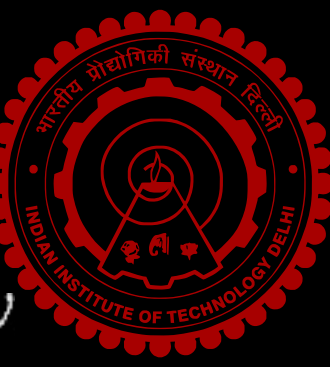
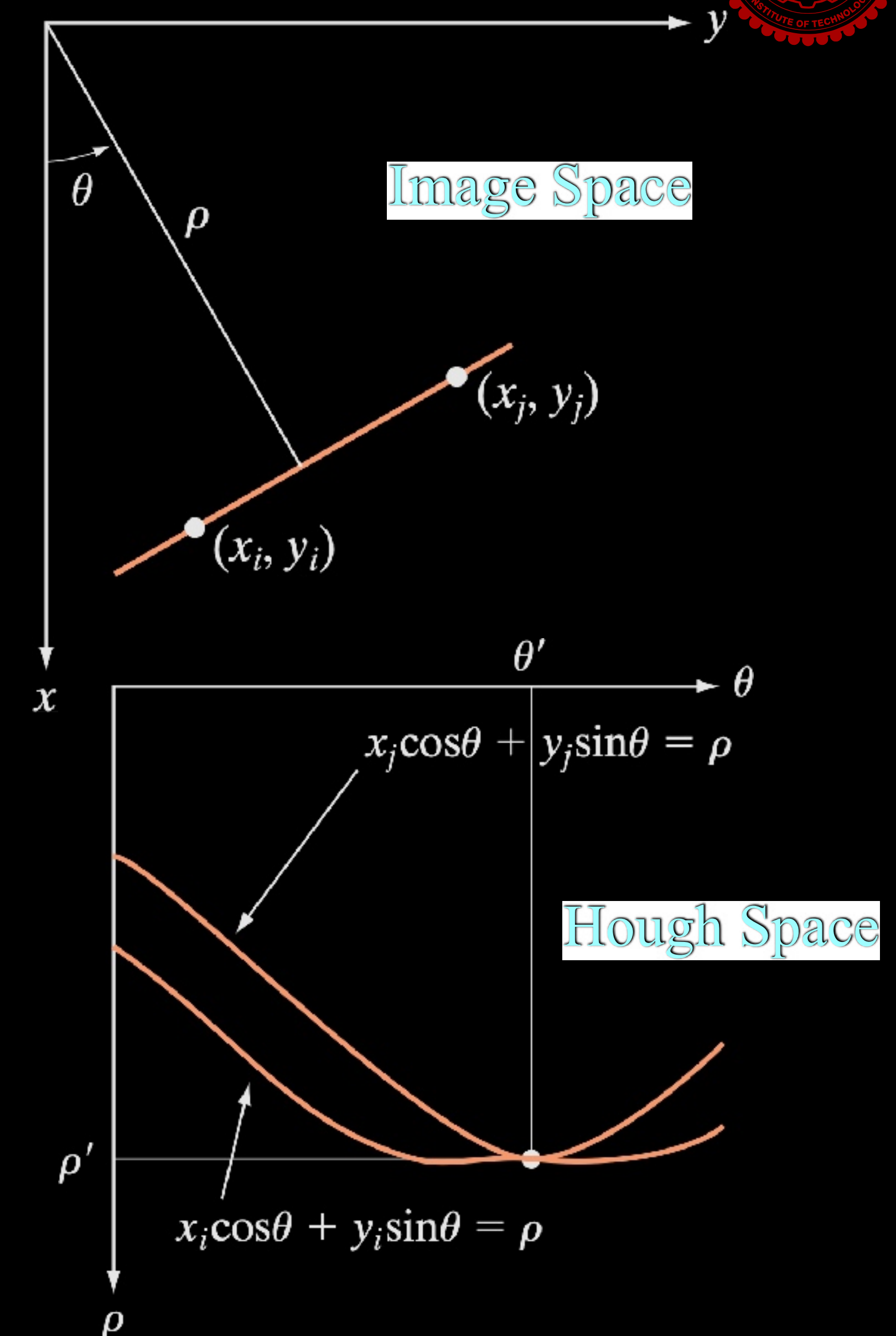


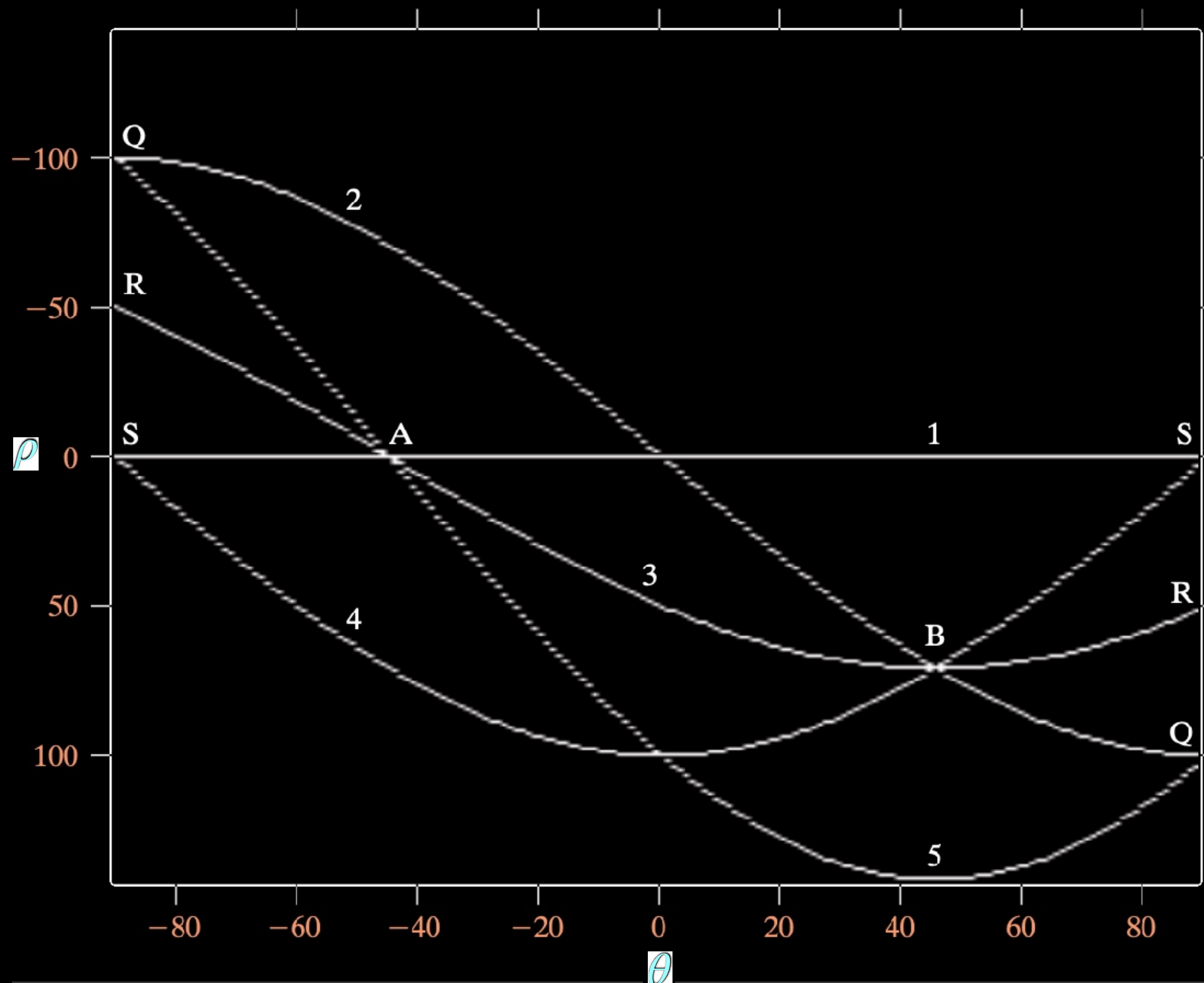
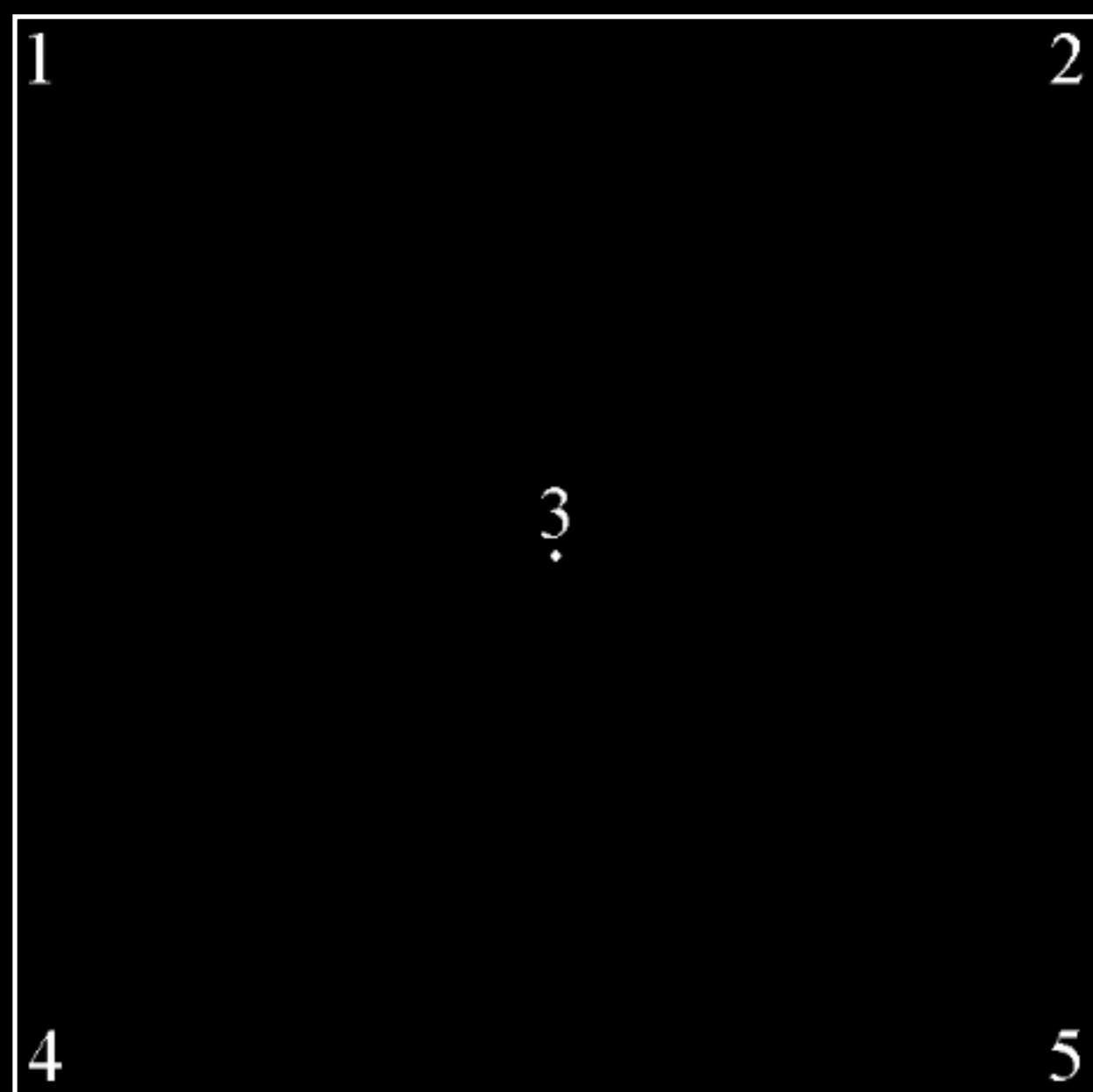
Hough Transform

