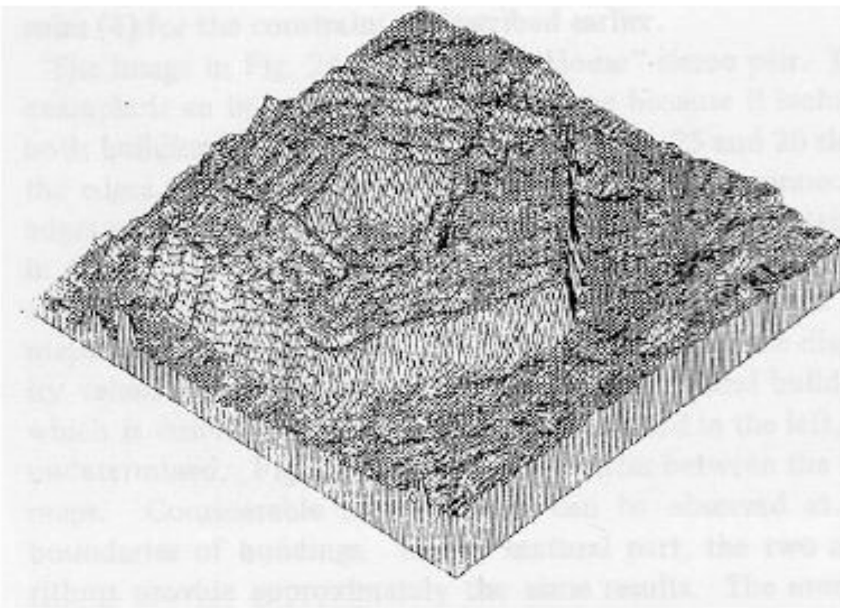
Image Filtering and Stereo

Multi-Camera Geometry

Multi-Camera Geometry

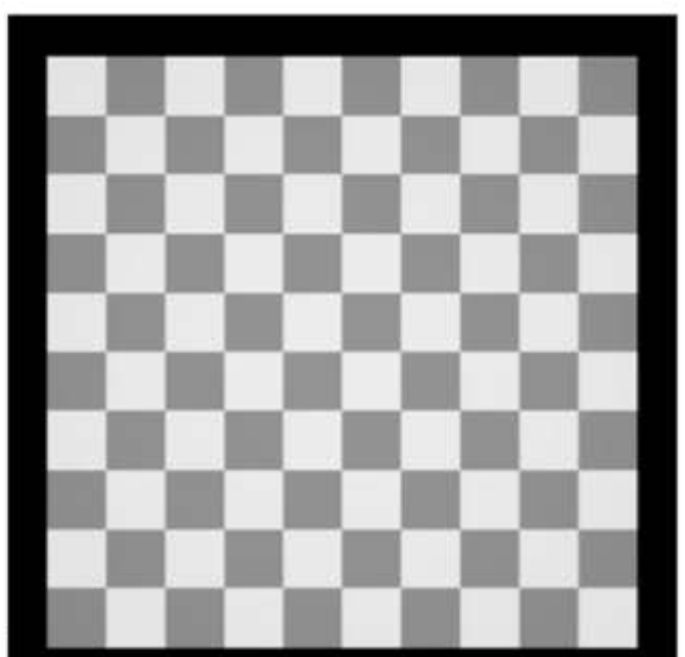


Multi-Camera Geometry

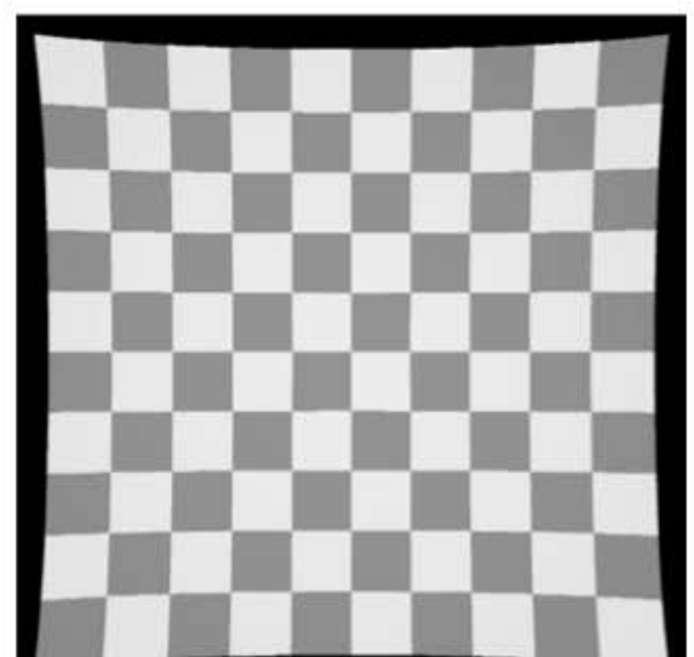
Thin Lens Model

Geometric Distortion

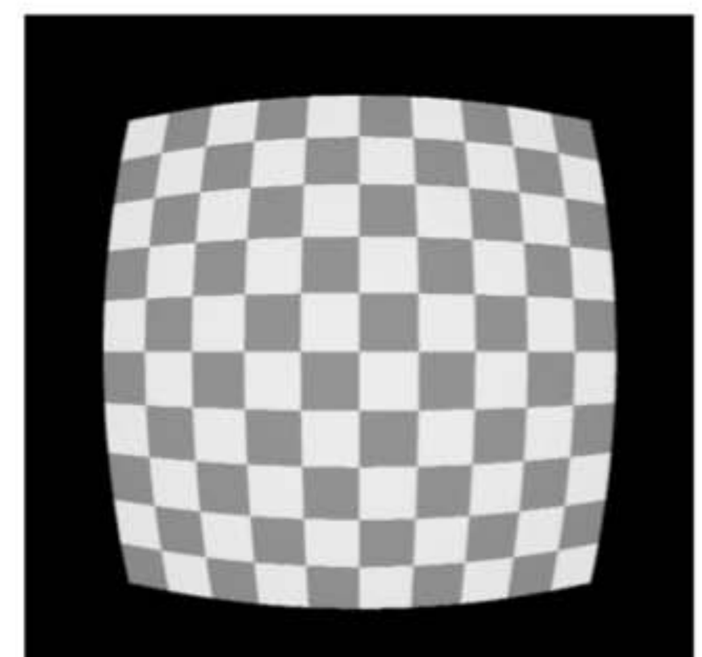
No distortion



Pincushion Distortions



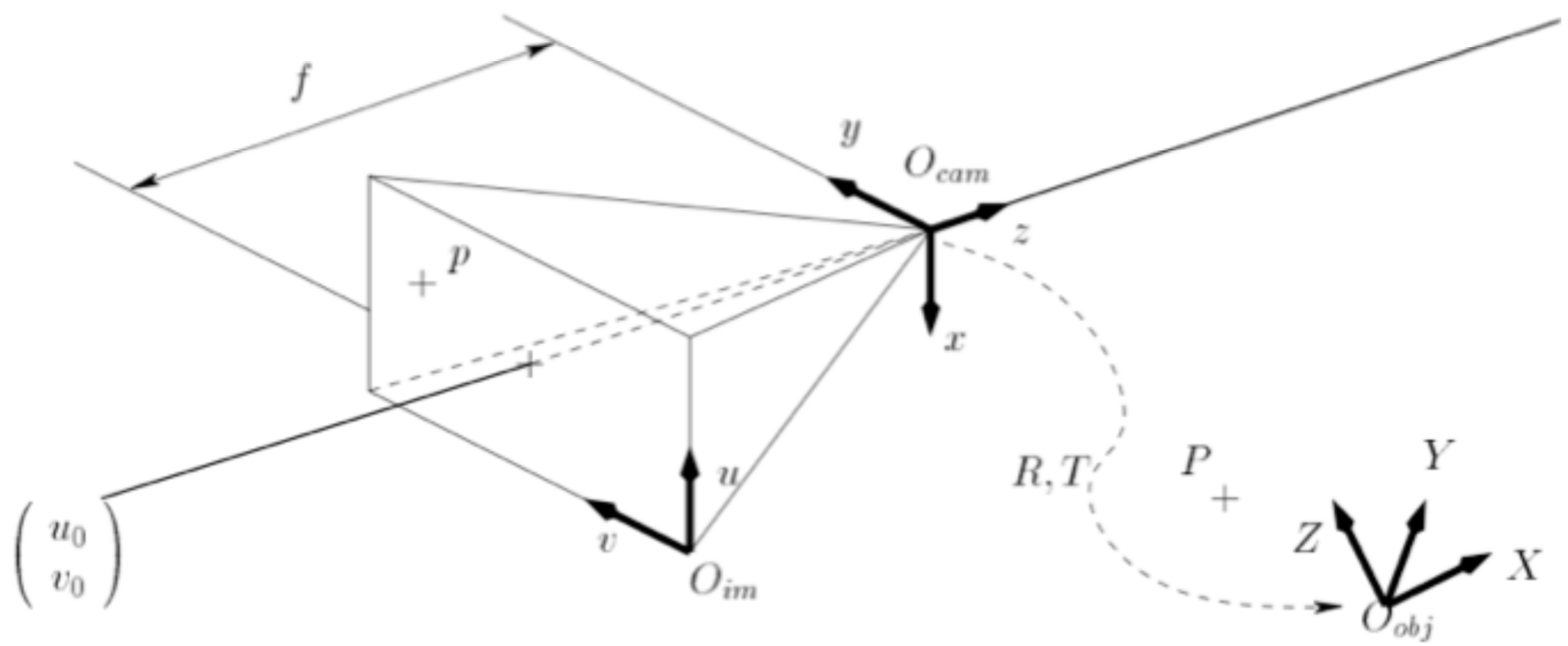
Barrel Distortions



Camera Parameters

- Determine the intrinsic parameters of a camera (with lens)
- What are Intrinsic Parameters?
  - Focal Length f
  - Pixel size s_x , s_y (k_u , k_v)
  - Distortion coefficients k_1 , k_2...
  - Image center u_0 , v_0

Camera Model



$$\alpha_u = k_u f$$

$$\alpha_v = k_v f$$

Coord. of Pixel in the image →

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \underbrace{\begin{pmatrix} \alpha_u & 0 & u_o & 0 \\ 0 & \alpha_v & v_o & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{I} \underbrace{\begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix}}_{E} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

← Coord. Of Point in the "World" frame

E: Transf. between Camera frame and "World" frame

s = depth of 3D point in the scene (*scale factor*)

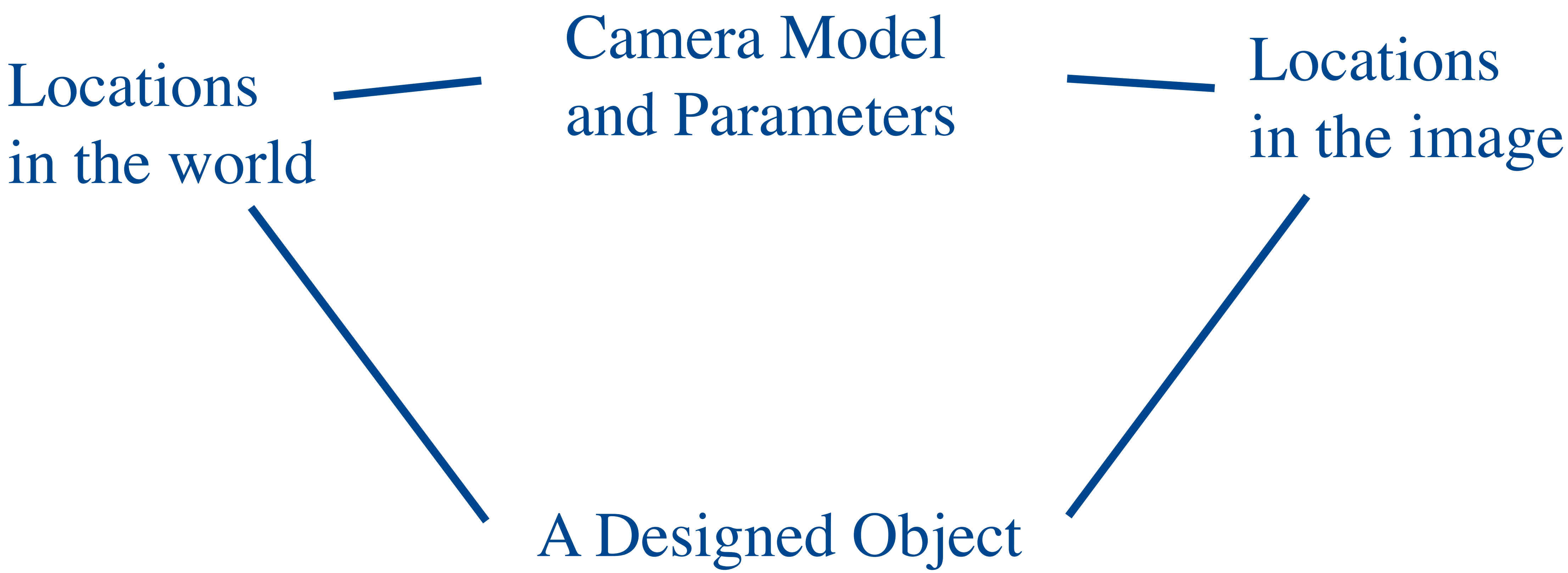
[Devy 2003]

Camera Model

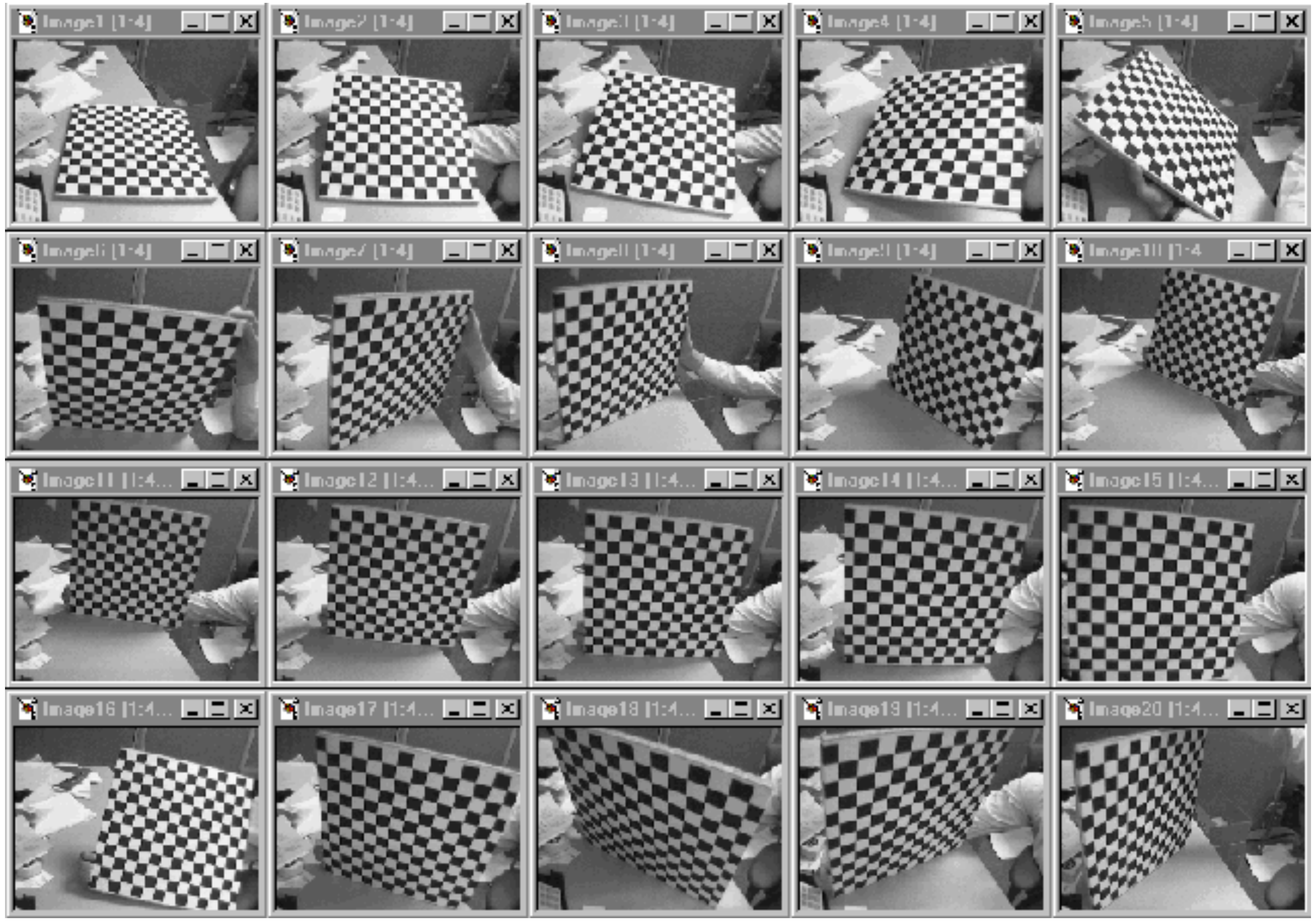
$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = M \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} = I\,E \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

- M = Matrix of Perspective Projection
- I = Matrix of Intrinsic Parameters
- E = Matrix of Extrinsic Parameters (Rotation + Translation)

