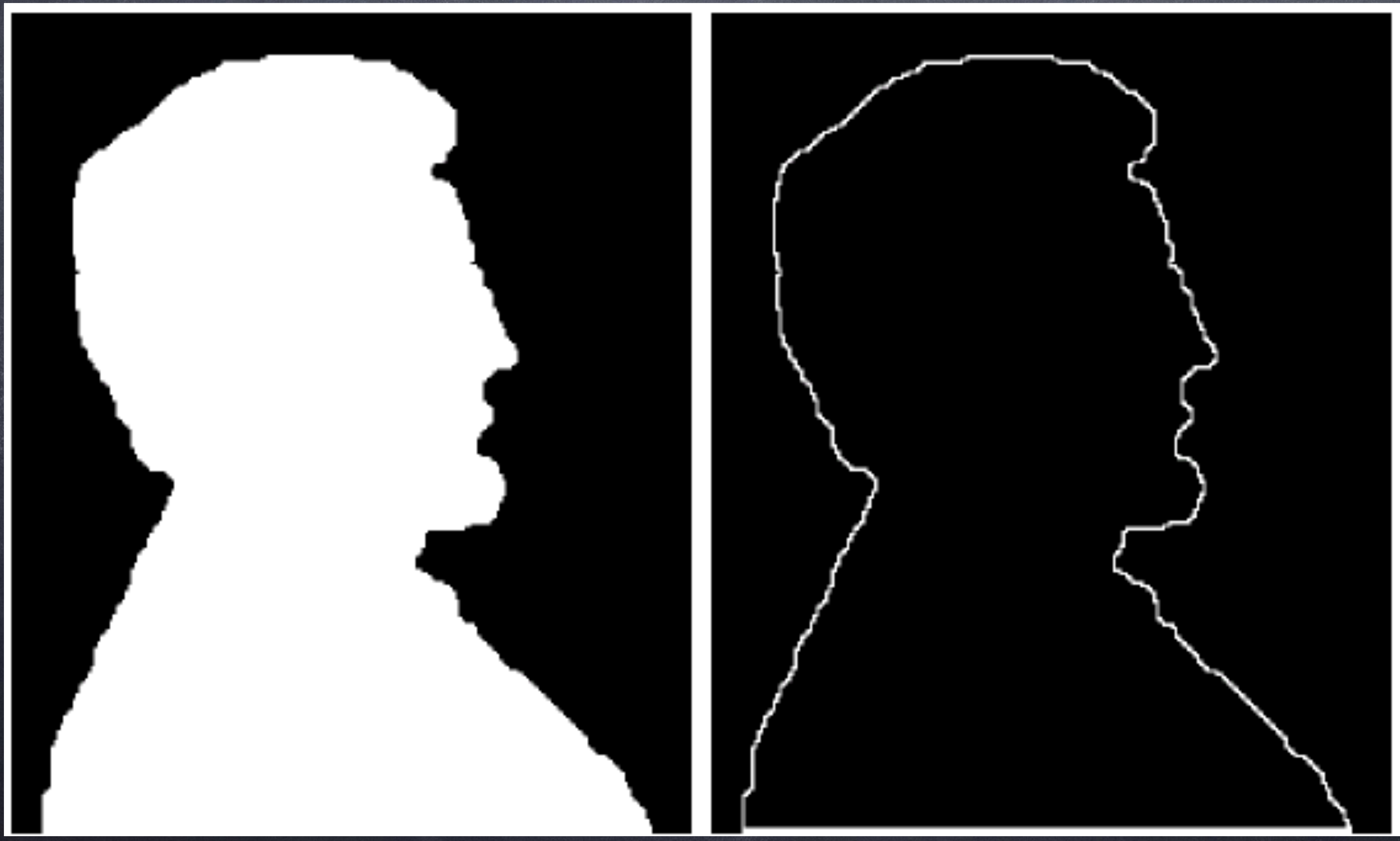
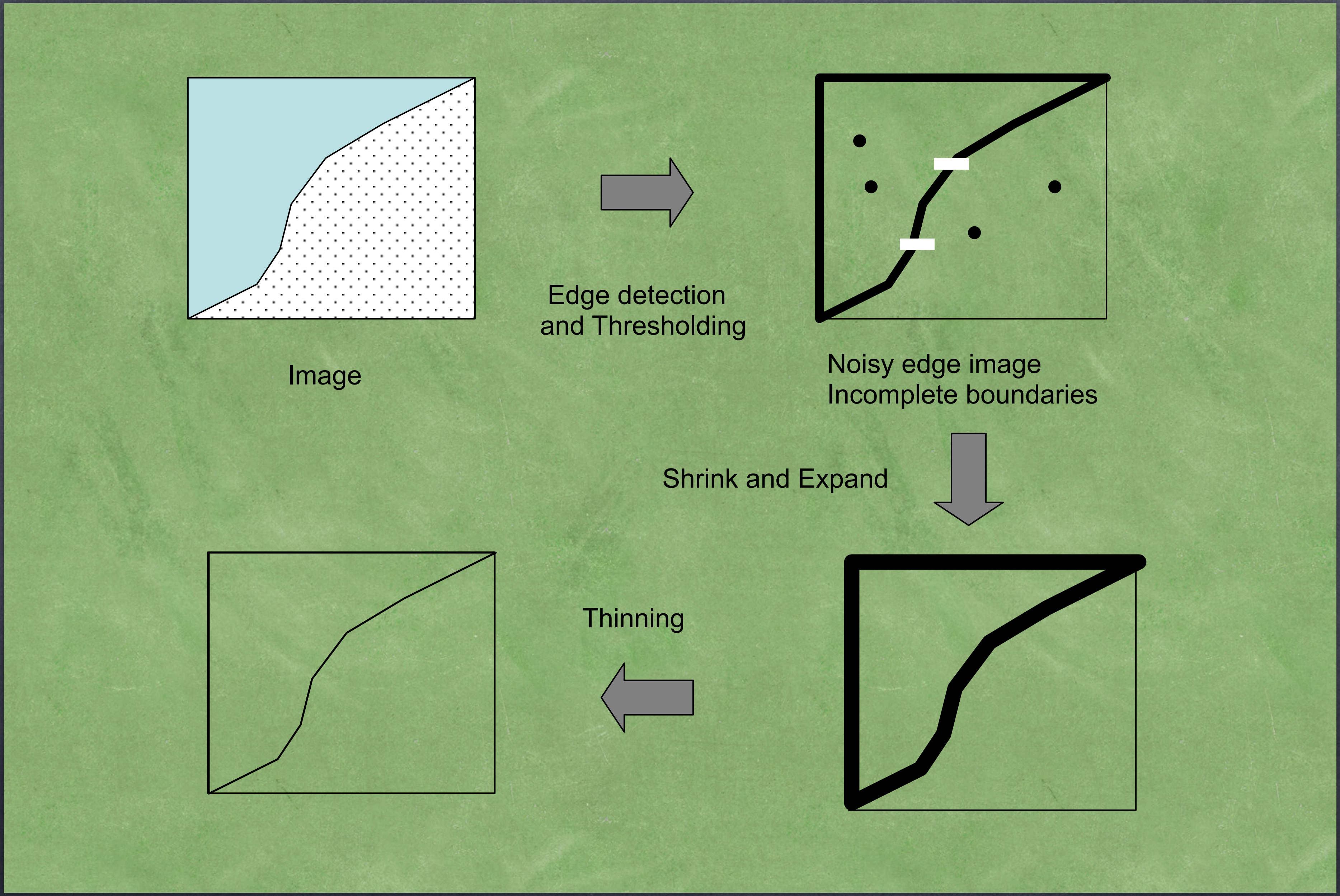
Computer Vision

Question: boundary extraction



Preprocessing Edge Images



Line Detection

- How we extract a line?

Line Detection

- Masks that extract lines of different directions.

FIGURE 10.3 Line masks.