- Find straight lines passing through many “edge” pixels
- Recall *normal representation* of line

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

- Consider parameter (aka dual) space: (θ, ρ)
 - Line in image space $\rightarrow$ point in parameter space
 - Point in image space $\rightarrow$ curve in parameter space
 - Same line through many image points
 $\rightarrow$ Hough space point where many curves meet





Hough Transform

Implementation

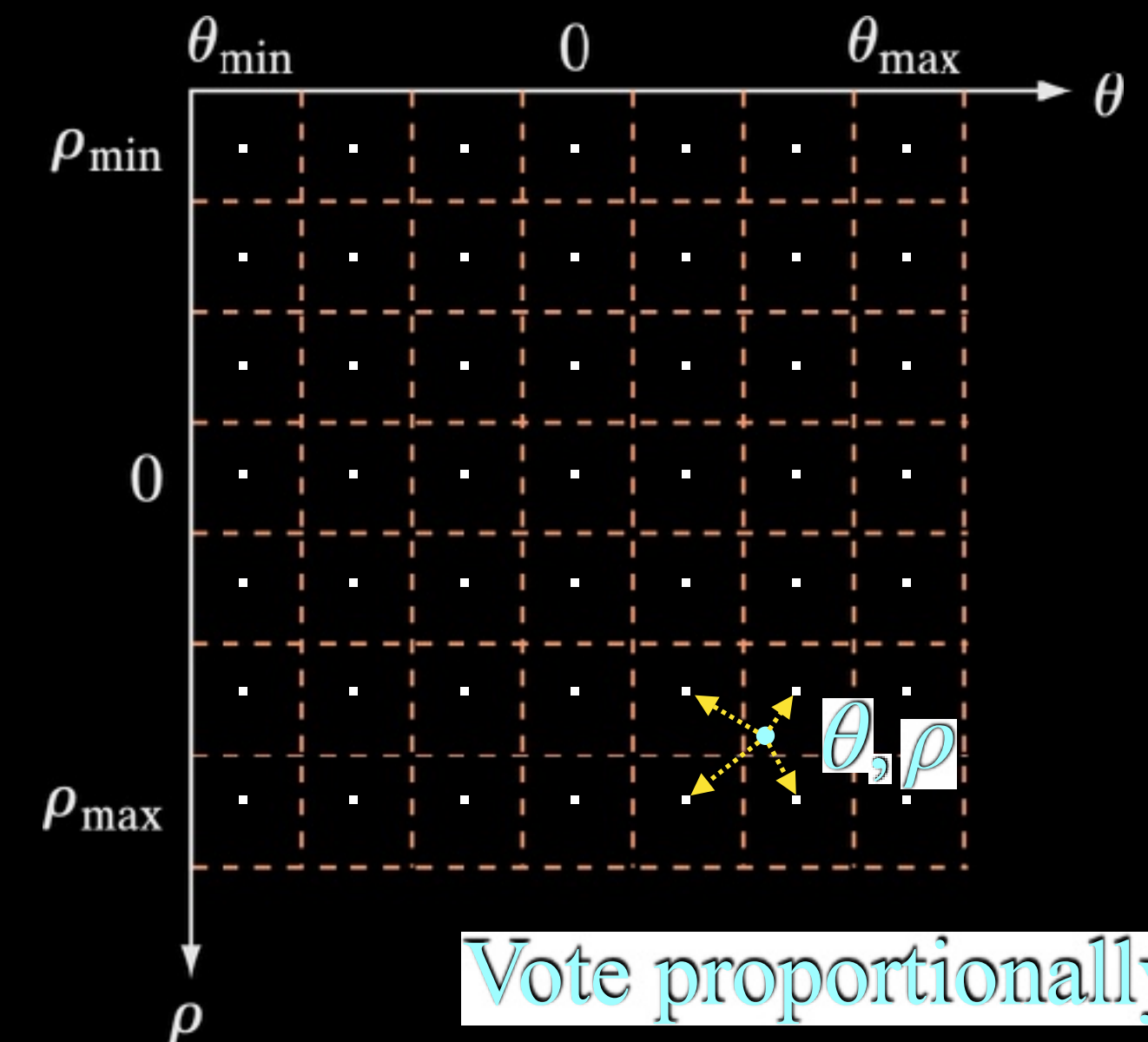
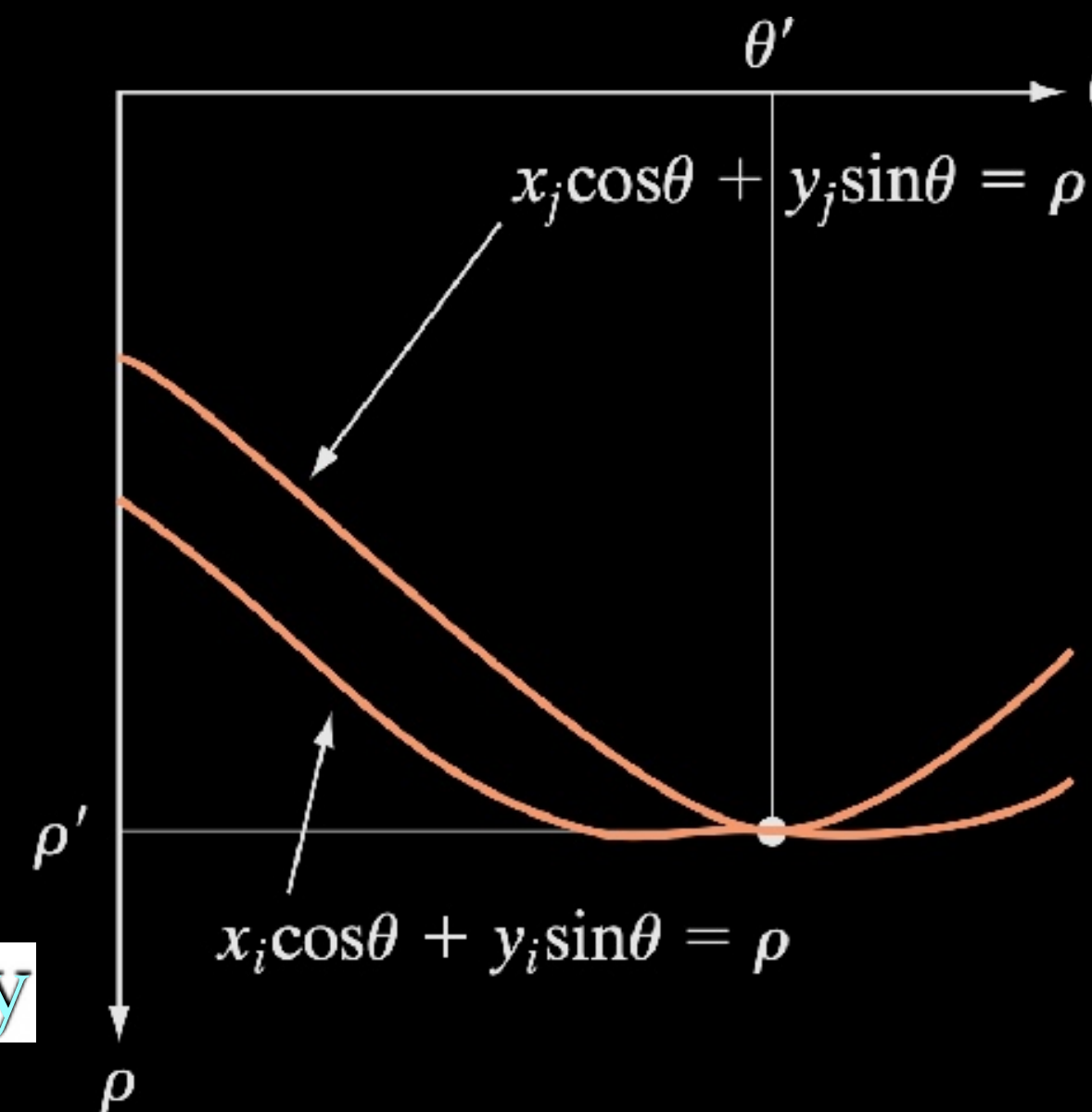
- Discretize (ρ, θ) into bins
- Count number of edge pixels contributing to each bin