Ideas for Camera Calibration



Camera Calibration



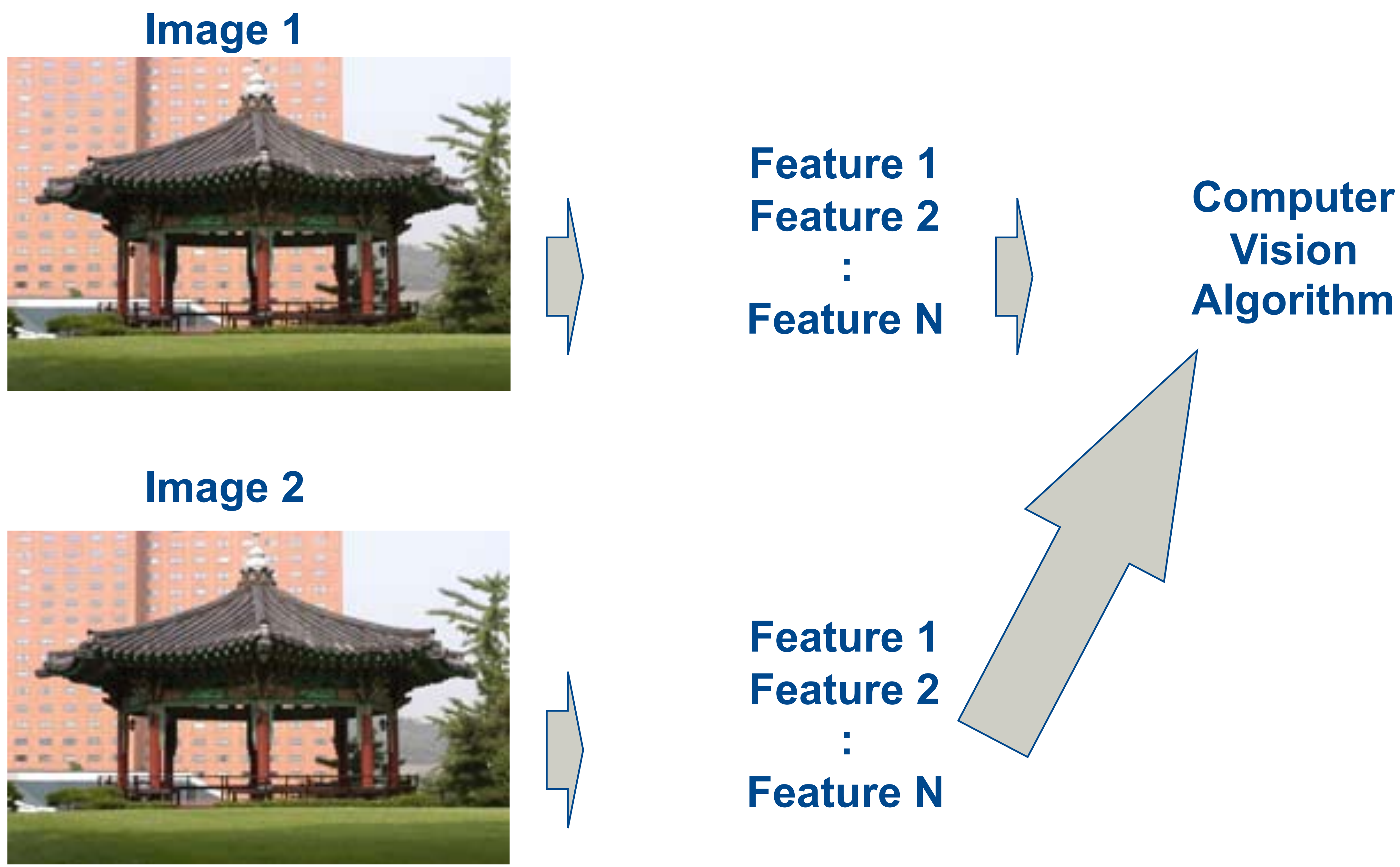
Camera Calibration Toolbox for Matlab

- http://www.vision.caltech.edu/bouguetj/calib_doc/index.html

Features in Computer Vision

- What is a feature?
  - Location of sudden change
- Why use features?
  - Information content high
  - Invariant to change of view point, illumination
  - Reduces computational burden

Image Feature Simplification

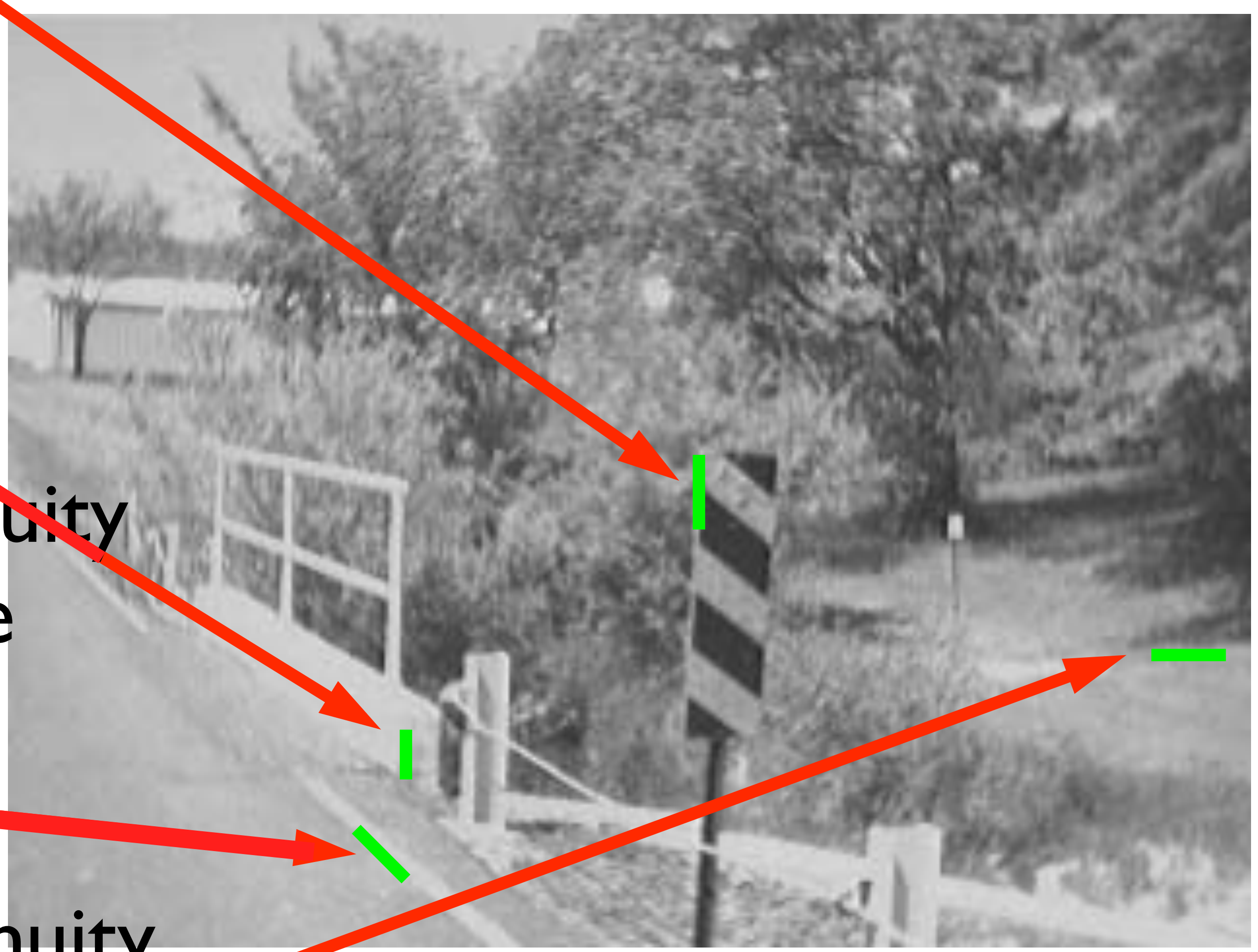


What makes for GOOD features?

- Invariance
  - View point (scale, orientation, translation)
  - Lighting condition
  - Object deformations
  - Partial occlusion
- Other Characteristics
  - Uniqueness
  - Sufficiently many
  - Tuned to the task

First Feature: Edge

- Depth discontinuity
- Surface orientation discontinuity
- Reflectance discontinuity (i.e., change in surface material properties)
- Illumination discontinuity (e.g., shadow)

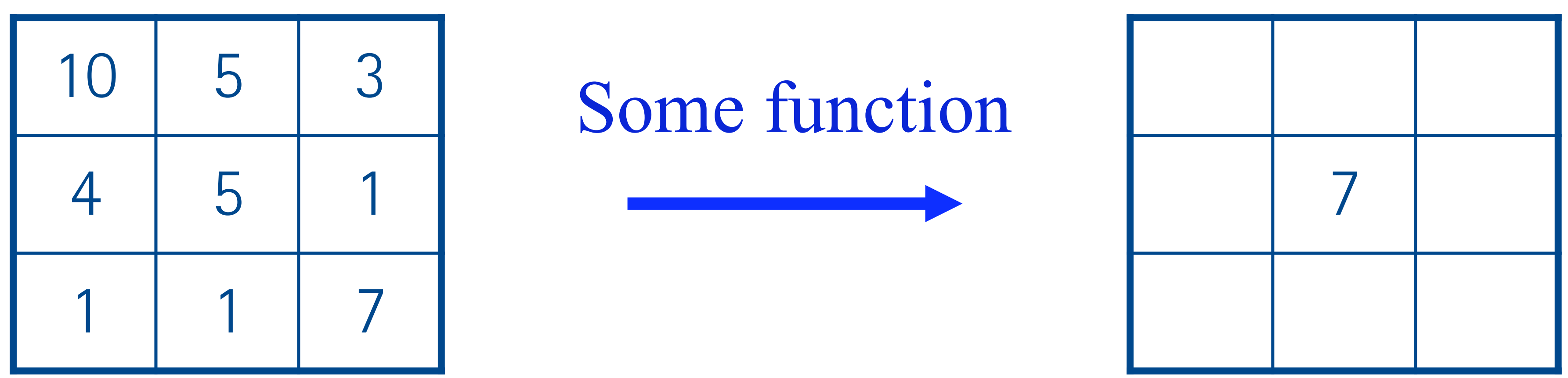


How to find Edges?

Basic Filtering → Edge Detection

Basic Image Filtering

- Modify the pixels in an image based on some function of a local neighborhood of the pixels

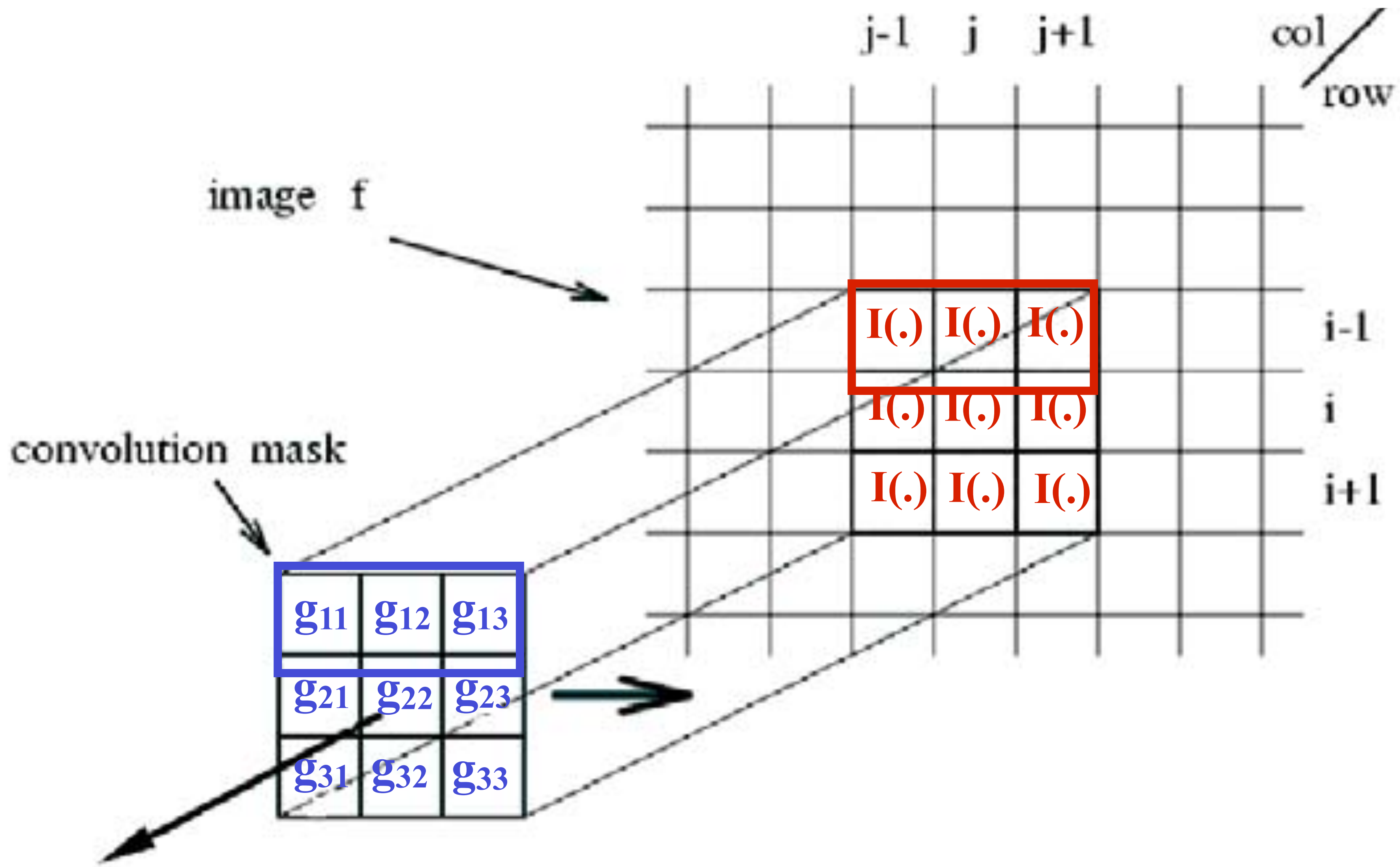


Linear Filtering

- Linear case is simplest and most useful
  - Replace each pixel with a linear combination of its neighbors.
- The prescription for the linear combination is called the convolution kernel.

$$\begin{bmatrix} 10 & 5 & 3 \\ 4 & 5 & 1 \\ 1 & 1 & 7 \end{bmatrix} \otimes \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 1.0 & 0.5 \end{bmatrix}}_{\text{kernel}} = \begin{bmatrix} \; & \; & \; \\ \; & 7 & \; \\ \; & \; & \; \end{bmatrix}$$

Linear Filter = Convolution



$$f(i,j) = g_{11}\, I(i-1,j-1) + g_{12}\, I(i-1,j) + g_{13}\, I(i-1,j+1) + g_{21}\, I(i,j-1) + g_{22}\, I(i,j) + g_{23}\, I(i,j+1) + g_{31}\, I(i+1,j-1) + g_{32}\, I(i+1,j) + g_{33}\, I(i+1,j+1)$$

Linear Filter = Convolution

$$f[m,n] = I \otimes g = \sum_{k,l} I[m-k, n-l]\, g[k,l]$$

$$\text{with} \quad \sum_{k,l} g[k,l] = 1$$

- Example on the web: www.jhu.edu/~signals/convolve
- Matlab function: conv(1D) or conv2(2D)

