Image Features and Matching

CSE473/573 - Computer Vision and Image Processing

Junsong Yuan

jsyuan@buffalo.edu

2018 Fall

Image Features

Image Features - Motivation

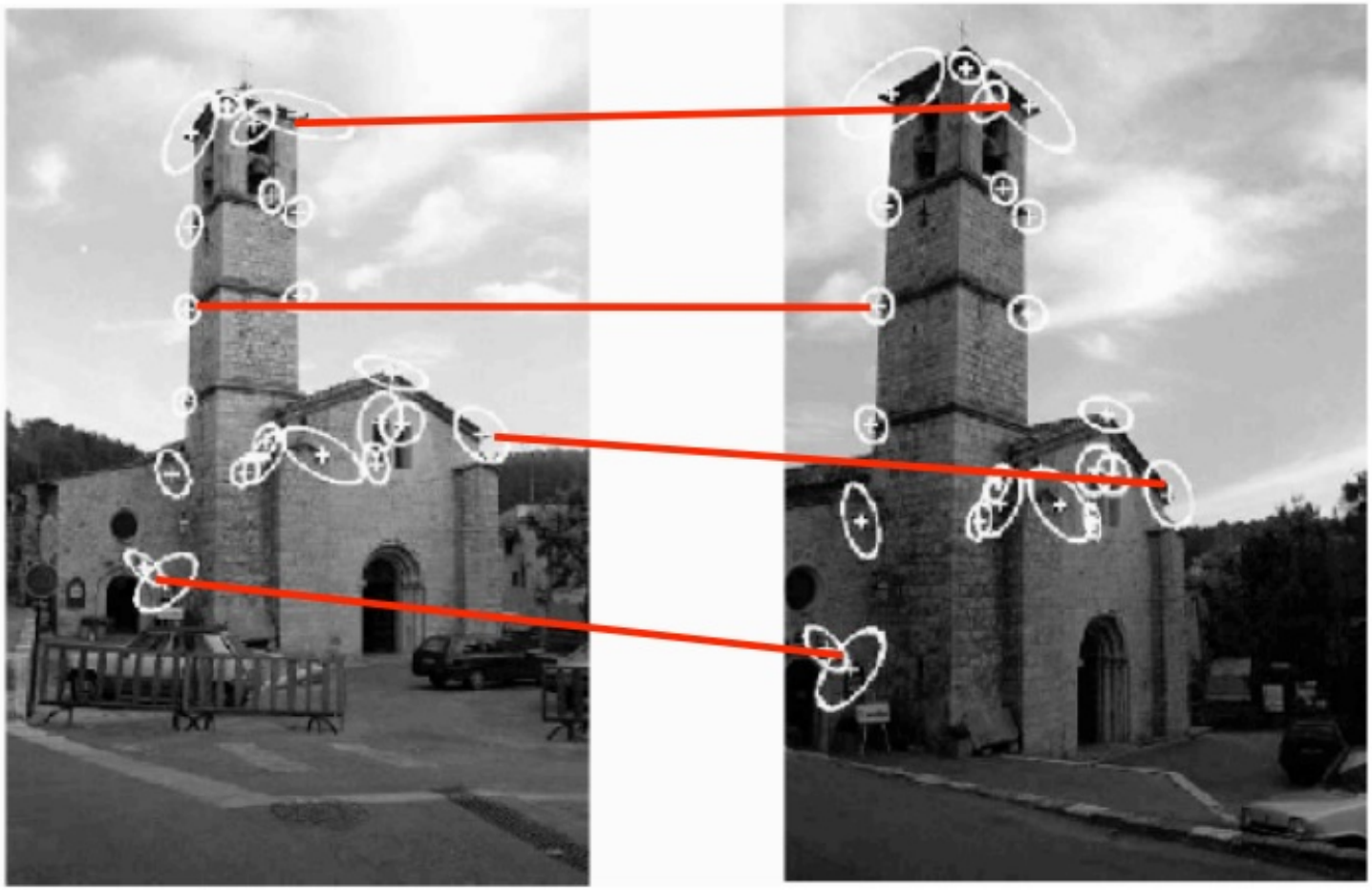
Find this book in the image below

Transformations:

- Orientation
- Scale
- Occlusion

Image Features - Motivation

Vision tasks such as stereo and motion estimation require finding corresponding features across two or more views.