Informed by gradient direction: both-way

for **each** edge pixel (x, y) :
for each angle θ :

increment count in bin (ρ, θ) , where $\rho = x \cos \theta + y \sin \theta$

- Bins with the highest local votes (local peaks) represent the actual lines
 - Still useful: vote thresholding, non-maxima suppression
 - More efficient: Progressive Probabilistic Hough Transform (PPHT)



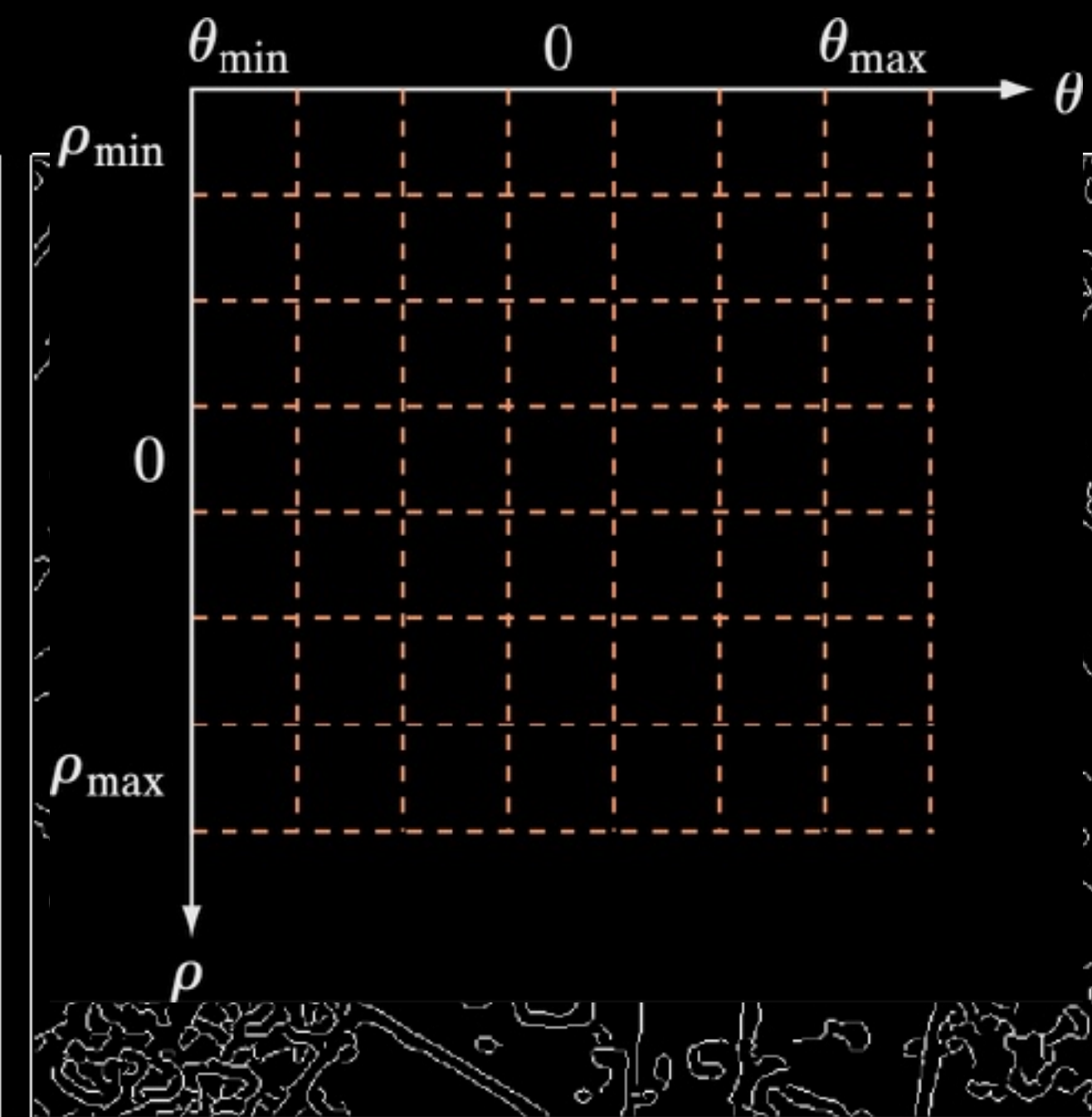
Vote proportionally

$$\frac{\rho - \rho_i}{\rho_{i+1} - \rho_i}, \frac{\rho_{i+1} - \rho}{\rho_{i+1} - \rho_i}$$