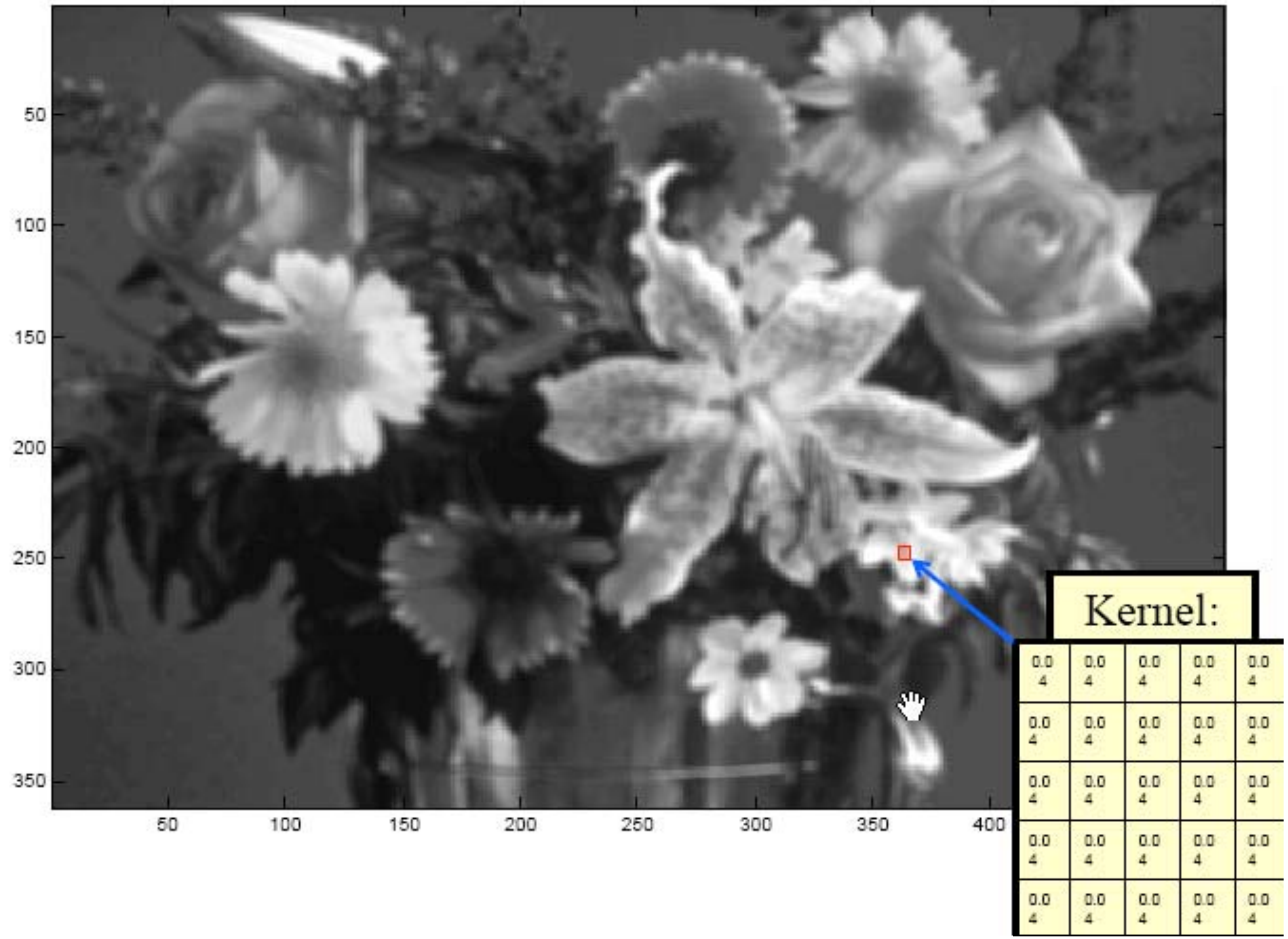
Original Image

Slight Blurring

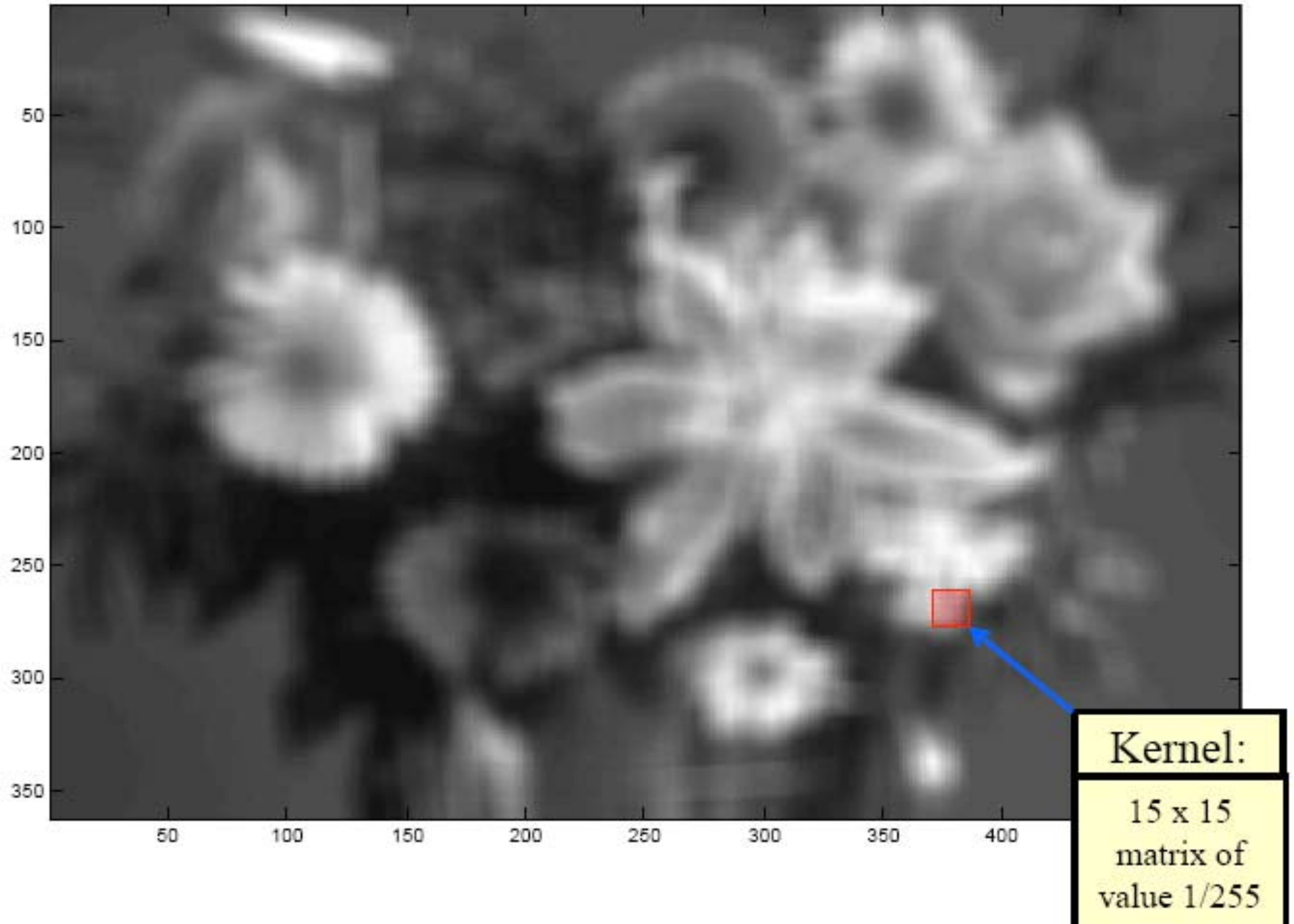
Kernel:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

More Blurring



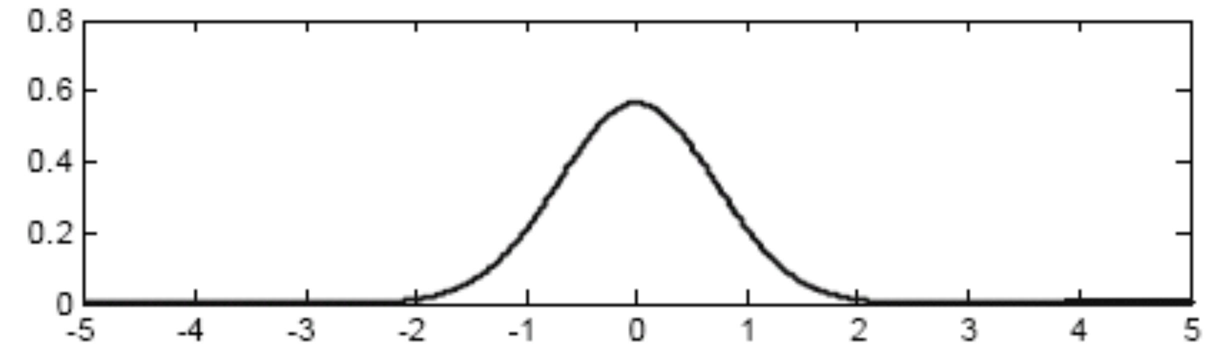
Lots of Blurring



Gaussian

1-D:

$$g(x) = e^{-\frac{x^2}{2\sigma^2}}$$

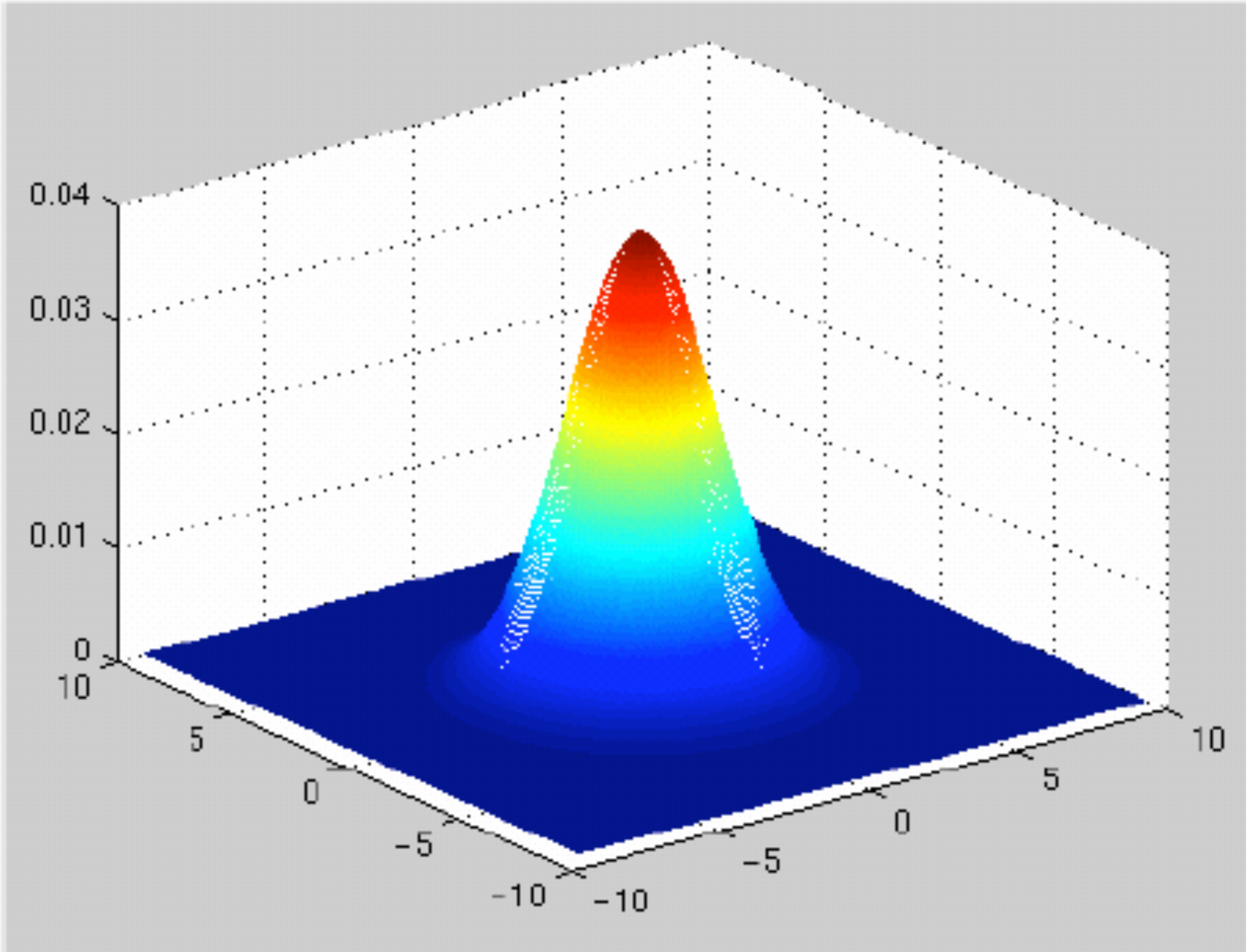


2-D:

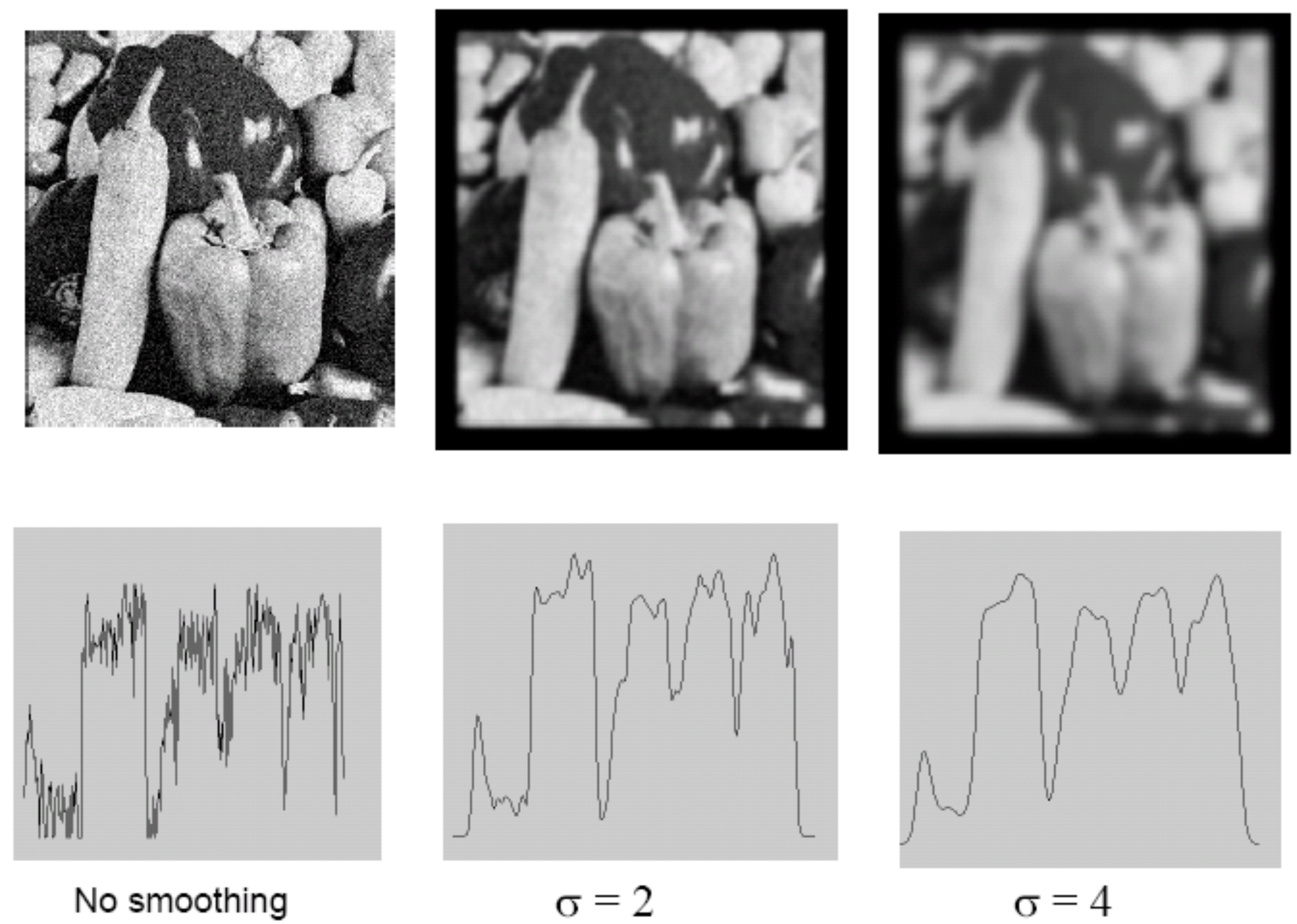
$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Slight abuse of notations:
We ignore the normalization constant such that

$$\int g(x)dx = 1$$



Gaussian Smoothing to Remove Noise



Some kernels

1	1	1
1	-2	1
-1	-1	-1

Prewitt 1

5	5	5
-3	0	-3
-3	-3	-3

Kirsch

-1	$-\sqrt{2}$	-1
0	0	0
1	$\sqrt{2}$	1

Frei & Chen

1	1	1
0	0	0
-1	-1	-1

Prewitt 2

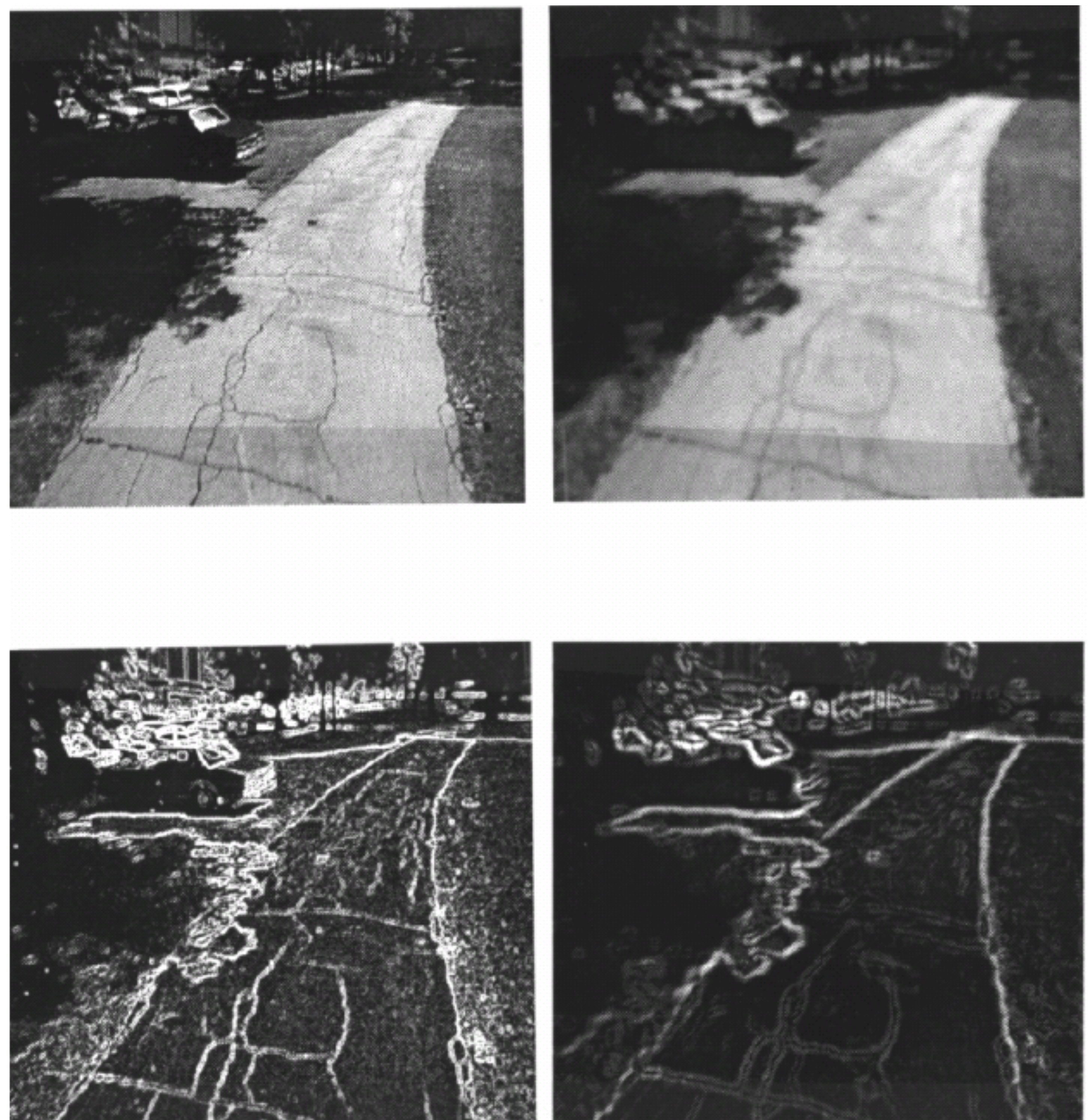
1	2	1
0	0	0
-1	-2	-1

Sobel

These kernels are Gradient operators

- Edges are discontinuities of intensity in images
- Correspond to local maxima of image gradient
- Gradient computed by convolution
- General principle applies:
    - Slight smoothing: Good localization, poor detection
    - More smoothing: Poor localization, good detection