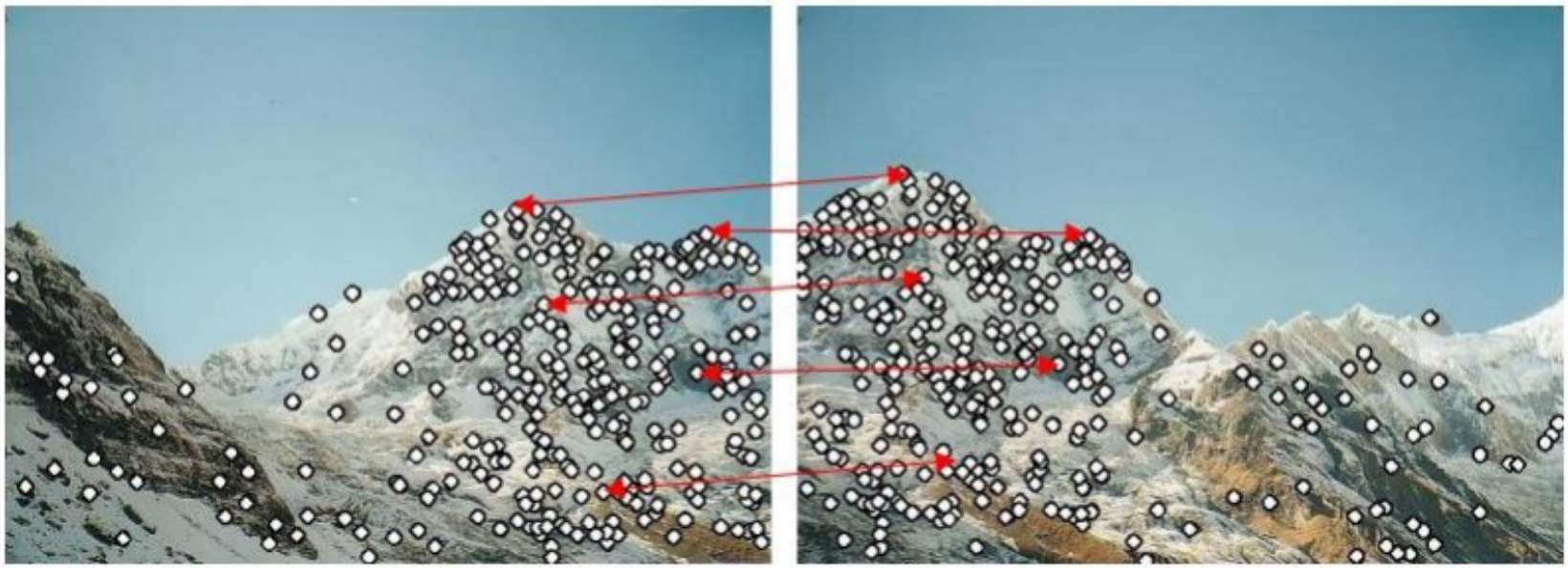
Panorama Creation

First step is to match and align the images

Find features to do it.

Panorama Creation

Model fitting and RANSAC (Random Sample Consensus)



Panorama Creation

Using geometric transformation

Advantages of Local Features

Locality

- features are local, so robust to occlusion and clutter

Distinctiveness:

- can differentiate a large database of objects

Quantity

- hundreds or thousands in a single image

Efficiency

- real-time performance achievable

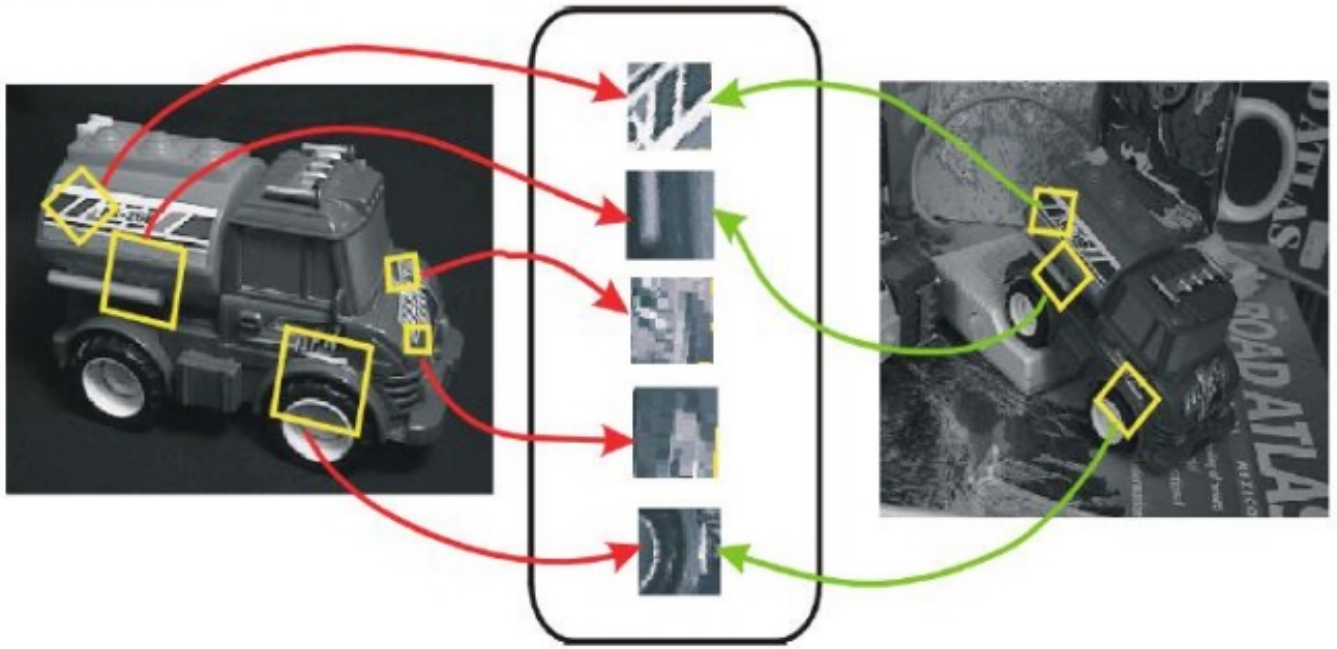
Generality

- exploit different types of features in different situations

Source: S. Seitz slides.

