Lec. 4

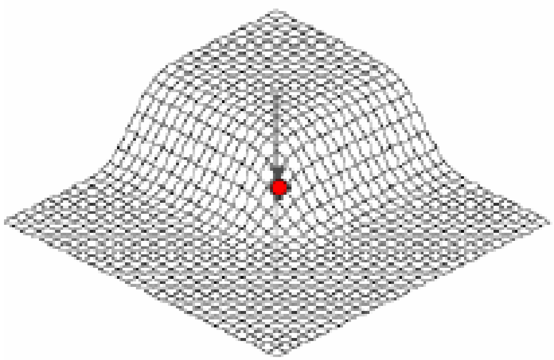
Interest Points, Line and Corners Detection

Invariant Local Image Features

Assist. Prof. Dr. Saad Albawi

What is an interest point

- Expressive texture
  - The point at which the direction of the boundary of object changes abruptly
  - Intersection point between two or more edge segments



Interest Points

- What do we mean with Interest Point Detection in an Image
- Goal: Find Same features between multiple images taken from different position or time

For image registration, need to obtain correspondence between images.

Basic idea:

- detect feature points, also called keypoints
- match feature points in different images

Want feature points to be detected consistently and matched correctly.

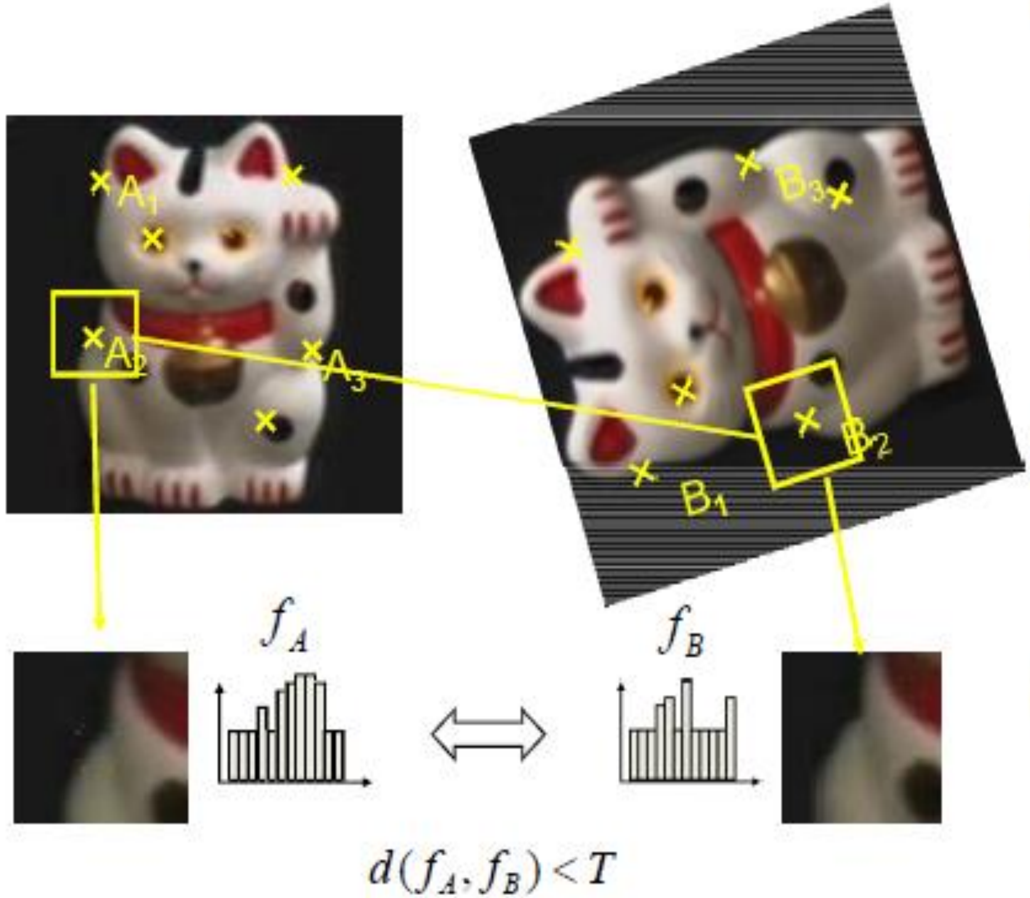
Properties of Interest Point Detectors

- Detect all (or most) true interest points
- No false interest points
- Well localized.
- Robust with respect to noise.
- Efficient detection

Applications

- Image alignment
- Image Stitching
- 3D reconstruction
- Object recognition
- Indexing and database retrieval
- Object tracking
- Robot navigation

Overview of Keypoint Matching



1. Find a set of distinctive key-points
2. Define a region around each keypoint
3. Extract and normalize the region content
4. Compute a local descriptor from the normalized region
5. Match local descriptors

K. Grauman, B. Leibe

Not all patches are created equal

Intuition: this would be a good patch for matching, since it is very distinctive (there is only one patch in the second frame that looks similar)

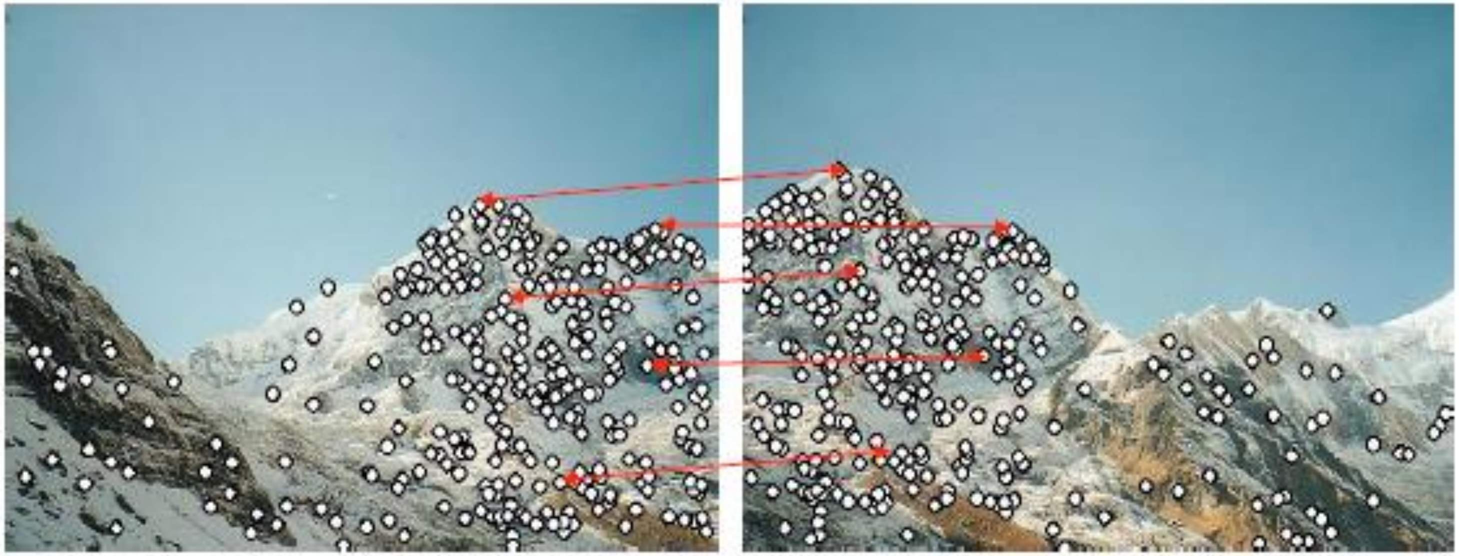
Not all patches are created equal

Intuition: this would be a bad patch for matching, since it is **not** very distinctive (there are many similar patches in the second frame)

Matching with Features

Detect feature points in both images

Find corresponding pairs



Matching with Features

- Detect feature points in both images
- Find corresponding pairs
- Use these matching pairs to align images - the required mapping is called a homography.

Matching with Features

- Problem 1:
  – Detect the same point independently in both images

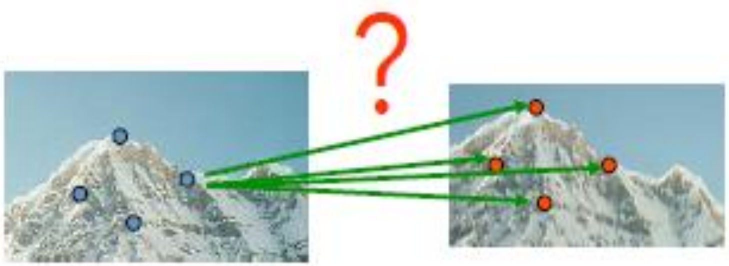
counter-example:

no chance to match!

We need a repeatable detector

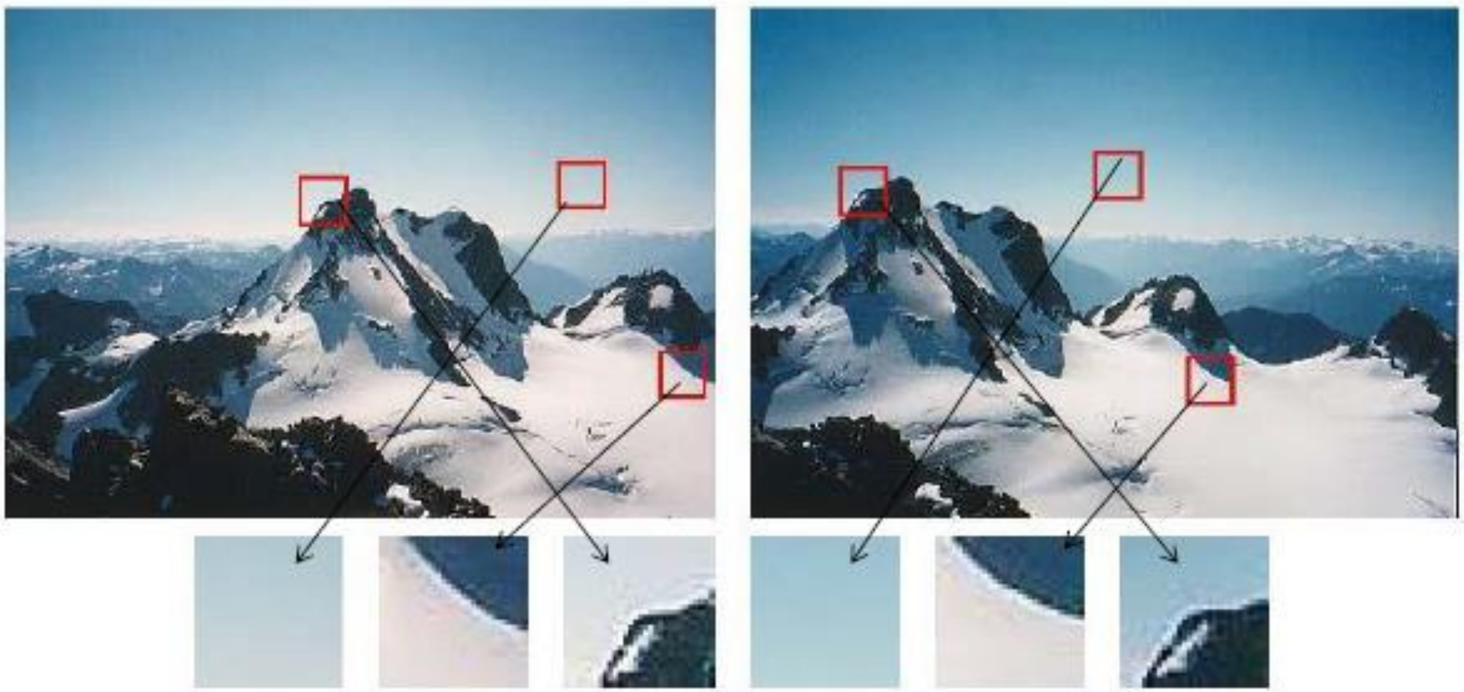
Matching with Features

- Problem 2:
  - For each point correctly recognize the corresponding one



We need a reliable and distinctive descriptor

Some patches can be localized or matched with higher accuracy than others.



Line Identification

Identifying parametric edges

- Can we identify lines?
- Can we identify curves?
- More general
  - Can we identify circles/ellipses?
- Voting scheme called Hough Transform

The Hough Transform

- A mathematical method designed to find lines in images.
- It can be used for linking the results of edge detection, turning potentially sparse, broken, or isolated edges into useful lines that correspond to the actual edges in the image.

How to identify lines?

- For each edge point
  - Add intensity to the corresponding line in Hough space
- Each edge point votes on the possible lines through them
- If a line exists in the image space, that point in Hough space will get many votes and hence high intensity
- Find maxima in Hough space
- Find lines by equations y – mx+b

Basic Hough Transform

1. Initialize H[d, θ]=0
2. for each edge point I[x,y] in the image

    for $\theta = 0$ to 180

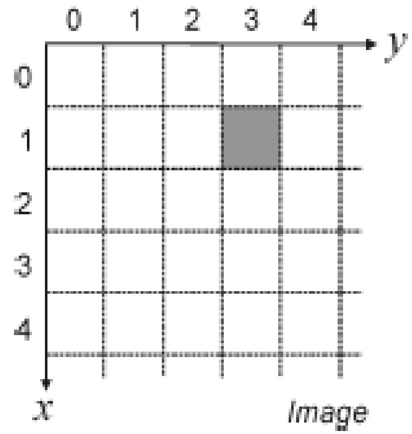
    $d = x\cos\theta + y\sin\theta$

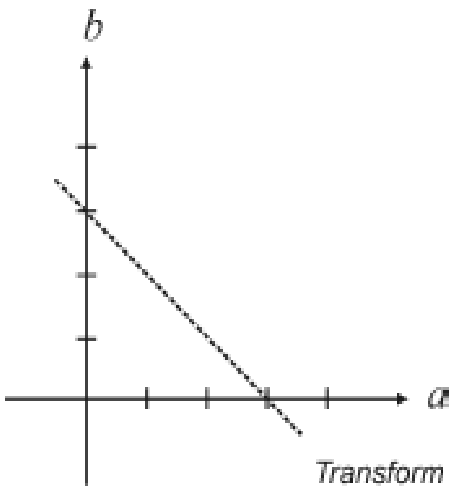
    H[d, θ] += 1
3. Find the value(s) of (d, θ) for max H[d, θ]

A similar procedure can be used for identifying circles, squares, or other shape with appropriate change in Hough parameterization.

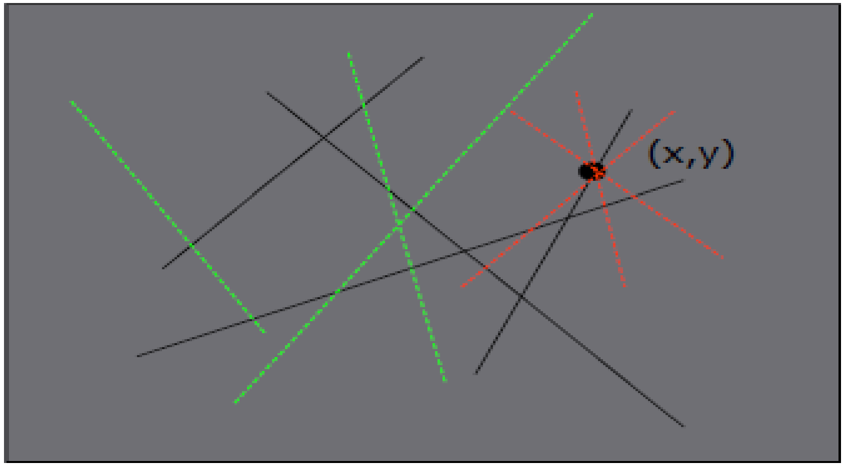
The Hough transform

- Let *(x,y)* be the coordinates of a point in a binary image (containing thresholded edge detection results).
- The Hough transform stores in an *accumulator array* all pairs *(a,b)* that satisfy the equation $y = ax + b$. The *(a,b)* array is called the *transform array*.
  - Example:, the point *(x,y)* = (1,3) in the input image will result in the equation $b = -a + 3$, which can be plotted as a line that represents all pairs *(a,b)* that satisfy this equation.