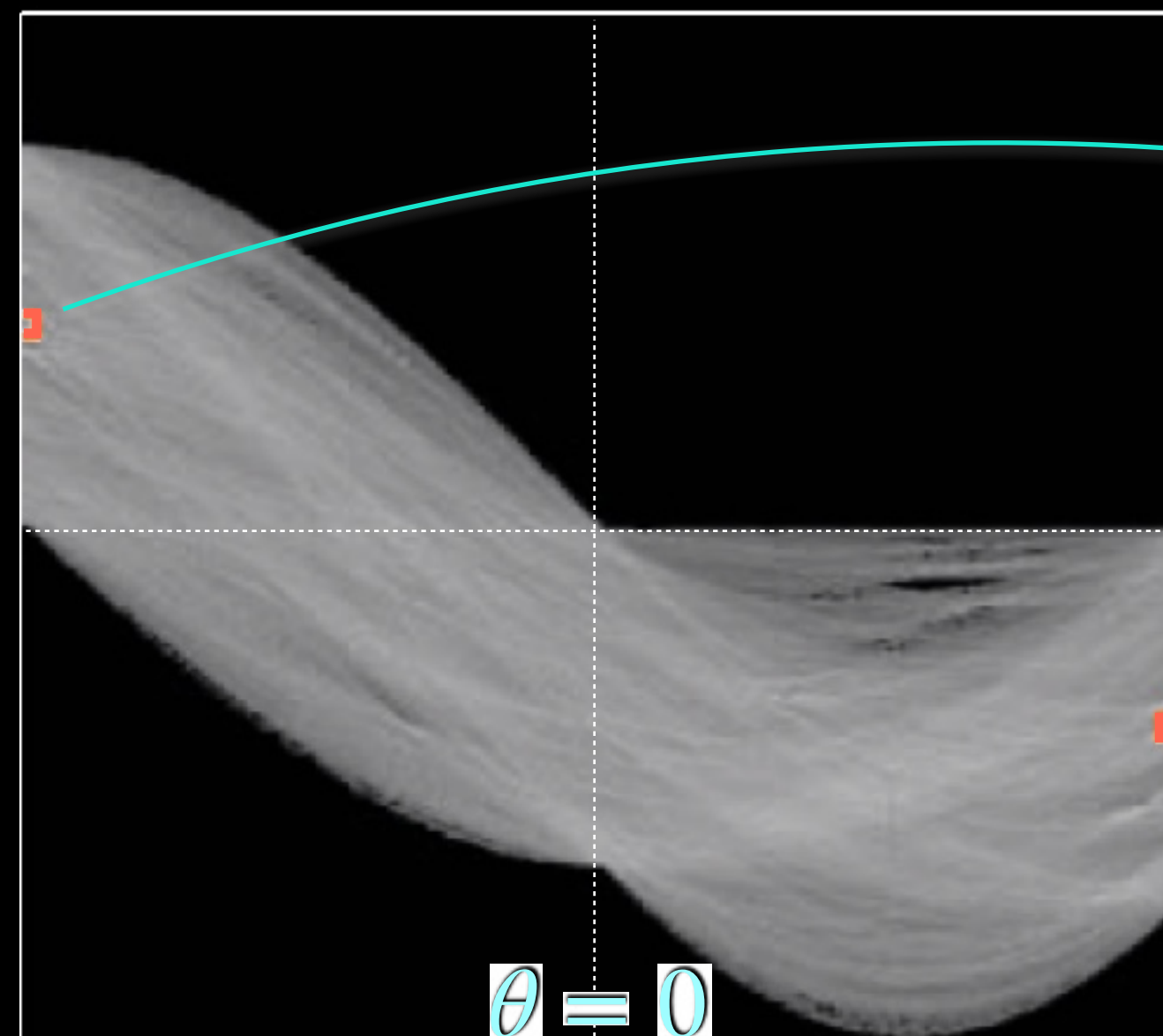
How to compute

true (θ, ρ) with max votes from samples

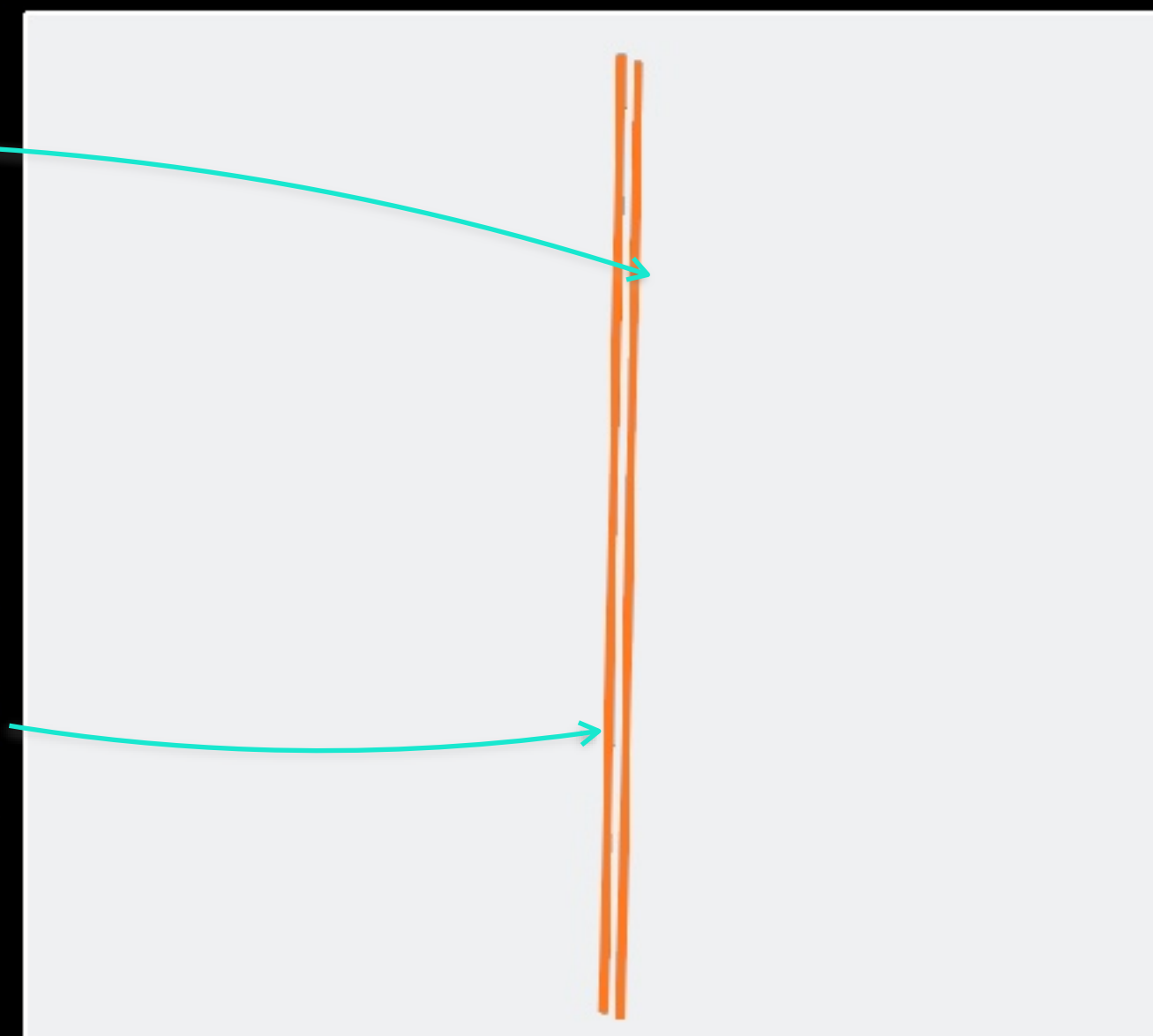
Input Image



$\rho = 0$



Hough Space



Corresponding Lines



Lines on the Image



PPHT: Progressive Probabilistic Hough Transform

- Random sampling: Select a random pixel from the *edge pool*
- Accumulator voting: Cast votes in the Hough-space accumulator array
- Early termination: If a bin reaches a set threshold, assume a line exists
 - Line completion (Connected components): Starting at pixels that voted for the line, “trace” along the line to neighbors: mark and connect “continuous” edge pixels
 - Pool shrinking: Remove marked pixels from *edge pool*
 - Progressive loop: Clear accumulator corresponding to the output line
- Iterate: Continue sampling until the edge pool is empty



Image Features

Edges and other general features

- Features are data that concisely say something about a given image
- Can help answer questions like:
 - Are two Images related?
 - Which parts of two images are related?
- Applications
 - Object/Face Matching, Abnormality detection, Image stitching, Detecting roads and lanes, obstacles, vehicles and pedestrians

Overlap



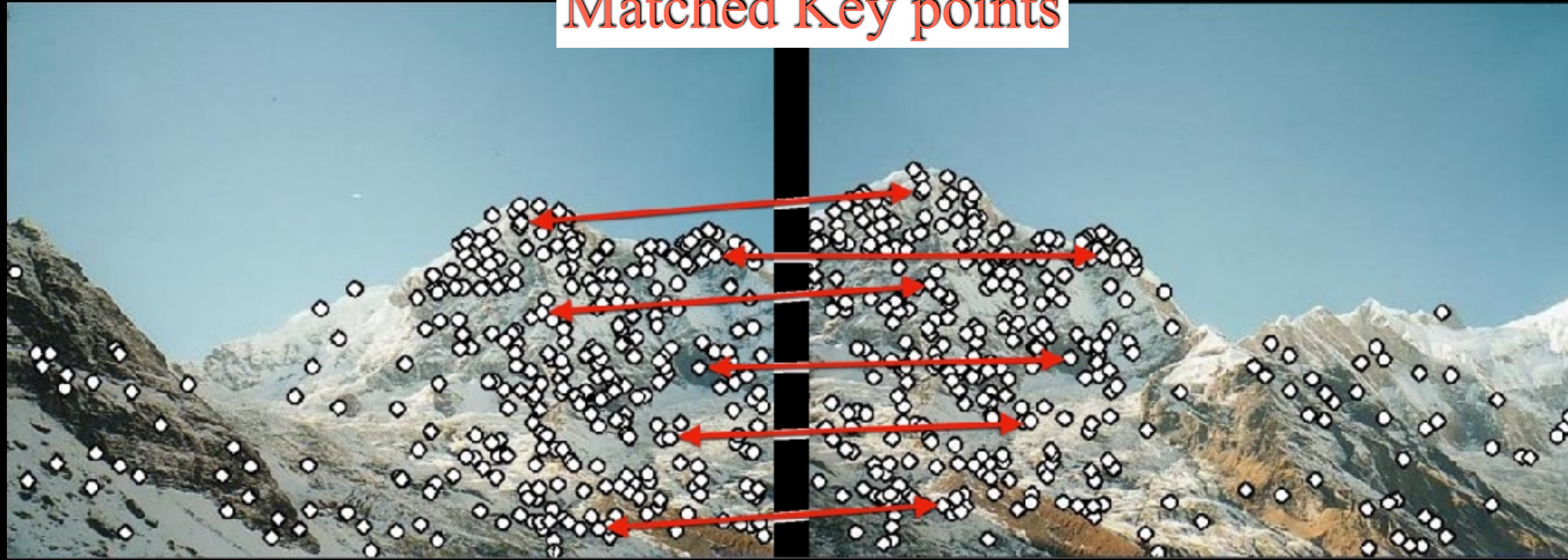
Merged



Similarity of two patches A and B :