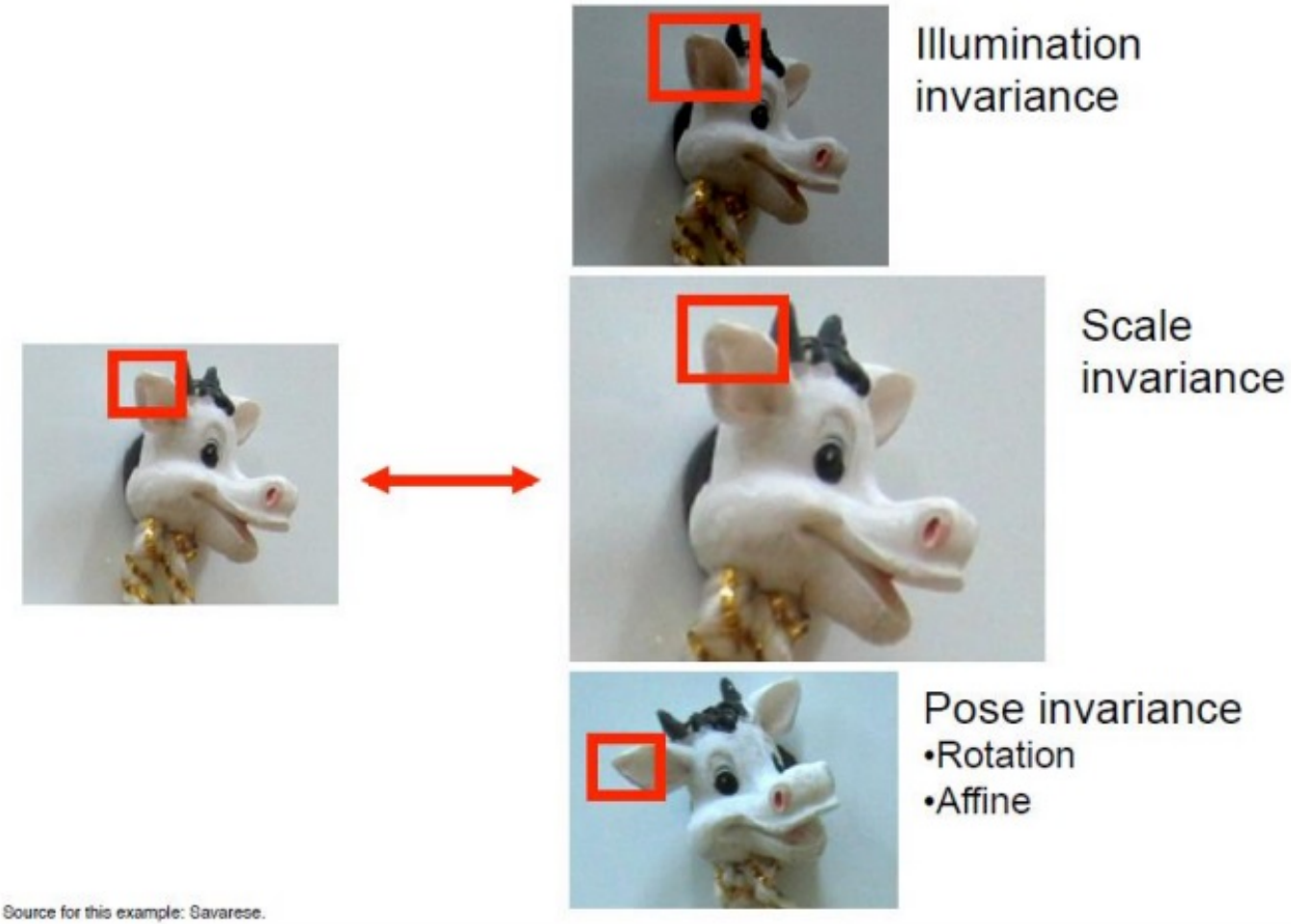
Challenges

- Repeatability
- Uniqueness
- Invariance

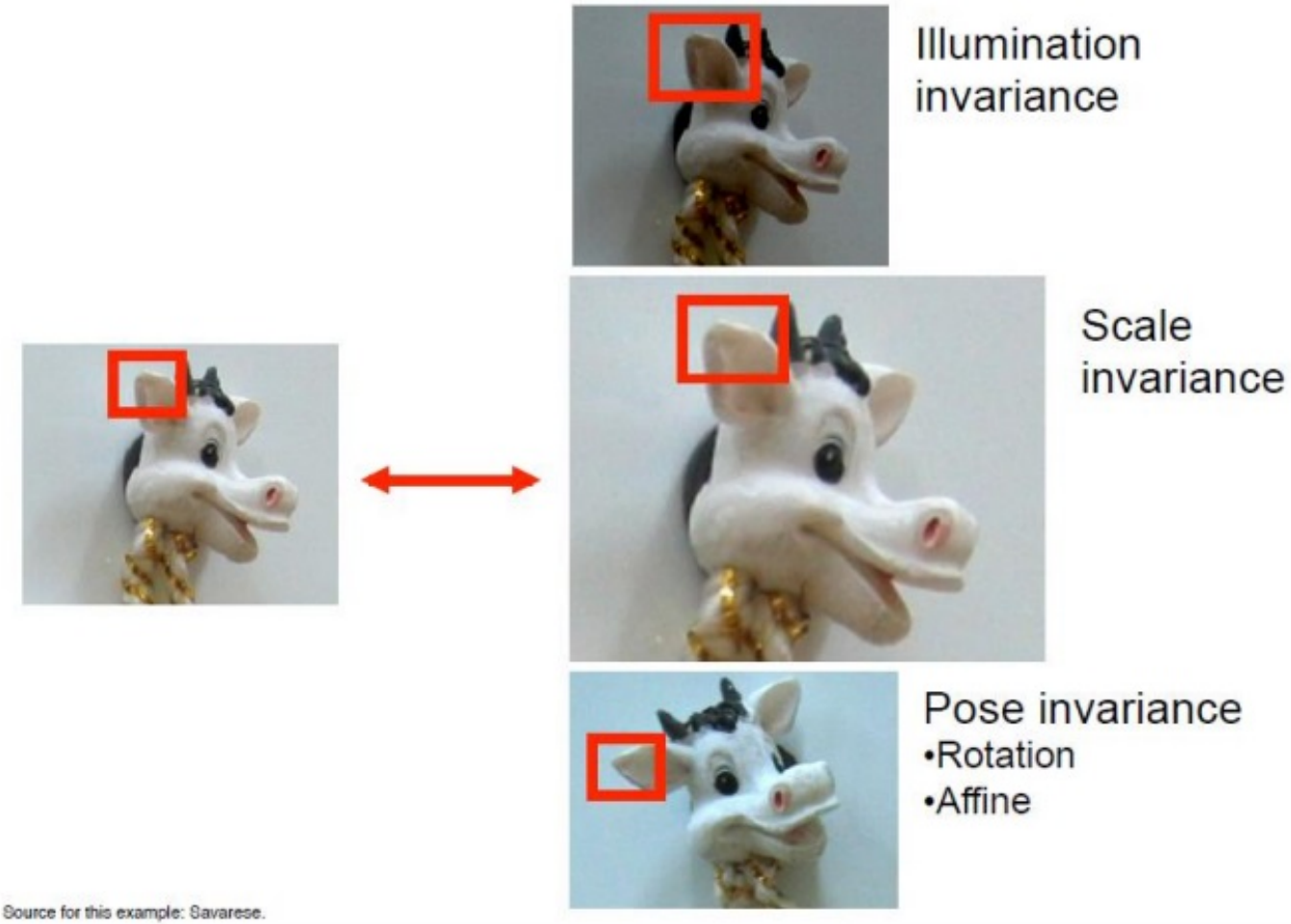


Sources for this example: I. Kokkinos, D. Frolova, D. Simakov, S. Seitz.