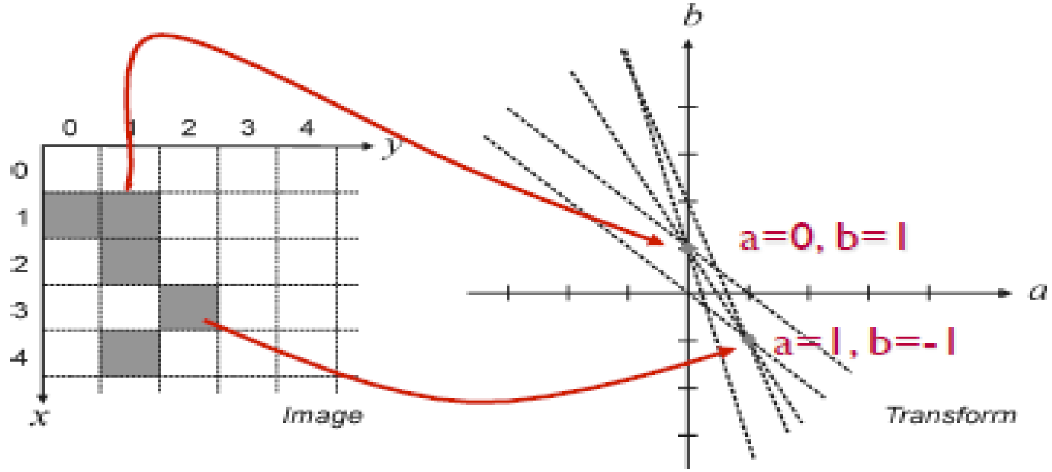
Hough Transform



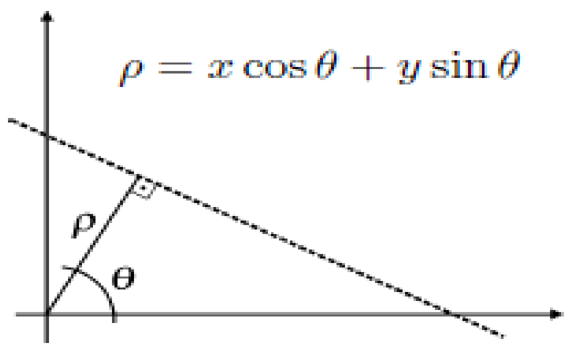
- Only a few lines can pass through (x,y)
  - $mx+b$
- Consider (m,b) space
- Red lines are given by a line in that space
  - $b = y - mx$
- Each point defines a line in the Hough space
- Each line defines a point (since same m,b)

I

- Since each point in the image will map to a line in the transform domain, repeating the process for other points will result in many intersecting lines, one per point.
- The meaning of two or more lines intersecting in the transform domain is that the points to which they correspond are aligned in the image.
- The points with the greatest number of intersections in the transform domain correspond to the longest lines in the image.



- Describing lines using the equation *y* = *ax* + *b* (where *a* represents the gradient) poses a problem, though, since vertical lines have infinite gradient.
- This limitation can be circumvented by using the *normal representation* of a line, which consists of two parameters: ρ and θ.



Hough Transform Example

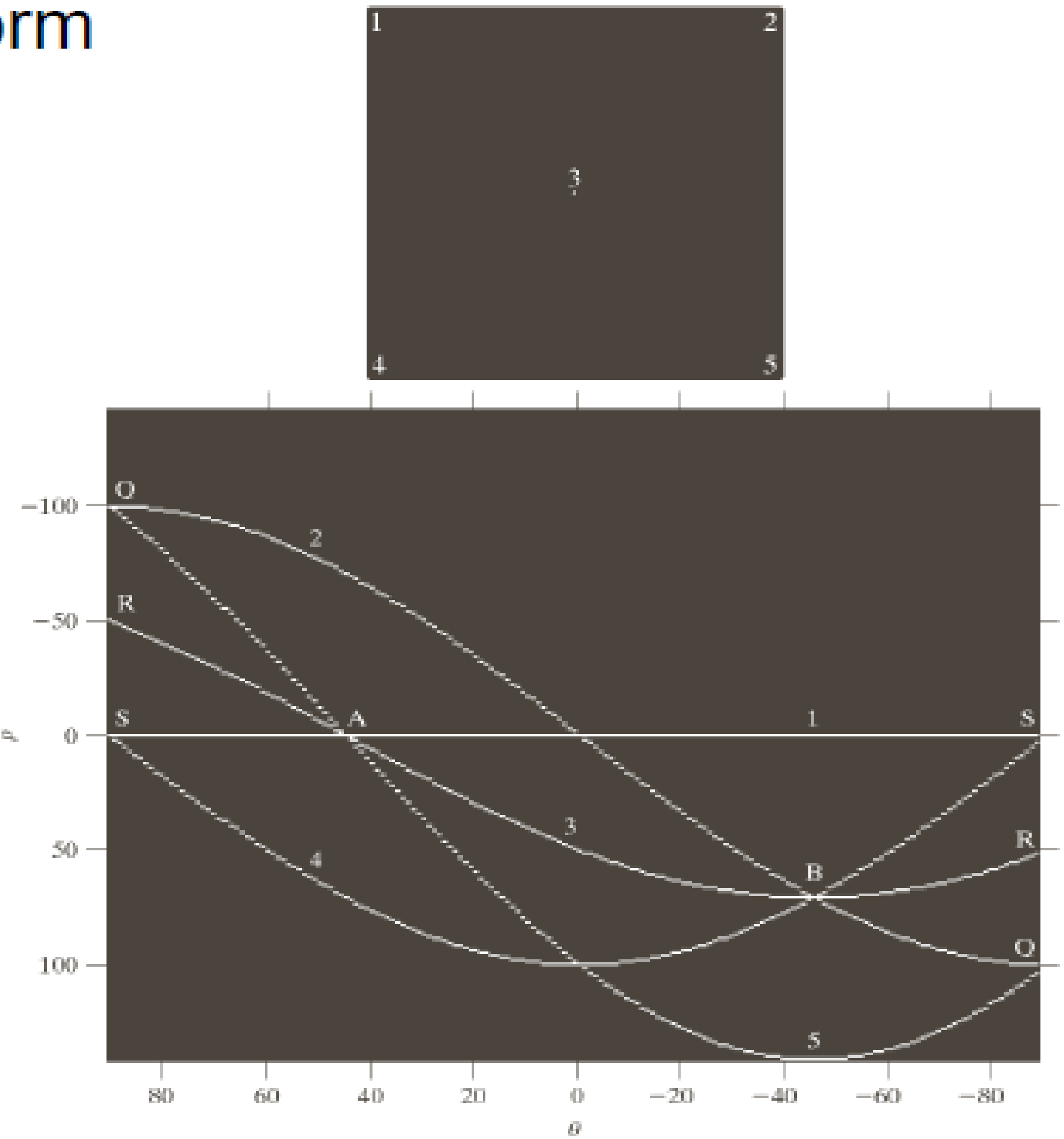
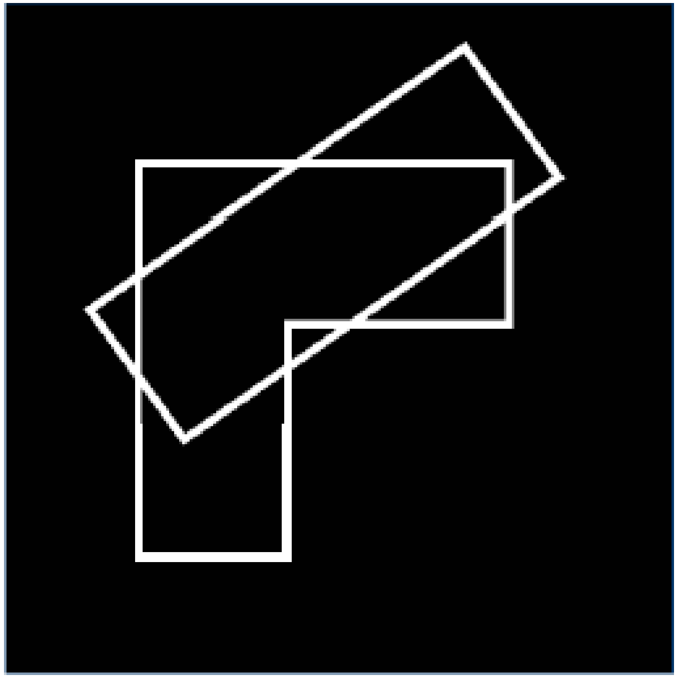


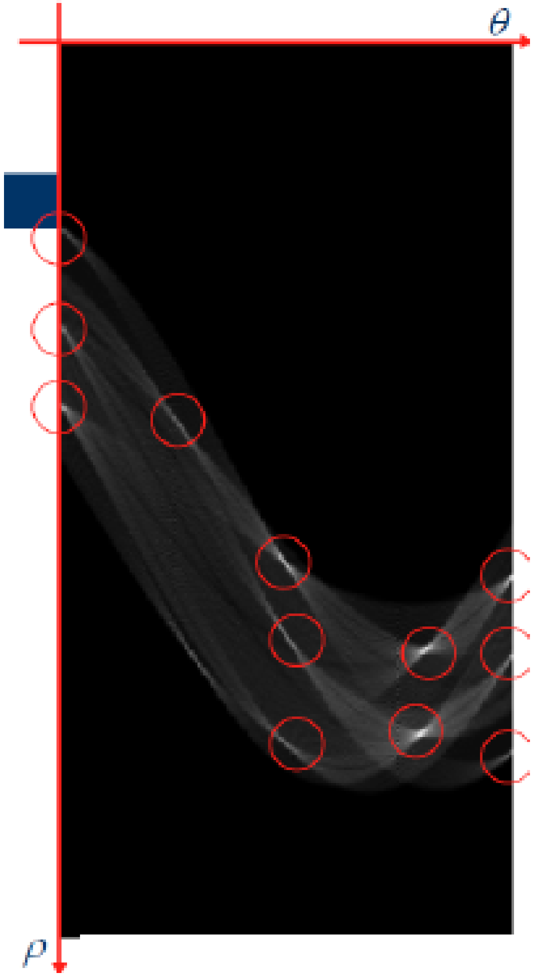
FIGURE 10.33
(a) Image of size 101 × 101 pixels, containing five points.
(b) Corresponding parameter space. (The points in (a) were enlarged to make them easier to see.)

The Hough Transform Algorithm

1. Create a 2D array corresponding to a discrete set of values for ρ and θ. Each element in this array is referred to as an *accumulator cell.*
   1. Increments too big: May not distinguish different lines
   2. Increments oo small: Noise may cause lines to be missed
2. For each pixel (*x*,*y*) in the image and for each chosen value of θ, compute $x \cos \theta + y \sin \theta$ and write the result in the corresponding position (ρ, θ) in the accumulator array.
3. The highest values in the (ρ, θ) array will correspond to the most relevant lines in the image.

Line FittingHough Transform

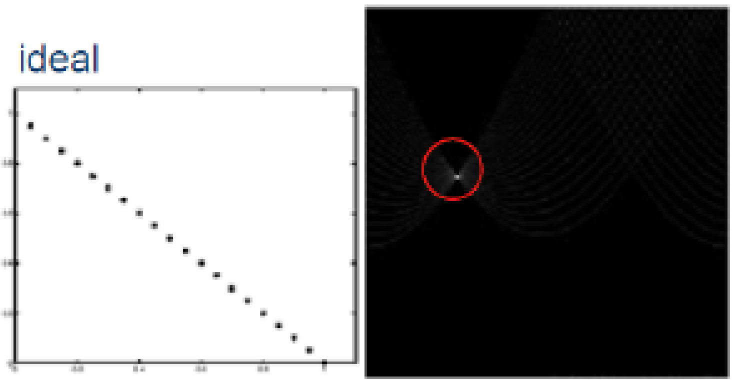


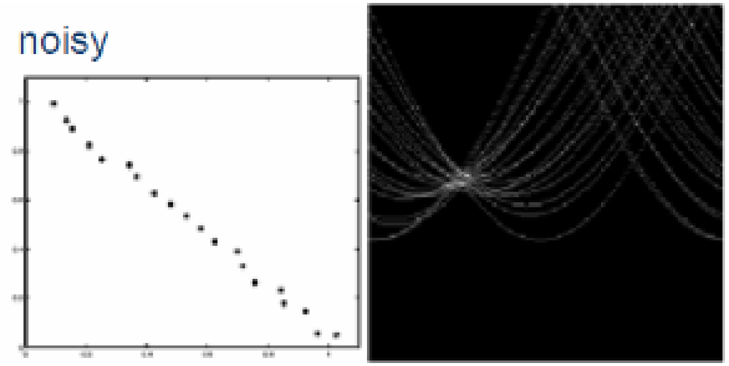


Noise vs. Increments

- ρ and θ increments too big: May not distinguish different lines

- ρ and θ increments too small: Noise may cause lines to be missed

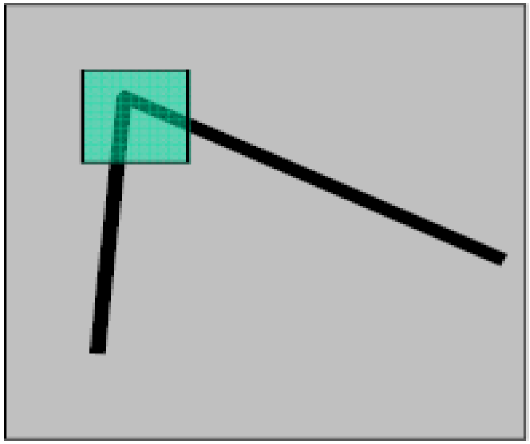




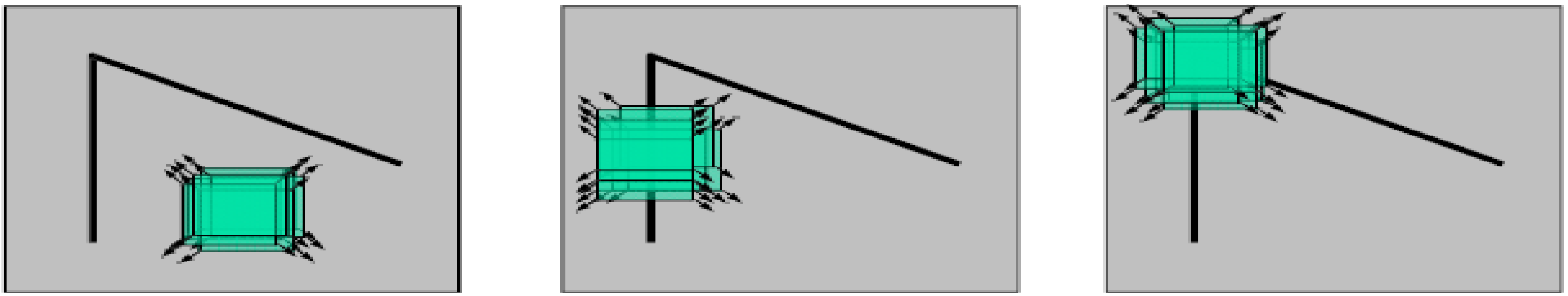
Corner Detection

Harris Corner Detector

- Corner point can be recognized in a window
- Shifting a window in any direction should give a large change in intensity



Basic Idea



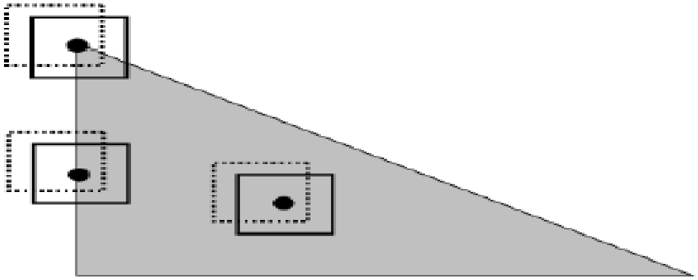
"flat" region:
no change in
all directions