Smoothing Effects



Canny Edge Detector

Canny's Result

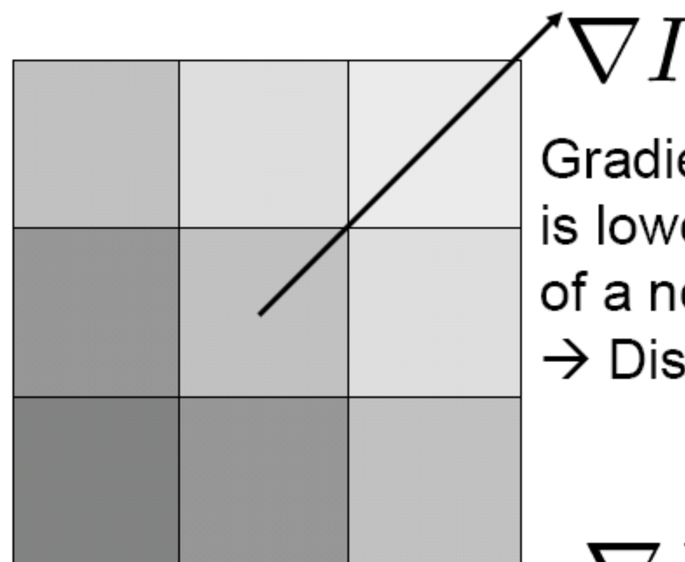
- Given a filter *f*, define the two objective functions:
  - A(*f*) large if *f* produces good localization
  - B(f) large if f produces good detection
- Problem: Find a family of f that maximizes the compromise criterion A(*f*)B(*f*) under the constraint that a single peak is generated by a step edge.
- Solution: Unique solution, a close approximation is the Gaussian derivative.

Next Steps

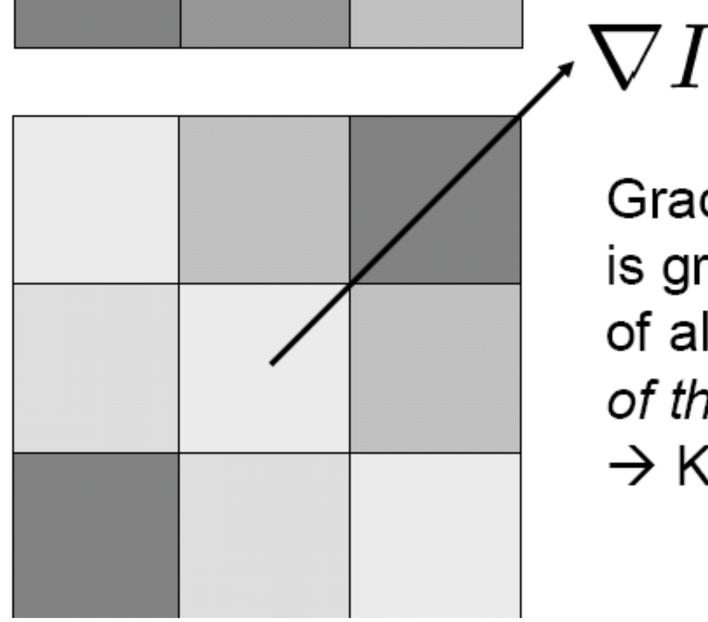
- The gradient magnitude enhances the edges but 2 problems remain:
  - What threshold should we use to retain only the "real" edges?
  - Even if we had a perfect threshold, we would still have poorly localized edges. How to optimally localize contours?

- Solution:
  - Non-local maxima suppression
  - Hysteresis thresholding

Non-Local Maxima Suppression

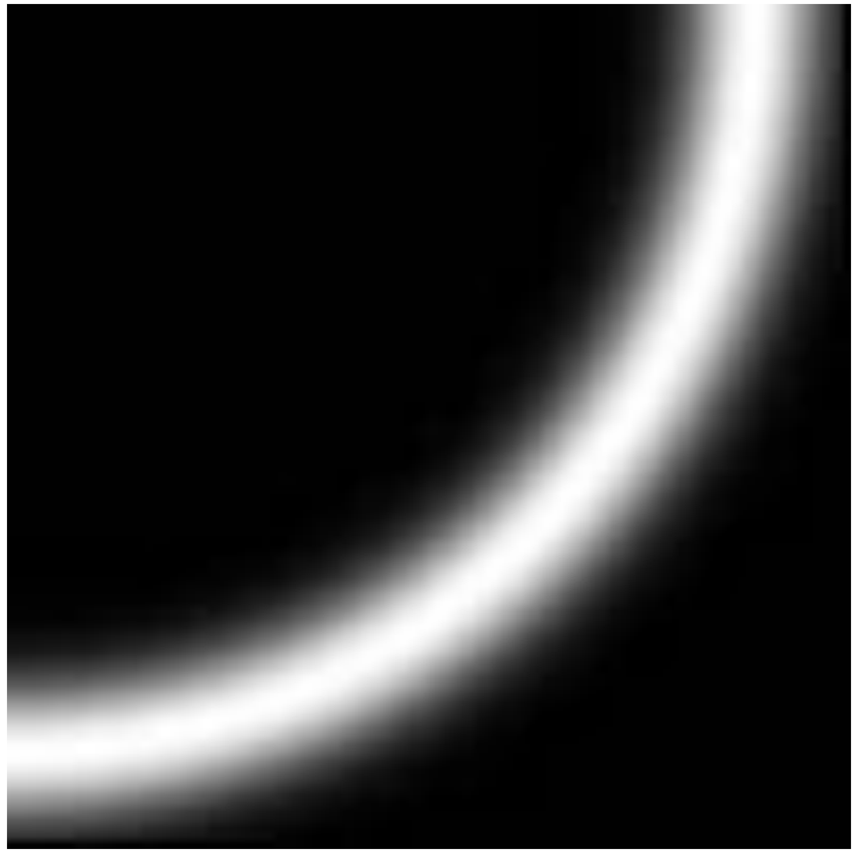


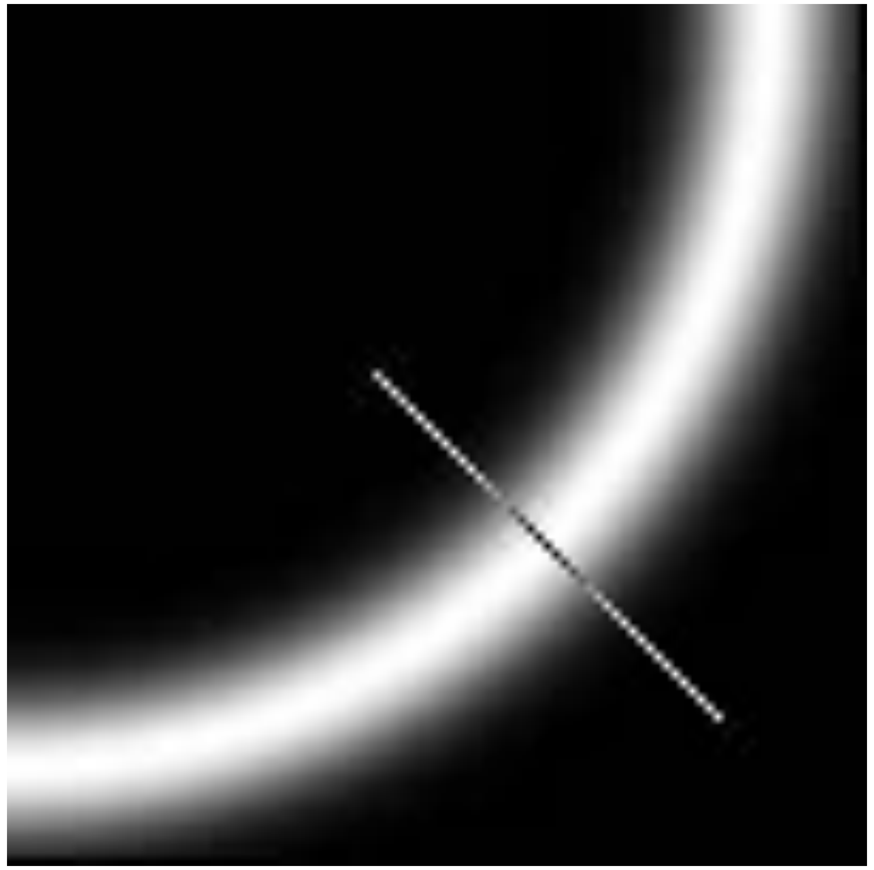
Gradient magnitude at center pixel
is lower than the gradient magnitude
of a neighbor *in the direction of the gradient*
→ Discard center pixel (set magnitude to 0)



Gradient magnitude at center pixel
is greater than gradient magnitude
of all the neighbors *in the direction*
of the gradient
→ Keep center pixel unchanged

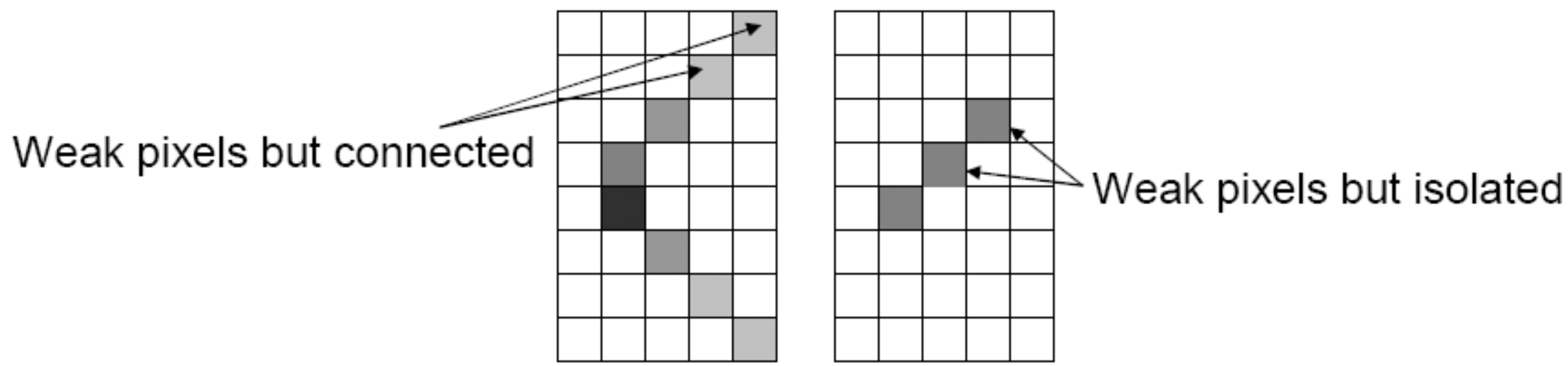
Non-Local Maxima Suppression

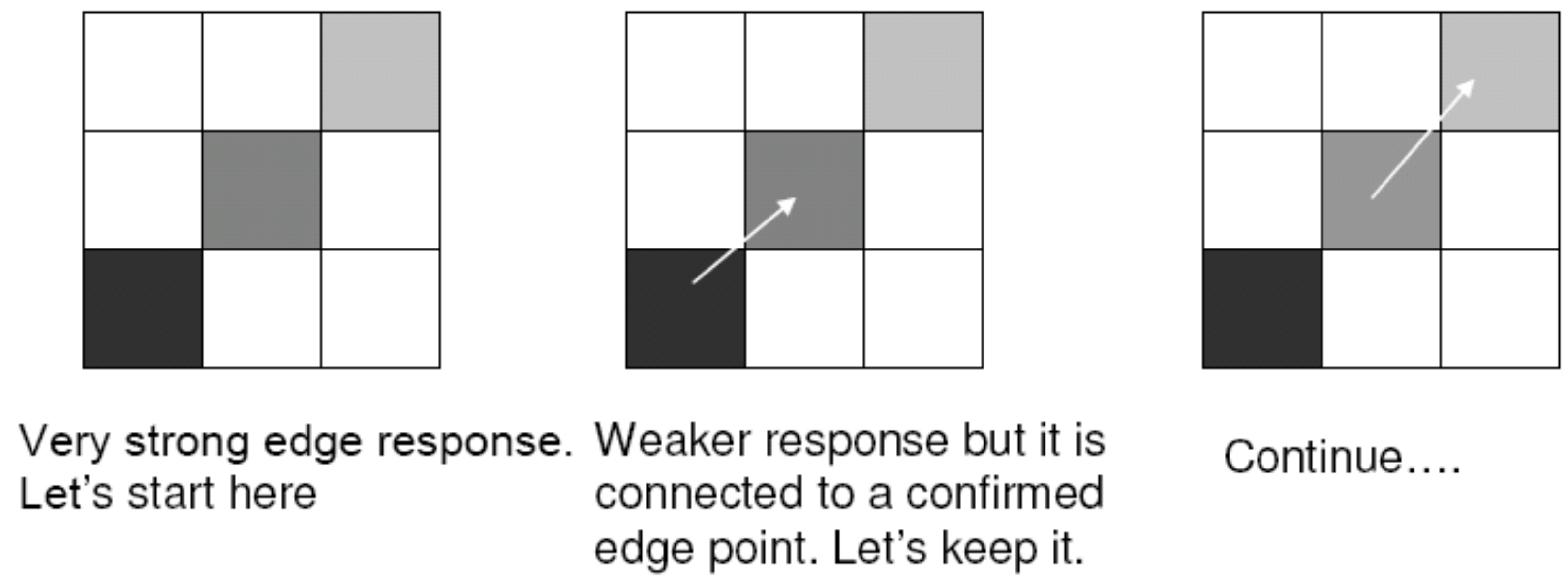




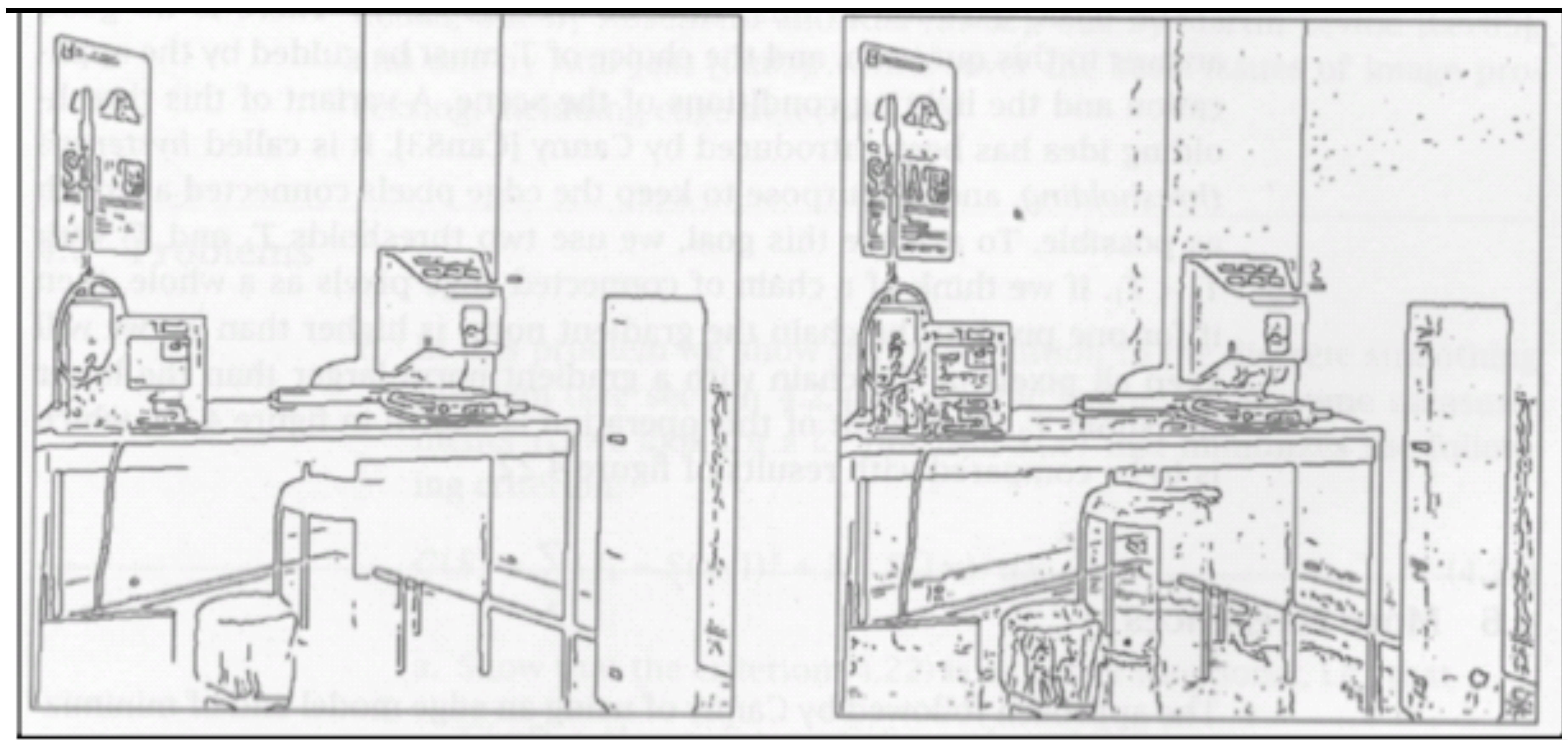
- Select the single maximum point across the width of an edge