Challenges

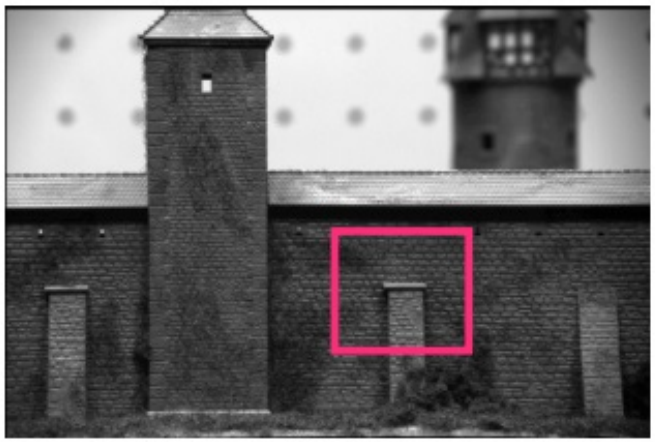

Illumination invariance
Scale invariance
Pose invariance
•Rotation
•Affine
Source for this example: Savarese.

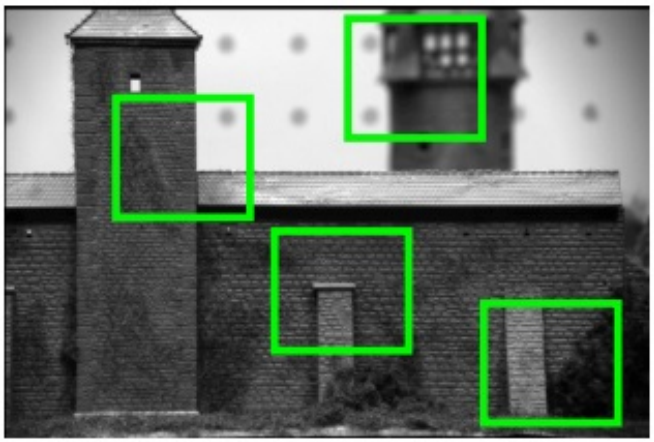
Challenges


Illumination invariance
Scale invariance
Pose invariance
•Rotation
•Affine
Source for this example: Savarese.

What features would you pick?

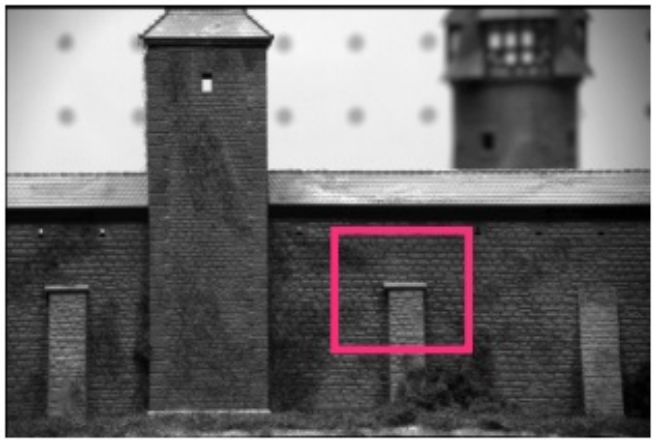
Elements to be matched are image patches of fixed size

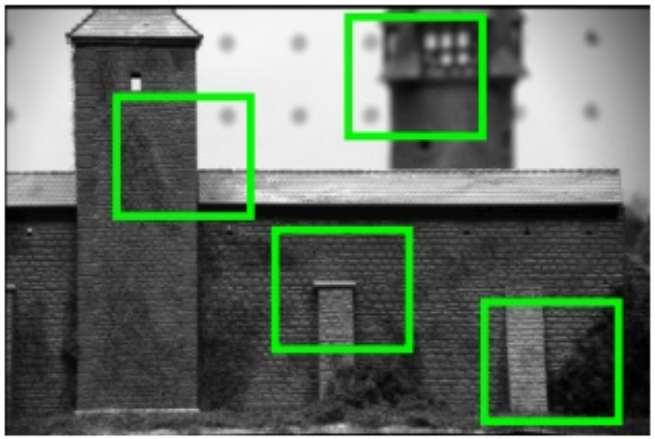




Task: find the best (most similar) patch in a second image

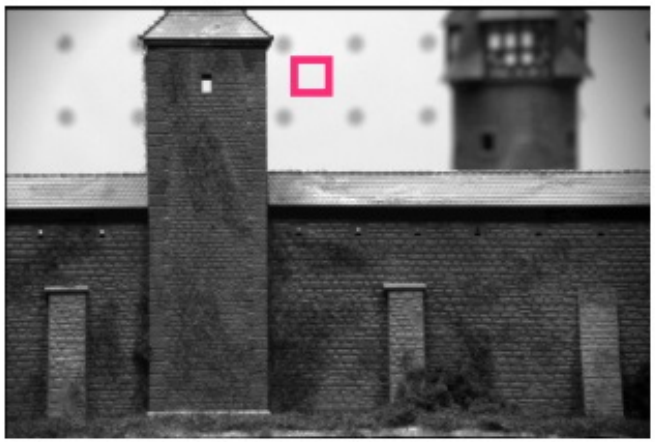
What features would you pick?