"edge":
no change along
the edge direction

"corner":
significant change
in all directions

Corner Detection: Basic Idea

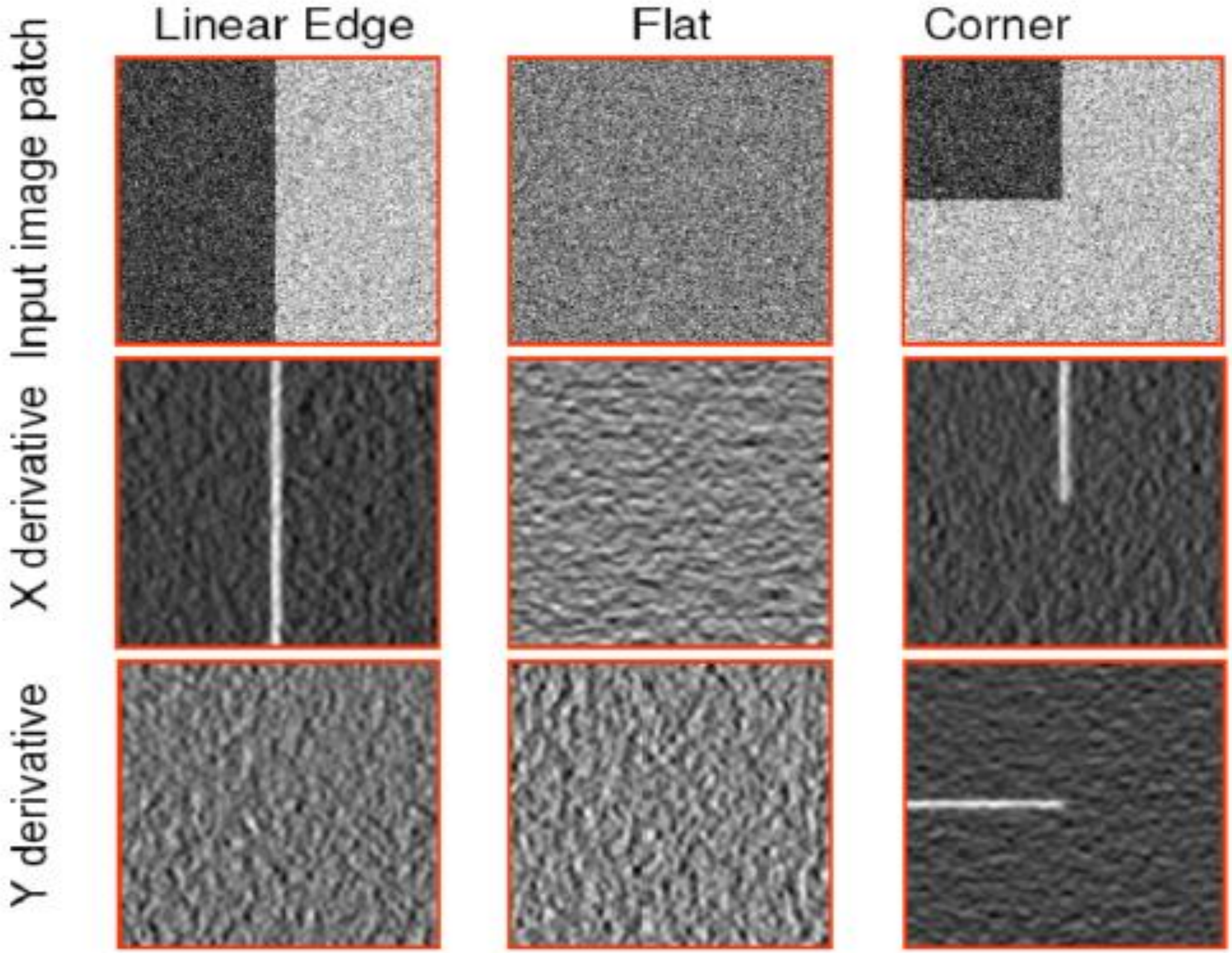
- Where two edges meet
- Where X and Y gradients are both high??



Harris Corner Detector

- A corner is a point around which the gradient has two or more dominant directions
- Corners can be repeatably detected under varying illumination and view point changes

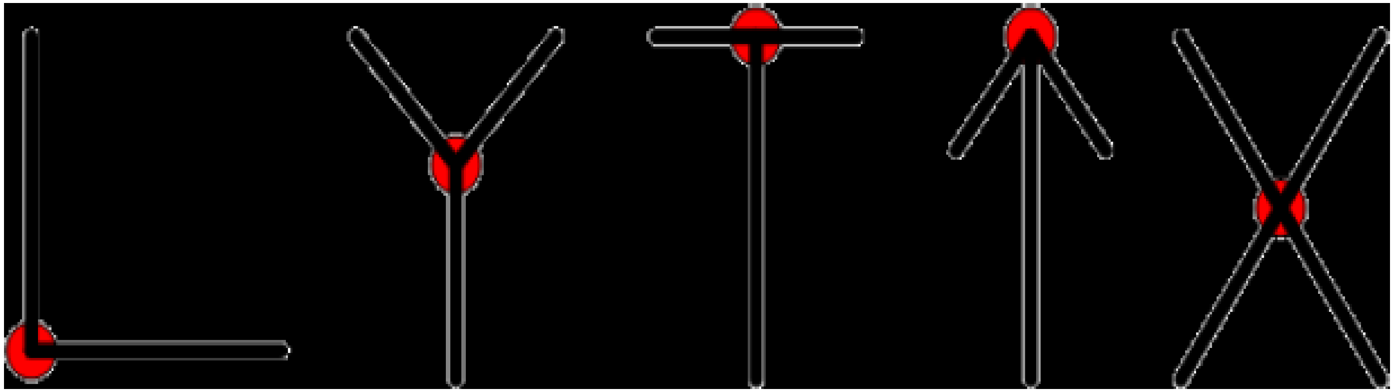
- the corner can be defined as
  1. an intersection of two edges
  2. a (important) point where two dominant directions (gradients) exist
- every corner is an important point, but not the other way around
- a corner detection algorithm needs to be very robust



Obrázek: Different regions and their derivatives.

Corner Types

Example of L-junction, Y-junction, T-junction, Arrow-junction, and X-junction corner types



Correlation

$$f \otimes h = \sum_k \sum_l f(k,l)h(k,l)$$

$f =$ Image

$h =$ Kernel

f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

$\otimes$

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$$f * h = f_1h_1 + f_2h_2 + f_3h_3 + f_4h_4 + f_5h_5 + f_6h_6 + f_7h_7 + f_8h_8 + f_9h_9$$

$$f \otimes h = \sum_k \sum_l f(k,l)h(k,l)$$

Cross correlation

$$f \otimes f = \sum_k \sum_l f(k,l)f(k,l)$$

Auto correlation

Mathematics of Harris Detector

- Change of intensity for the shift (u,v)

$$E(u,v) = \sum_{x,y} [\underbrace{I(x+u, y+v)}_{\text{shifted intensity}} - \underbrace{I(x,y)}_{\text{intensity}}]^2$$

Auto-correlation

Taylor Series

$f(x)$ Can be represented at point a in terms of its derivatives

$$f(x) = f(a) + (x-a) f_x + \frac{(x-a)^2}{2!} f_{xx} + \frac{(x-a)^3}{3!} f_{xxx} + \ldots$$

Express $I(x+u, y+v)$ at (x, y) :

$$I(x+u, y+v) = I(x, y) + I_x(x+u-x) + I_y(y+v-y)$$

$$I(x+u, y+v) = I(x, y) + I_x u + I_y v$$

$$E(u,v) = \sum_{x,y}[\underbrace{I(x+u, y+v)}_{\text{shifted intensity}} - \underbrace{I(x, y)}_{\text{intensity}}]^2$$

$$E(u,v) = \sum_{x,y}[\underbrace{I(x, y) + uI_x + vI_y}_{\text{shifted intensity}} - \underbrace{I(x, y)}_{\text{intensity}}]^2$$

Taylor Series

$$E(u,v) = \sum_{x,y}[uI_x + vI_y]^2$$

$$E(u,v) = \sum_{x,y}\left[\begin{pmatrix} u & v \end{pmatrix}\begin{pmatrix} I_x \\ I_y \end{pmatrix}\right]^2$$

$$E(u,v) = \sum_{x,y}\begin{pmatrix} u & v \end{pmatrix}\begin{pmatrix} I_x \\ I_y \end{pmatrix}\begin{pmatrix} I_x & I_y \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix}$$

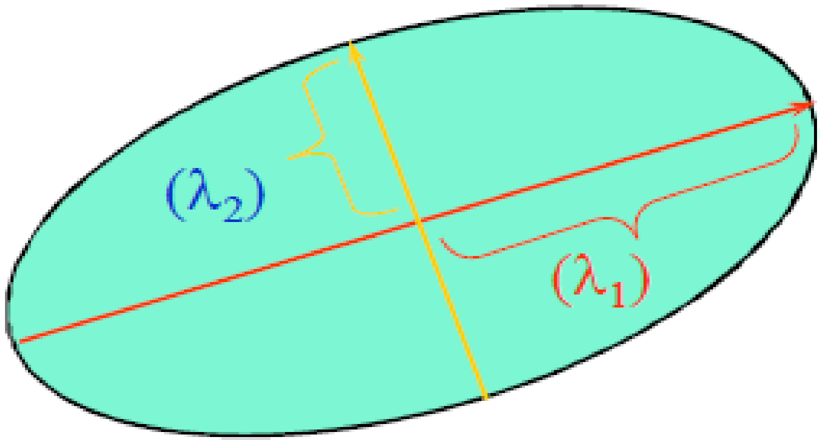
$$M = \sum_{x,y}\begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

$$E(u,v) = \begin{pmatrix} u & v \end{pmatrix}\left[\sum_{x,y}\begin{pmatrix} I_x \\ I_y \end{pmatrix}\begin{pmatrix} I_x & I_y \end{pmatrix}\right]\begin{pmatrix} u \\ v \end{pmatrix}$$

$$E(u,v) = \begin{pmatrix} u & v \end{pmatrix} M \begin{pmatrix} u \\ v \end{pmatrix}$$

$$E(u,v) = (u \quad v) M \begin{pmatrix} u \\ v \end{pmatrix} \qquad M = \sum_{x,y} \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

- E(u,v) is an equation of an ellipse.
- Let λ_1 and λ_2 be eigenvalues of M



Eigen Vectors and Eigen Values

The eigen vector, x, of a matrix A is a special vector, with the following property

$$Ax = \lambda x$$

Where λ is called eigen value

To find eigen values of a matrix A first find the roots of:

$$\det(A - \lambda I) = 0$$

Then solve the following linear system for each eigen value to find corresponding eigen vector

$$(A - \lambda I)x = 0$$

Example

$$A = \begin{bmatrix} -1 & 2 & 0 \\ 0 & 3 & 4 \\ 0 & 0 & 7 \end{bmatrix}$$

Eigen Values

$$\lambda_1 = 7,\ \lambda_2 = 3, \lambda_3 = -1$$

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 4 \\ 4 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Eigen Vectors

$$\det(A - \lambda I) = 0$$

$$\det\left(\begin{bmatrix} -1 & 2 & 0 \\ 0 & 3 & 4 \\ 0 & 0 & 7 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) = 0$$

$$\det\left(\begin{bmatrix} -1-\lambda & 2 & 0 \\ 0 & 3-\lambda & 4 \\ 0 & 0 & 7-\lambda \end{bmatrix}\right) = 0$$

$$(-1-\lambda)((3-\lambda)(7-\lambda)-0) = 0$$

$$(-1-\lambda)(3-\lambda)(7-\lambda) = 0$$

$$\lambda = -1, \quad \lambda = 3, \quad \lambda = 7$$

Eigen Vectors

$\lambda = -1$

$$(A - \lambda I)x = 0$$

$$\left(\begin{bmatrix} -1 & 2 & 0 \\ 0 & 3 & 4 \\ 0 & 0 & 7 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right)\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 2 & 0 \\ 0 & 4 & 4 \\ 0 & 0 & 8 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$0 + 2x_2 + 0 = 0$$

$$0 + 4x_2 + 4x_3 = 0$$

$$0 + 0 + 8x_3 = 0$$

$$x_1 = 1, \quad x_2 = 0, \quad x_3 = 0$$

$$\mathbf{x_1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Autocorrelation Calculation

$$R(u,v) = \sum_{(x,y)\in W} I(x+u, y+v) I(x,y)$$

Autocorrelation can be approximated by sum-squared-difference (SSD):

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

SSD Calculation

Let $A = \begin{bmatrix} \sum_W I_x^2 & \sum_W I_x I_y \\ \sum_W I_x I_y & \sum_W I_y^2 \end{bmatrix}$ and $u = \begin{bmatrix} u \\ v \end{bmatrix}$

then $E(u,v) = u^T A u$

Harris Corner Detector : Mathematics

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

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

$$M = \sum w(x,y) \Delta I (\Delta I)^T$$

- Window function can be simply w = 1
- Product of first derivatives of the image

$$E(u,v) = \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

Eigenvalues and Eigenvectors of the Auto-correlation Matrix

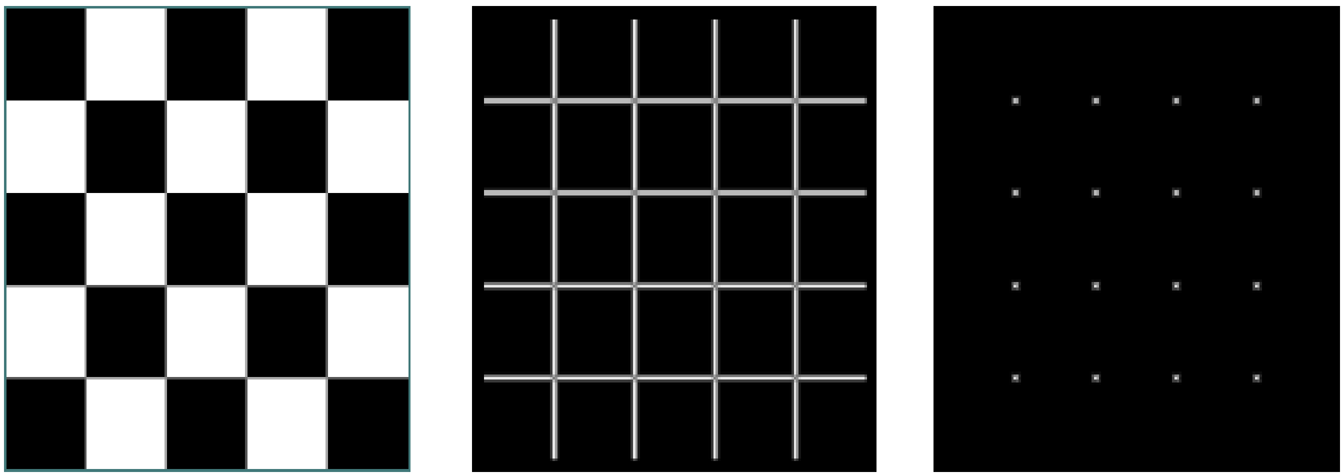
lower limit → $\lambda_- \leq E(u,v) = u^T A u \leq \lambda_+$ ← upper limit

where λ_+ and λ_- are the two eigenvalues of A.

The eigenvector e_+ corresponding to λ_+ gives the direction of **largest** increase ***E***,

while the eigenvector e_- corresponding to λ_- gives the direction of **smallest** increase in ***E***.