$$\frac{1}{N} \sum \|A_i - B_i\|^2$$

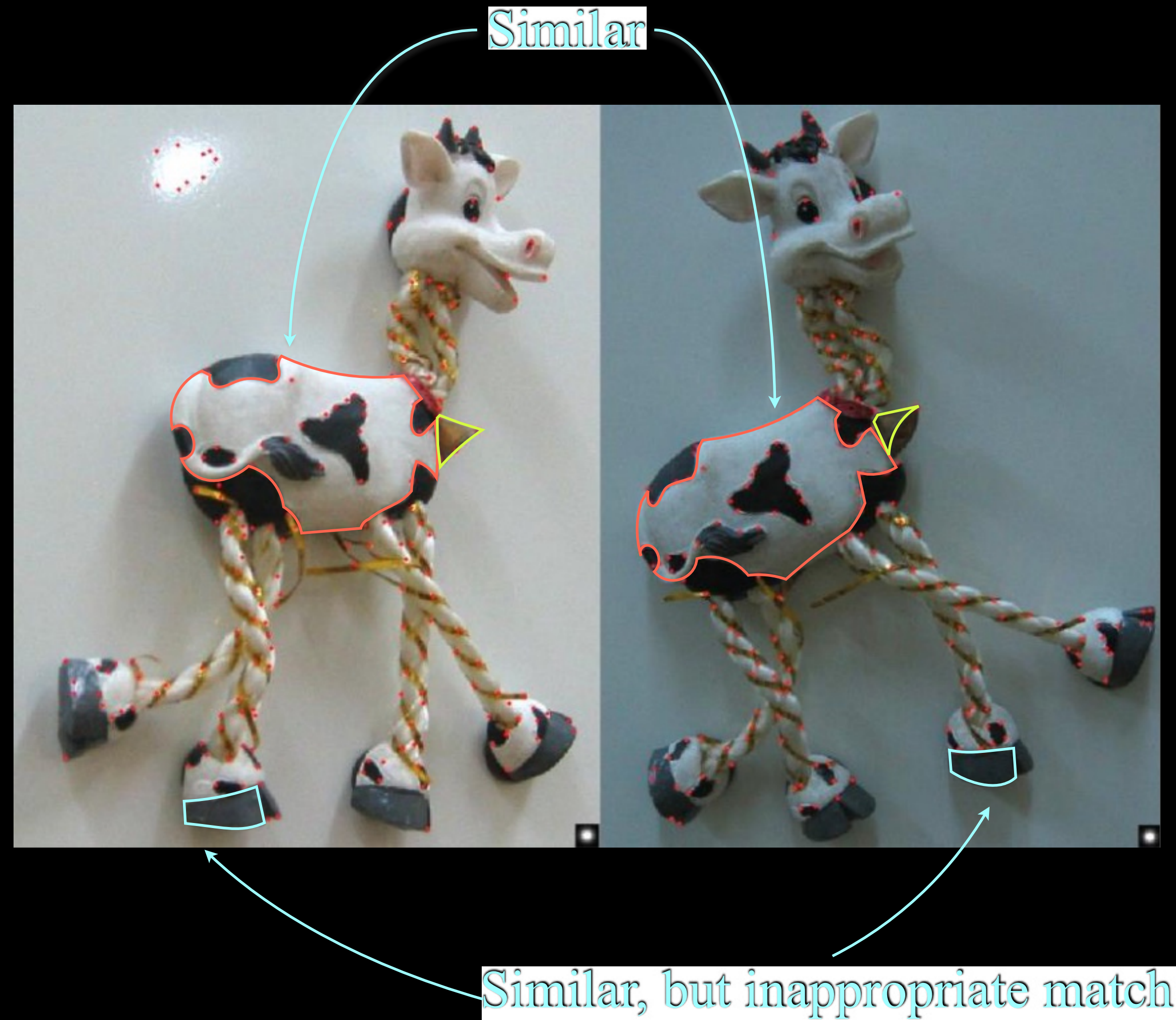
Matched Key points



Merged



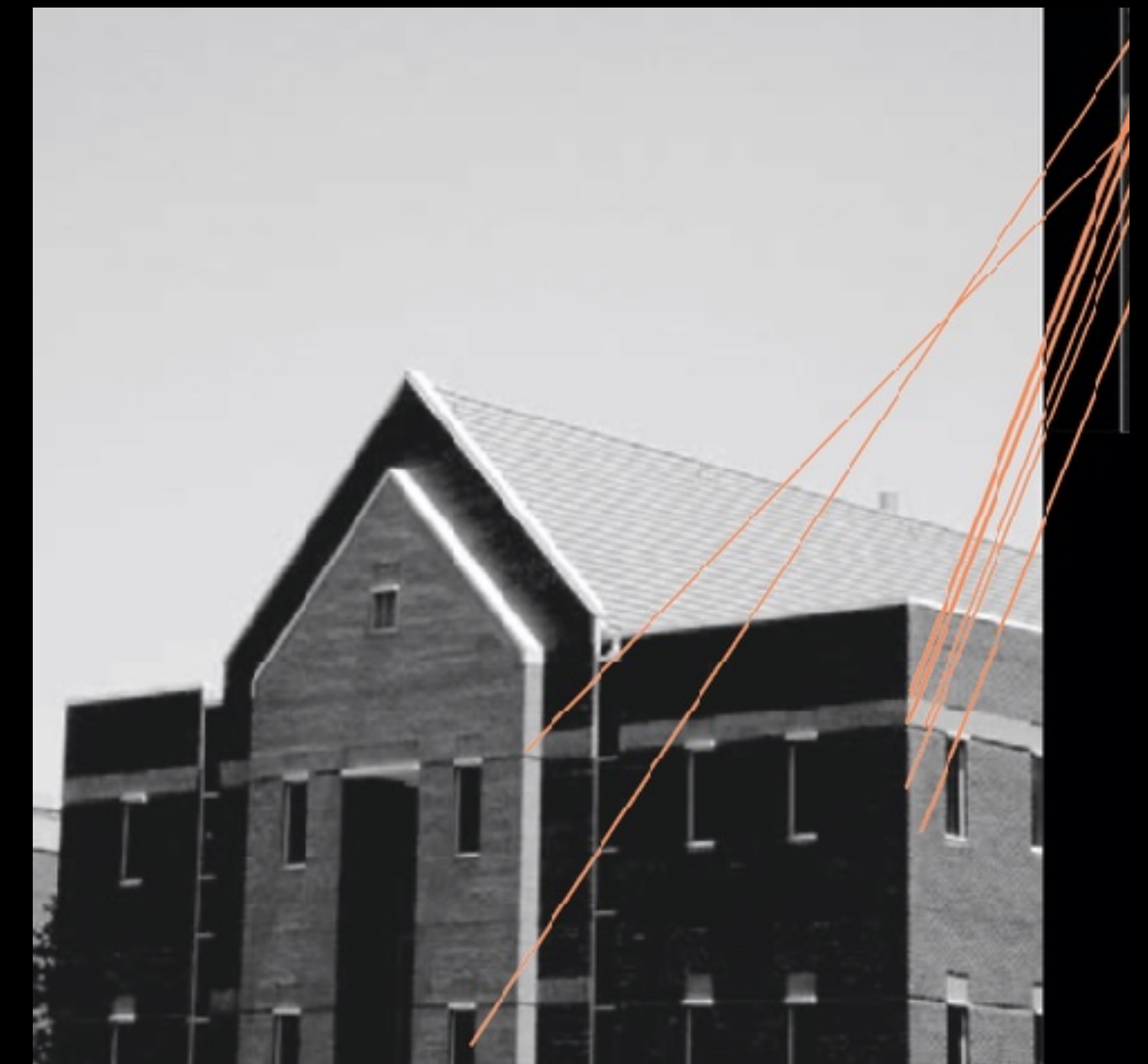
Brown & Lowe 2007

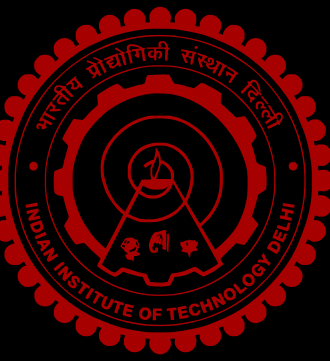


General Image Features

Data that concisely say something about the specific image

- Instead of color arrays, a list of features may describe the image more semantically
 - Region, corners, lines/circles/curves
 - Feature descriptor = array of values
- Can help with classification, recognition, matching
 - *e.g.*, we can divide image into parts and to match two parts, we may use a descriptor for the part or find key-points: 'distinguishing' features in the part





Features Variance

- Impact of transformation (*e.g.*, change of location/rotation/scale, lighting, viewpoint):
 - Feature is *invariant* if its values are unchanged on transformation
 - Feature is *covariant* if its values are transformed by a corresponding amount
- *e.g.*, perimeter of region is invariant to translation, and covariant to scaling
- For matching, we use key-points covariant to geometrical transformations, and invariant to illumination