Horizontal

−1	−1	−1
2	2	2
−1	−1	−1

+45°

−1	−1	2
−1	2	−1
2	−1	−1

Vertical

−1	2	−1
−1	2	−1
−1	2	−1

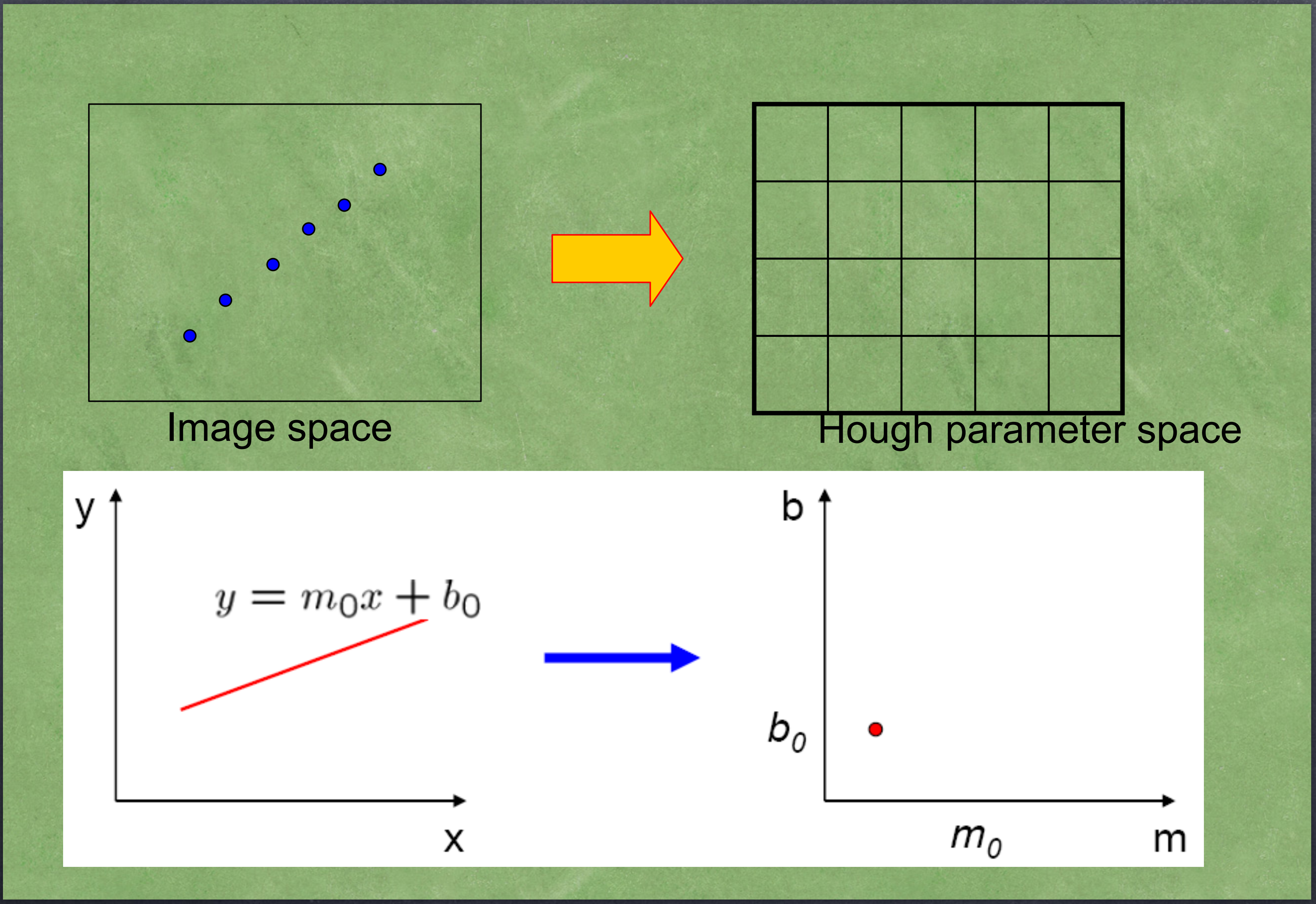
−45°

2	−1	−1
−1	2	−1
−1	−1	2

Can we do it better?