λ_+ for Edges, λ_- for Corners

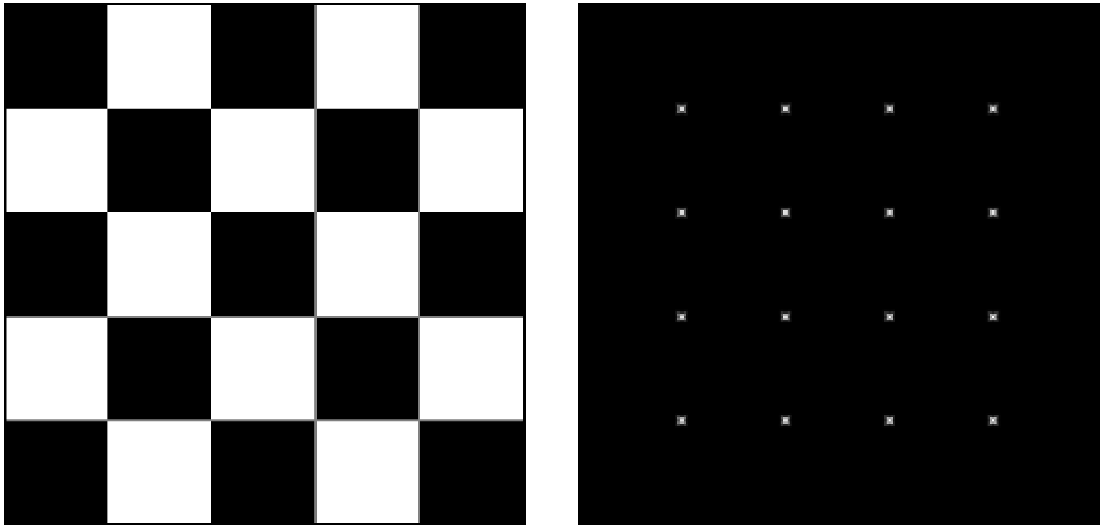


I λ_+ λ_-

Feature detection (interest point detection) summary

Here's what you do

- Compute the gradient at each point in the image
- Create the ***H*** matrix from the entries in the gradient
- Compute the eigenvalues
- Find points with large λ_- (i.e., $\lambda_- >$ threshold)
- Choose points where λ_- is a local maximum as interest points

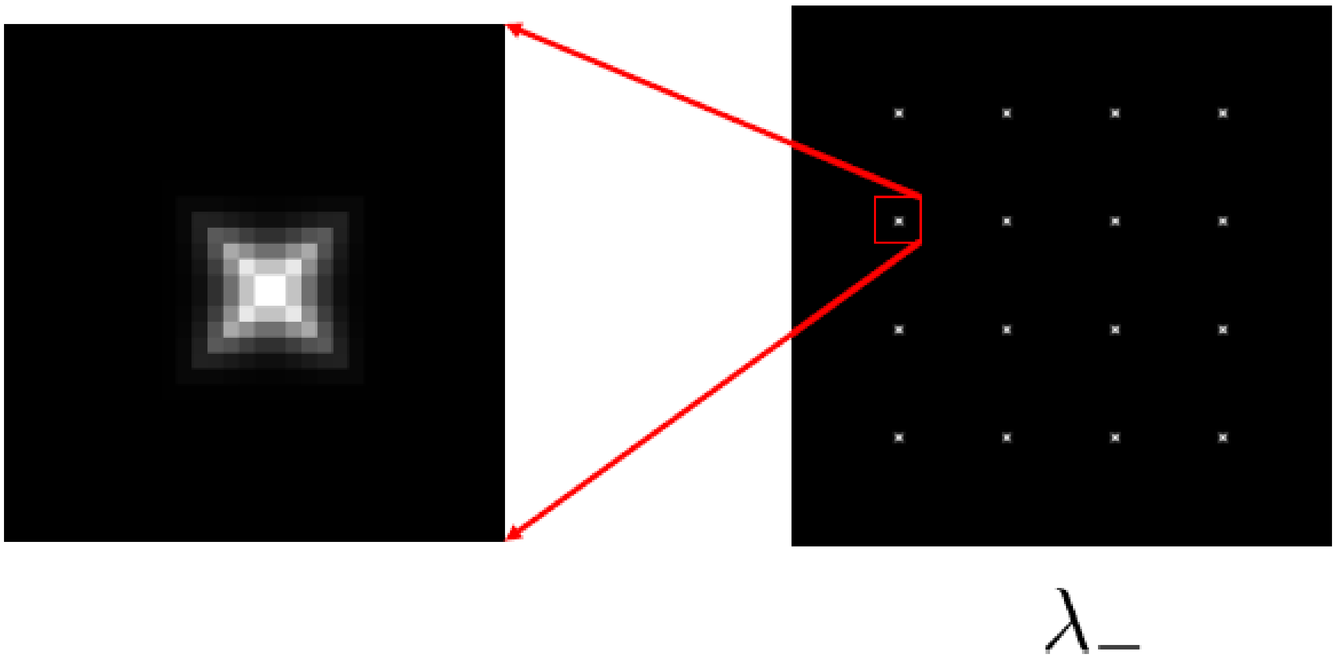


I λ_-

Feature detection summary

Here's what you do

- Compute the gradient at each point in the image
- Create the ***H*** matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response (λ_- > threshold)
- Choose those points where λ_- is a local maximum as features (interest points)



(1) If both λ_i are small, then feature does not vary much in any direction. $\Rightarrow$ uniform region (bad feature)

(2) If the larger eigenvalue $\lambda_1 \gg \lambda_2$, then the feature varies mainly in the direction of $\mathbf{v}_1$. $\Rightarrow$ edge (bad feature)

(3) If both eigenvalues are large, then the feature varies significantly in both directions. $\Rightarrow$ corner or corner-like (good feature)

(4) In practice, I has a maximum value (e.g., 255).
So, λ_1, λ_2 also have an upper bound.
So, only have to check that $\min(\lambda_1, \lambda_2)$ is large enough.

Harris Detector: Maths & Intuition

Window-averaged squared change of intensity induced by shifting the image data by [u,v]:

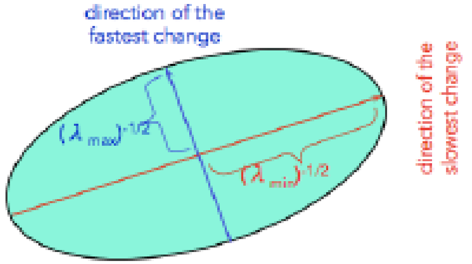
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Ellipse

Window function

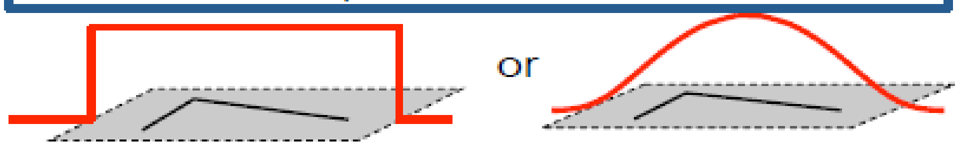
Shifted intensity

Intensity



For nearly constant patches, this will be near 0.
For very distinctive patches, this will be larger.
Hence... we want patches where E(u,v) is LARGE.

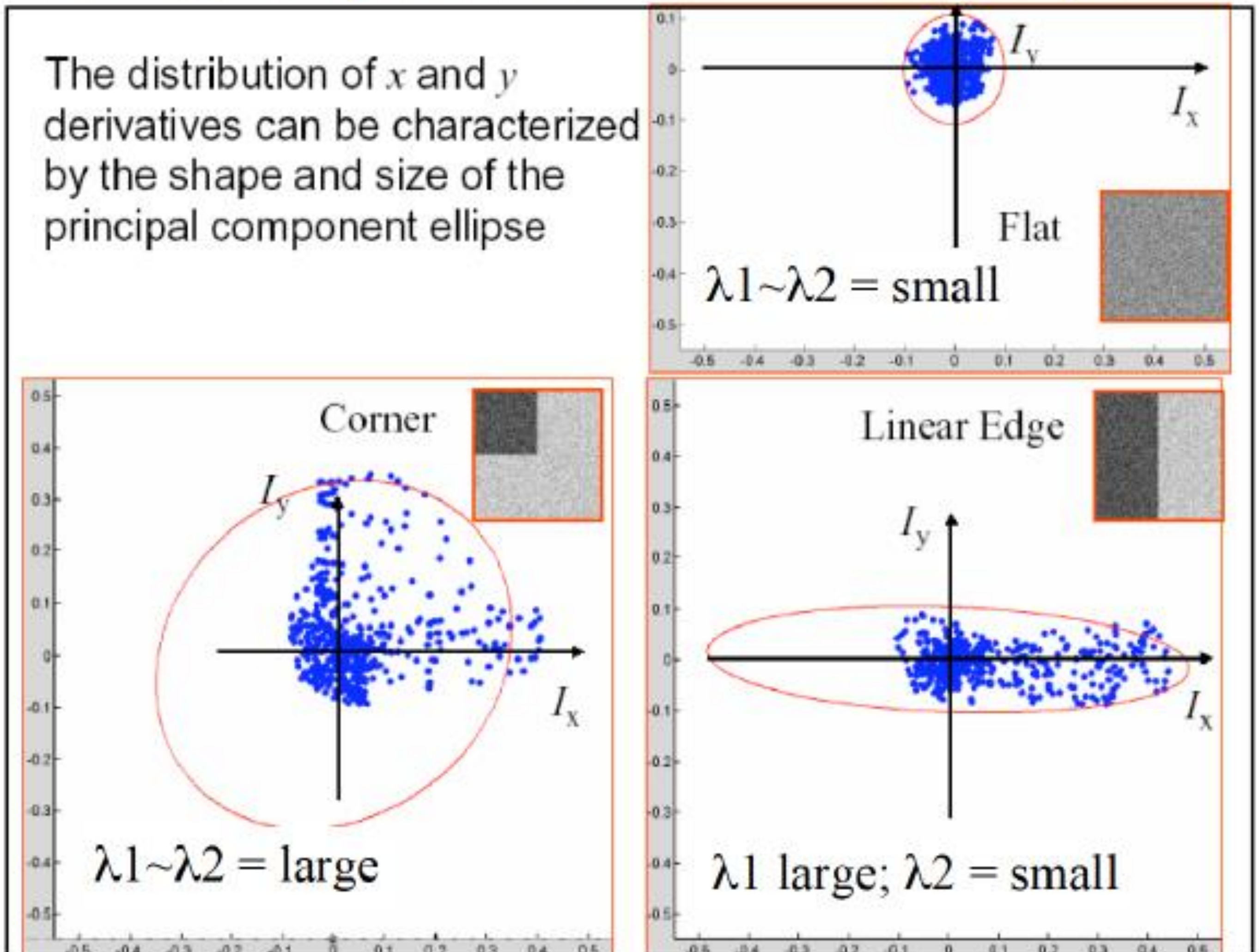
Window function w(x,y) =



1 in window, 0 outside

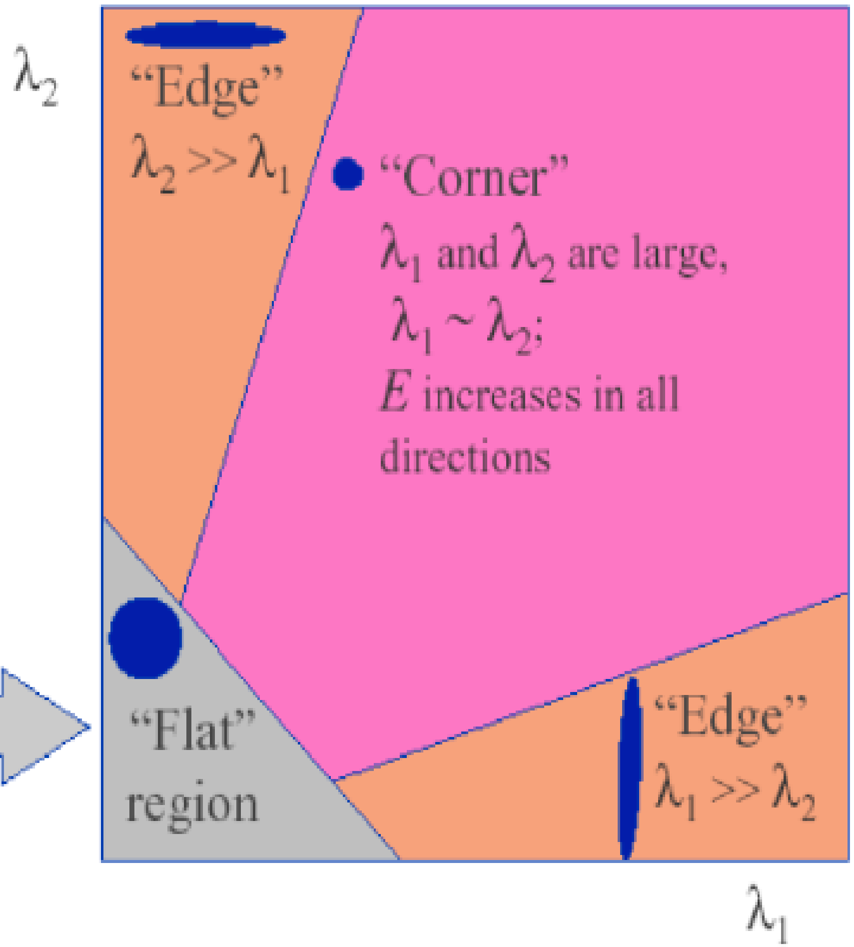
Gaussian

Fitting ellipses to each set of points



Classification via Eigenvalues

Classification of image points using eigenvalues of M:



λ_1 and λ_2 are small; E is almost constant in all directions

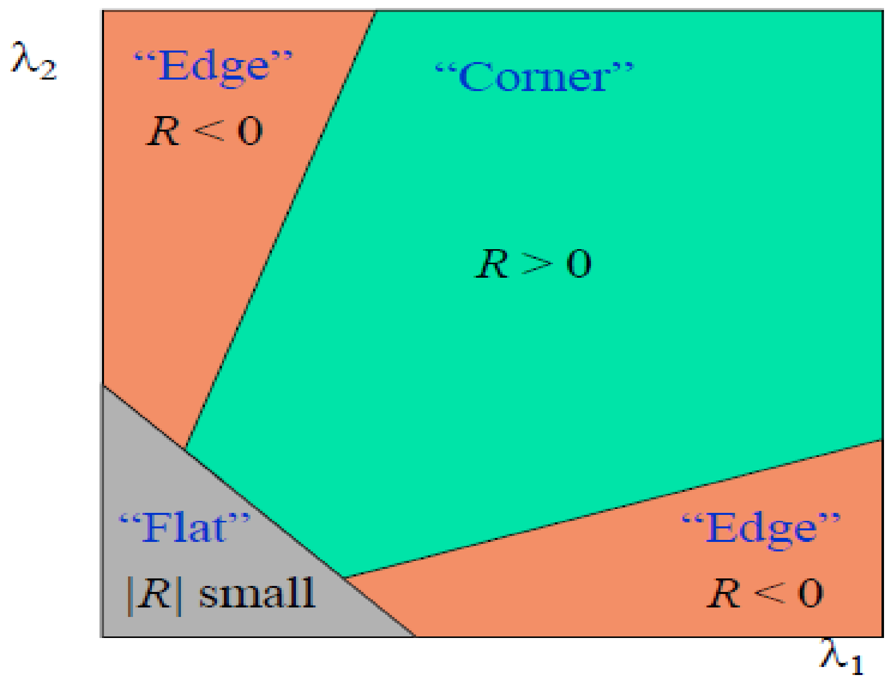
- Measure of cornerness in terms of λ_1, λ_2

$$M = SDS^{-1} \qquad D = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

$$R = \det D - k(trace\, D)^2 \qquad R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2$$

(k – empirical constant, k = 0.04-0.06)

- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region



Algorithm : Harris Corner Detector

1. Computer x and y derivatives I_x and I_y of the input image
2. Computer products of derivatives $I_x I_x$, $I_x I_y$ and $I_y I_y$
3. For each pixel, compute the matrix M in a local neighborhood
4. Compute the corner response R at each pixel
5. Threshold the value of R to select corners
6. Perform non-maximum suppression

Selecting Good Features

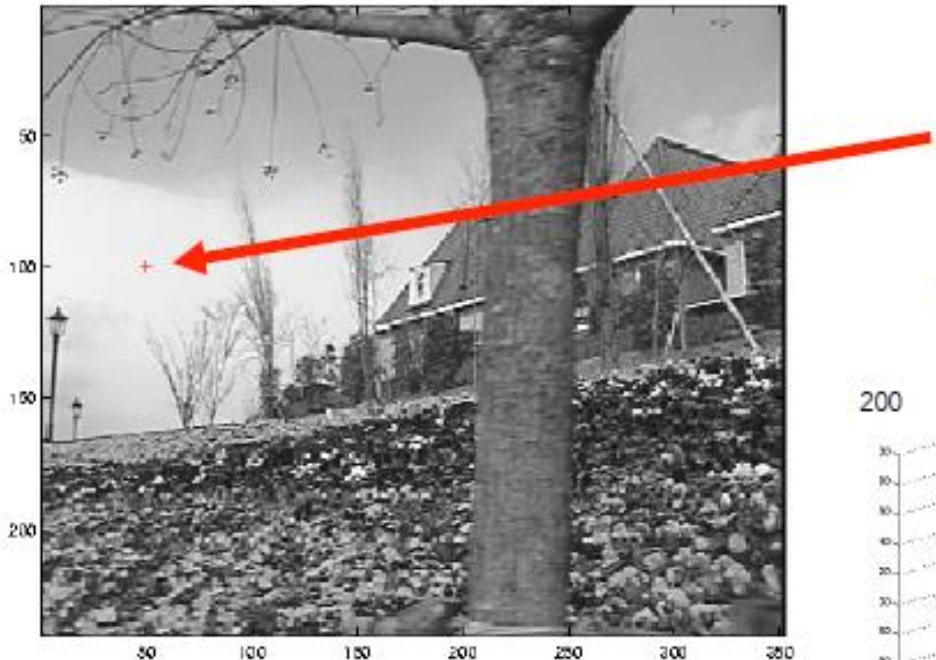
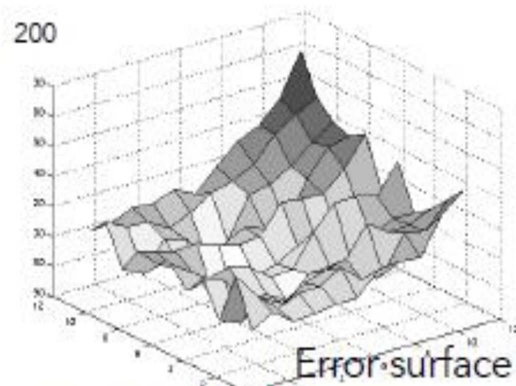