Hysteresis Thresholding





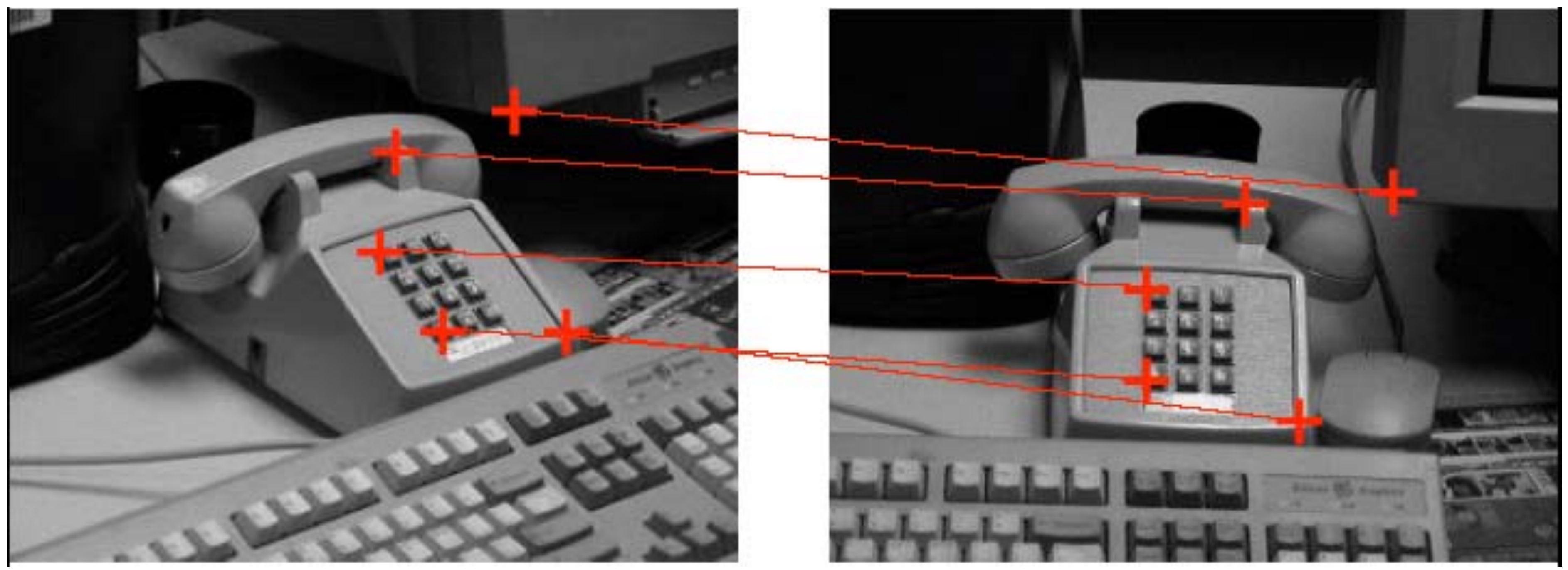
Varying Thresholds



Canny Edge Detector Algorithm

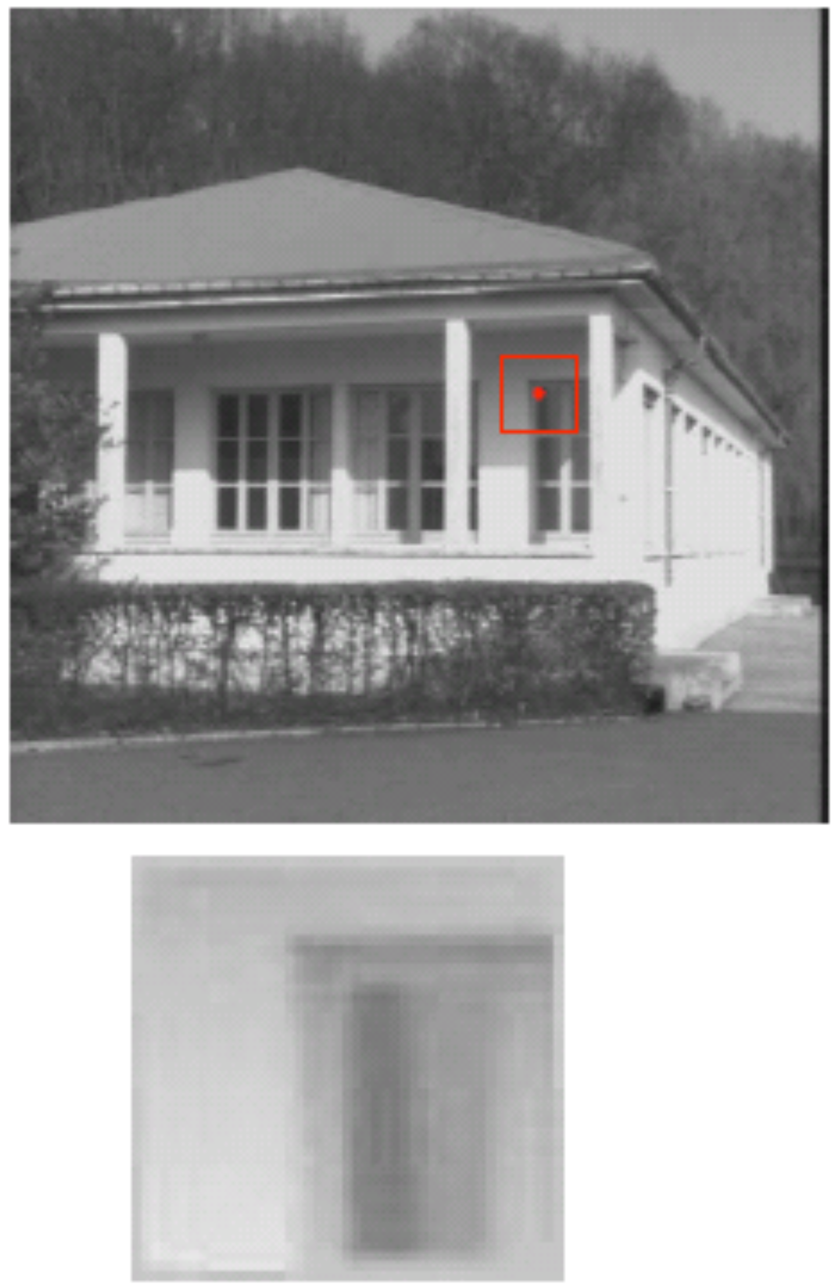
- Apply derivative of Gaussian
- Non-maximum suppression
    - Thin multi-pixel wide "ridges" down to single pixel width
- Linking and thresholding
    - Low, high edge-strength thresholds
    - Accept all edges over low threshold that are connected to edge over high threshold

Finding Correspondences Between Images

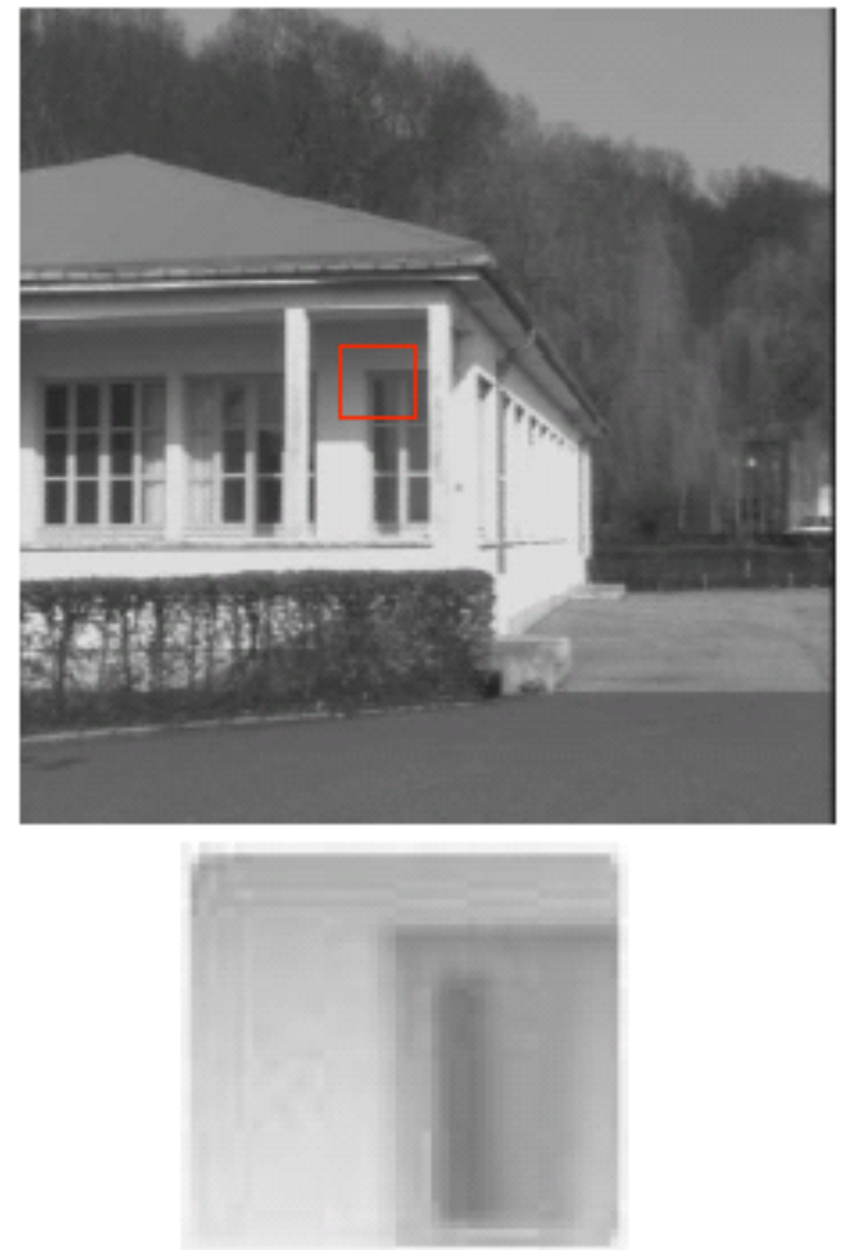


- First step toward 3-D reconstruction
- First step toward tracking
- Object Recognition: finding correspondences between feature points in “training” and “test” images.