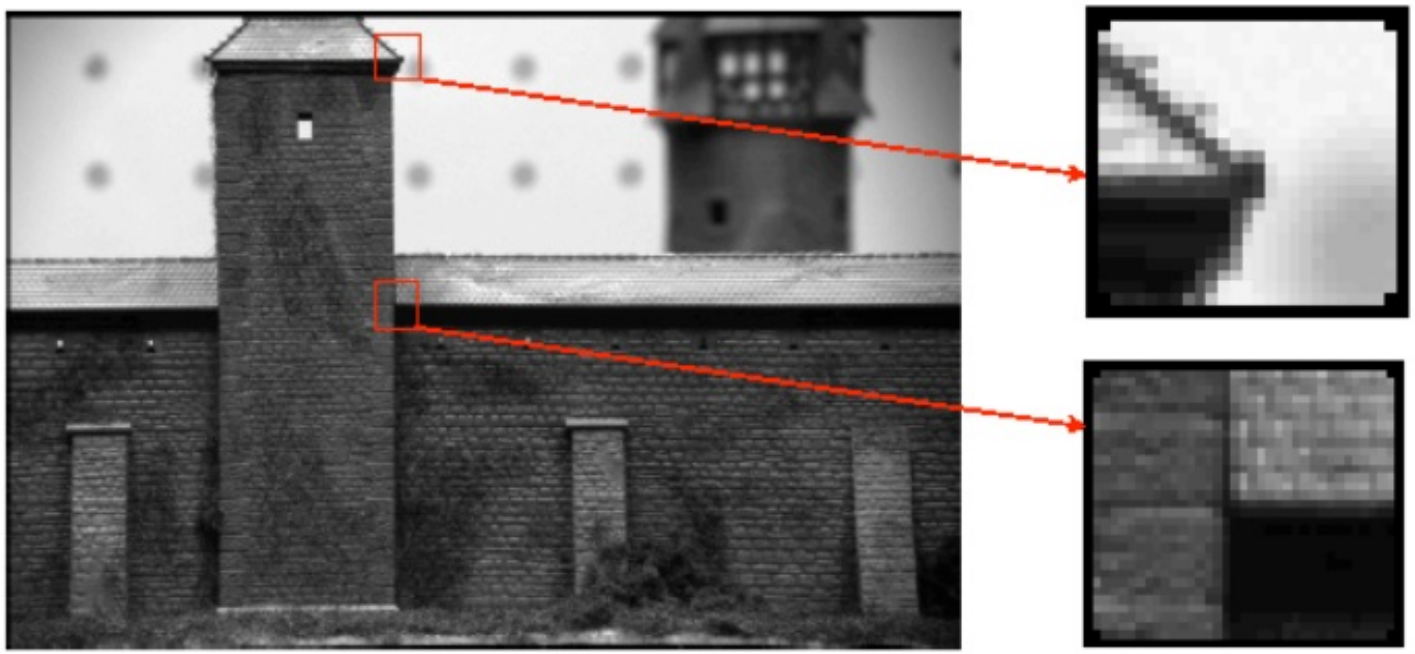
Inituition: this would be a good patch for matching, since it is very distinctive (there is only one patch in the second frame that looks similar).

What features would you pick?



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

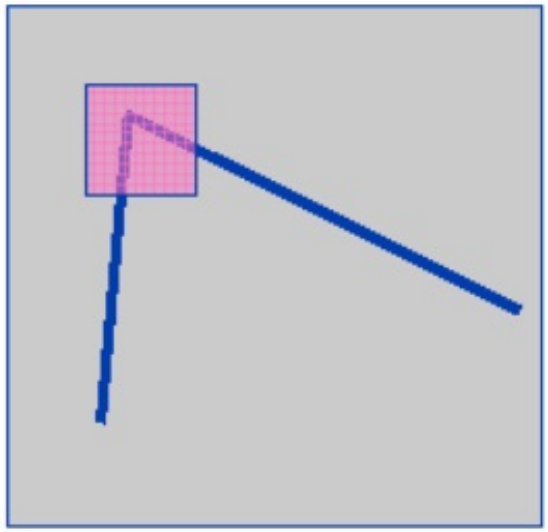
Corners as Features



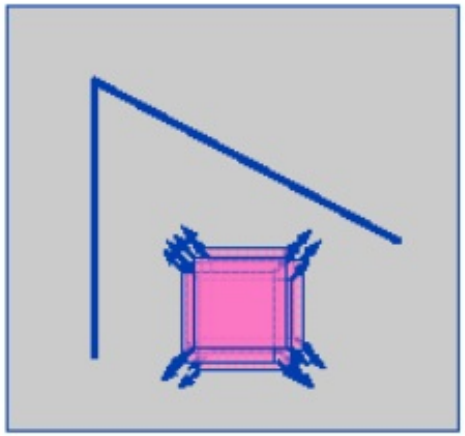
- Intuitively, junctions of contours.
- Generally more stable features over changes of viewpoint
- Intuitively, large variations in the neighborhood of the point in all directions
- They are good features to match!

Corners as Features

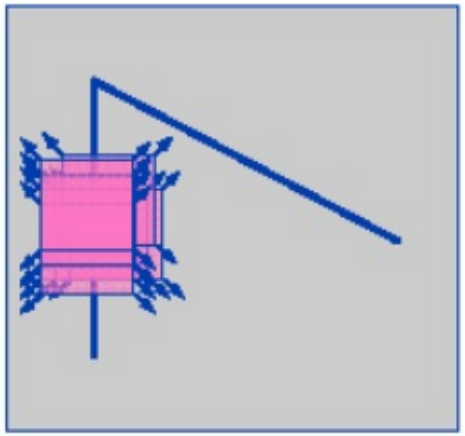
- We should easily recognize the point by looking at intensity values within a small window
- Shifting the window in *any direction* should yield a *large change* in appearance.



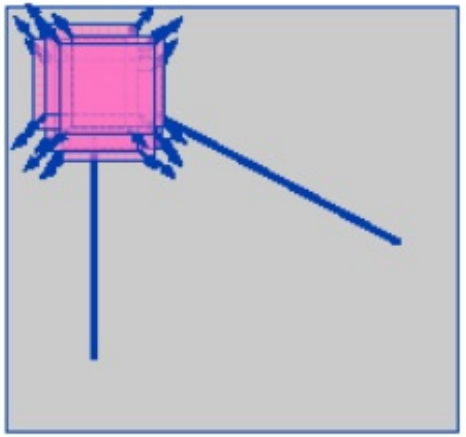
Harris Corner Detetctor



"flat" region:
no change in
all directions



"edge":
no change along
the edge direction



"corner":
significant change
in all directions

Harris corner detector gives a mathematical approach for determining which case holds.

Harris Detector - Math

Change of intensity 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

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.