Image patch

(contrast auto-scaled)

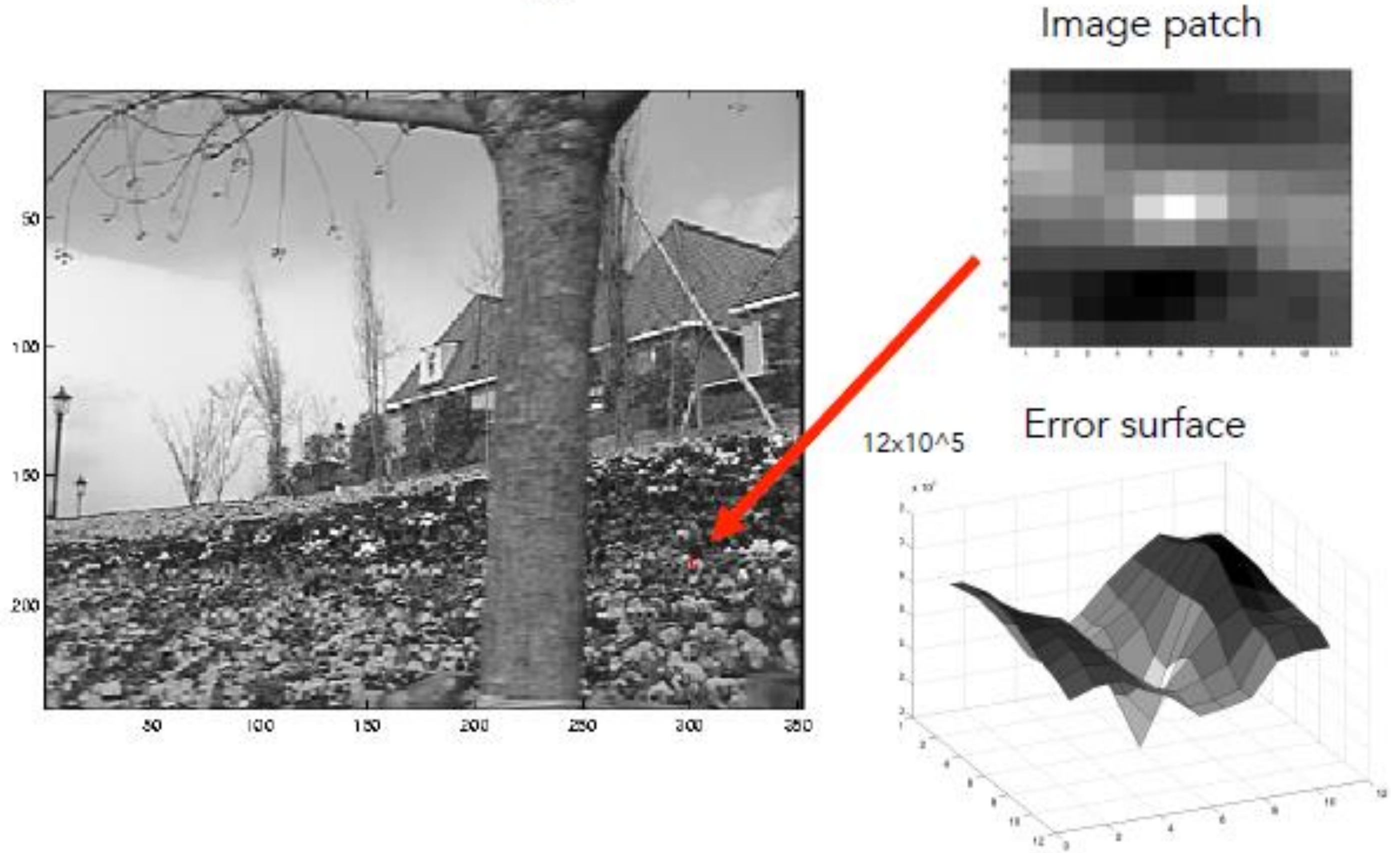


Error surface

(vertical scale exaggerated relative to previous plots)

small λ_1, small λ_2

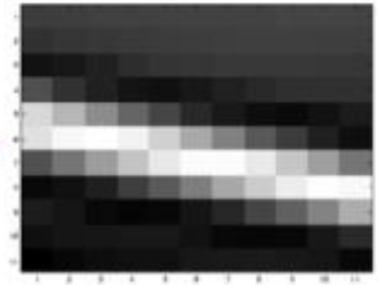
Selecting Good Features



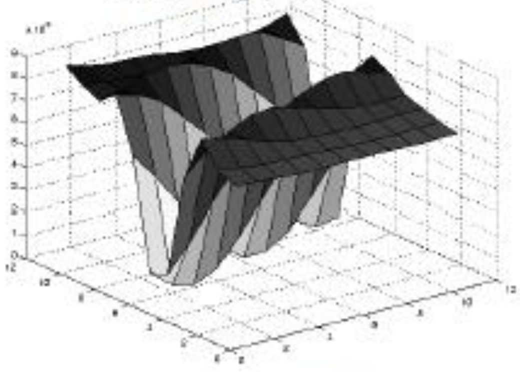
λ_1 and λ_2 are large

Selecting Good Features

Image patch



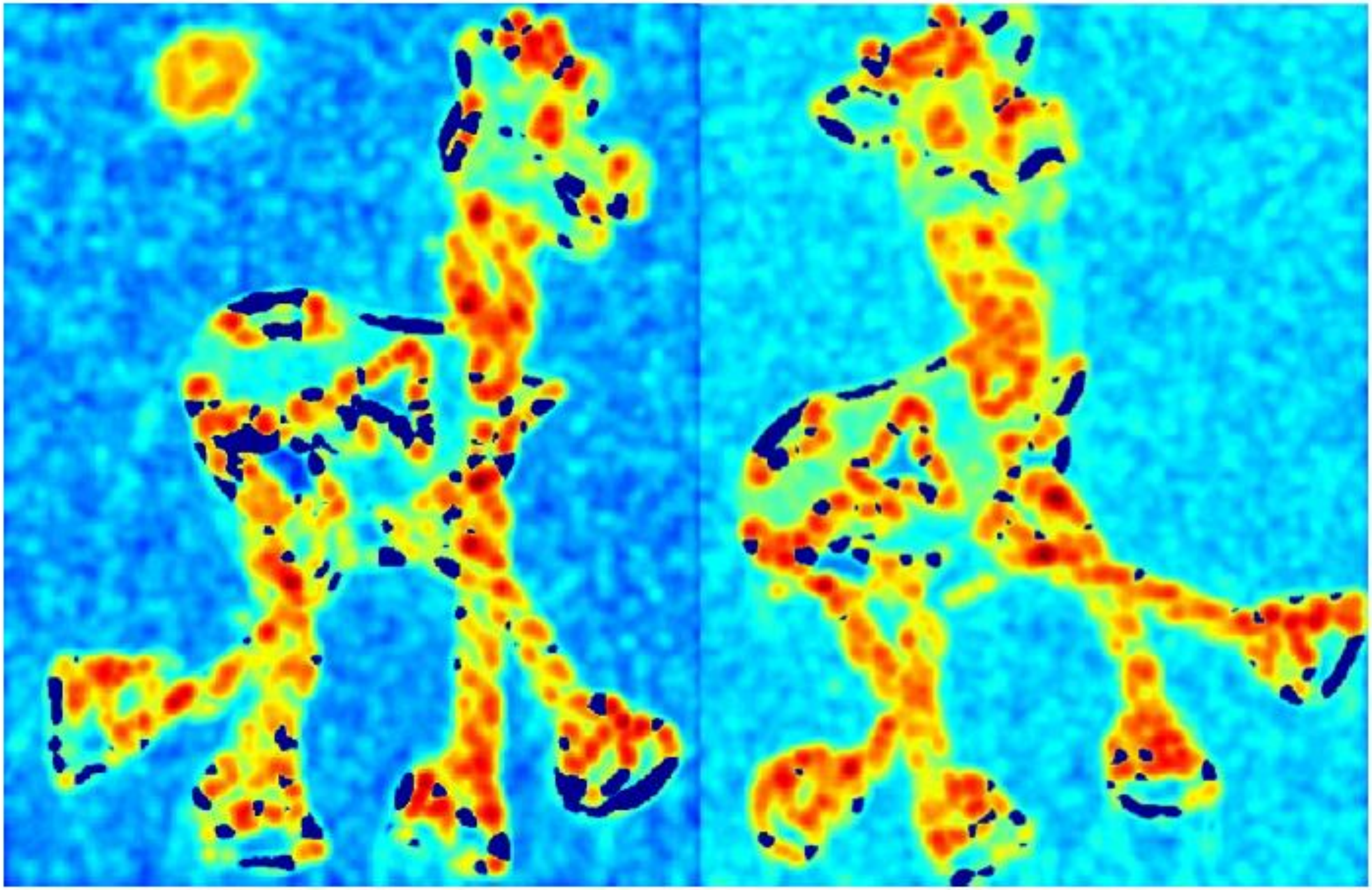
9x10^5 Error surface



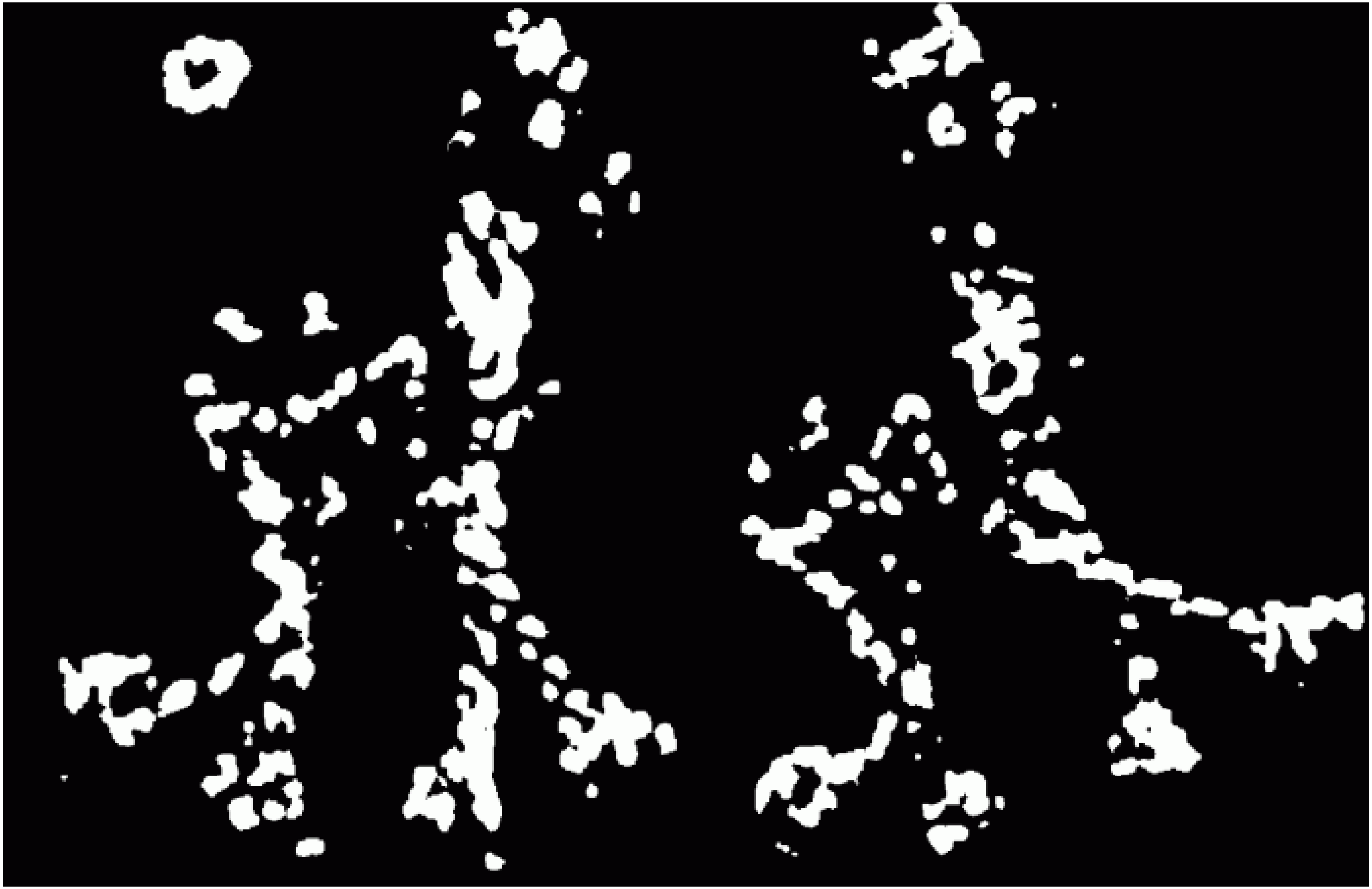
large λ_1, small λ_2

Harris detector example

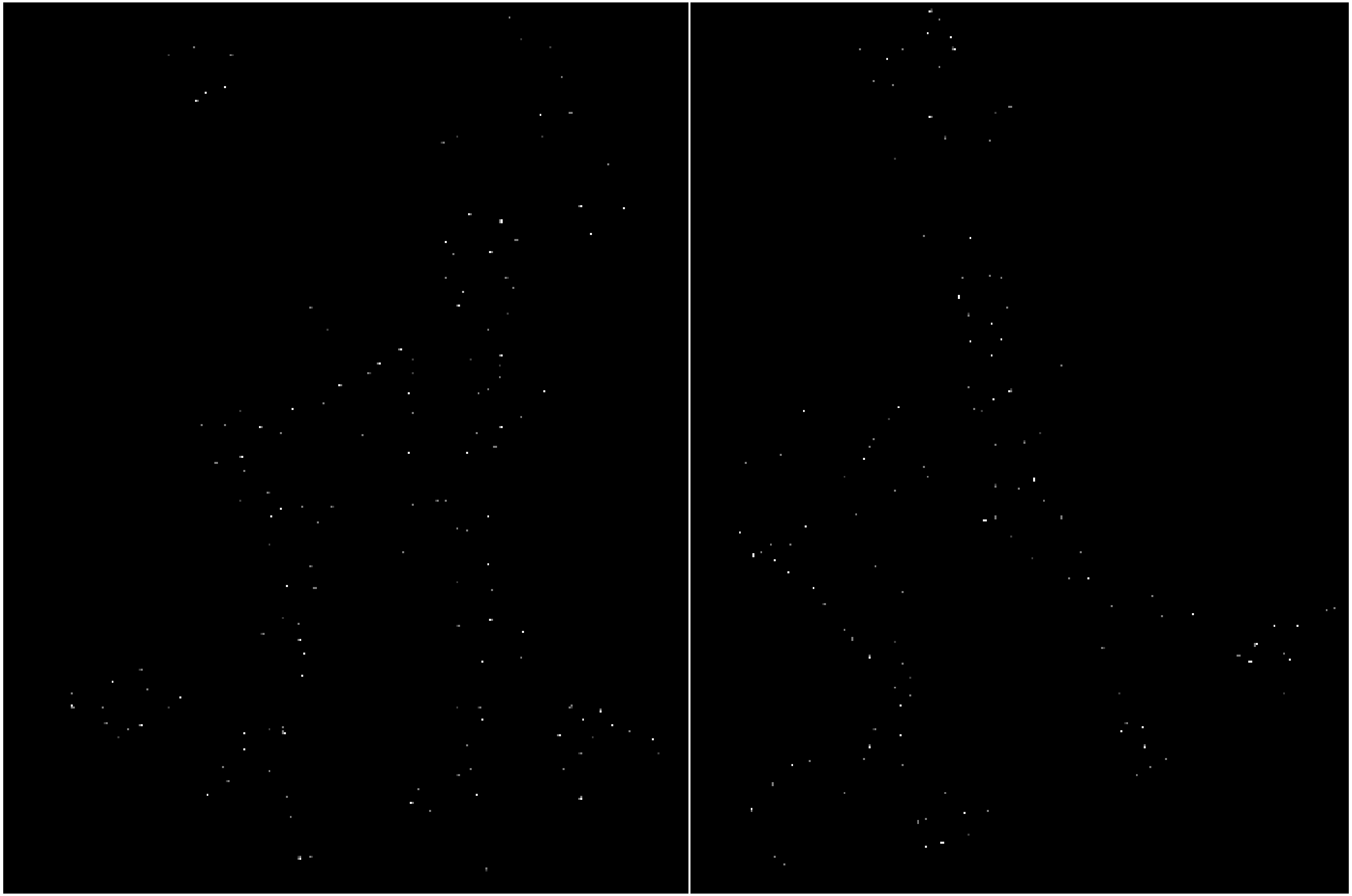
f_{Harris} value (red high, blue low)