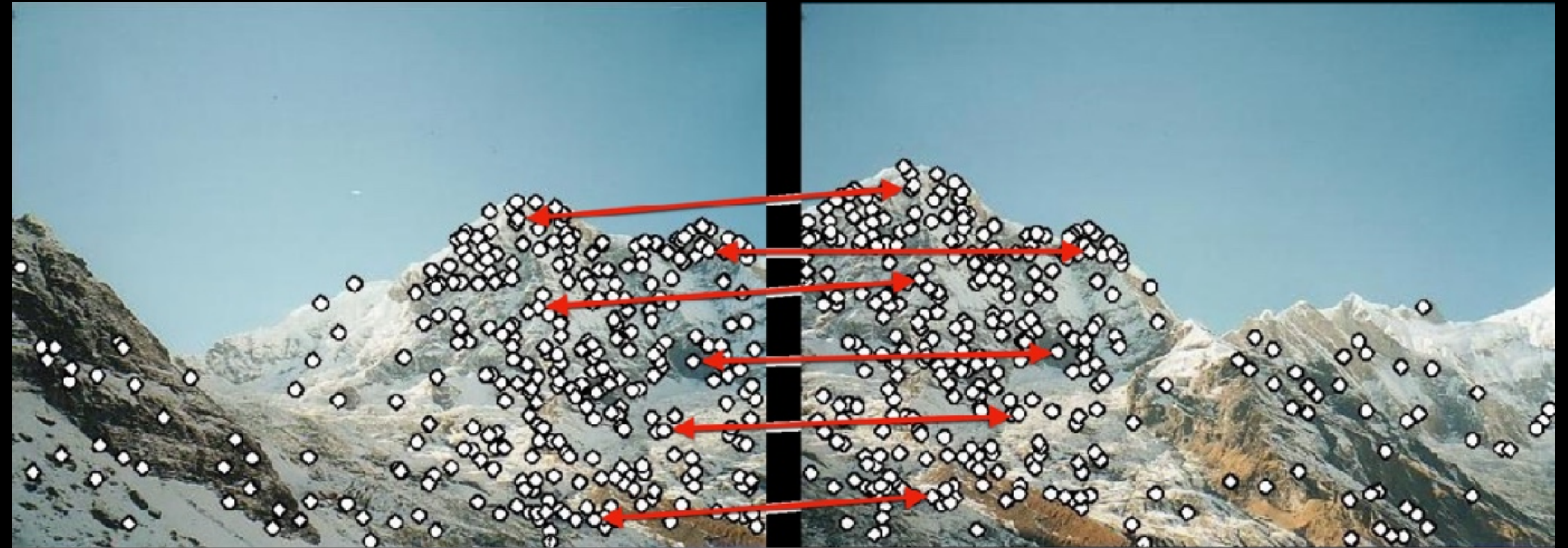
Features Variance

- Impact of transformation (*e.g.*, change of location/rotation/scale, lighting, viewpoint):
 - Feature is *invariant* if its values are unchanged on transformation
 - Feature is *covariant* if its values are transformed by a corresponding amount
- *e.g.*, perimeter of region is invariant to translation, and covariant to scaling
- For matching, we use key-points covariant to geometrical transformations, and invariant to illumination



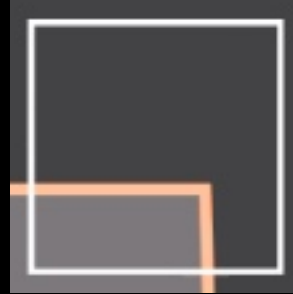
Key Point Detection

- Points that are
 - *repeatable* (same point can be found in different images)
 - *distinctive* (can be distinguished from other points in the image)
- Applications: image alignment, 3D reconstruction, object recognition, etc.



Good Feature Points

- Corners $\Rightarrow$ Gradient high in 'all' directions



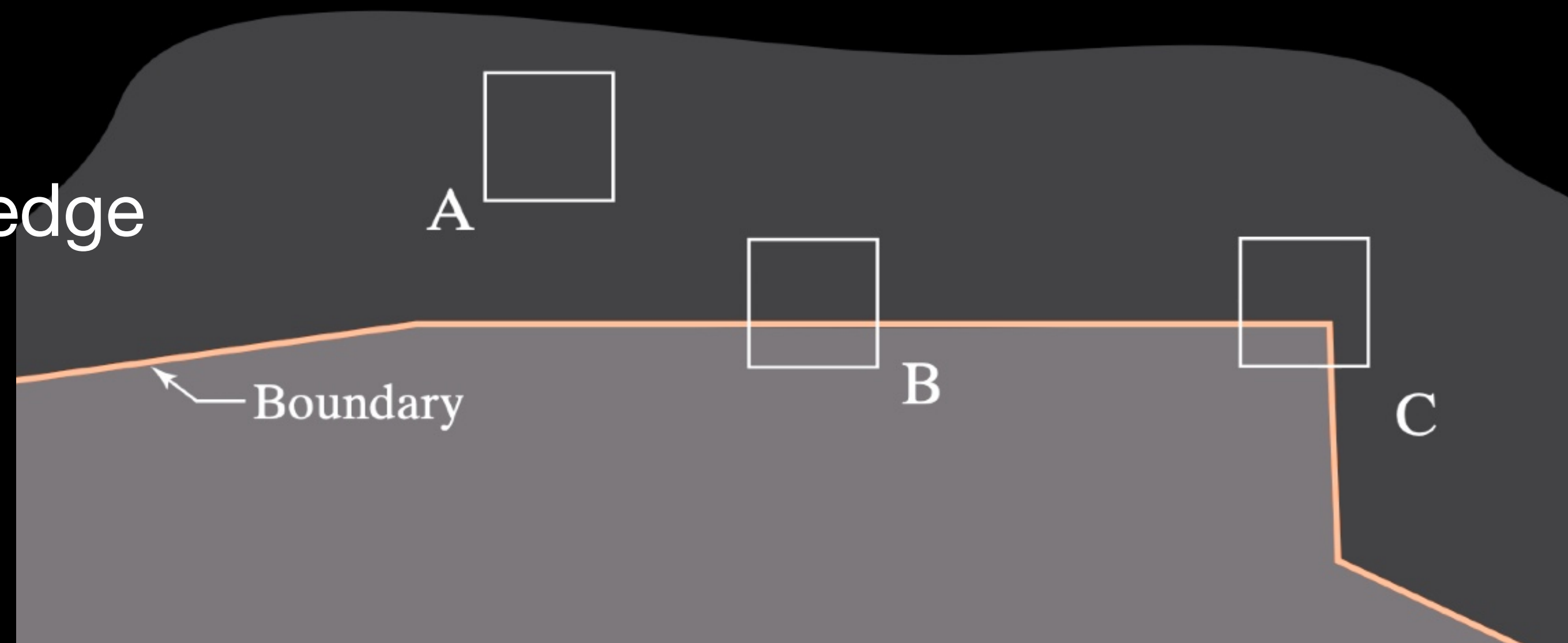
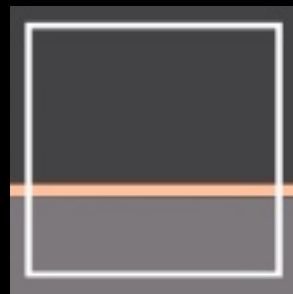
Easiest to identify in the image

- *i.e.*, significant change in pixel values

- Flat regions $\Rightarrow$ Gradient low in all directions



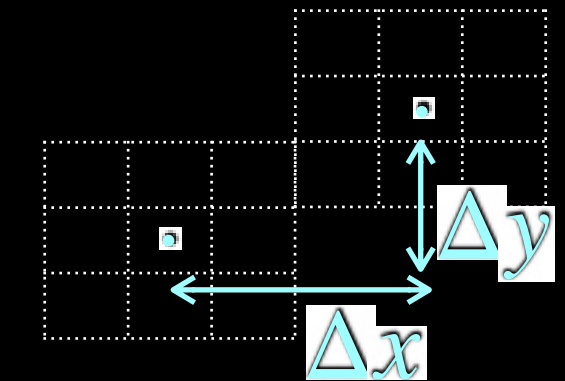
- Edges $\Rightarrow$ Gradient high across the edge



Distinctiveness of Point Features

Consider a patch of image centered at (x, y) , and a neighboring patch at $(x + \Delta x, y + \Delta y)$

Weighted sum of squared differences between two patches:



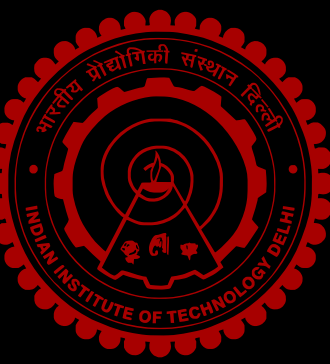
$$C(\Delta x, \Delta y) = \sum_{s,t \in \text{patch}} w(s, t) (c(s + \Delta x, t + \Delta y) - c(s, t))^2$$

$$\approx \sum_{s,t} w(s, t) (\nabla_x c(s, t) \Delta x + \nabla_y c(s, t) \Delta y)^2$$

$$= \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \underbrace{\left(\sum_{s,t} w(s, t) \begin{bmatrix} \nabla_x c(s, t)^2 & \nabla_x c(s, t) \nabla_y c(s, t) \\ \nabla_x c(s, t) \nabla_y c(s, t) & \nabla_y c(s, t)^2 \end{bmatrix} \right)}_{\mathbf{M}} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