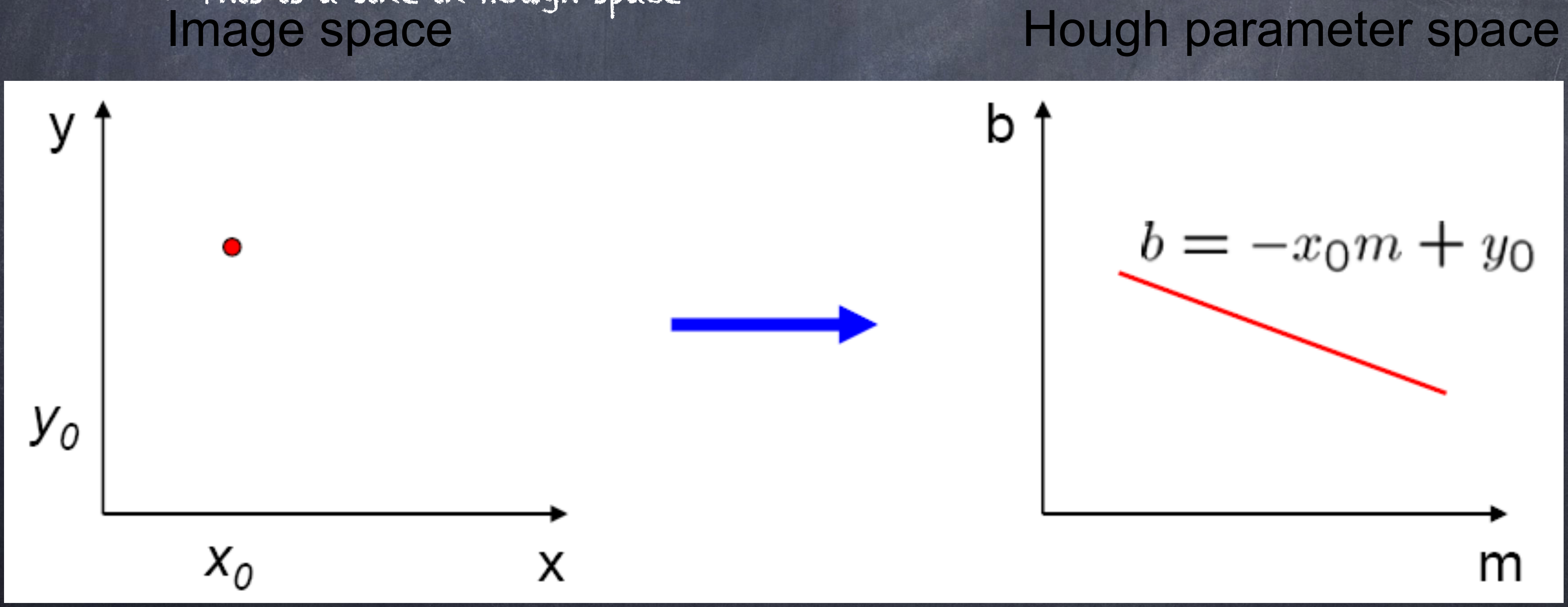
- Hough Transform

Hough Transform



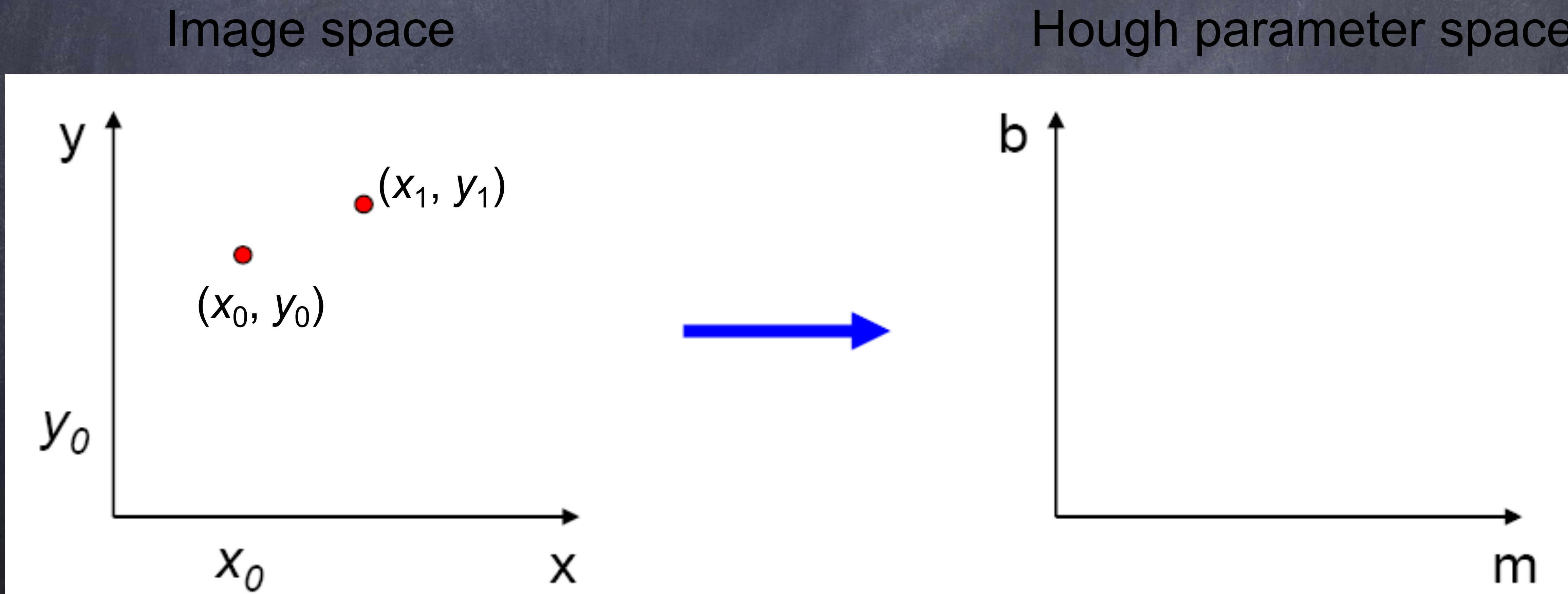
Parameter space representation

- 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



Parameter space representation

- Where is the line that contains both (x_0, y_0) and (x_1, y_1)?