Taylor Series

$$f(x+u,y+v) = f(x,y) + uf_x(x,y) + vf_y(x,y) +$$

First partial derivatives

$$\frac{1}{2!}\left[u^2 f_{xx}(x,y) + uvf_{xy}x,y + v^2 f_{yy}(x,y)\right] +$$

Second partial derivatives

$$\frac{1}{3!}\left[u^3 f_{xxx}(x,y) + u^2 v f_{xxy}(x,y) + uv^2 f_{xyy}(x,y) + v^3 f_{yyy}(x,y)\right]$$

Third partial derivatives

$+ \ldots$ (Higher order terms)

First order approx

$$f(x+u,y+v) \approx f(x,y) + uf_x(x,y) + vf_y(x,y)$$

Harris Corner Derivation

$$\sum [I(x+u,y+v)-I(x,y)]^2$$

$$\approx \sum [I(x,y)+uI_x+vI_y-I(x,y)]^2 \quad \text{First order approx}$$

$$= \sum u^2I_x^2+2uvI_xI_y+v^2I_y^2$$

$$= \sum \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \quad \text{Rewrite as matrix equation}$$

$$= \begin{bmatrix} u & v \end{bmatrix} \left(\sum \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}$$

Harris Corner Derivation

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

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

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

Windowing function - computing a weighted sum (simplest case, w=1)

Note: these are just products of components of the gradient, Ix, Iy

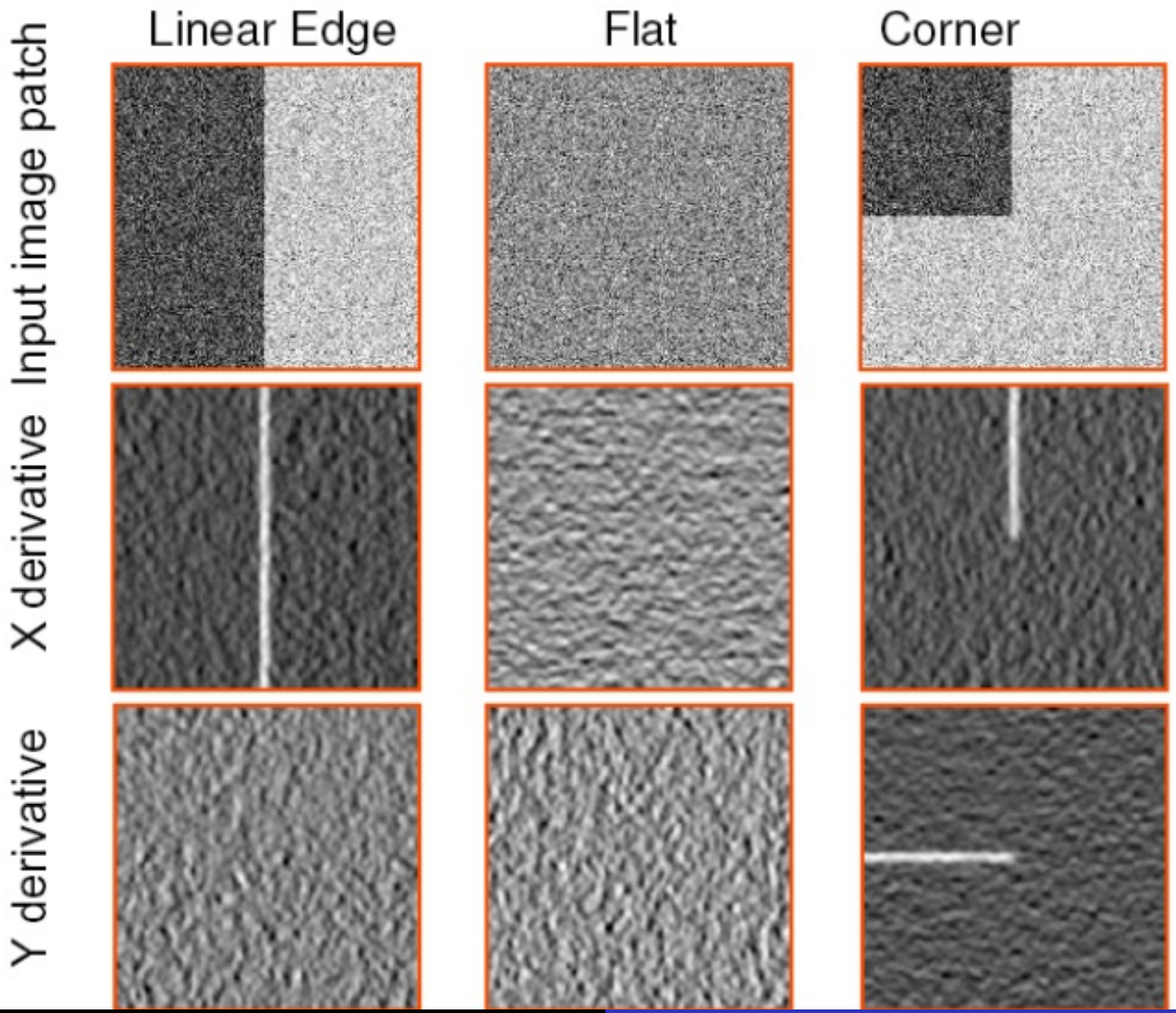
Harris Detector - Intuition

Treat gradient vectors as a set of (dx,dy) points with a center of mass defined as being at (0,0).