Finding Correspondences

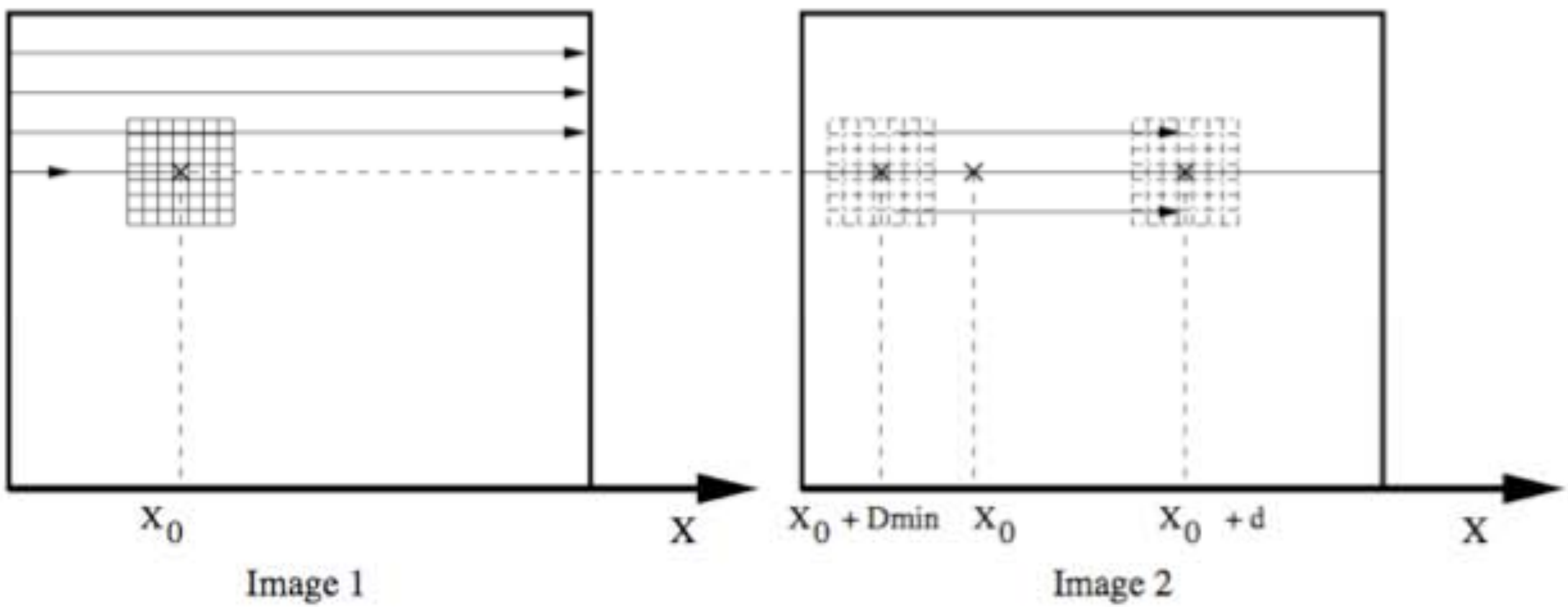


$W(\mathbf{p}_l)$

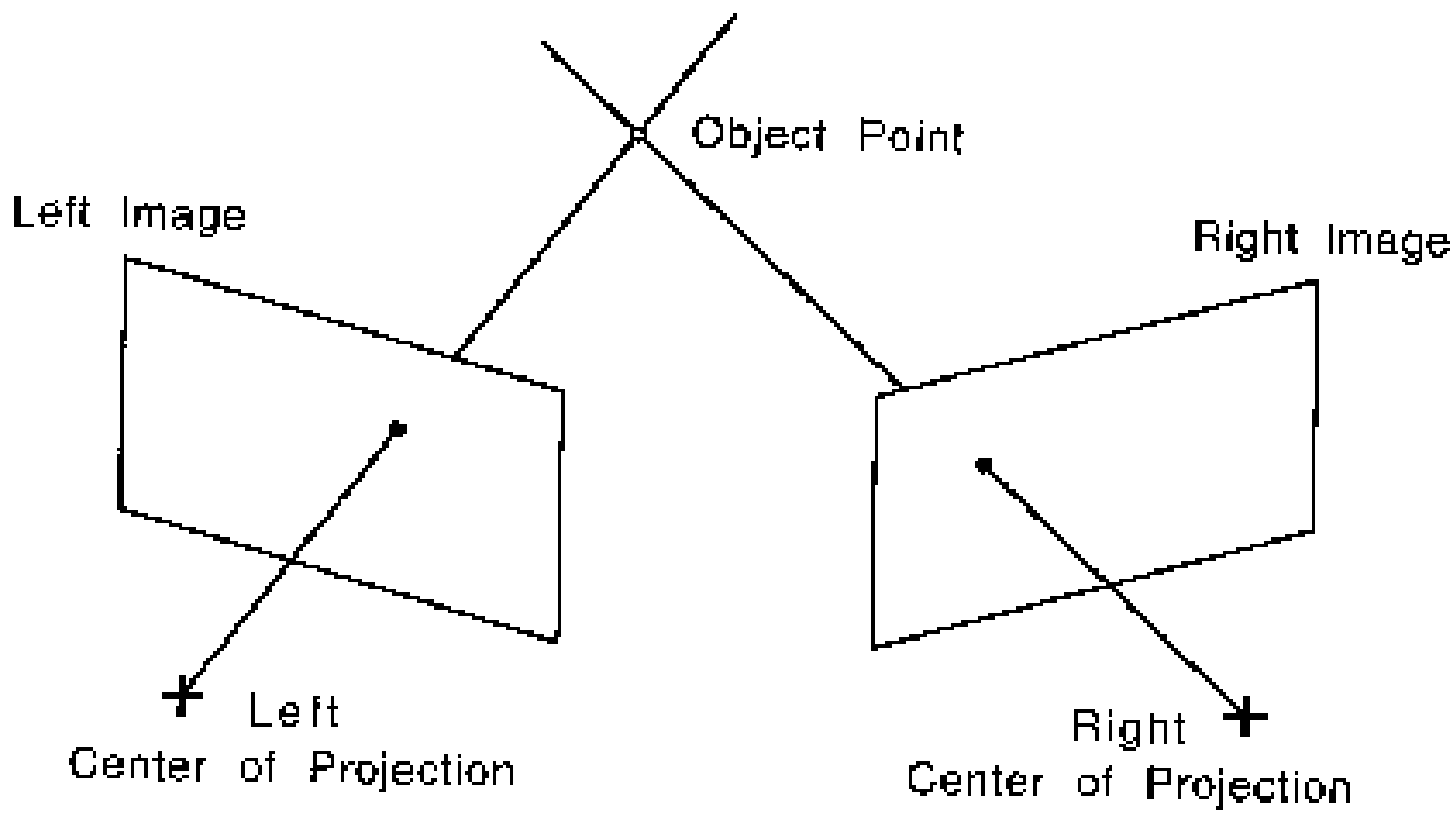


$W(\mathbf{p}_r)$

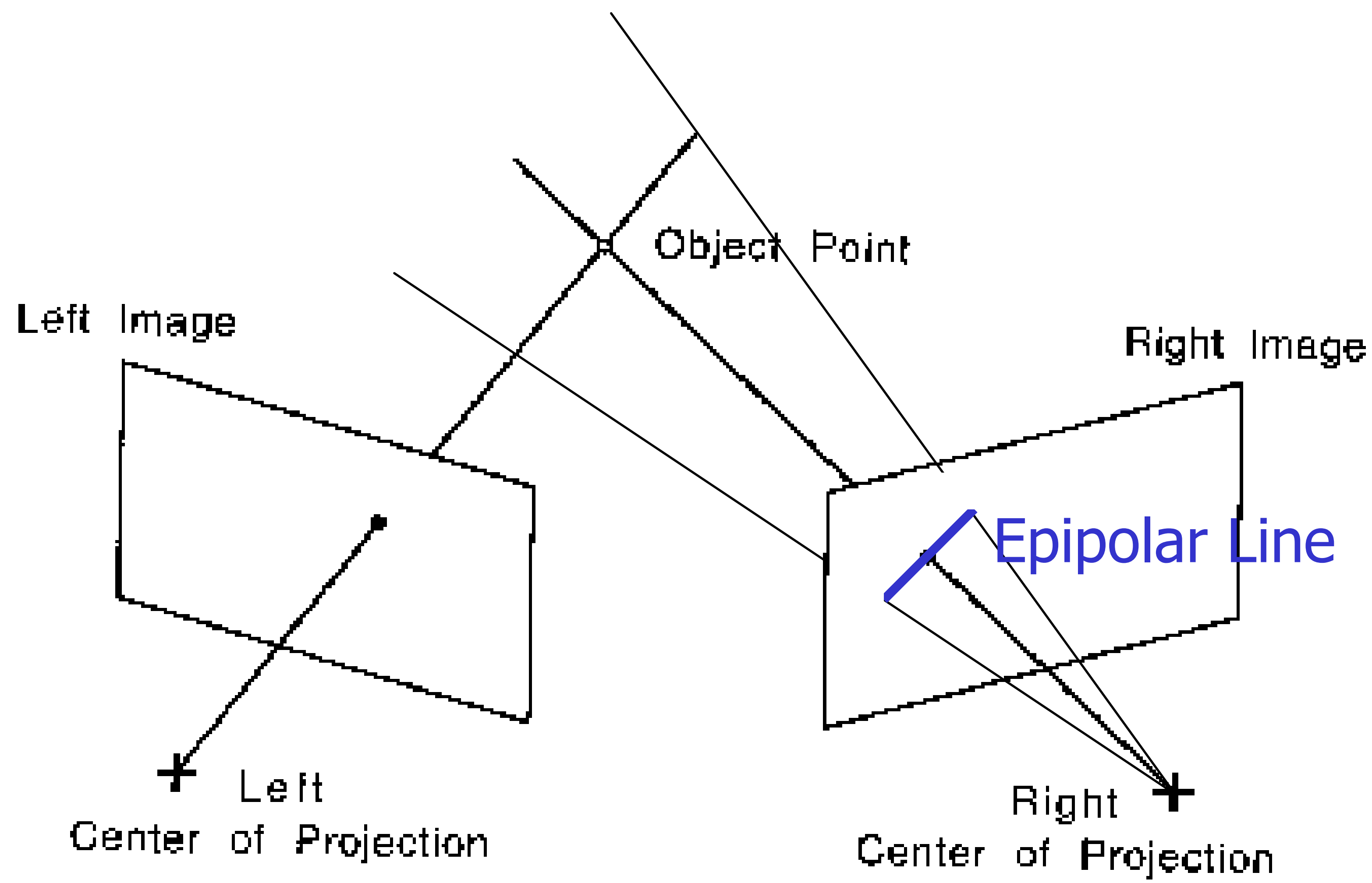
Disparity



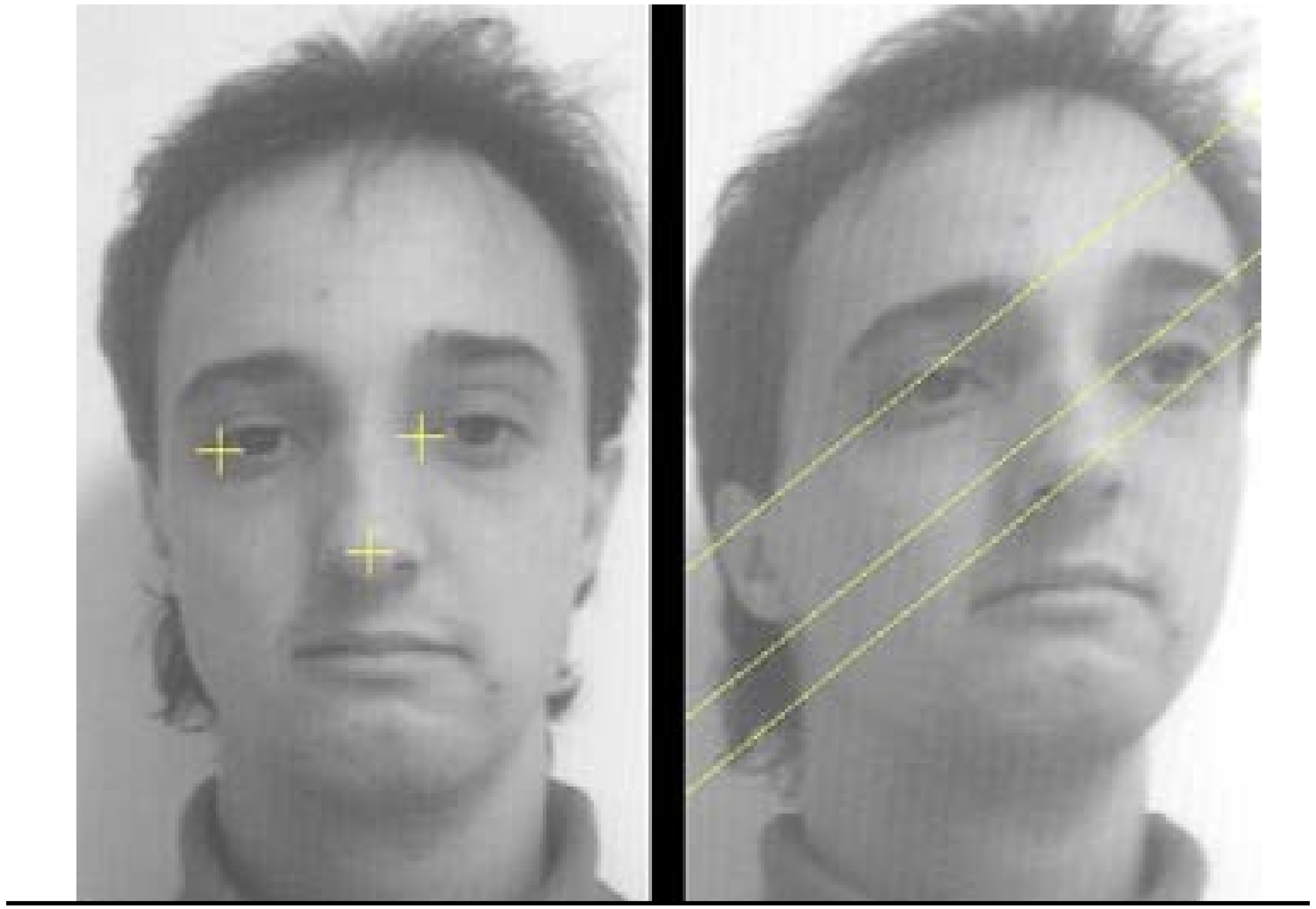
Triangulation



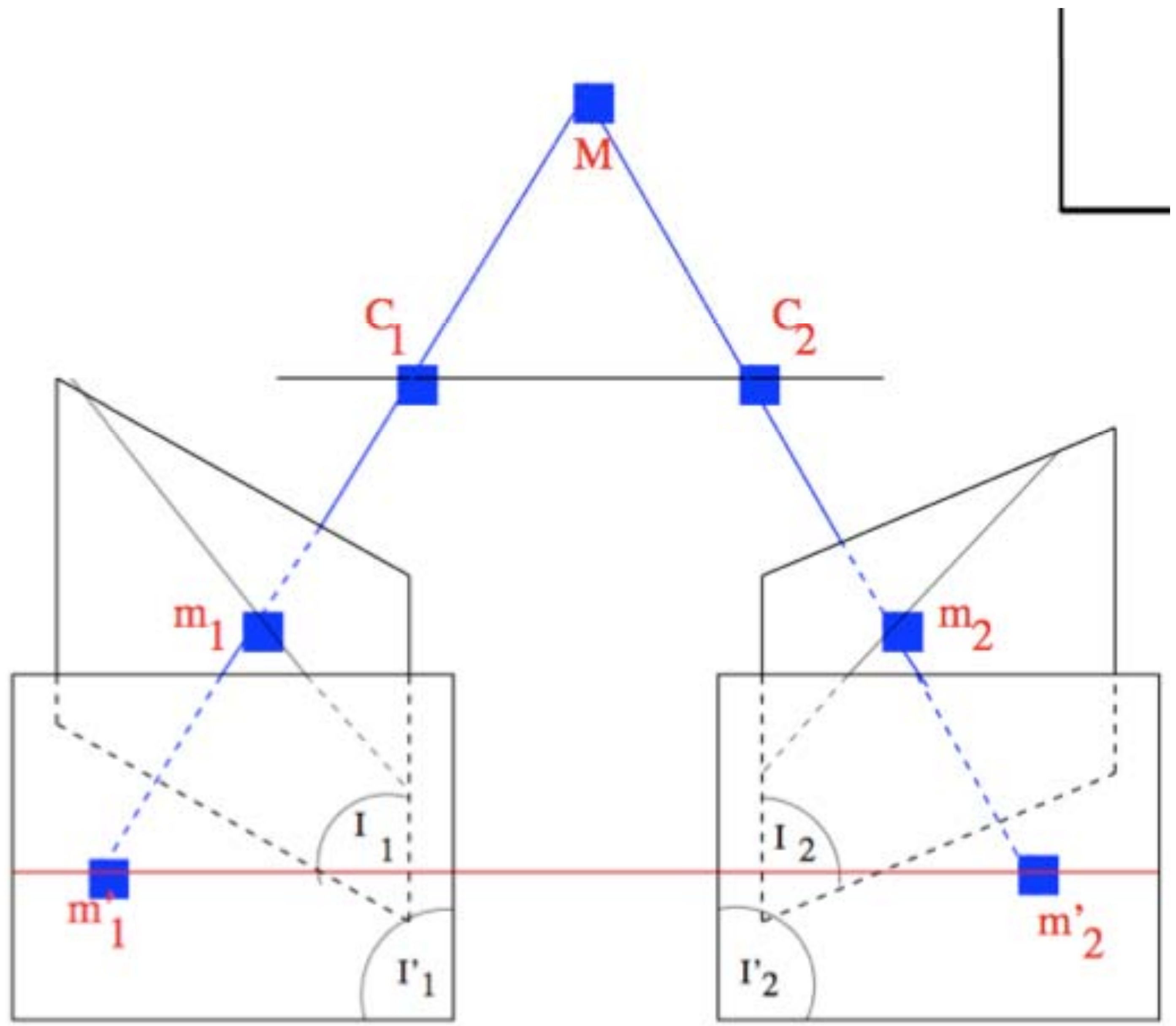
Epipolar Line



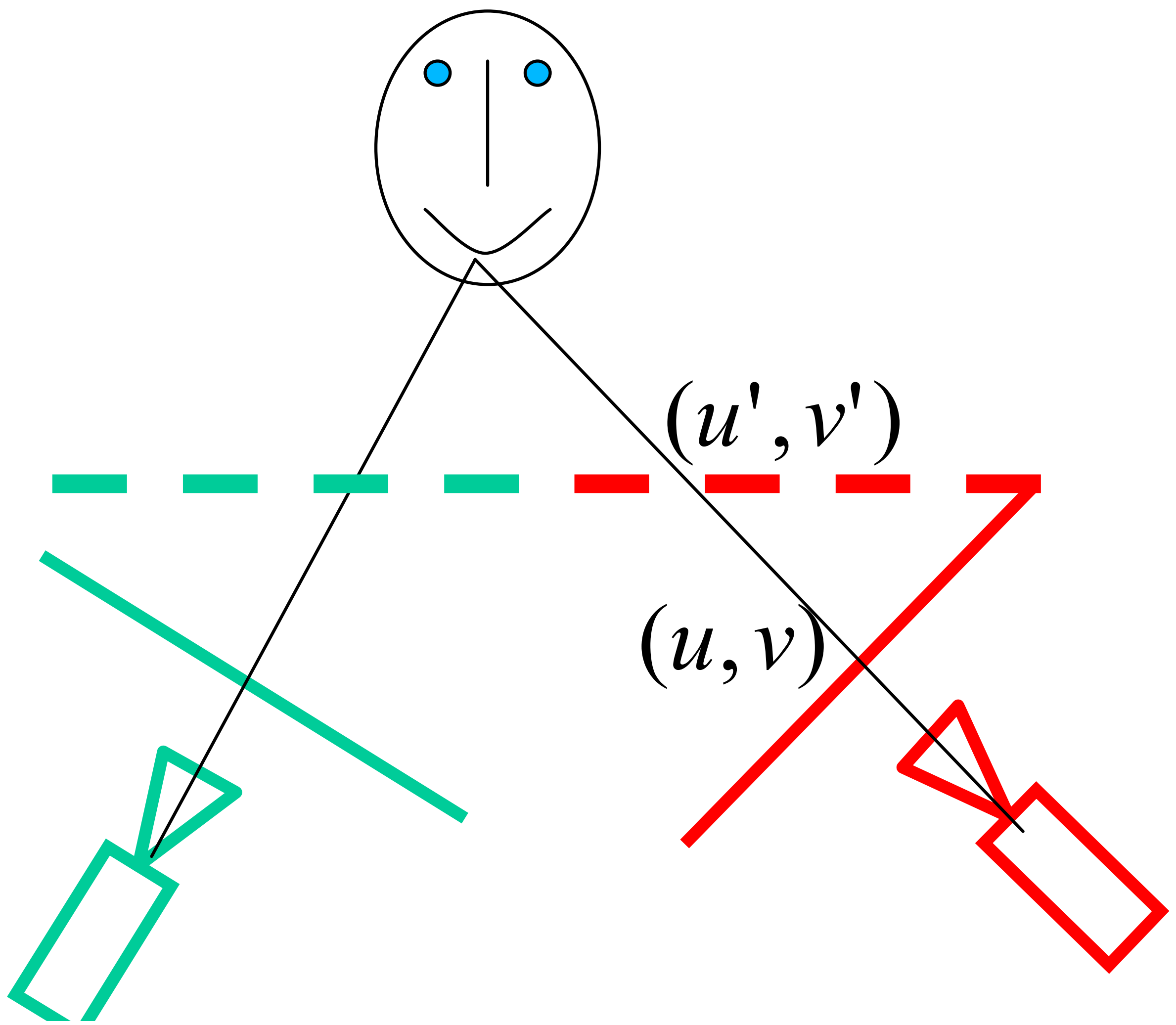
Epipolar Line



Rectification



Rectification

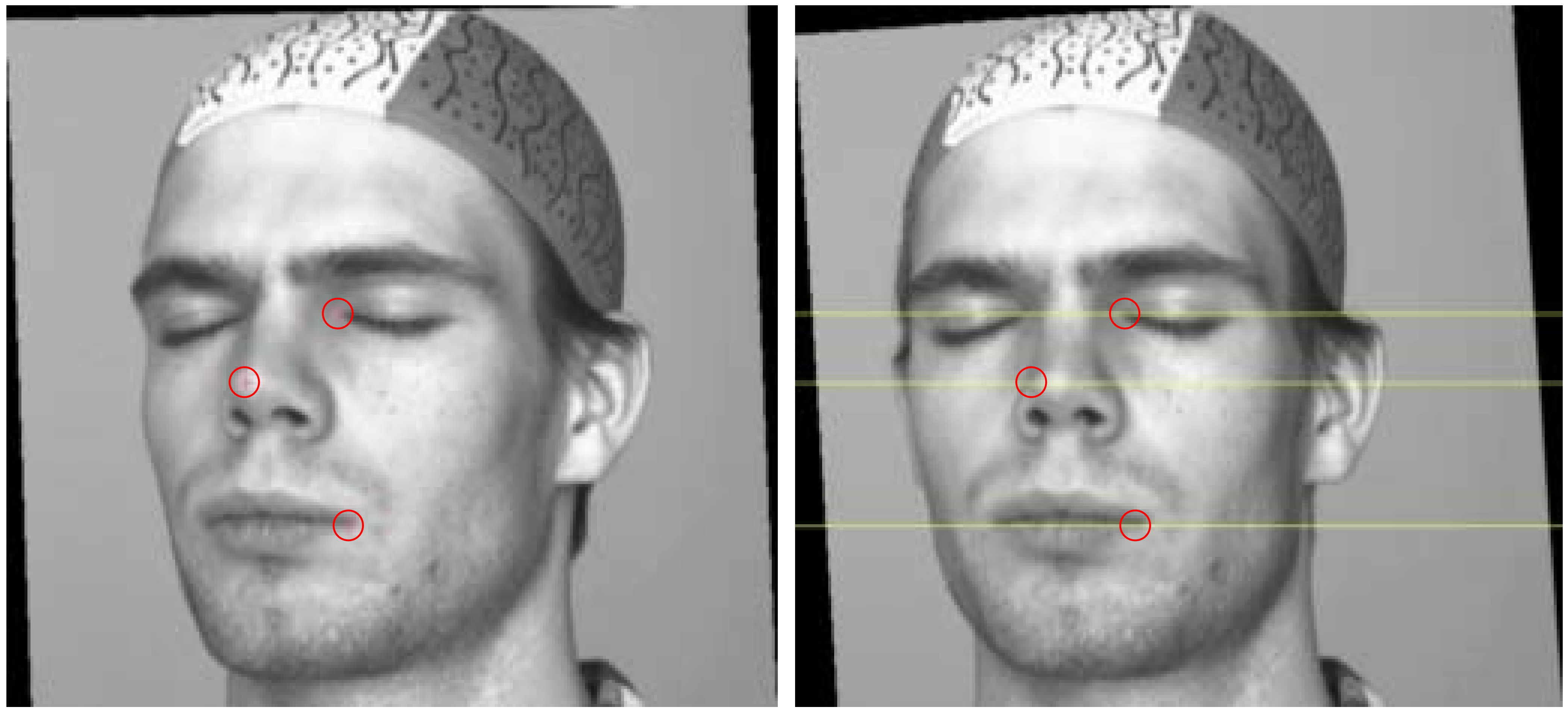


$$\begin{bmatrix} U' \\ V' \\ W' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