Threshold (f_{Harris} > threshold value)



Find local maxima of f_{Harris}



Harris features (in red)

Other Version of Harris Detectors

$$R = \lambda_1 - \alpha\lambda_2$$

Triggs

$$R = \frac{\det(D)}{trace(D)} = \frac{\lambda_1\lambda_2}{\lambda_1 + \lambda_2}$$

Szeliski **(Harmonic mean)**

$$R = \lambda_1$$

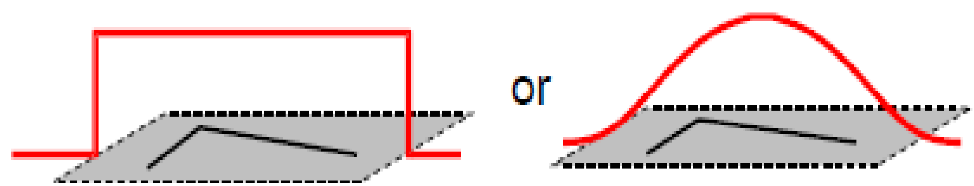
Shi-Tomasi

Change in appearance of window $w(x,y)$ 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 → $w(x,y)$
- Shifted intensity → $I(x+u, y+v)$
- Intensity → $I(x,y)$

Window function $w(x,y)$ =

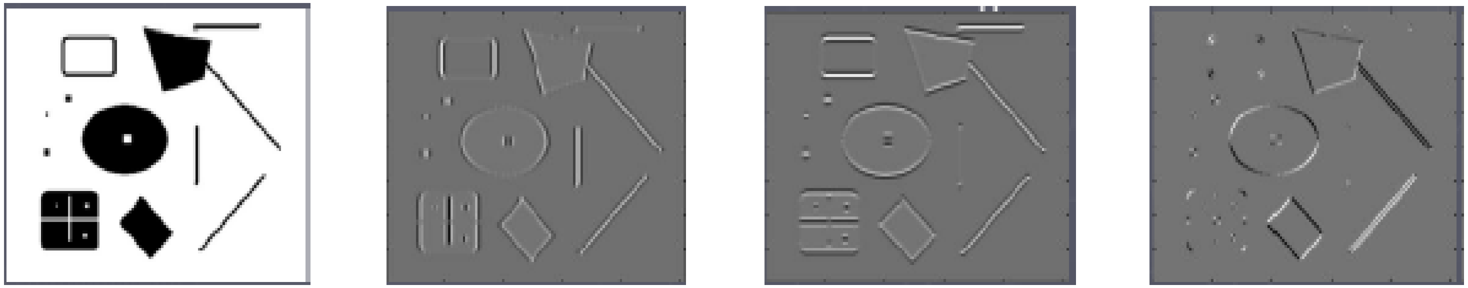


1 in window, 0 outside

Gaussian

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

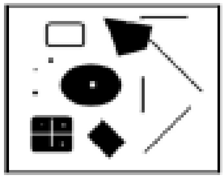
2 x 2 matrix of image derivatives (averaged in neighborhood of a point).



Notation: $I_x \Leftrightarrow \frac{\partial I}{\partial x}$ $I_y \Leftrightarrow \frac{\partial I}{\partial y}$ $I_x I_y \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$

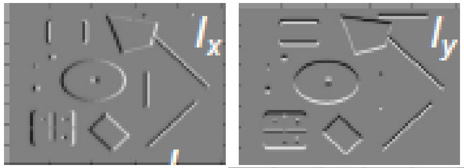
Harris Detector [Harris88]

- Second moment matrix



$$\mu(\sigma_I,\sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_xI_y(\sigma_D) \\ I_xI_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

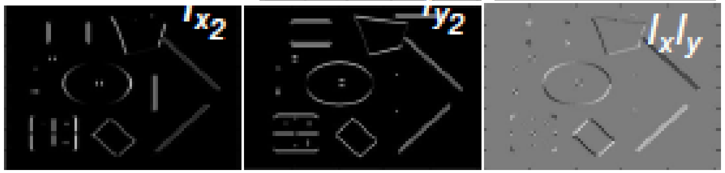
1. Image derivatives (optionally, blur first)



$$\det M = \lambda_1\lambda_2$$

$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

2. Square of derivatives



3. Gaussian filter $g(\sigma_I)$

4. Cornerness function – both eigenvalues are strong

$$har = \det[\mu(\sigma_I,\sigma_D)] - \alpha[\operatorname{trace}(\mu(\sigma_I,\sigma_D))^2] =$$

$$g(I_x^2)g(I_y^2) - [g(I_xI_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$

5. Non-maxima suppression

Slide Credit: James Hays

Invariance and covariance

- We want corner locations to be *invariant* to photometric transformations and *covariant* to geometric transformations
  - **Invariance:** image is transformed and corner locations do not change
  - **Covariance:** if we have two transformed versions of the same image, features should be detected in corresponding locations

Invariant / Covariant

- A function f( ) is **invariant** under some transformation T( ) if its value does change when the transformation is applied to its argument:

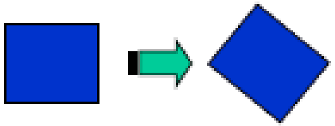
$$\text{if } f(x) = y \text{ then } f(T(x))=y$$

- A function f( ) is **covariant** when it commutes with the transformation T( ):

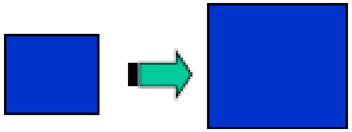
$$\text{if } f(x) = y \text{ then } f(T(x))=T(f(x))=T(y)$$

Invariance to Geometric/Photometric Changes

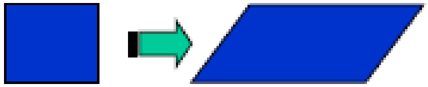
- Is the Harris detector invariant to geometric and photometric changes?
- Geometric
  - Rotation



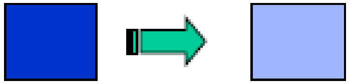
  - Scale



  - Affine



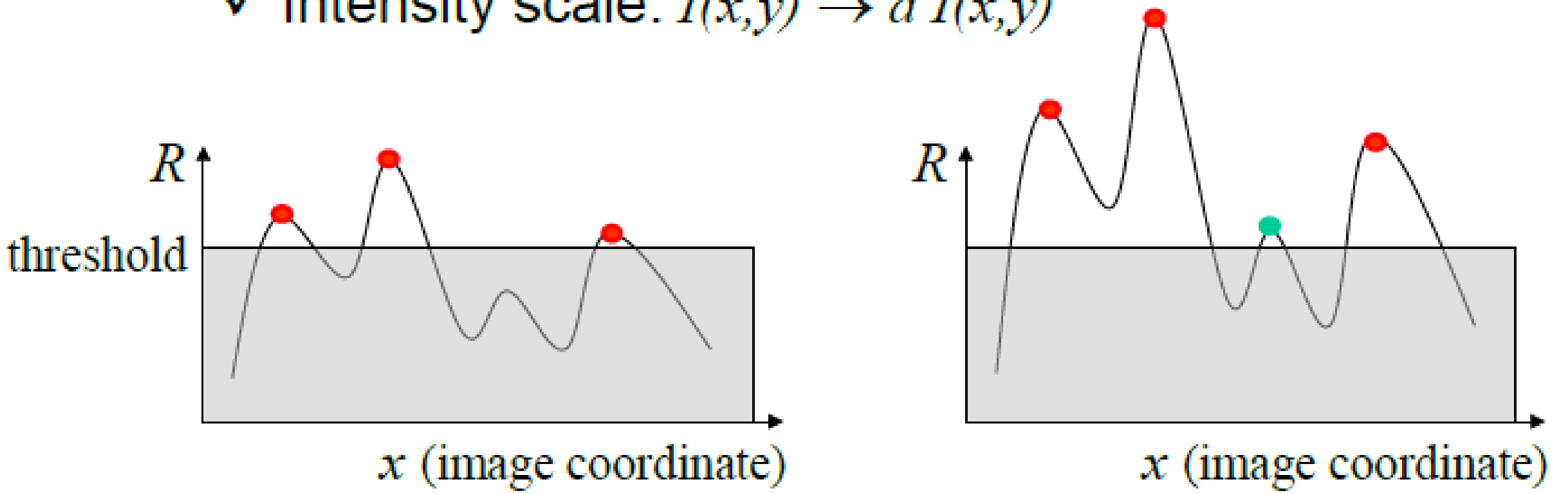
- Photometric



  - Affine intensity change: $I(x,y) \rightarrow a\,I(x,y) + b$

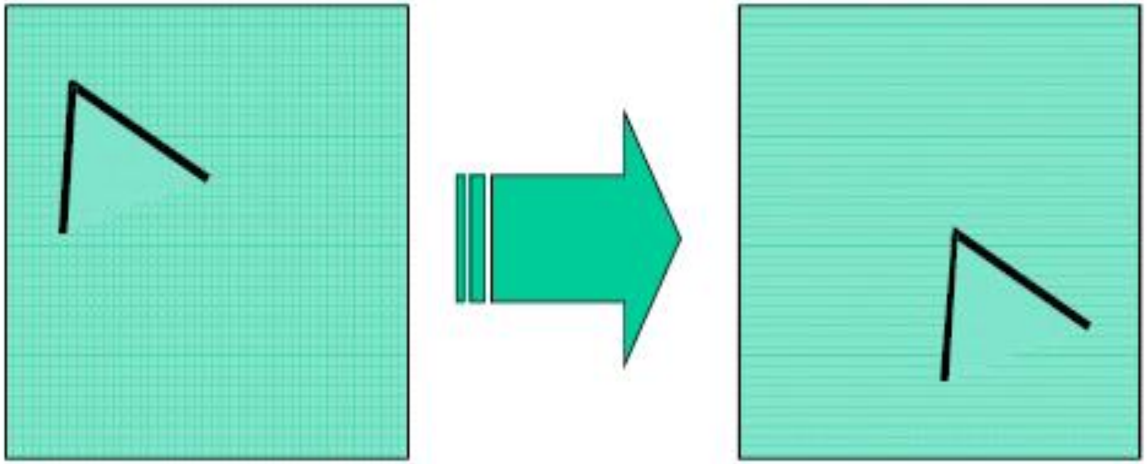
Harris Detector: Photometric Changes

- Affine intensity change
  - ✓ Only derivatives are used => invariance to intensity shift $I(x,y) \rightarrow I(x,y) + b$
  - ✓ Intensity scale: $I(x,y) \rightarrow a\, I(x,y)$



Partially invariant to affine intensity change

Image translation

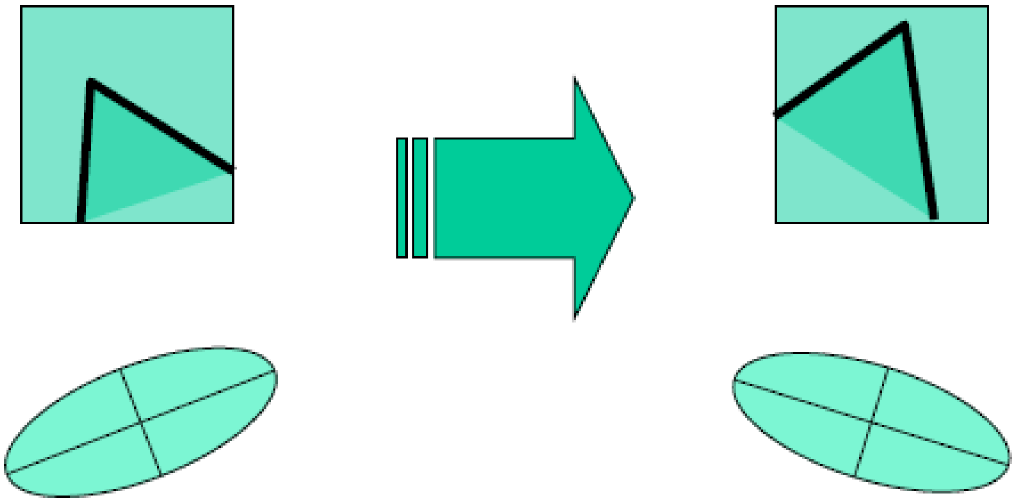


- Derivatives and window function are shift-invariant

Corner location is covariant w.r.t. translation

Harris Detector: Rotation Invariance

- Rotation

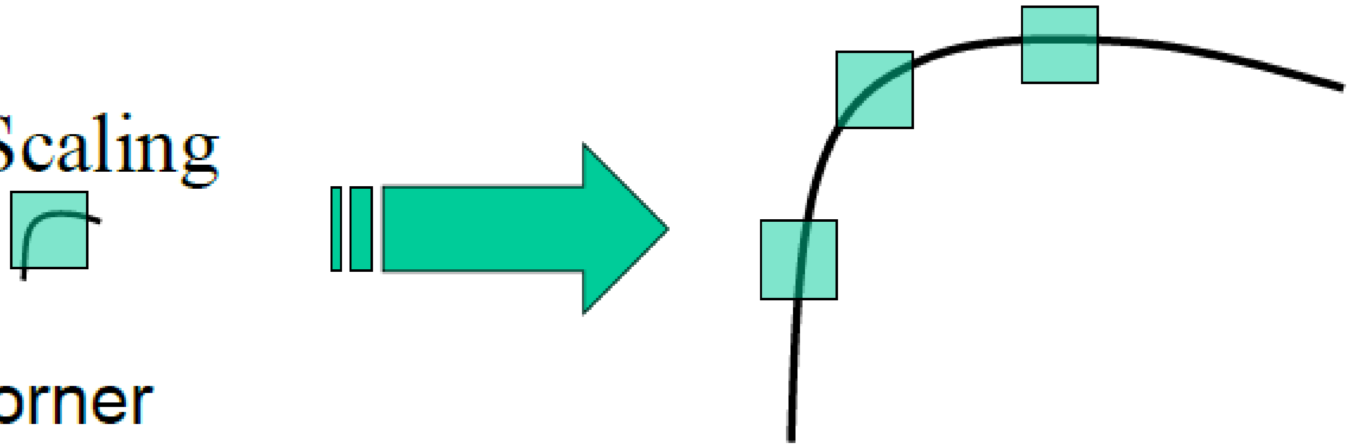


Ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner response R is invariant to image rotation

Harris Detector: Scale Invariance

- Scaling



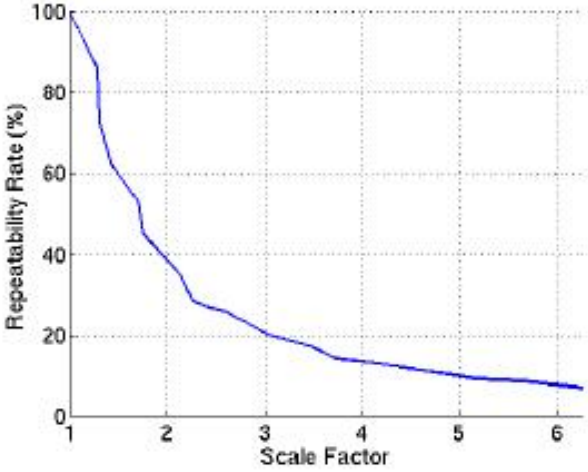
Corner

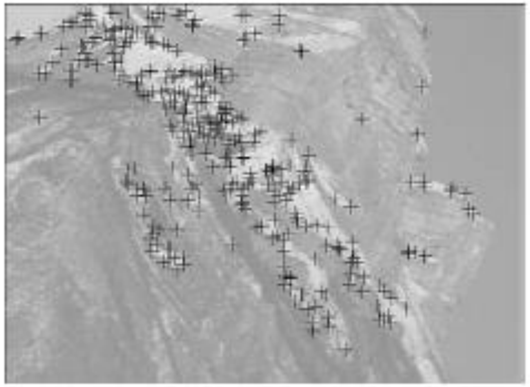
All points will be classified as edges

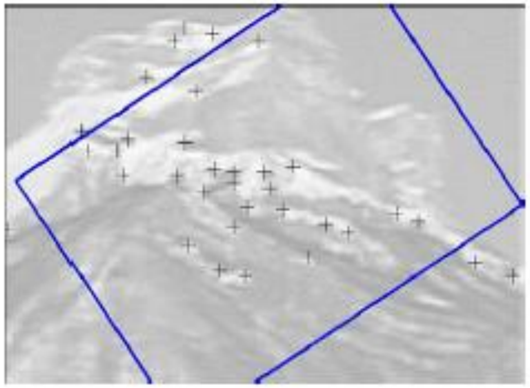
Not invariant to scaling (and affine transforms)

Harris Detector: Disadvantages

- Sensitive to:
  - Scale change
  - Significant viewpoint change
  - Significant contrast change







How to handle scale changes?