Fit an ellipse to that set of points via scatter matrix

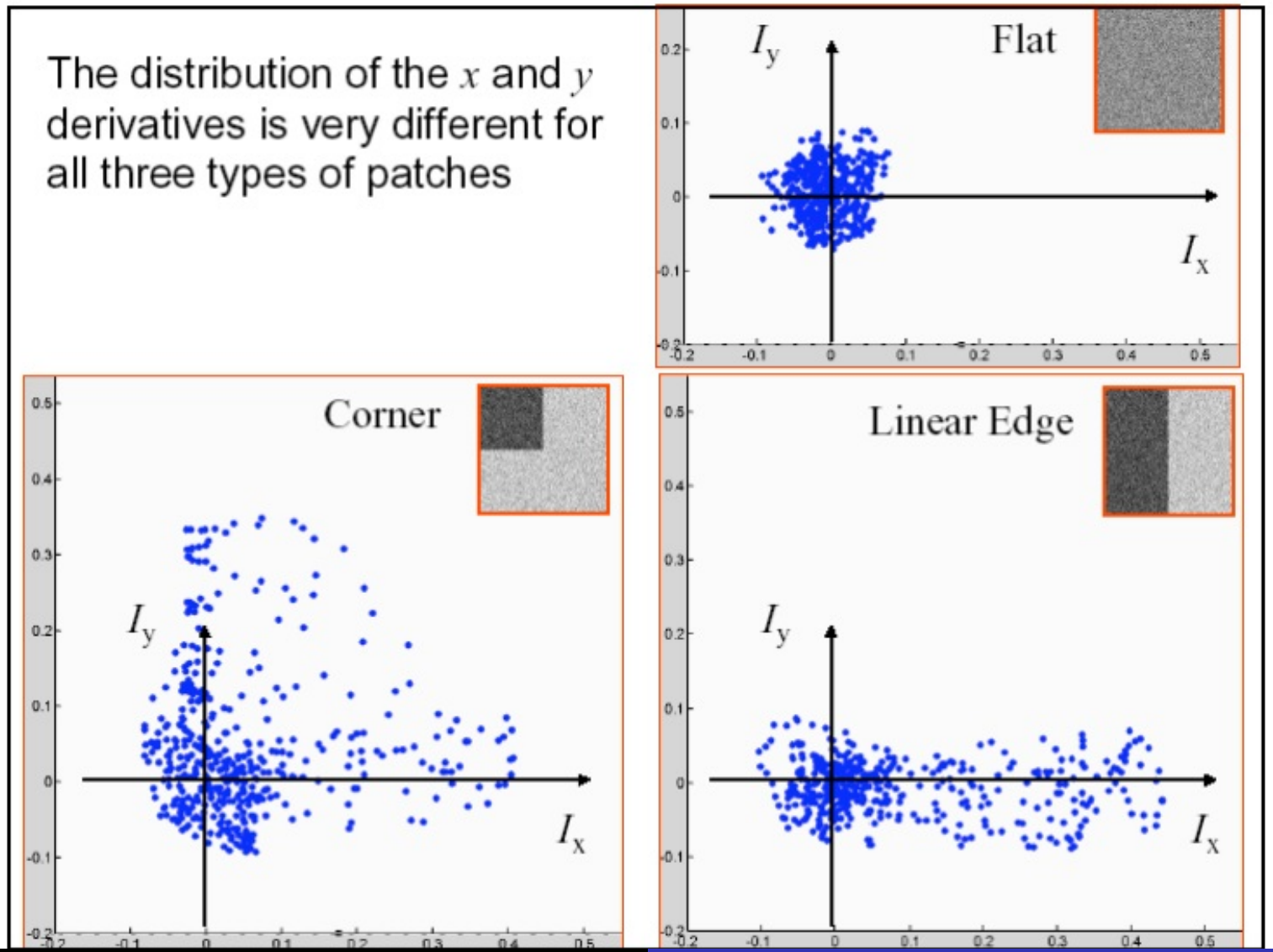
Analyze ellipse parameters for varying cases…

Example Images



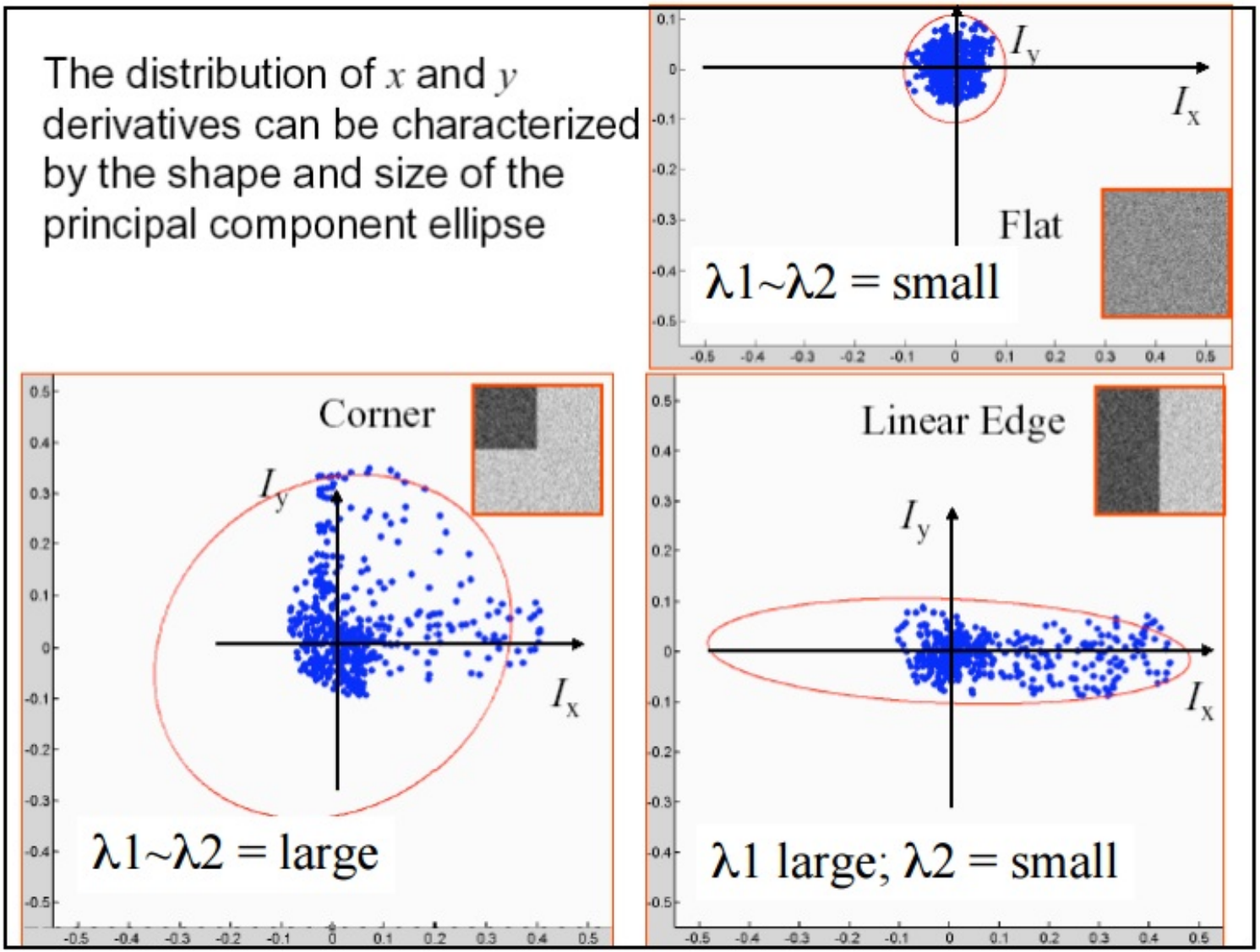
Plotting the gradients

The distribution of the x and y derivatives is very different for all three types of patches