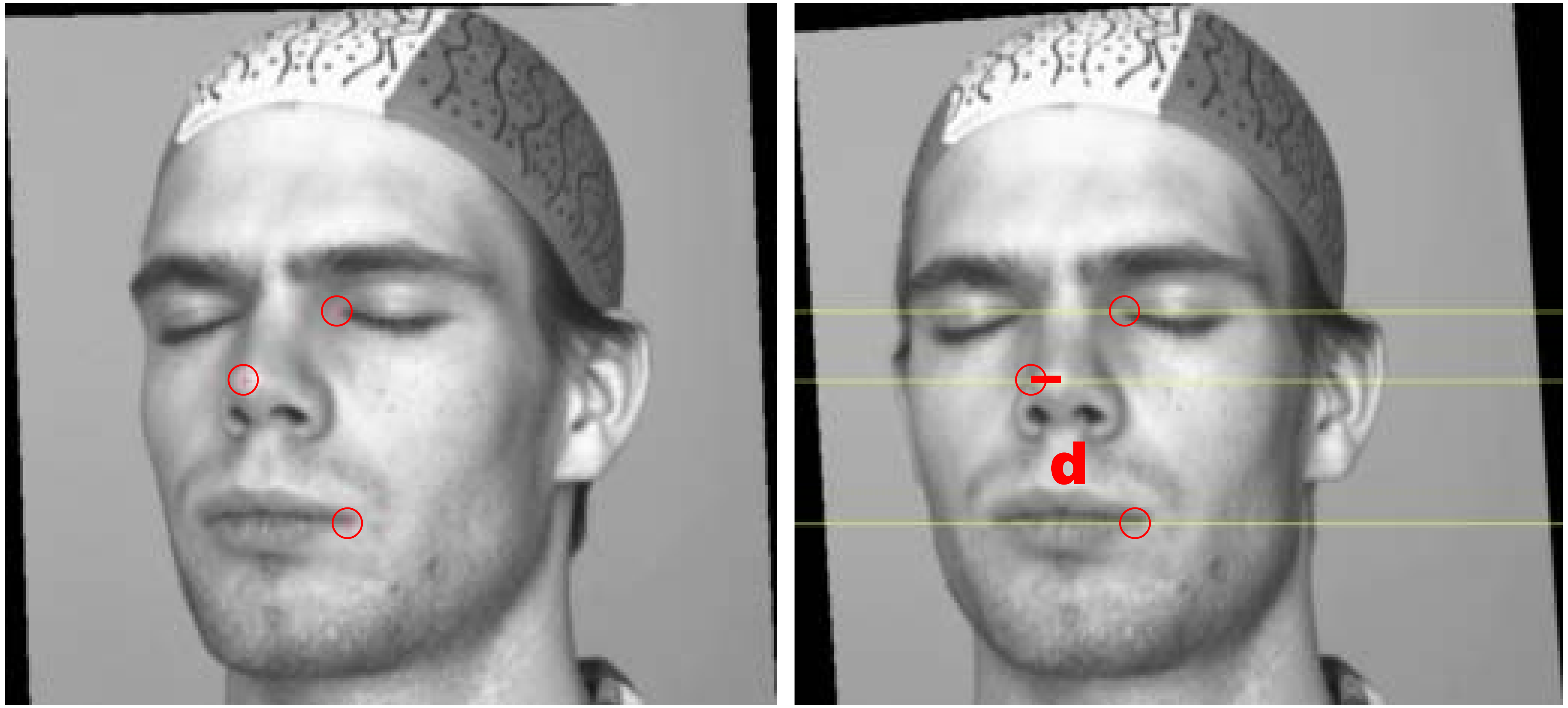
$$u' = U' / W'$$

$$v' = V' / W'$$

Rectification



Disparity



Sum of Squared Differences

- Subtract pattern and image pixel by pixel and add squares:

$$ssd(u,v) = \sum_{(x,y)\in N} \left[I(u+x, v+y) - P(x,y) \right]^2$$

- If identical ssd=0, otherwise ssd >0 Look for minimum of ssd with respect to u and v.

SSD

$$ssd(u,v) = \sum_{(x,y)\in N} \left[I(u+x, v+y) - P(x,y)\right]^2$$

$$= \sum_{(x,y)\in N} I(u+x, v+y)^2 + \sum_{(x,y)\in N} P(x,y)^2 - 2 \sum_{(x,y)\in N} I(u+x, v+y)P(x,y)$$

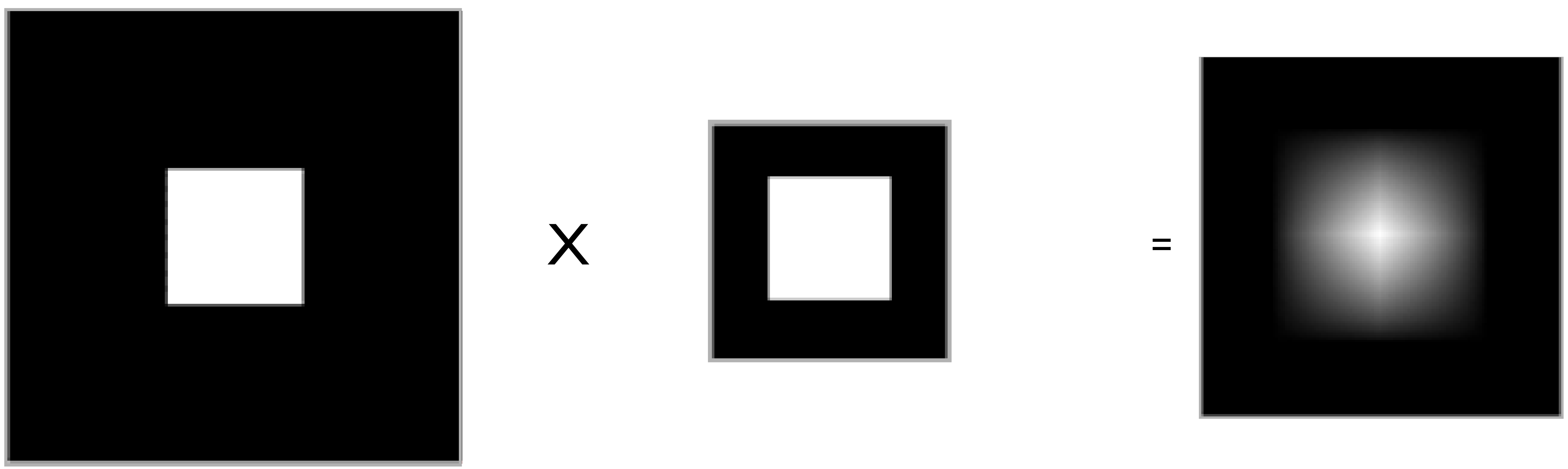
Sum of squares of the window (positive term)

Sum of squares of the pattern (CONSTANT term)

Correlation

- SSD is minimized when correlation is largest or patches are most similar

Simple Example

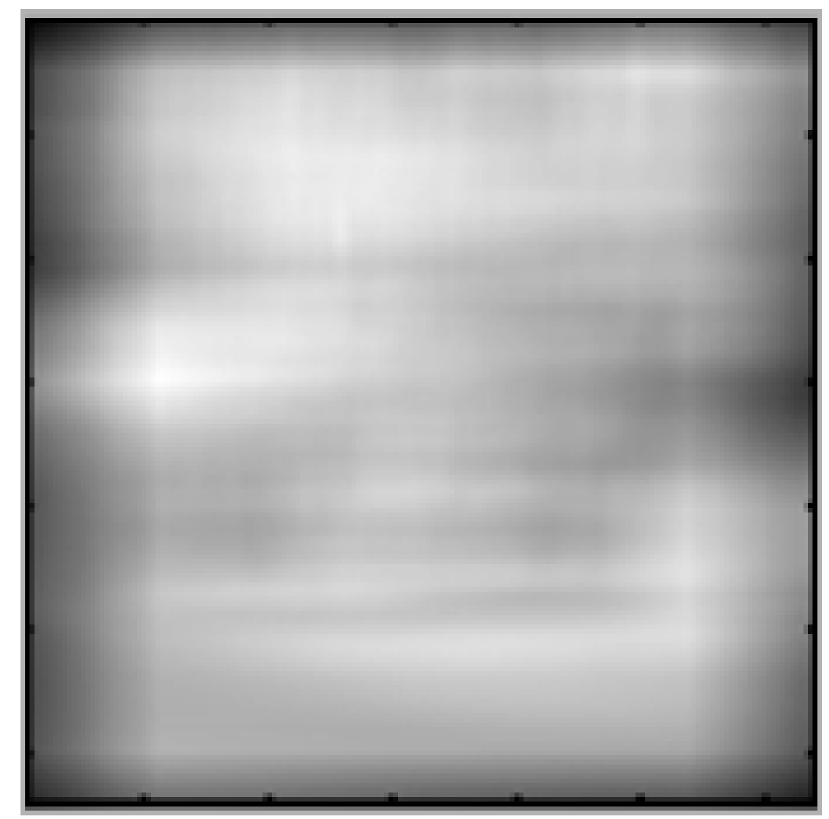


More realistic

Image

Pattern

Correlation



Normalized Cross-Correlation

$$ncc(u,v) = \frac{\sum_{(x,y)\in N}\left[I(u+x,v+y)-\bar{I}\right]\left[P(x,y)-\overline{P}\right]}{\sqrt{\sum_{(x,y)\in N}\left[I(u+x,v+y)-\bar{I}\right]^2 \sum_{(x,y)\in N}\left[P(x,y)-\overline{P}\right]^2}}$$

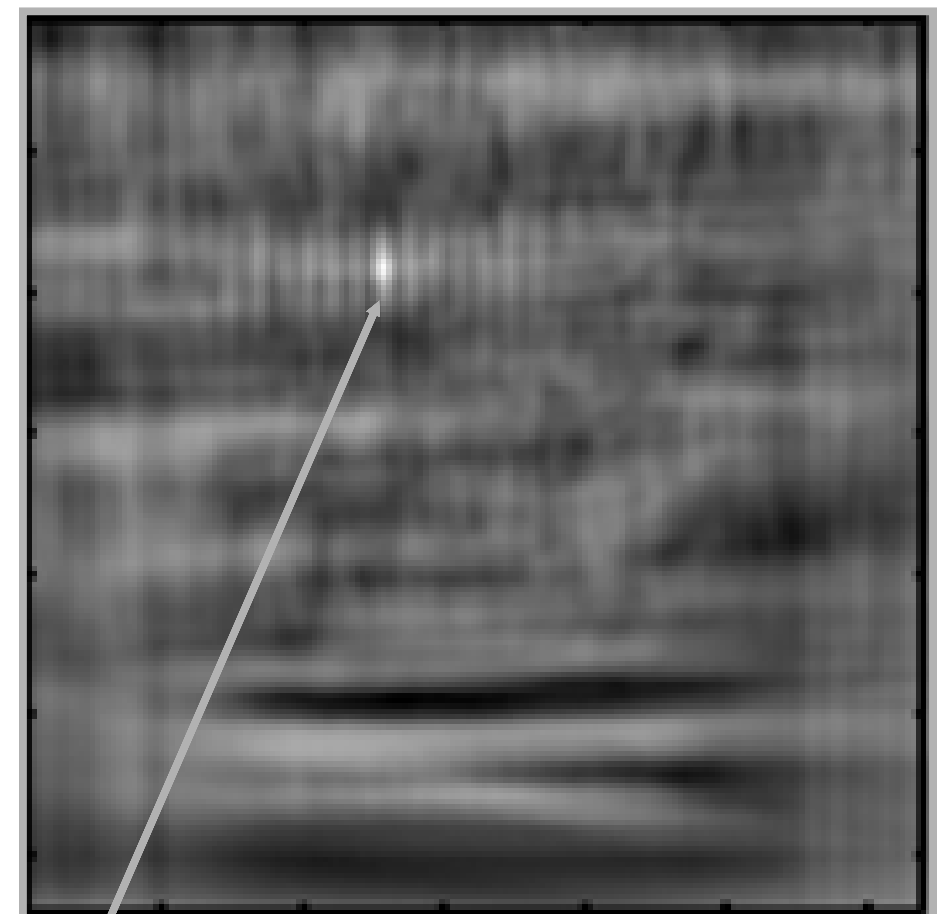
- Between -1 and 1
- Invariant to linear transforms
- Independent of the average gray levels of the pattern and the image window