- A_W must be adapted to scale changes.

- If the scale change is known, we can adapt the Harris detector to the scale change (i.e., set properly σ_I, σ_D).

- What if the scale change is unknown?

Multi-scale Harris Detector

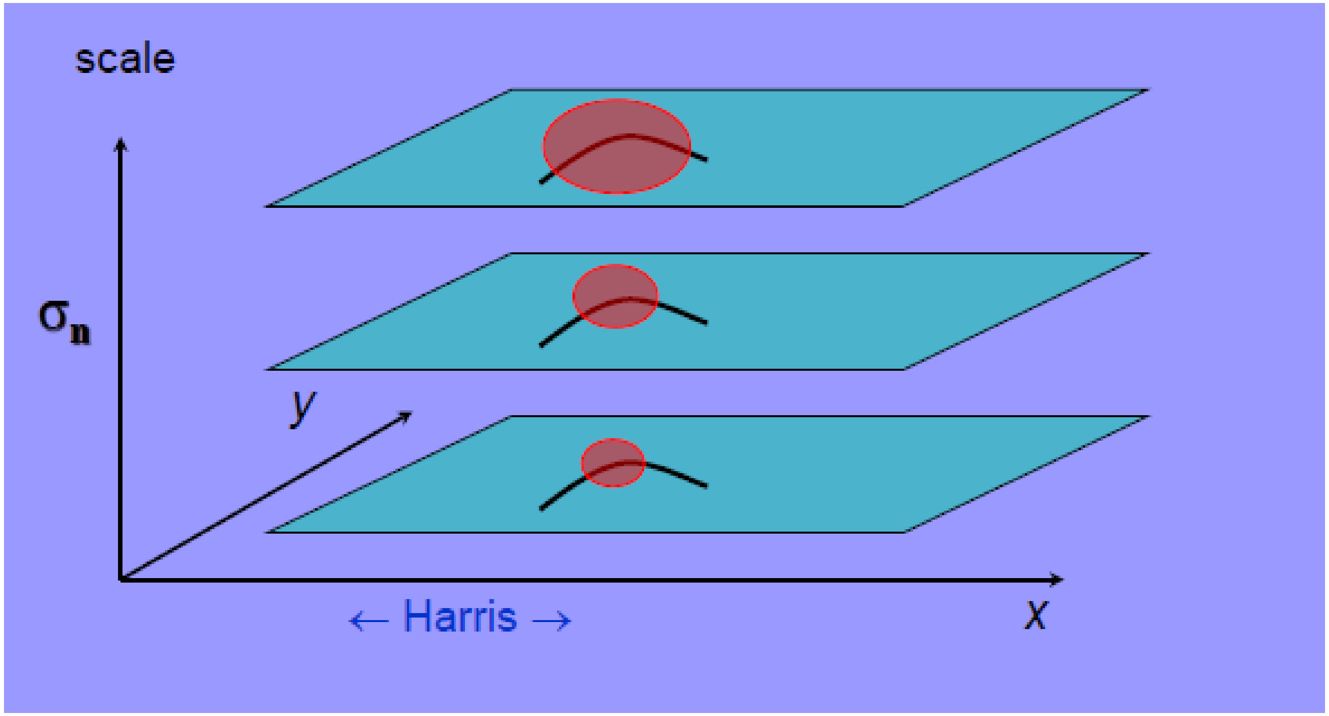
- Detects interest points at varying scales.

$$R(A_W) = \det(A_W(x,y,\sigma_I,\sigma_D)) - \alpha\, \mathrm{trace}^2(A_W(x,y,\sigma_I,\sigma_D))$$

$\sigma_n = k^n \sigma$

$\sigma_D = \sigma_n$

$\sigma_I = \gamma \sigma_D$



Invariant Local Image Features

Properties of good features

- **Local:** features are local, robust to occlusion and clutter (no prior segmentation!).
- **Accurate:** precise localization.

- **Invariant** (or **covariant**)
- **Robust:** noise, blur, compression, etc. do not have a big impact on the feature.

} **Repeatable**

- **Distinctive:** individual features can be matched to a large database of objects.
- **Efficient:** close to real-time performance.

Invariant Local Image Features

Advantages of invariant local features

- **Locality:** features are local, so robust to occlusion and clutter (no prior segmentation)
- **Distinctiveness:** individual features can be matched to a large database of objects
- **Quantity:** many features can be generated for even small objects
- **Efficiency:** close to real-time performance
- **Extensibility:** can easily be extended to wide range of differing feature types, with each adding robustness

Scale Invariance

In many applications, the scale of the object of interest may vary in different images.

Simple but inefficient solution:

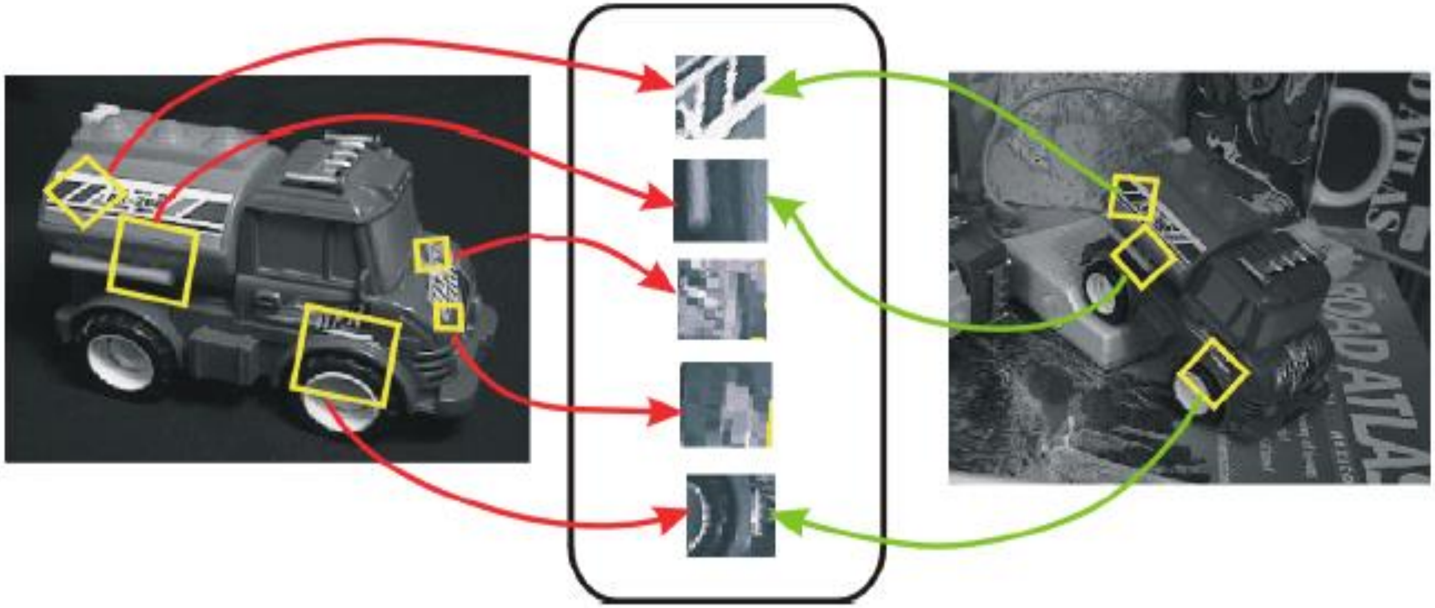
- Extract features at many different scales.
- Match them to the object's known features at a particular scale.

More efficient solution:

- Extract features that are invariant to scale.

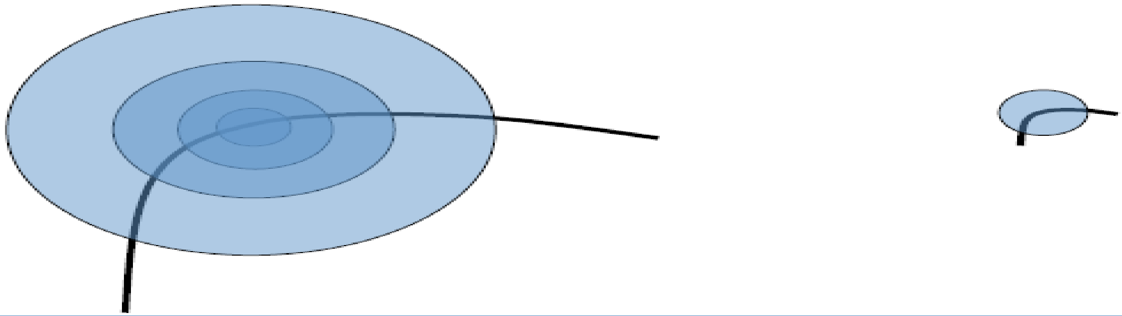
Invariant Local Features

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



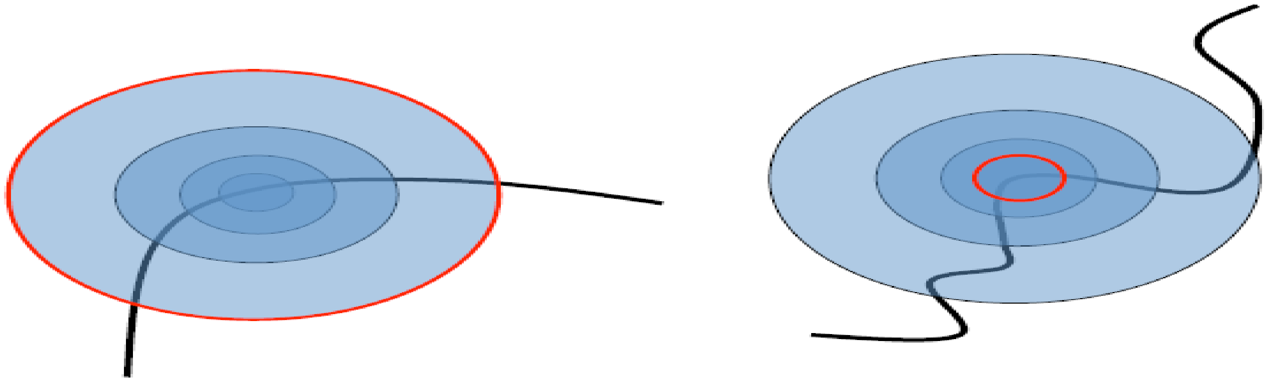
Scale Invariant Detection

- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images



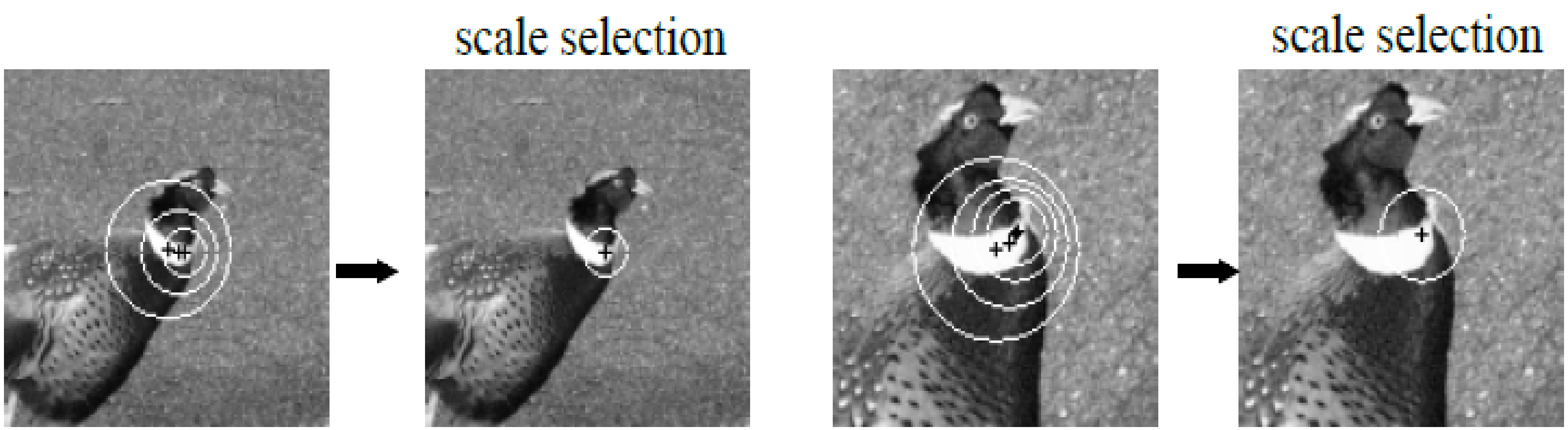
Scale Invariant Detection

- The problem: how do we choose corresponding circles ***independently*** in each image?



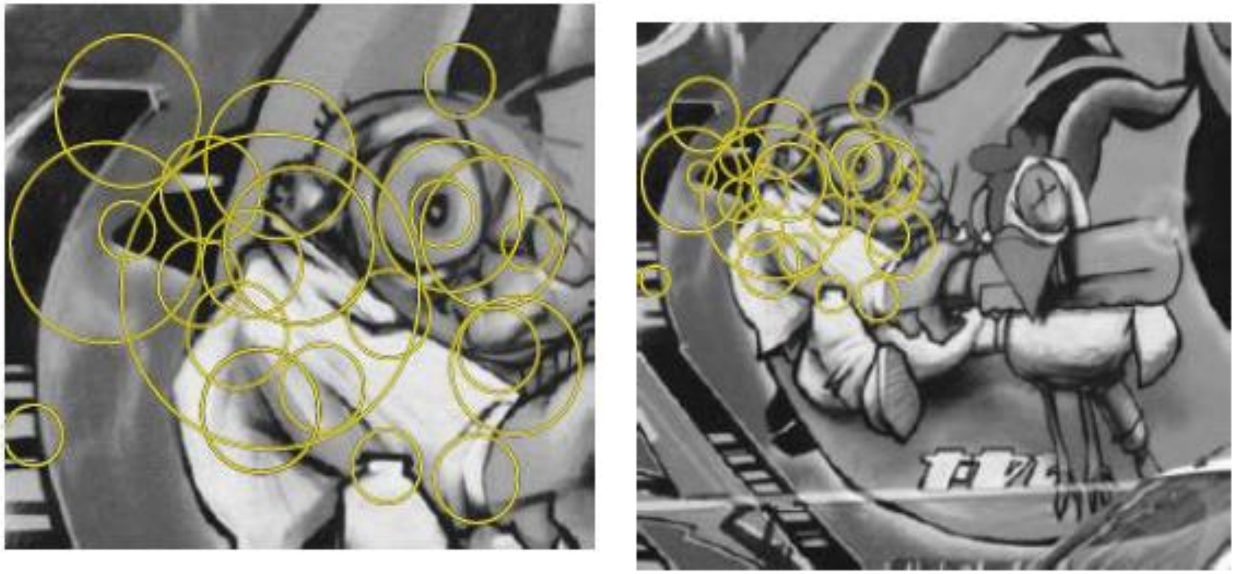
How to handle scale changes? (cont'd)

- Alternatively, use **scale selection** to find the characteristic scale of each feature.
- Characteristic scale depends on the feature's spatial extent (i.e., local neighborhood of pixels).



How to handle scale changes?

- Only a subset of the points computed in scale space are selected!

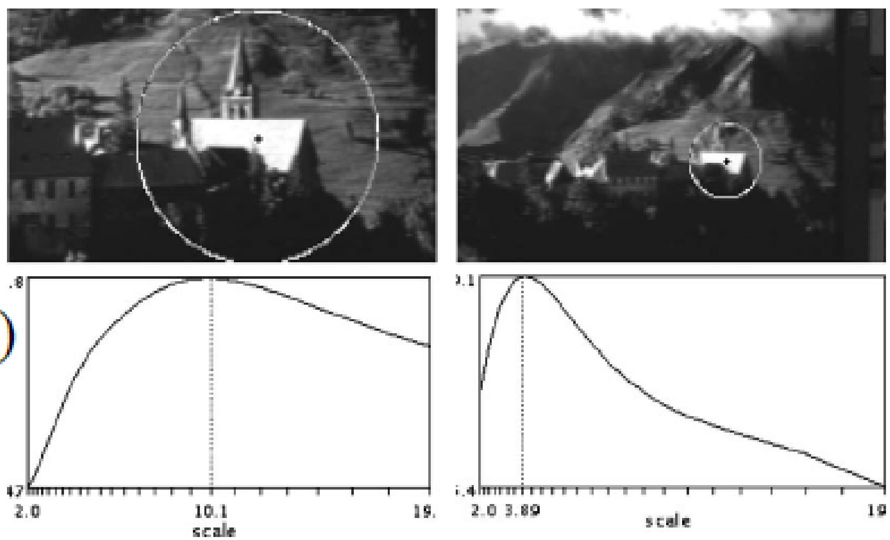


The size of the circles corresponds to the scale
at which the point was selected.