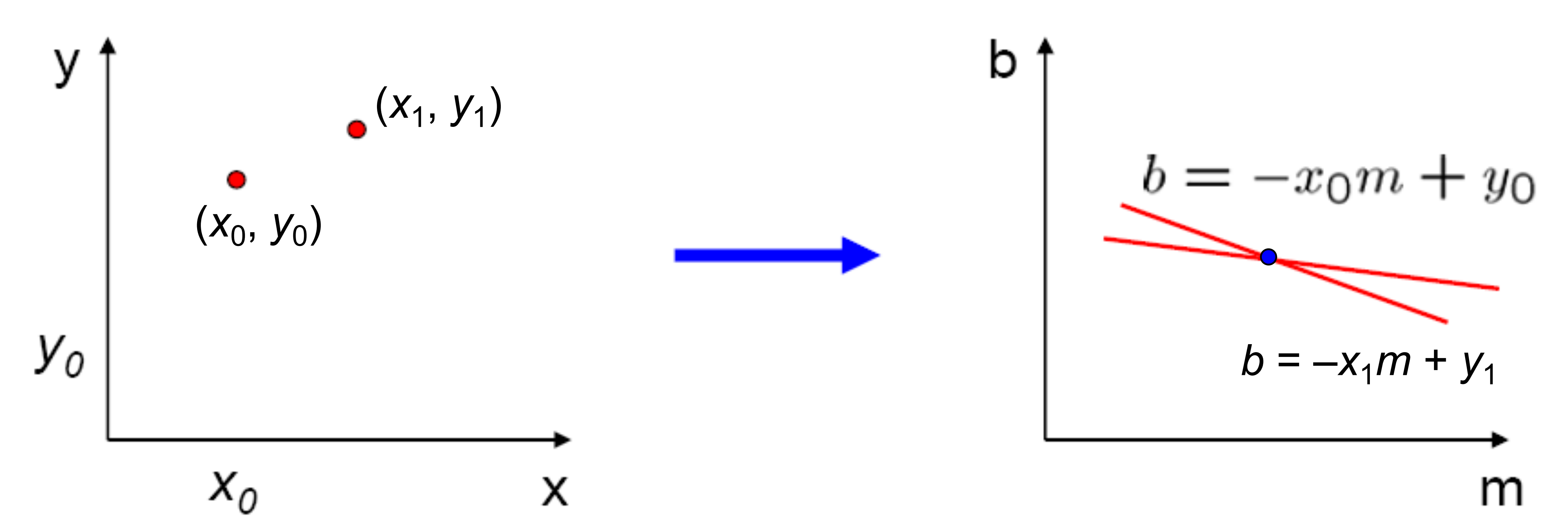


Parameter space representation

- 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

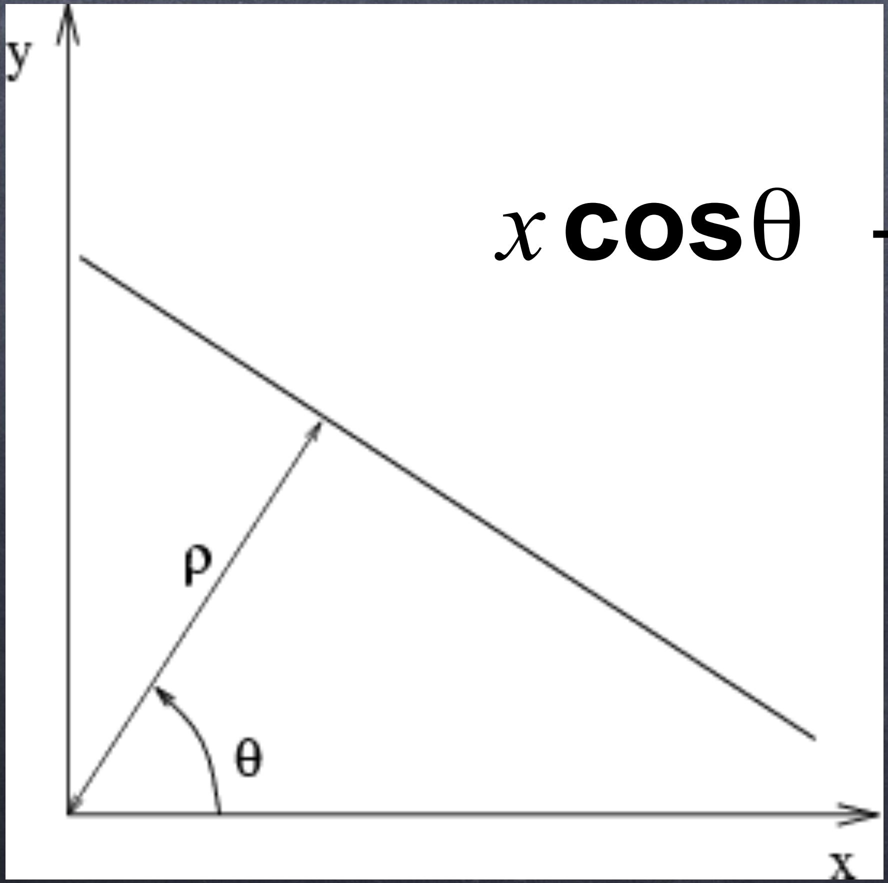


Parameter space representation

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

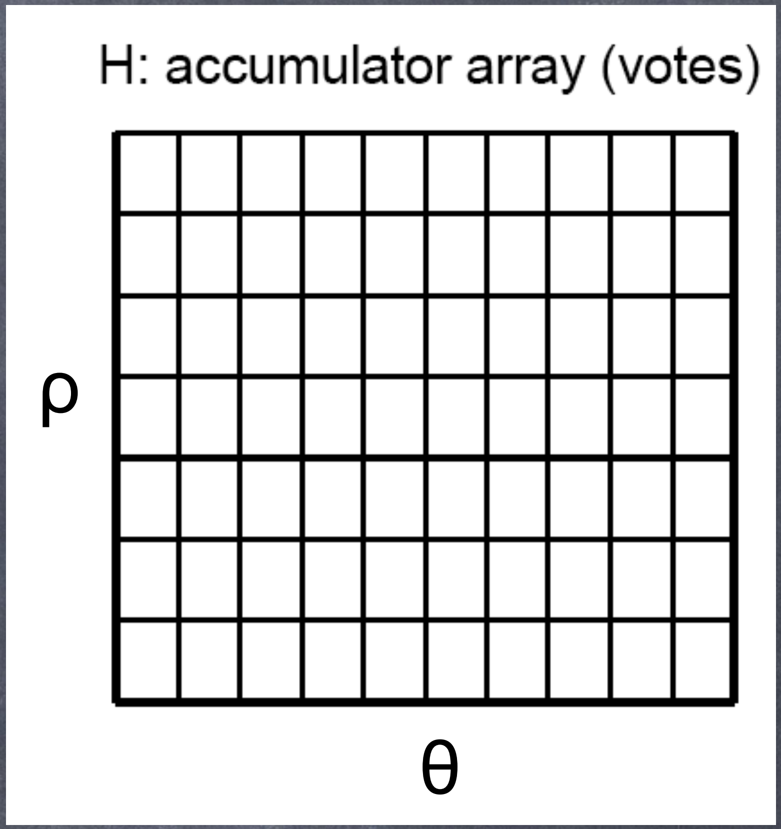
Parameter space representation

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



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

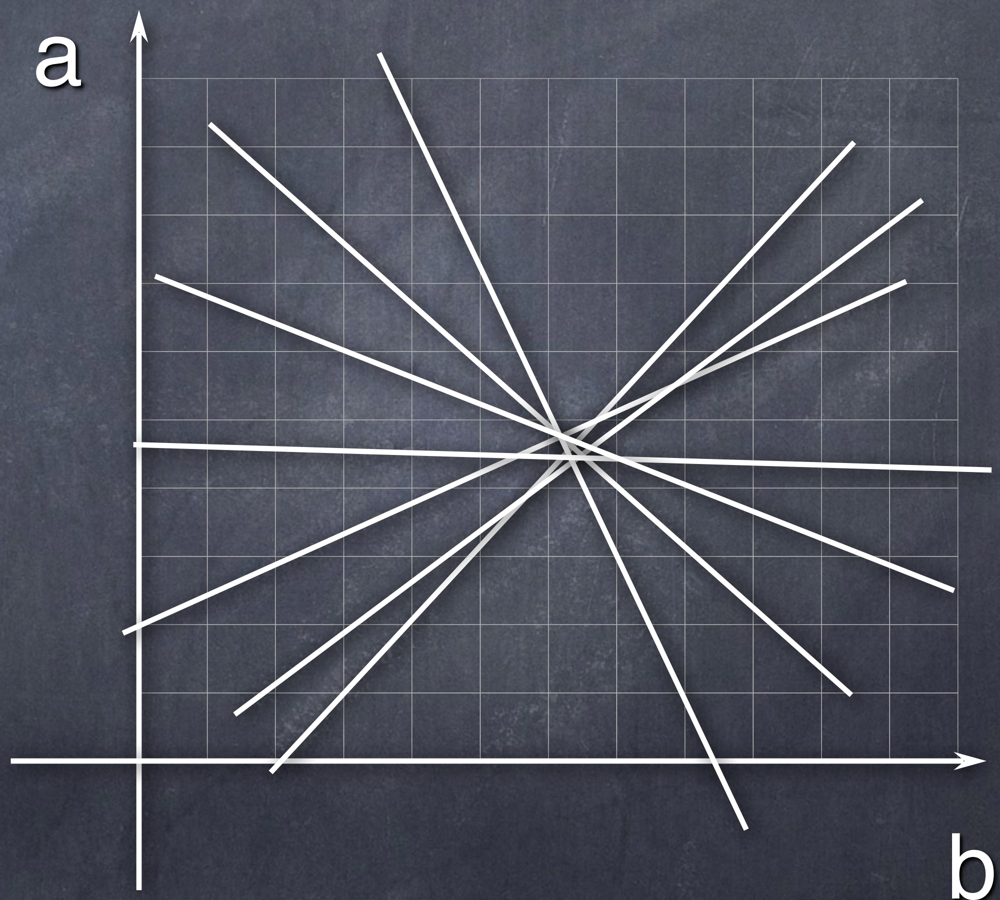
Algorithm outline



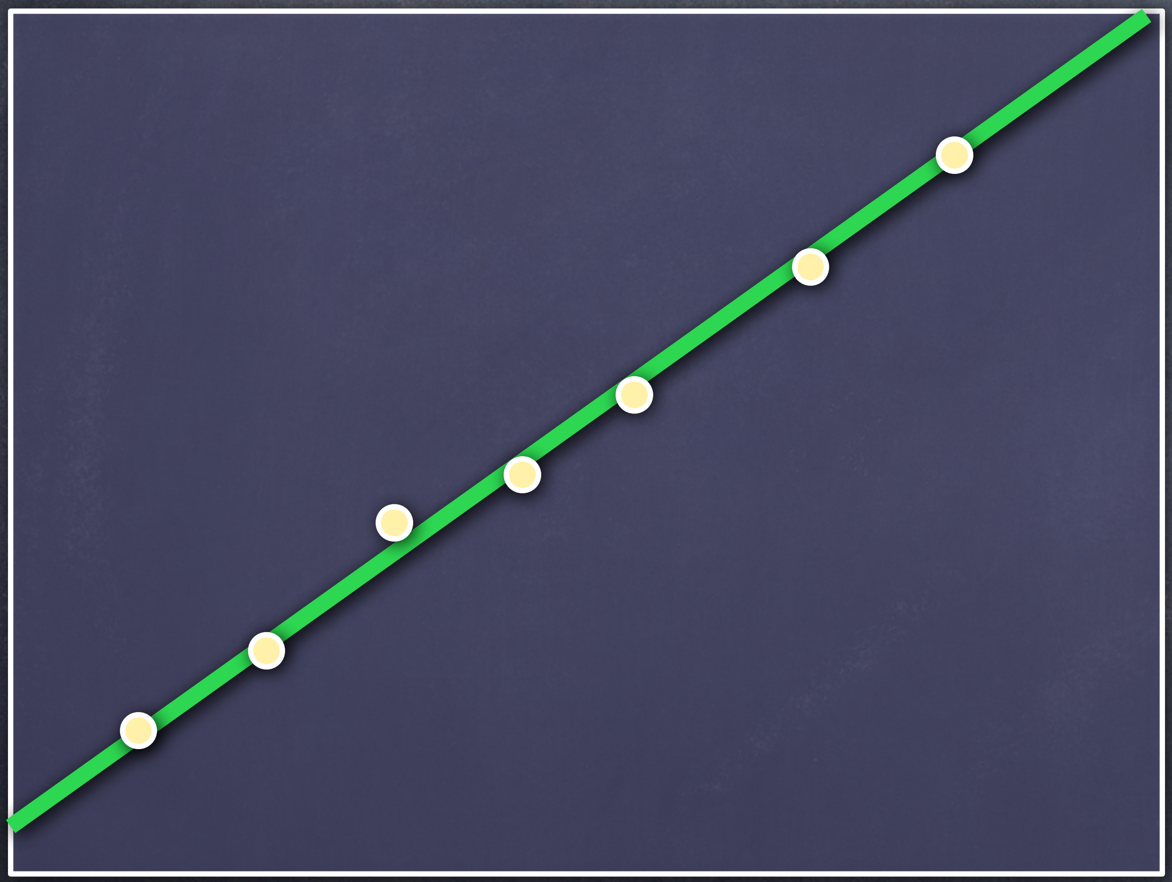
- Initialize accumulator H to all zeros
- For each edge point (x,y) in the image
  For θ = 0 to 180
      $\rho = x \cos\theta + y \sin\theta$
      $H(\theta, \rho) = H(\theta, \rho) + 1$
      end
  end
- Find the value(s) of (θ, ρ) where $H(\theta, \rho)$ is a local maximum
  - The detected line in the image is given by $\rho = x \cos\theta + y \sin\theta$