Shift vector $\mathbf{v}$

structure tensor or **Harris matrix**



$$C(\mathbf{v}) \approx \mathbf{v}^T \mathbf{M} \mathbf{v}, \text{ where } \mathbf{M} = \sum_{s,t} w(s,t) \begin{bmatrix} \nabla_x c(s,t)^2 & \nabla_x c(s,t) \nabla_y c(s,t) \\ \nabla_x c(s,t) \nabla_y c(s,t) & \nabla_y c(s,t)^2 \end{bmatrix}$$
$$= \sum_{s,t} w(s,t) \nabla c(s,t) \nabla c(s,t)^T$$

We want $C(\mathbf{v}) = \mathbf{v}^T \mathbf{M} \mathbf{v}$ to be large for all choices of shift $\mathbf{v} = (\Delta x, \Delta y)$

Since $\mathbf{M}$ is a symmetric, positive definite matrix,

$$\lambda_{\min} \|\mathbf{v}\|^2 \leq \mathbf{v}^T \mathbf{M} \mathbf{v} \leq \lambda_{\max} \|\mathbf{v}\|^2$$

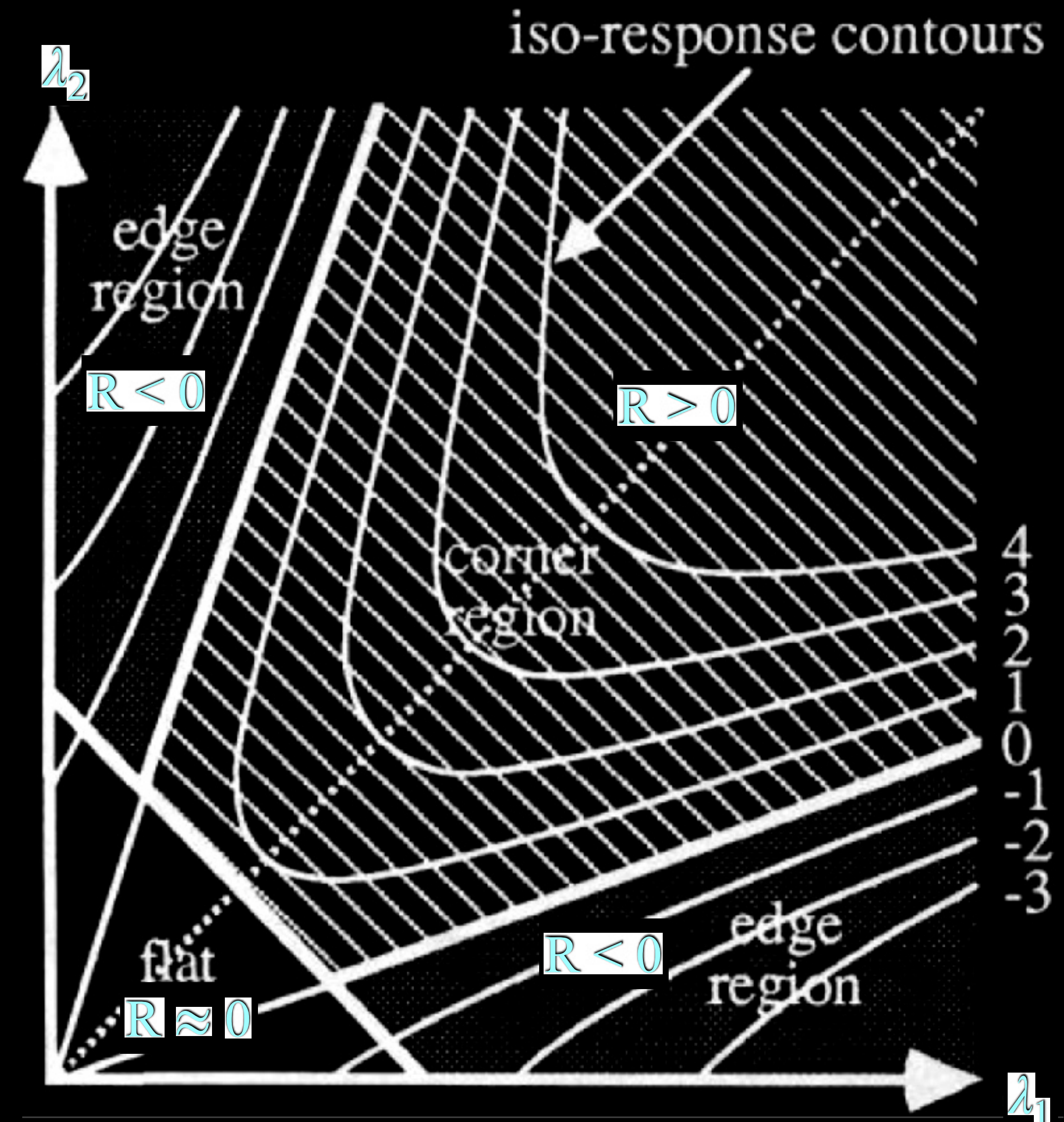
$\Rightarrow$ If all eigenvalues of $\mathbf{M}$ are large for an (x,y) , it is distinct from all neighbors

Corner Detector

- Large Eigenvalues of $\mathbf{M}$ indicate corner
 - **ST: Shi-Tomasi detector:** $\lambda_{\min}(\mathbf{M}) > \text{Th}$
 - **HS: Harris-Stephens detector:**

$$R = \det(\mathbf{M}) - \kappa \text{tr}(\mathbf{M})^2 > \text{Th}$$

(usually $\kappa \approx 0.04-0.06$)



Corner Detector

Invariant to rotation, translation, small luminosity changes.

Not scale invariant.



- Large Eigenvalues of $\mathbf{M}$ indicate corner

- ST: Shi-Tomasi detector:** $\lambda_{\min}(\mathbf{M}) > \text{Th}$

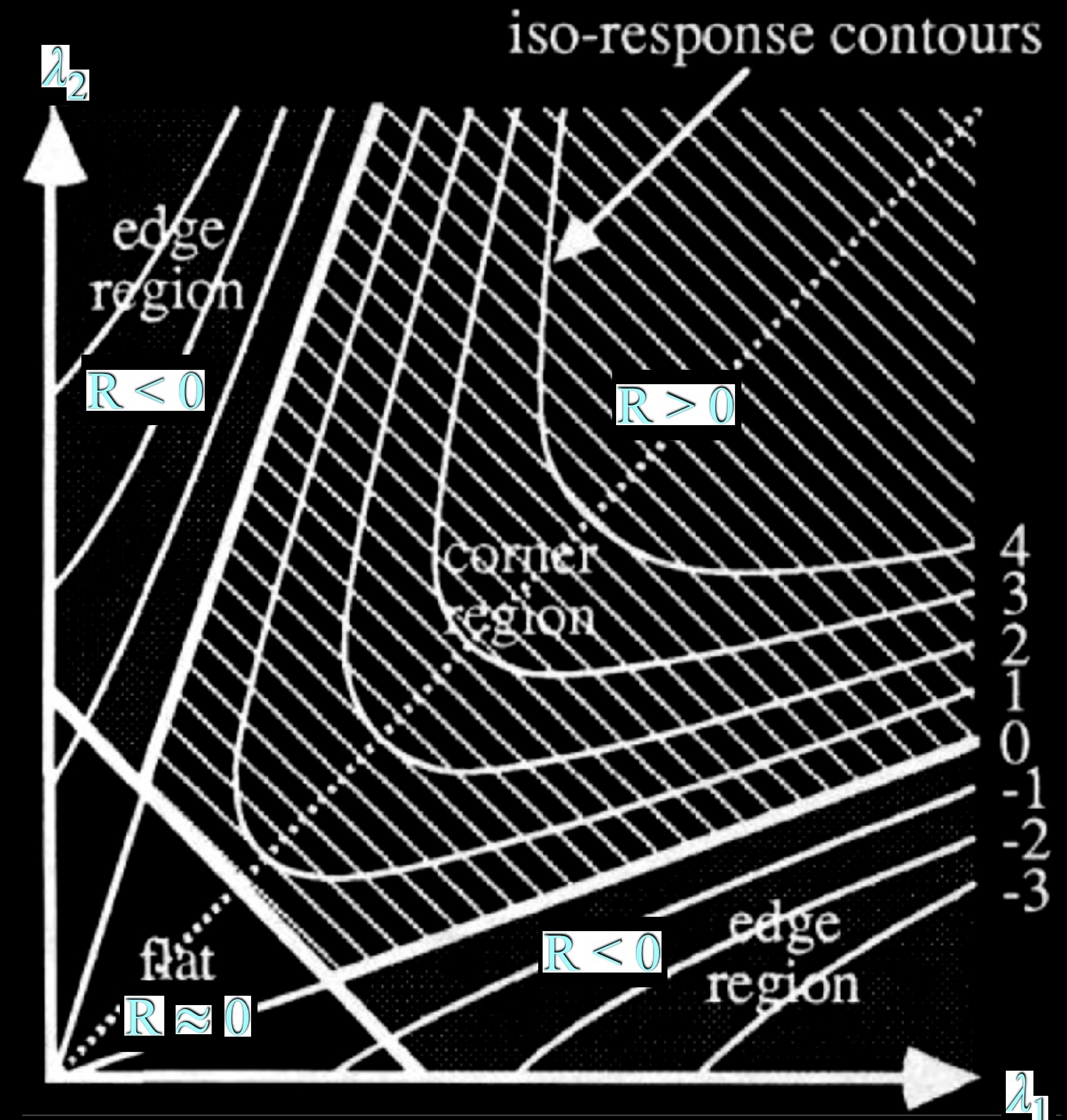
- HS: Harris-Stephens detector:**

$$\lambda_1 \lambda_2 \xleftarrow{\quad} R = \det(\mathbf{M}) - \kappa \text{tr}(\mathbf{M})^2 > \text{Th} \xrightarrow{\quad} \lambda_1^2 + \lambda_2^2$$