With Normalization

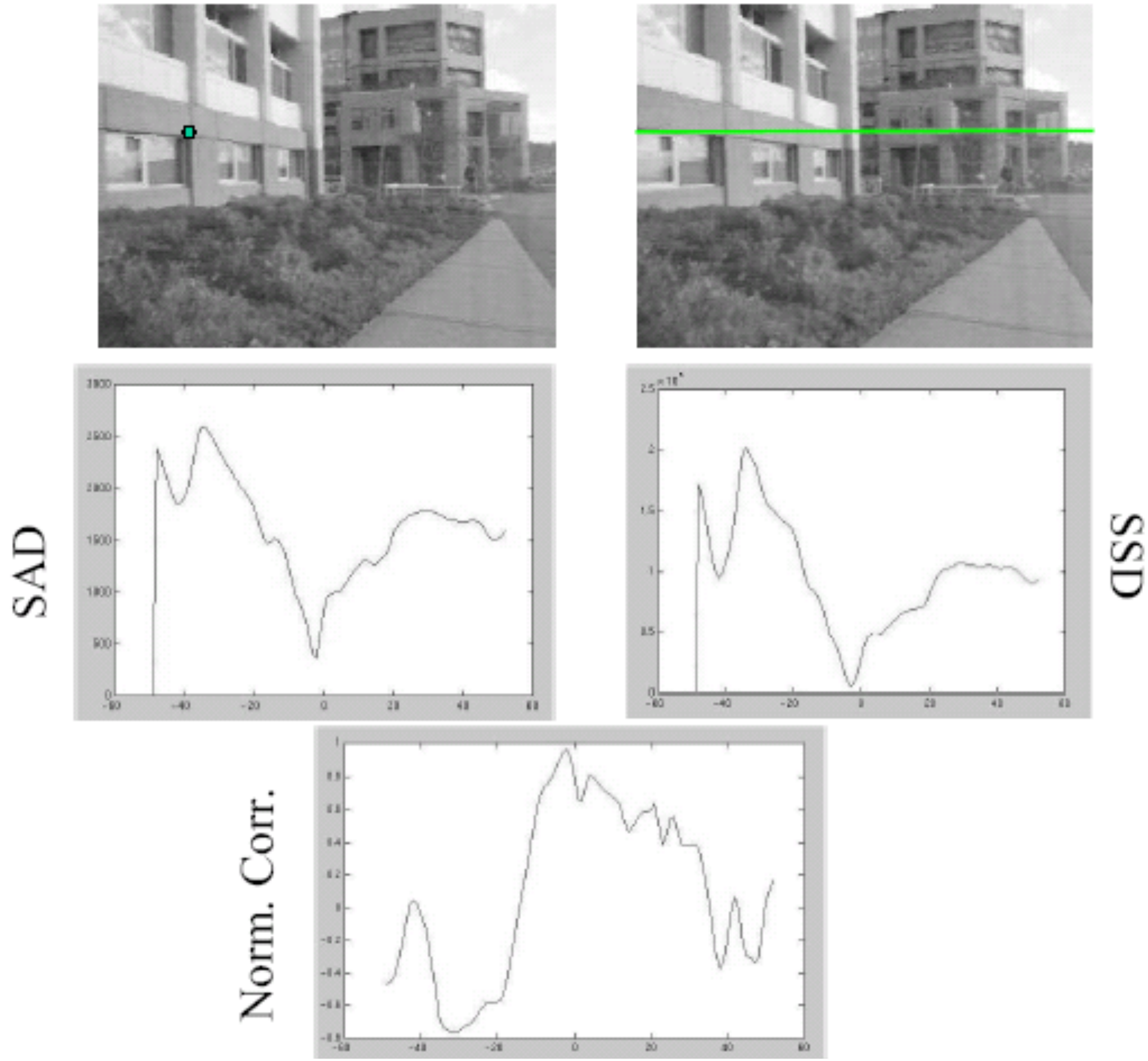
Image

Pattern

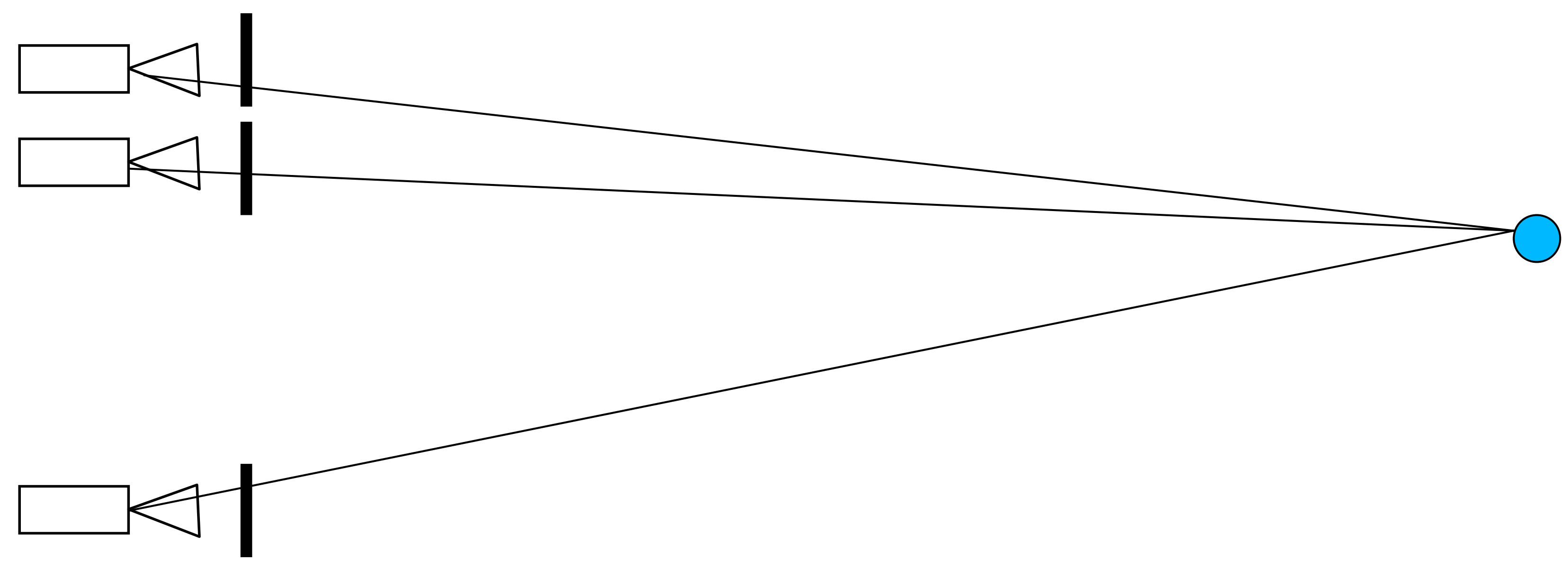
Normalized Correlation



Point of maximum correlation

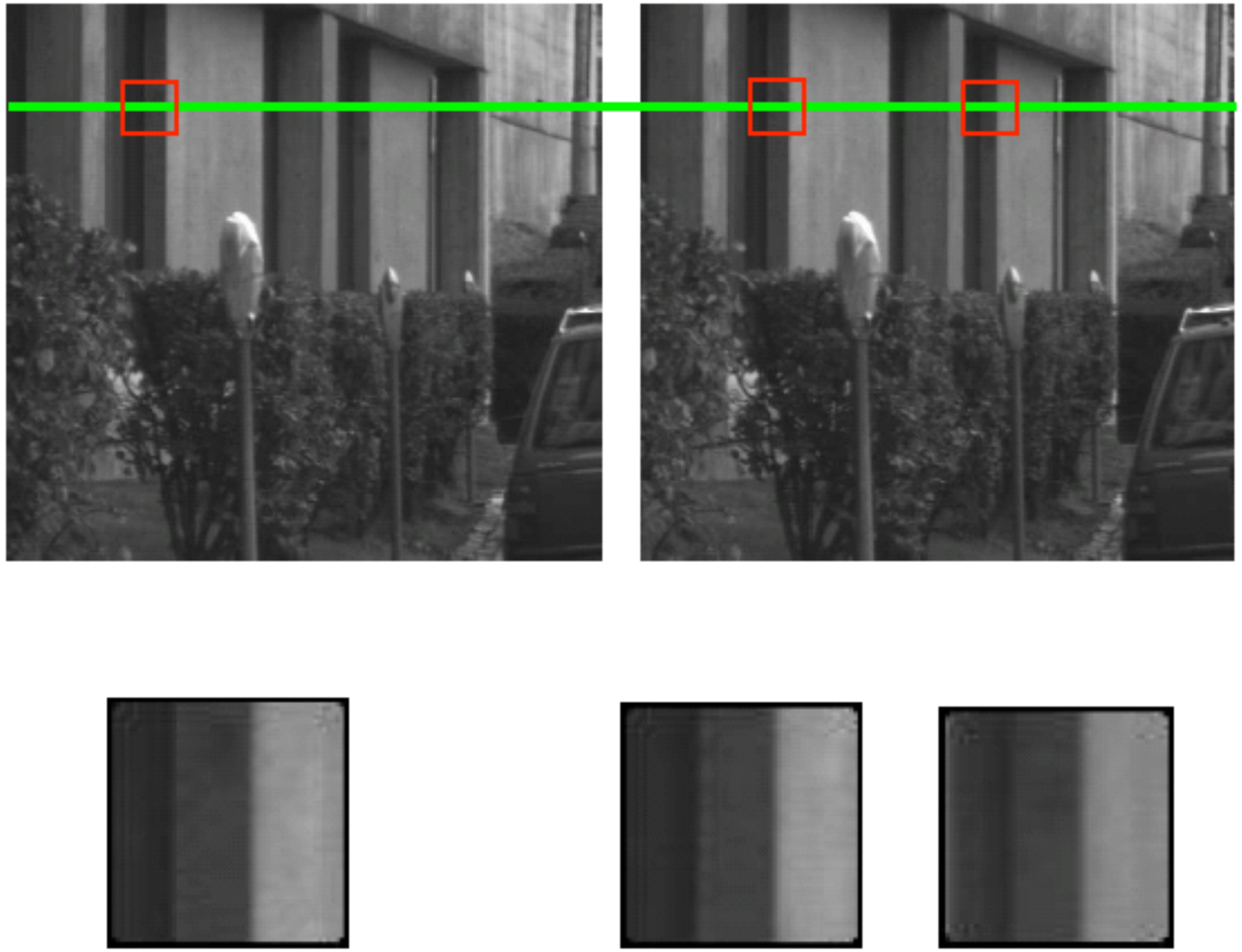

SAD
SSD
Norm. Corr.

Baseline



- Short Baseline
  - Good Matching
  - Few Occlusions
  - Poor Precision
- Long Baseline
  - More Difficult Matching
  - More Occlusions
  - Better Precision

Ambiguity



Stereo Matching Functions

SSD: (Sum of Squared Differences)

$$\psi(I_l(x,y), I_r(x+d,y)) = (I_l(x,y) - I_r(x-d,y))^2$$

SAD: (Sum of Absolute Differences)

$$\psi(I_l(x,y), I_r(x+d,y)) = \left|I_l(x,y) - I_r(x-d,y)\right|$$

Correlation:

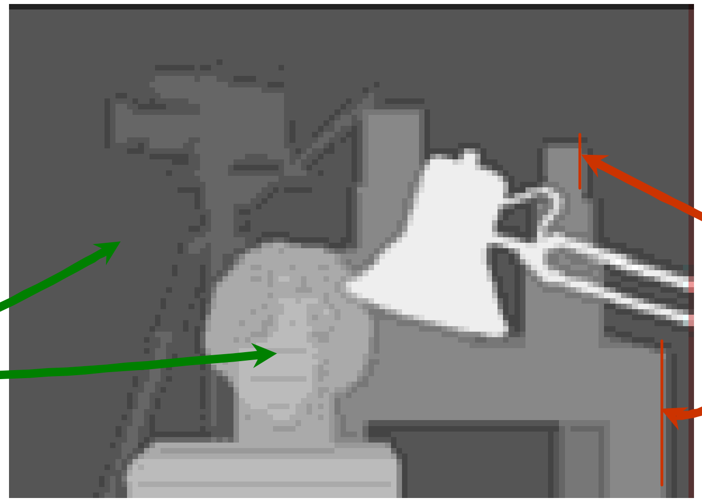
$$\psi(I_l(x,y), I_r(x+d,y)) = I_l(x,y).I_r(x-d,y)$$

Normalized Correlation:

$$\psi(I_l(x,y), I_r(x+d,y)) = \frac{I_l(x,y).I_r(x-d,y) - \bar{I}_l\bar{I}_r}{\sigma_l\sigma_r(d)}$$

Energy Minimization for Stereo

Disparity continuous in most places,

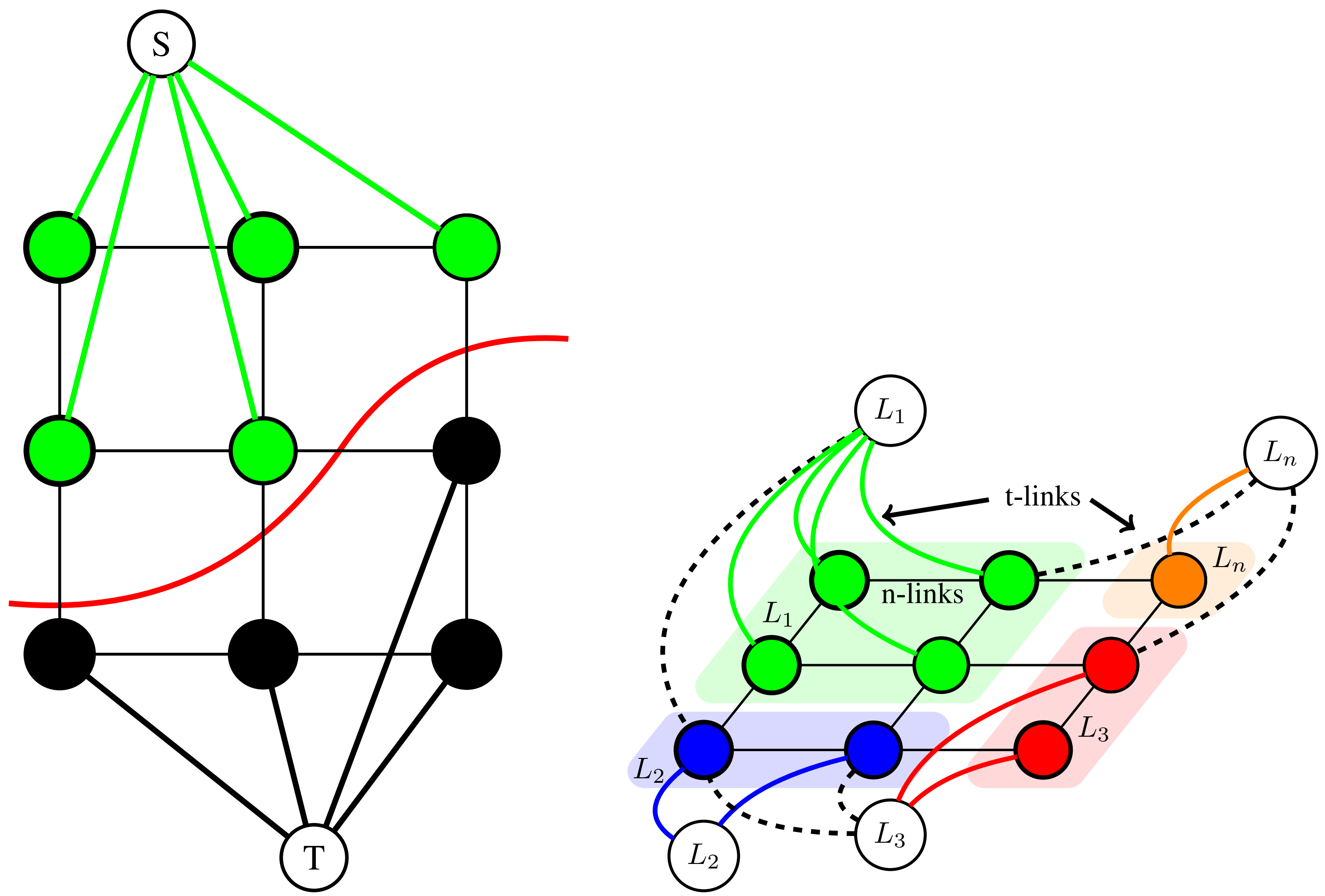


except at depth discontinuities

- Matching pixels should have similar intensities.
- Most nearby pixels should have similar disparities

$$\text{Minimize } \sum \left[I_1(x + D(x,y), y) - I_2(x,y) \right]^2 + \lambda \sum \left[D(x+1,y) - D(x,y) \right]^2 + \mu \sum \left[D(x,y+1) - D(x,y) \right]^2$$

Graph Cuts

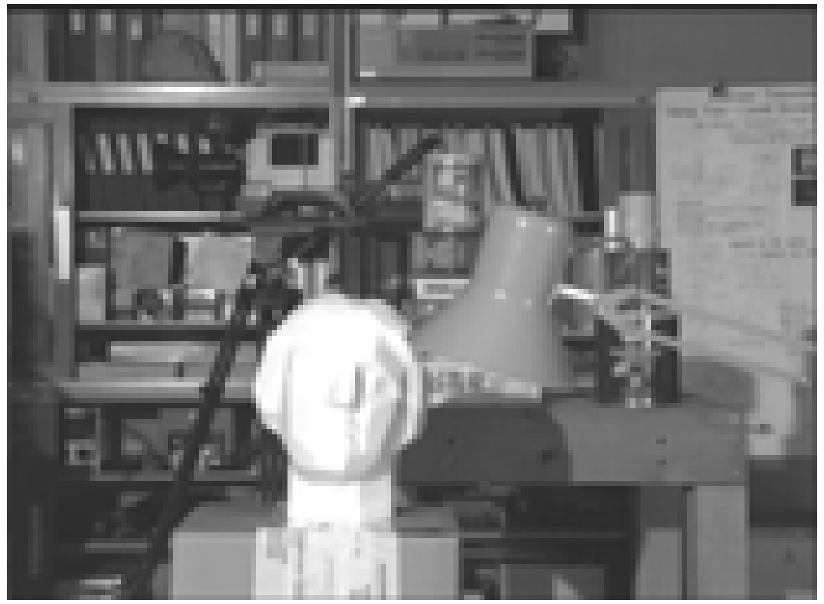


(a) Binary Seg (b) Multi-way Cut

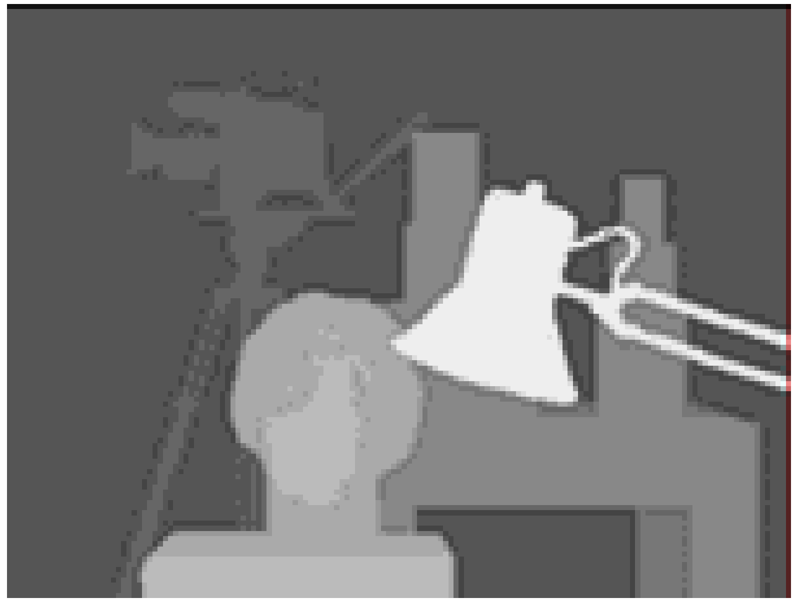
- Stereo is a labeling problem
- Graph cut corresponds to a labeling.
  - Assign edge weights cleverly so that the min-weight cut gives the minimum energy!

Graph Cuts Improvement

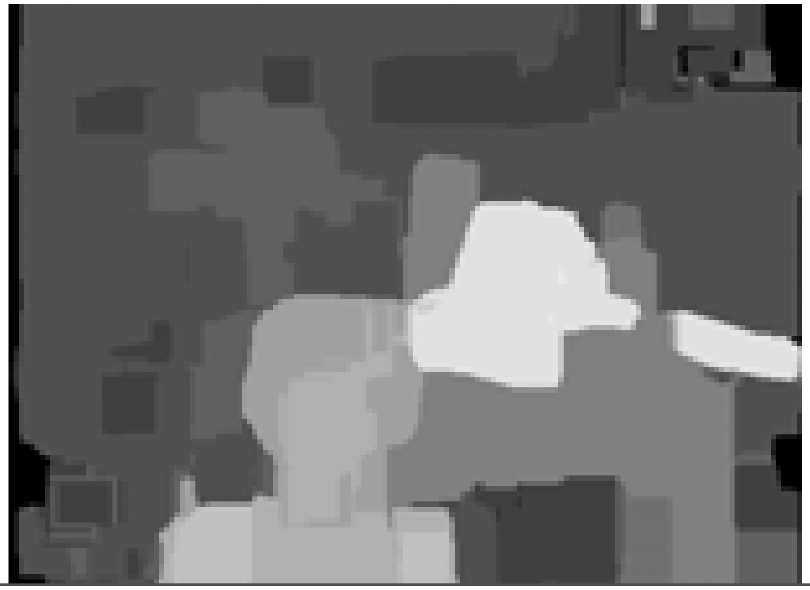
left image



true disparities



Normalized correlation



Graph Cuts