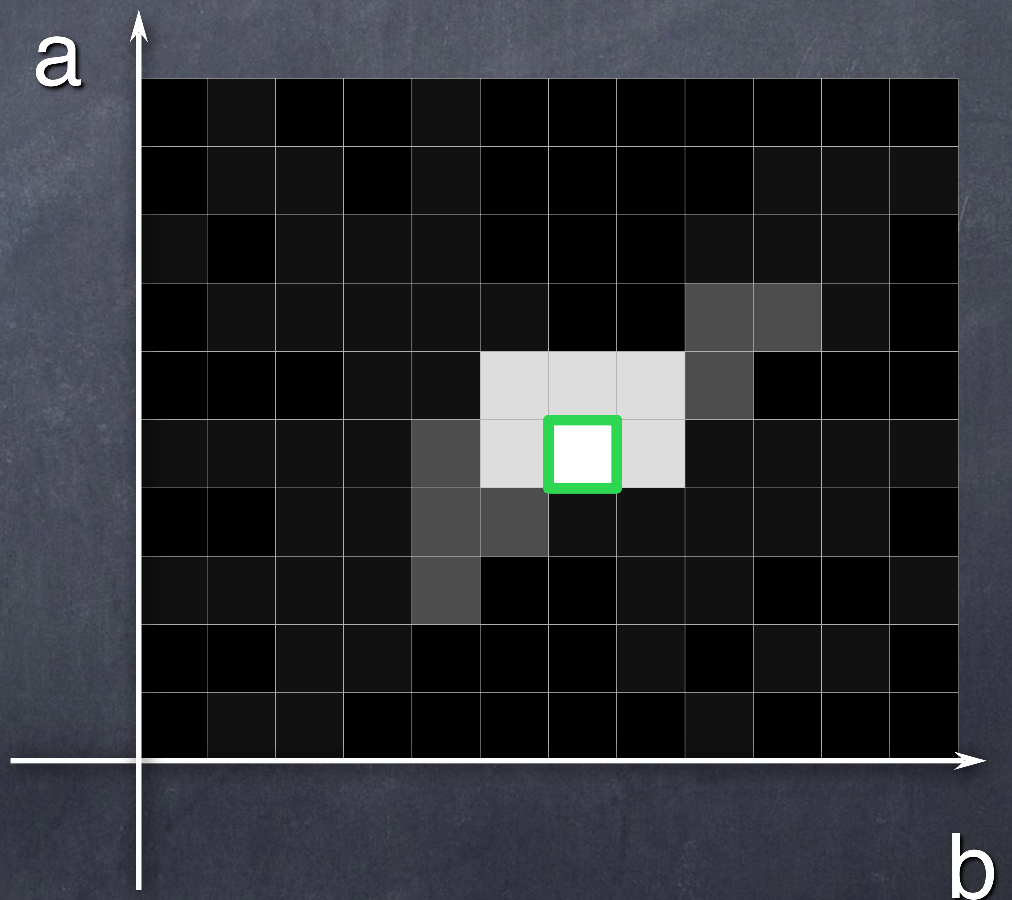
Can we further improve it? (hint: what is input to this algorithm?)

Hough Transform for Lines



Hough Transform for Lines



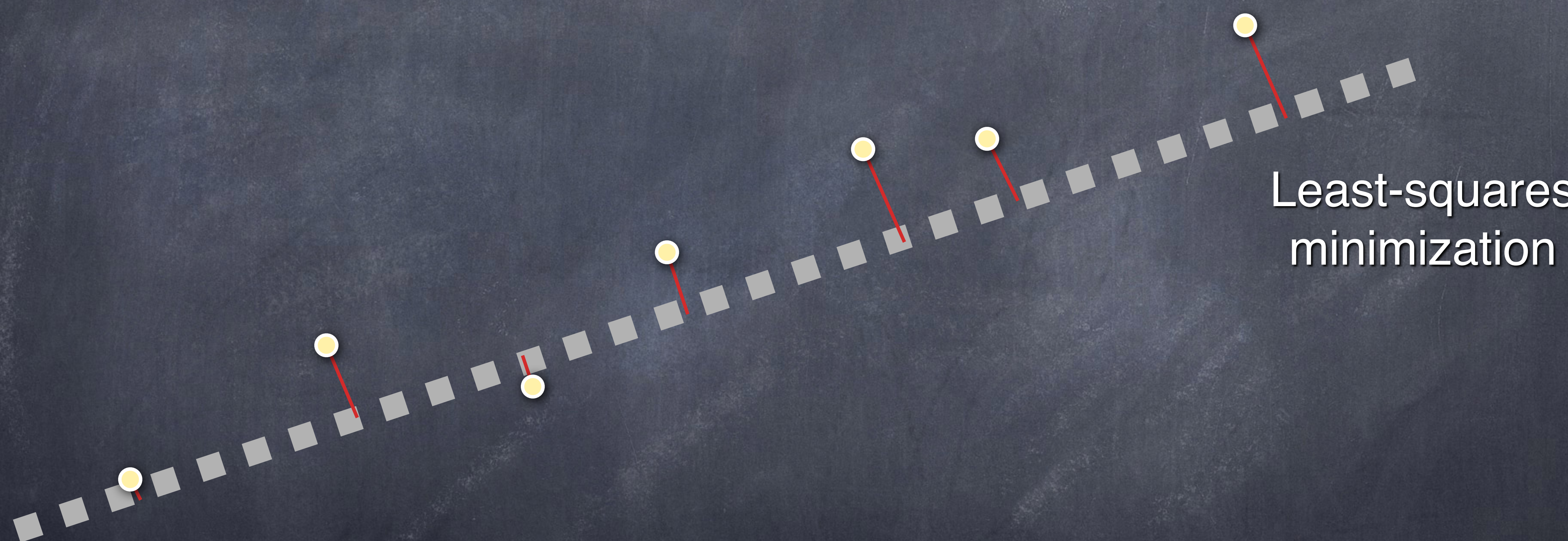


Fitting

- Output of Hough transform often not accurate enough
- Use as initial guess for fitting

Fitting Lines

Fitting Lines



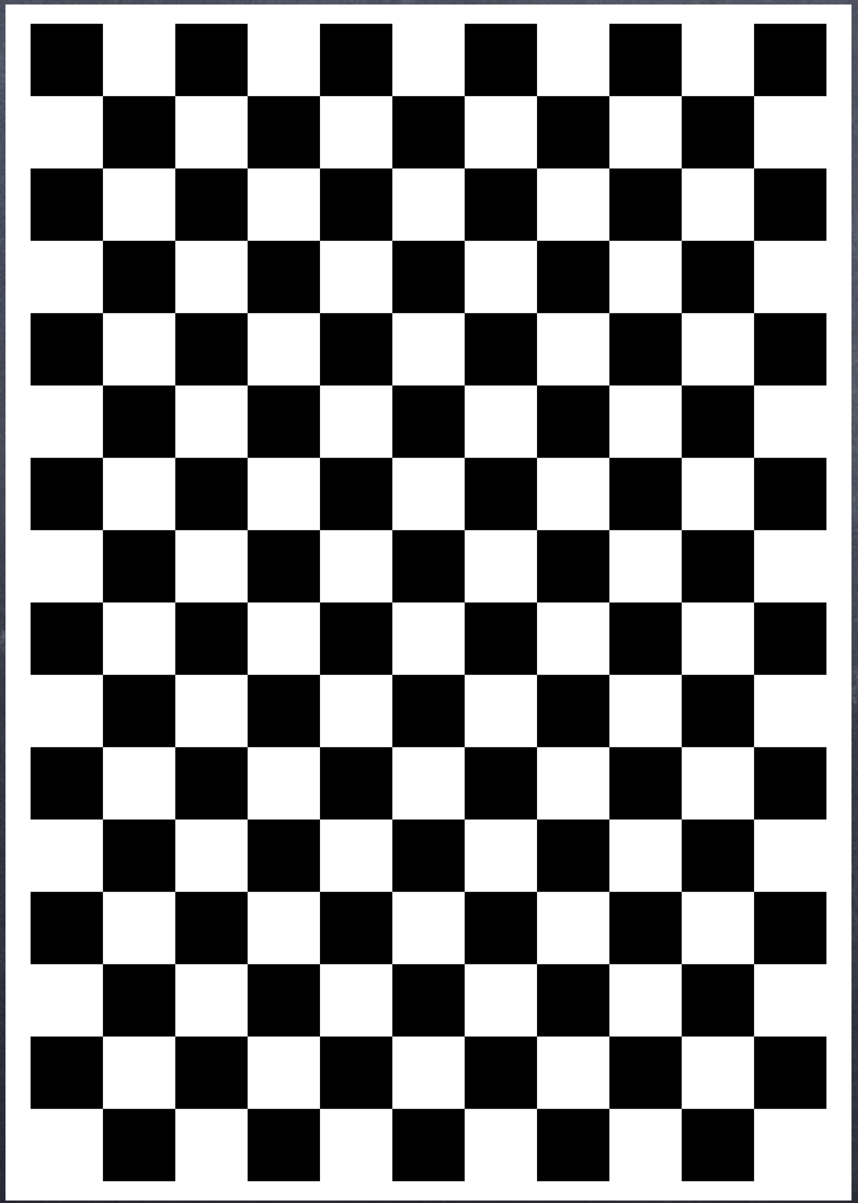
Fitting Lines

For mid-sem

- One of the potential questions for mid-sem exam is: how to use Hough Transform to detect circles?