(usually $\kappa \approx 0.04-.06$)

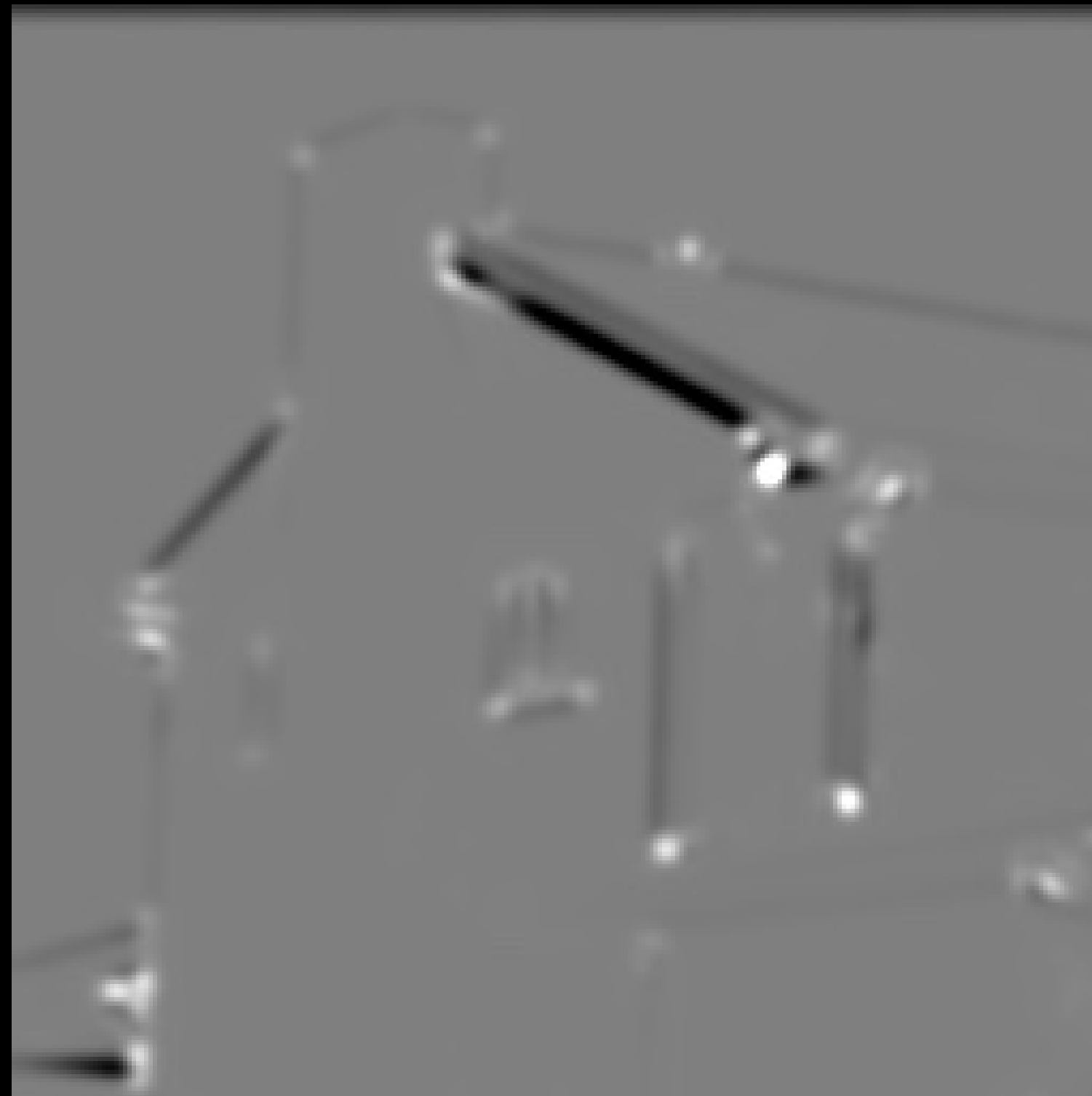
- Non-maxima suppression

- Pick corners as local maxima of “corneriness,”
i.e., HS or ST above some chosen threshold
 - Also, pre-blur to reduce corners due to noise



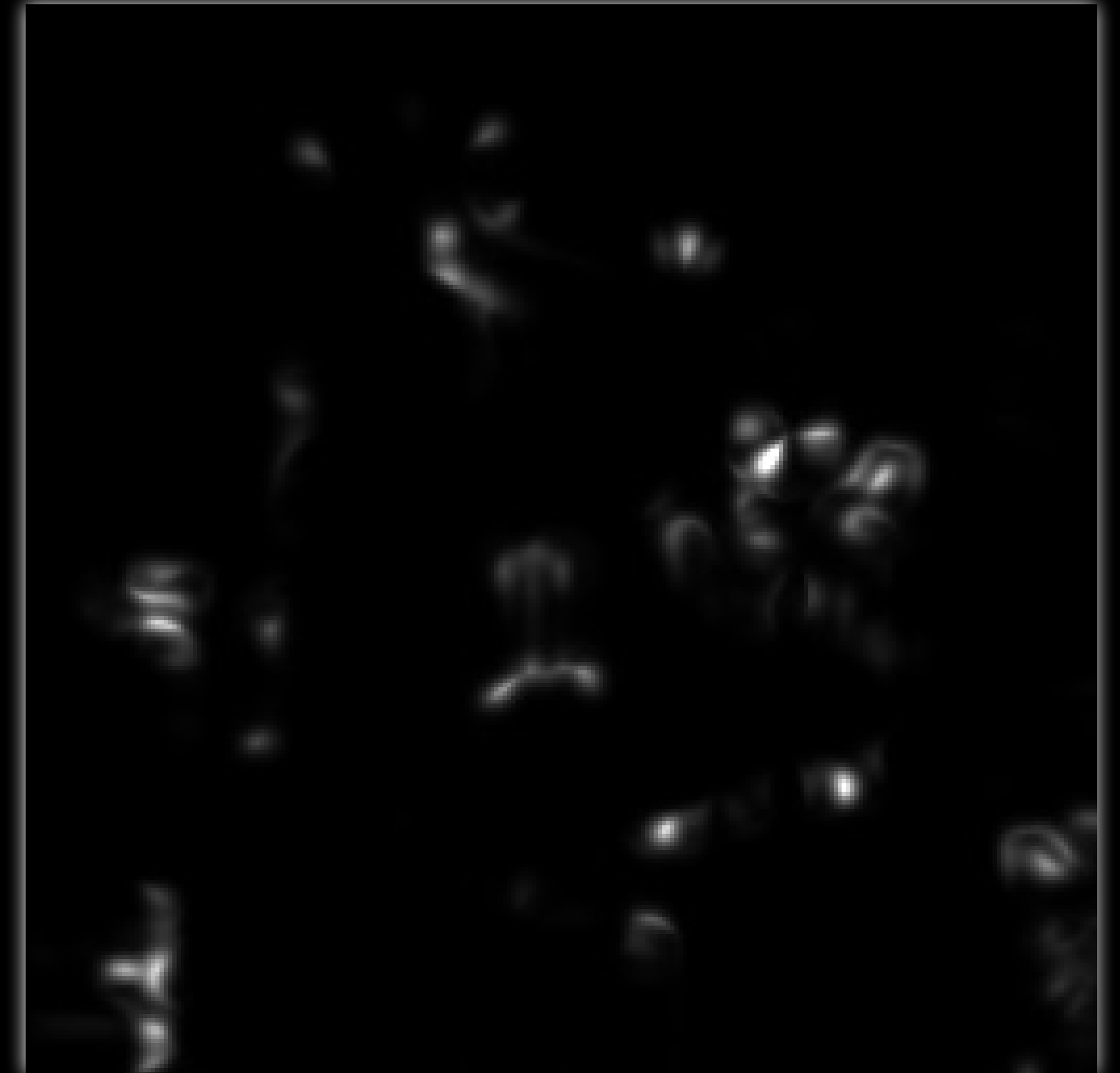


Original image



Harris-Stephens:
 $R(\mathbf{M}) = \det(\mathbf{M}) - k\text{tr}(\mathbf{M})^2$
(normalized to 0-1)

Edges are dark, Corners are bright



Shi-Tomasi:
 $\lambda_{\min}(\mathbf{M})$