Automatic Scale Selection

- Design a function $F(x,\sigma_n)$ which provides some local measure.
- Select points at which $F(x,\sigma_n)$ is maximal over σ_n.

$F(x,\sigma_n)$

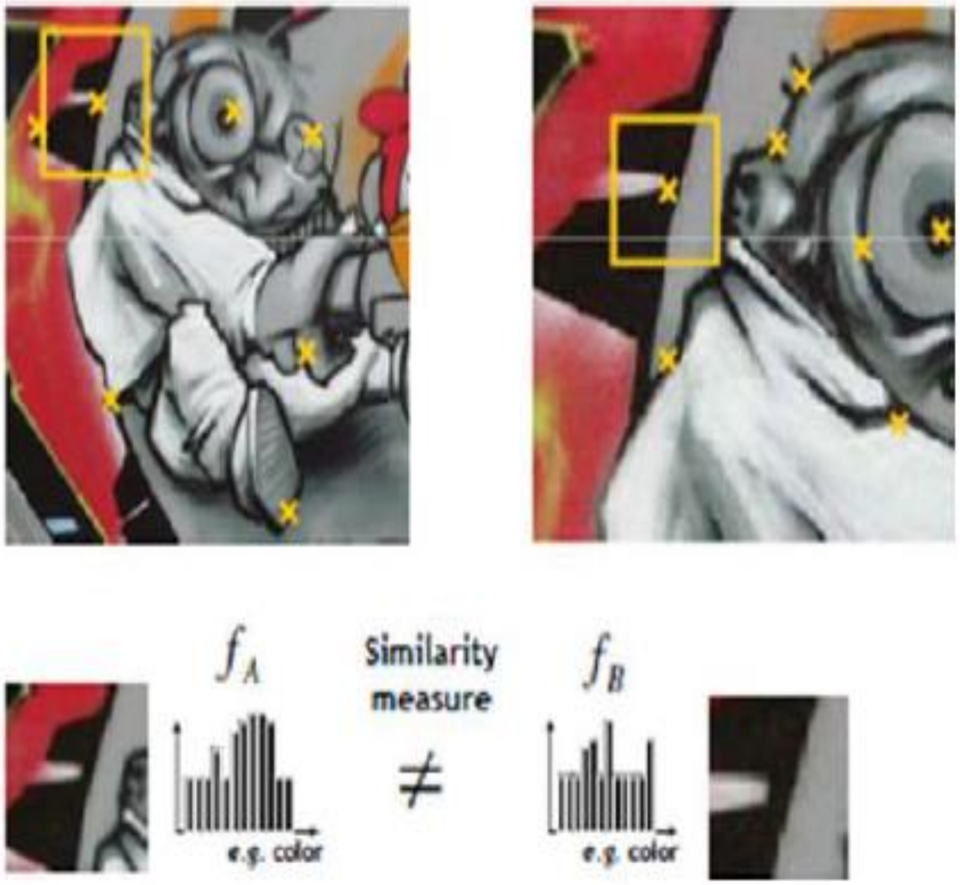


σ_n

max of $F(x,\sigma_n)$ corresponds to characteristic scale!

T. Lindeberg, "Feature detection with automatic scale selection" *International Journal of Computer Vision,* vol. 30, no. 2, pp 77-116, 1998.

Scale Different

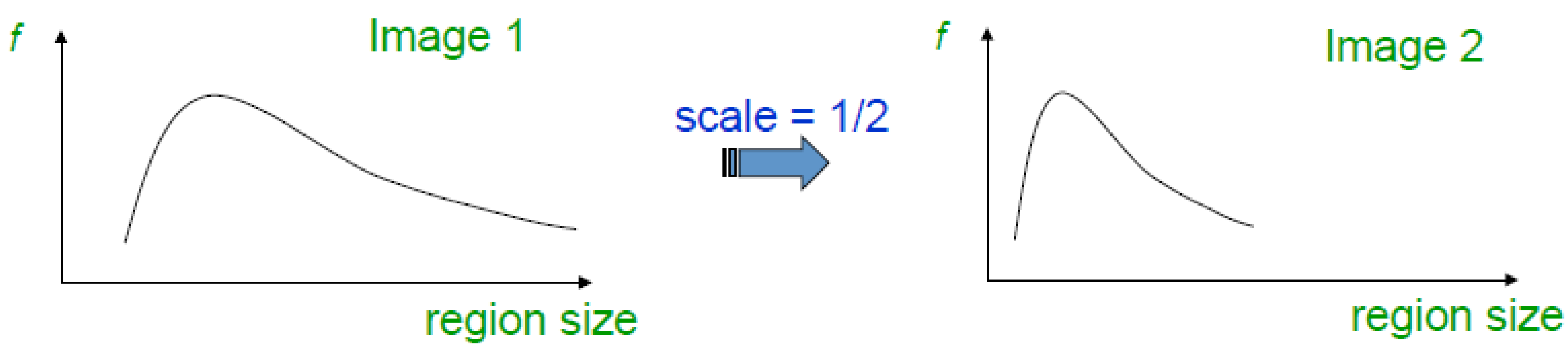


Scale Invariant Detection

- Solution:
  - Design a function on the region (circle), which is "scale invariant" (the same for corresponding regions, even if they are at different scales)

    Example: average intensity. For corresponding regions (even of different sizes) it will be the same.

  - For a point in one image, we can consider it as a function of region size (circle radius)

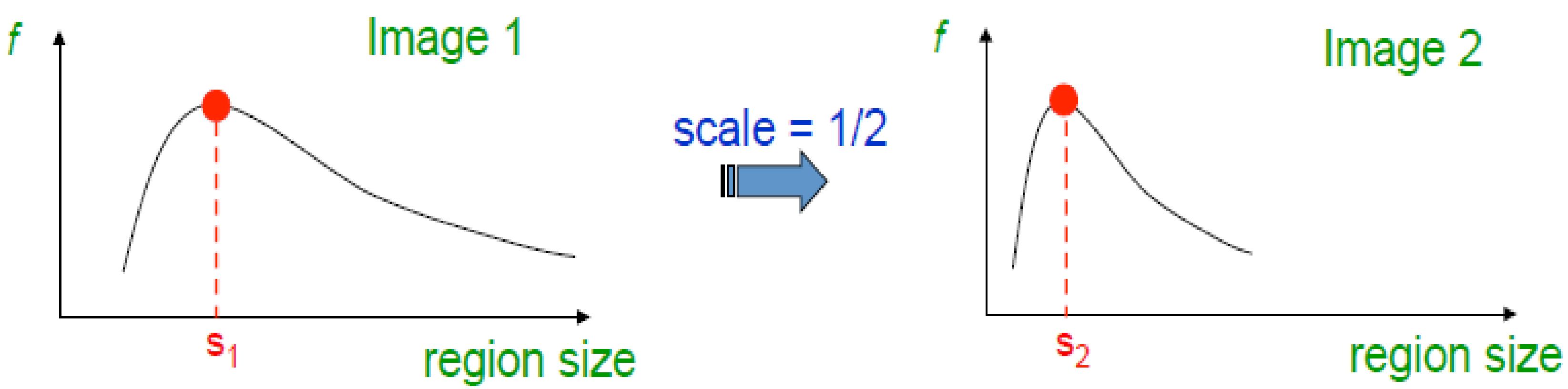


Scale Invariant Detection

- Common approach:
  Take a local maximum of this function

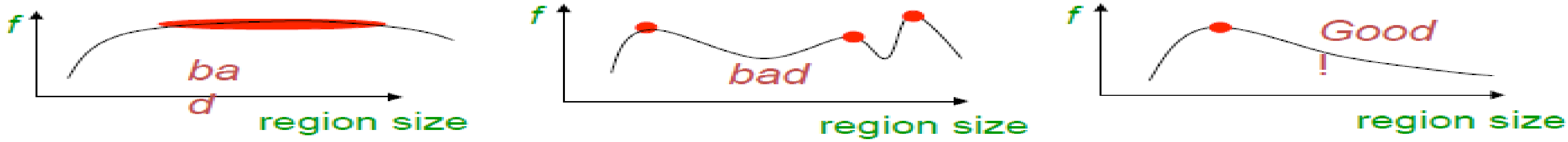
- Observation: region size, for which the maximum is achieved, should be *invariant* to image scale.

Important: this scale invariant region size is found in each image independently!



Scale Invariant Detection

- A "good" function for scale detection: has one stable sharp peak



- For usual images: a good function would be a one which responds to contrast (sharp local intensity change)

- Functions for determining scale $f = \text{Kernel} * \text{Image}$

Kernels:

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

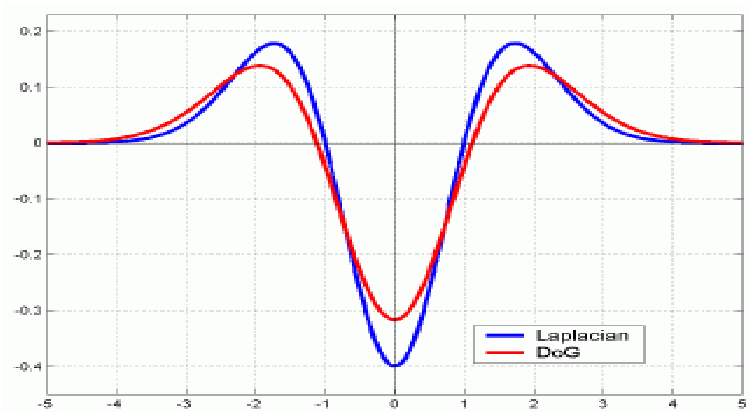
(Laplacian)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)

where Gaussian

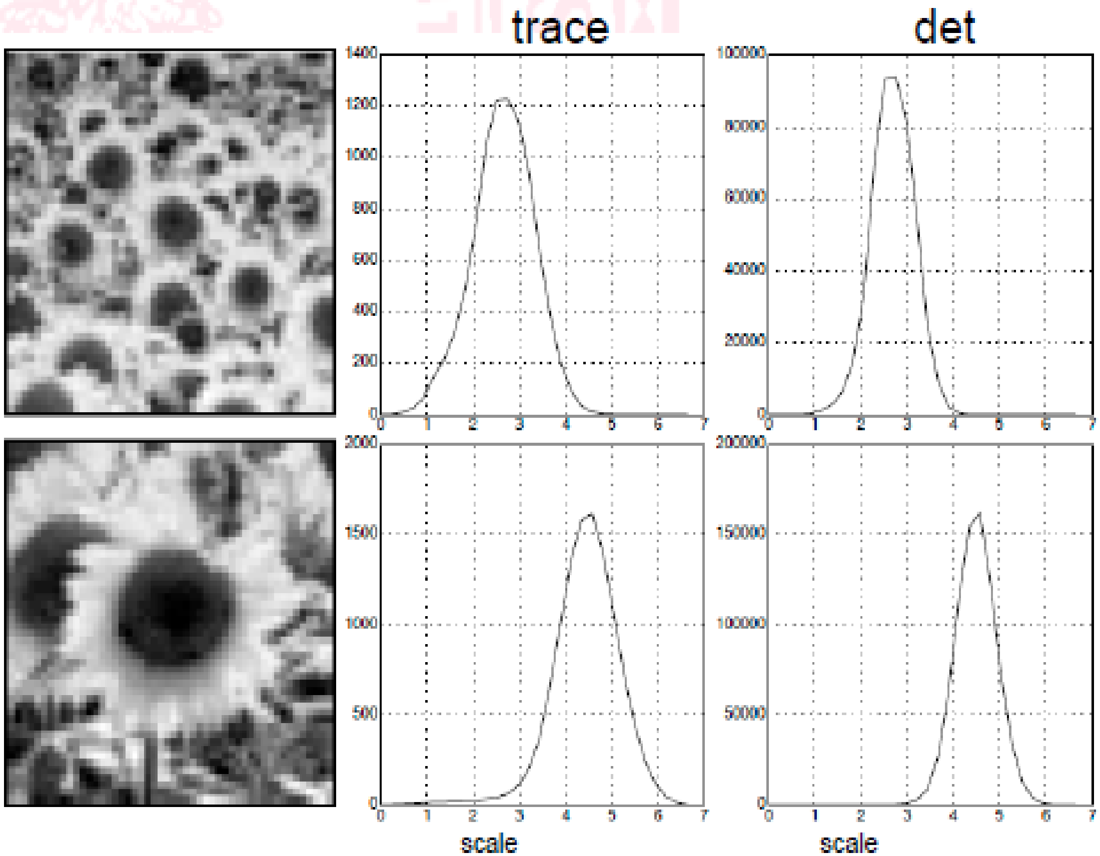
$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



Note: both kernels are invariant to *scale* and *rotation*

$$\det M = \lambda_1 \lambda_2$$

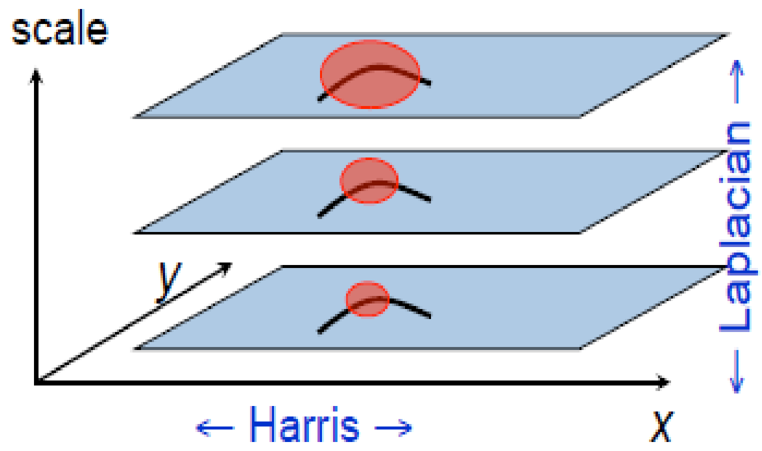
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$



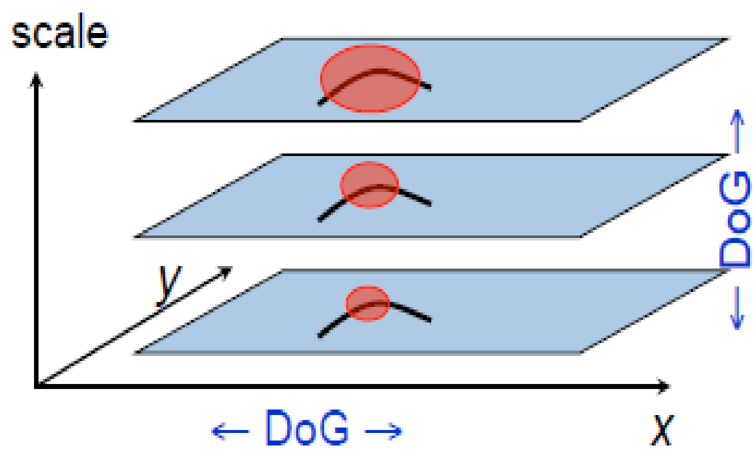
From Lindeberg 1998

Scale Invariant Detectors

- Harris-Laplacian[1]
  Find local maximum of:
  - Harris corner detector in space (image coordinates)
  - Laplacian in scale



- SIFT (Lowe)[2]
  Find local maximum of:
  - Difference of Gaussians in space and scale



Scale Invariant Detection: Summary

- Given: two images of the same scene with a large *scale difference* between them
- Goal: find *the same* interest points *independently* in each image
- Solution: search for *maxima* of suitable functions in *scale* and in *space* (over the image)

Methods:

1. Harris-Laplacian [Mikolajczyk, Schmid]: maximize Laplacian over scale, Harris' measure of corner response over the image
2. SIFT [Lowe]: maximize Difference of Gaussians over scale and space

How to achieve invariance in image matching

Two steps:

1. Make sure your feature *detector* is invariant
   - Harris is invariant to translation and rotation
   - Scale is trickier
     - common approach is to detect features at many scales using a Gaussian pyramid (e.g., MOPS)
     - More sophisticated methods find “the best scale” to represent each feature (e.g., SIFT)
2. Design an invariant feature *descriptor*
   - A descriptor captures the intensity information in a region around the detected feature point
   - The simplest descriptor: a square window of pixels
     - What’s this invariant to?
   - Let’s look at some better approaches…