Corner Detection

- How we detect corner (or key-points)?



Edges vs. Corners

- Edges = maxima in intensity gradient

Edges vs. Corners

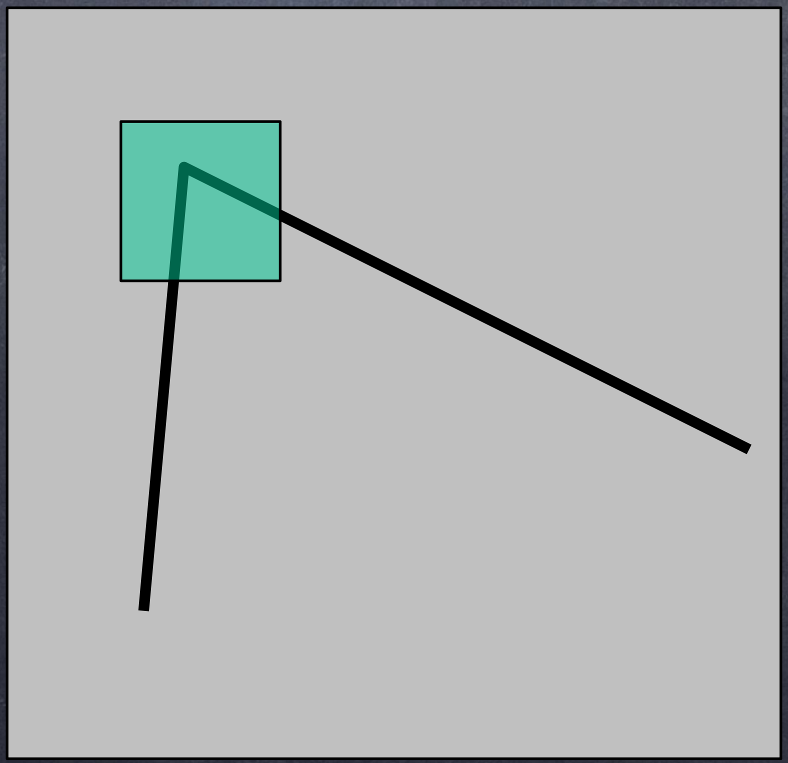
- Corners = lots of variation in direction of gradient in a small neighborhood

Detecting Corners

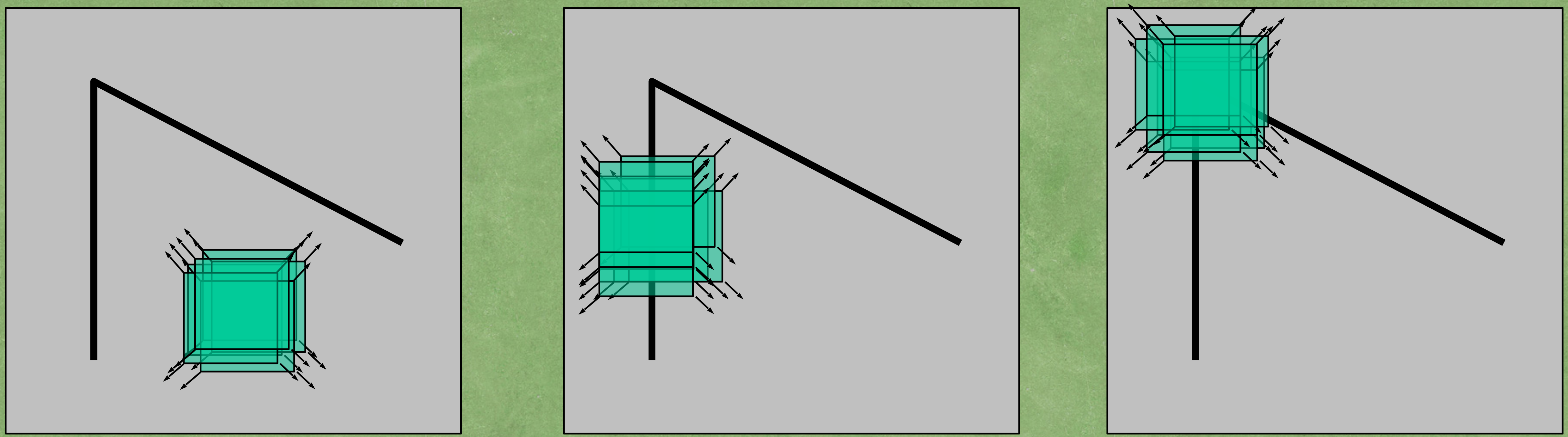
- How to detect this variation?
- Not enough to check average $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$

The Basic Idea

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



The Basic Idea



"flat" region:
no change as shift window in all directions

"edge":
no change as shift window along the edge direction

"corner":
significant change as shift window in all directions

Detecting Corners

- Claim: the following covariance matrix summarizes the statistics of the gradient

$$C = \begin{bmatrix} \sum f_x^{\,2} & \sum f_x f_y \\ \sum f_x f_y & \sum f_y^{\,2} \end{bmatrix} \qquad f_x = \frac{\partial f}{\partial x},\ f_y = \frac{\partial f}{\partial y}$$

Summations over local neighborhoods

Detecting Corners

- Examine behavior of C by testing its effect in simple cases
- Case #1: Single edge in local neighborhood

Case#1: Single Edge

- Let (*a*,*b*) be gradient along edge
- Compute *C*· (*a*,*b*):

$$C \cdot \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum f_x^{\ 2} & \sum f_x f_y \\ \sum f_x f_y & \sum f_y^{\ 2} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

$$= \sum (\nabla f)(\nabla f)^{\mathrm{T}} \begin{bmatrix} a \\ b \end{bmatrix}$$

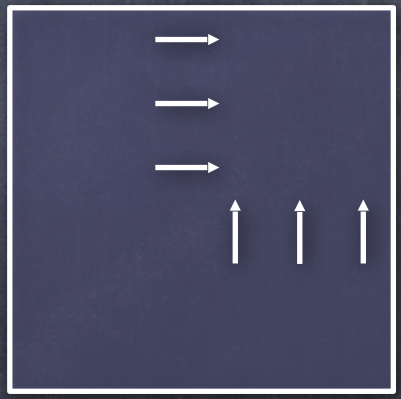
$$= \sum (\nabla f)\left(\nabla f \bullet \begin{bmatrix} a \\ b \end{bmatrix}\right)$$

Case #1: Single Edge

- However, in this simple case, the only nonzero terms are those where
- $\nabla f = (a,b)$
- So, $C \cdot (a,b)$ is just some multiple of (a,b)

Case #2: Corner

- Assume there is a corner, with perpendicular gradients (a,b) and (c,d)



Case #2: Corner

- What is $C \cdot (a,b)$?
  - Since $(a,b) \bullet (c,d) = 0$, the only nonzero terms are those where $\nabla f = (a,b)$
  - So, $C \cdot (a,b)$ is again just a multiple of (a,b)
- What is $C \cdot (c,d)$?
  - Since $(a,b) \bullet (c,d) = 0$, the only nonzero terms are those where $\nabla f = (c,d)$
  - So, $C \cdot (c,d)$ is a multiple of (c,d)

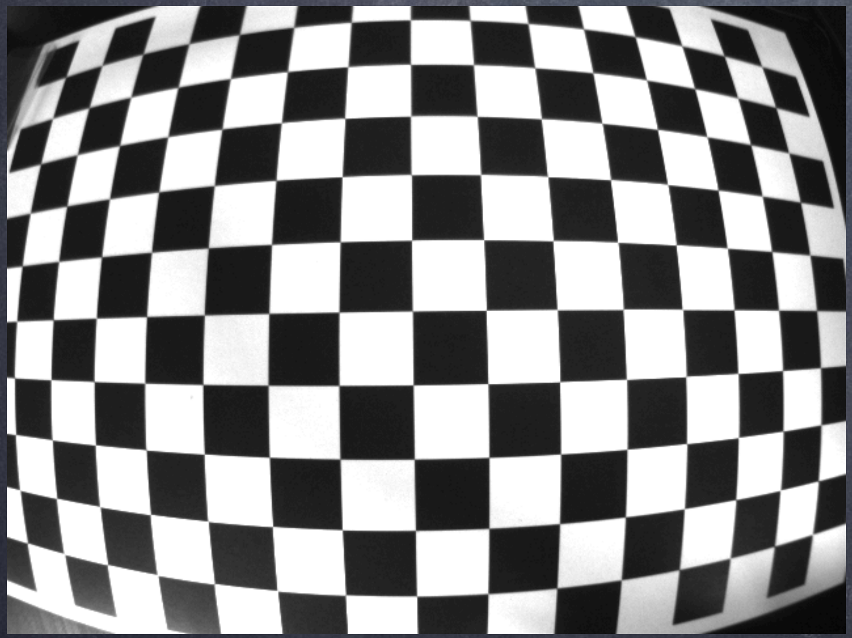
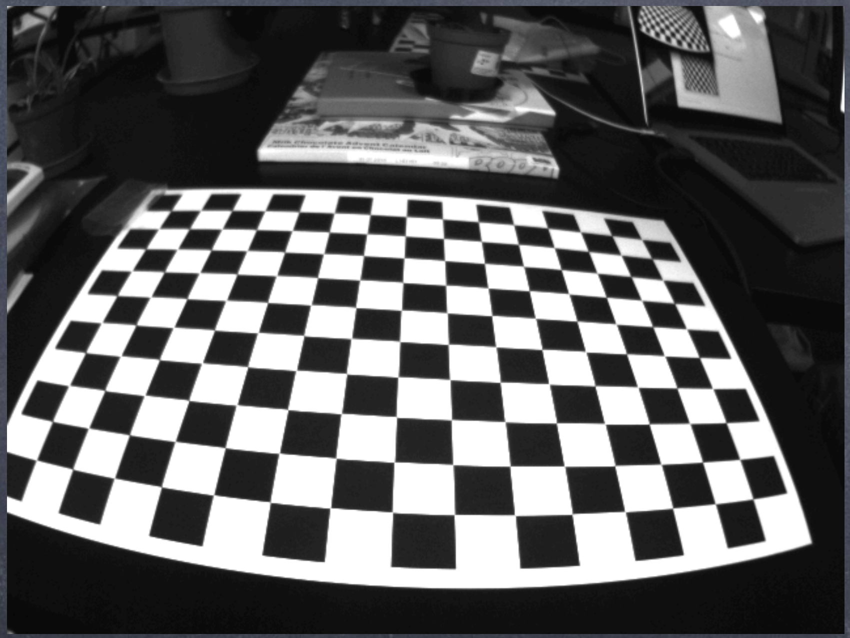
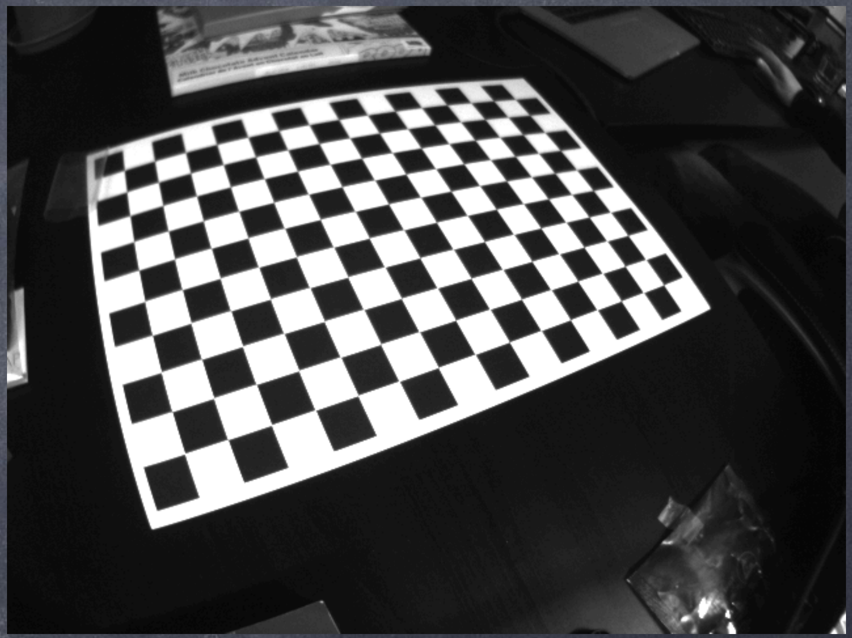
Corner Detection

- Eigenvectors and eigenvalues!
- In particular, if C has **one** large eigenvalue, there's an edge
- If C has **two** large eigenvalues, have corner

Corner Detection Implementation

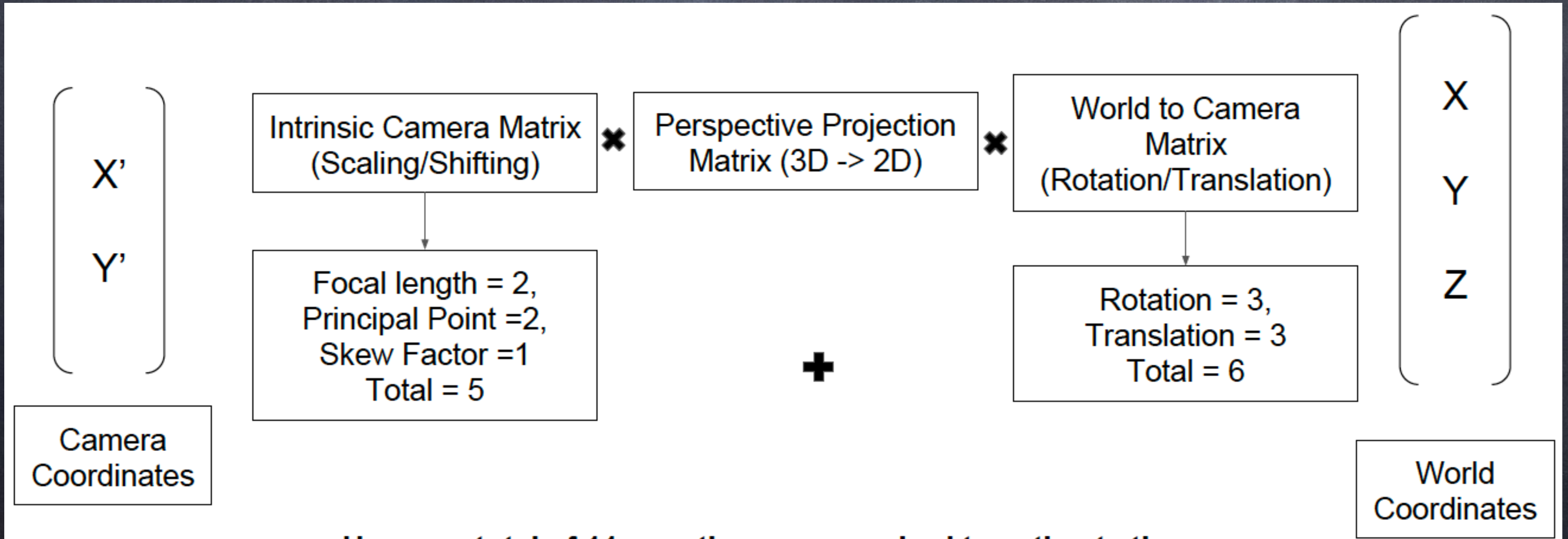
1. Compute image gradient
2. For each $m \times m$ neighborhood, compute matrix C
3. If both the eigenvalue is larger than threshold τ, record a corner
4. Nonmaximum suppression: only keep strongest corner in each $m \times m$ window

Coming back to Camera Calibration

Coming back to Camera Calibration

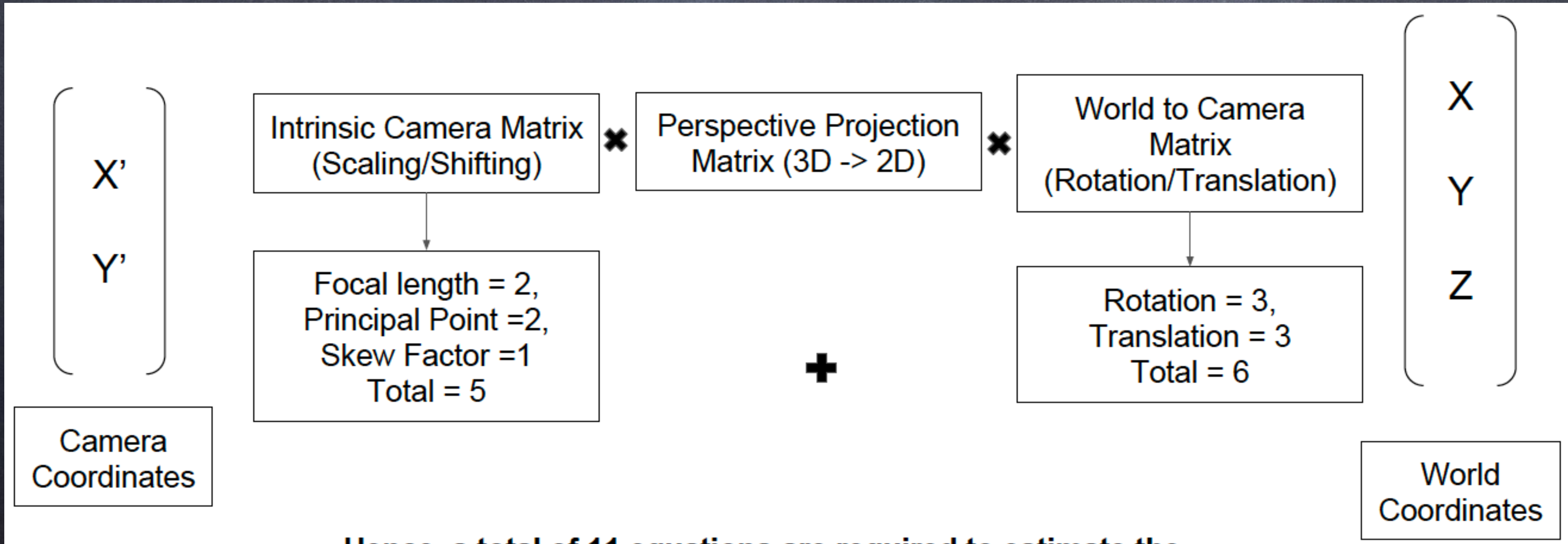
- Estimate the extrinsic and intrinsic camera parameters.



Hence, a total of 11 equations are required to estimate the parameters.

Coming back to Camera Calibration

- Using a set of known correspondences between point features in the world (X_w, Y_w, Z_w) and their projections on the image (x_{im}, y_{im})

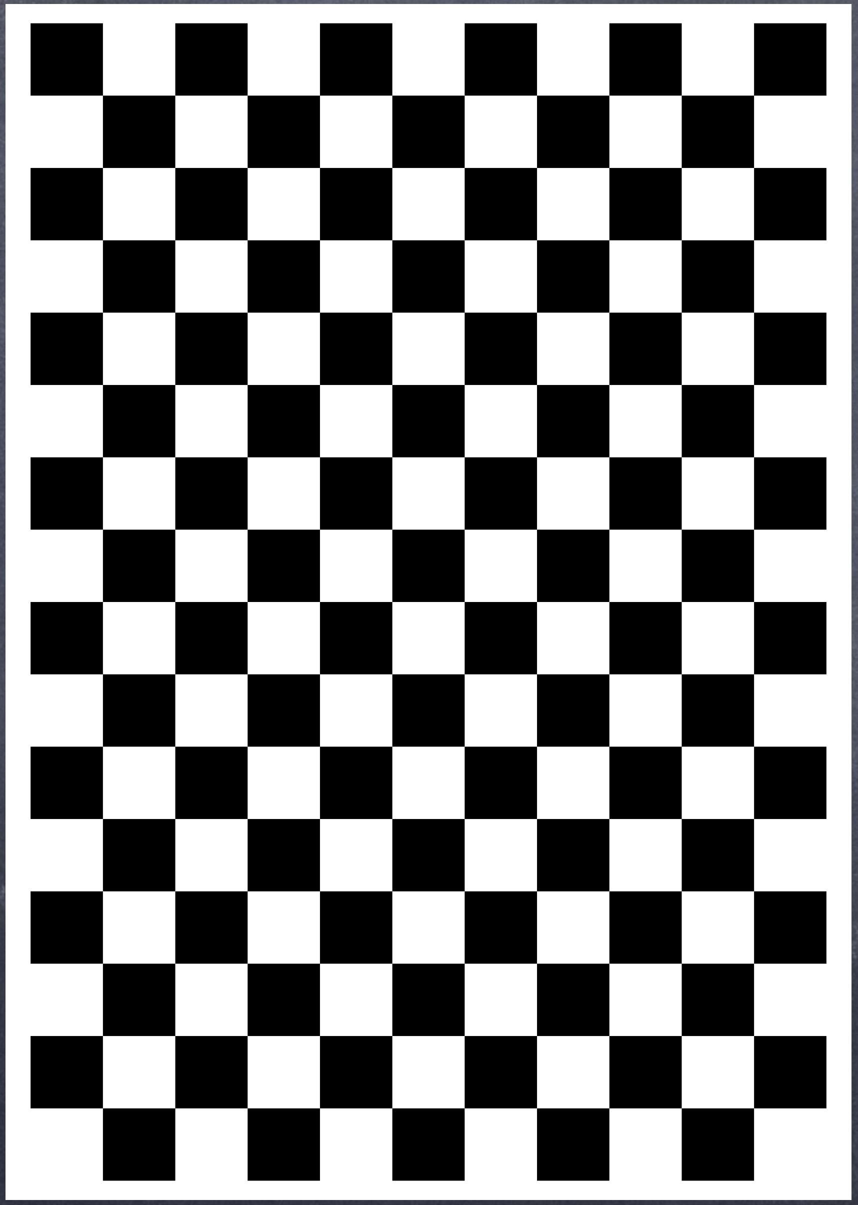


Hence, a total of 11 equations are required to estimate the parameters.

Camera Calibration

- Camera calibration requires two things: a physical calibration pattern and an algorithm which estimates the parameters

Pattern



Parameter Estimation

Indirect camera calibration

- Estimate the elements of the projection matrix.
- Compute the intrinsic/extrinsic camera parameters from the entries of the projection matrix.

$$M = M_{in}\ M_{ex} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}$$

$$\Downarrow$$

$$M = \begin{bmatrix} -f_x r_{11} + o_x r_{31} & -f_x r_{12} + o_x r_{32} & -f_x r_{13} + o_x r_{33} & -f_x T_x + o_x T_z \\ -f_y r_{21} + o_y r_{31} & -f_y r_{22} + o_y r_{32} & -f_y r_{23} + o_y r_{33} & -f_y T_y + o_y T_z \\ r_{31} & r_{32} & r_{33} & T_z \end{bmatrix}$$

Step 1: solve for m_{ij}'s

- M has 11 independent entries.
  - e.g., divide every entry by m_{11}

$$M = M_{in}\ M_{ex} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}$$

- Need at least 11 equations for computing M.
- Need at least 6 world-image point correspondences.

Next lecture

- Camera Calibration Contd.