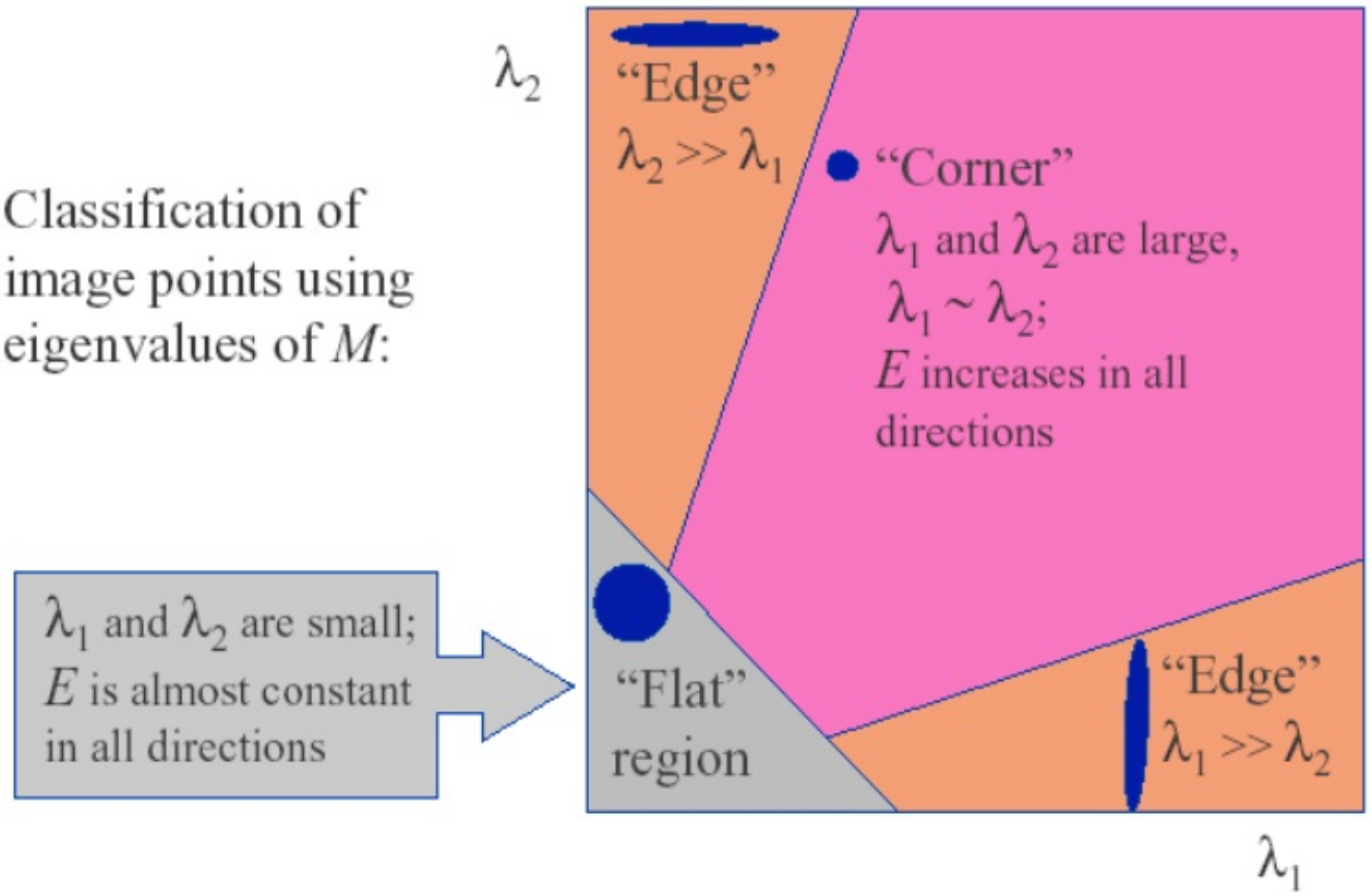


Fitting Ellipses to the set of points

The distribution of x and y derivatives can be characterized by the shape and size of the principal component ellipse



Classification via Eigen Values



Classification via Eigen Values

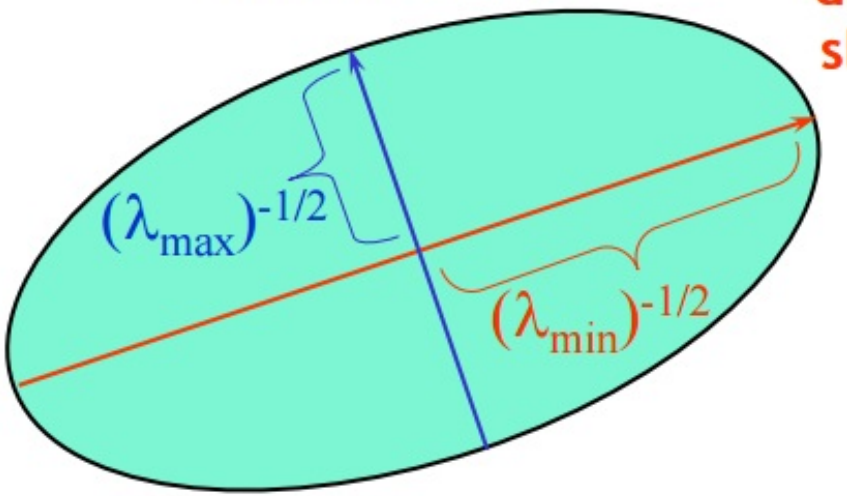
$$E(u,v) \cong [u,v] \quad M \quad \begin{bmatrix} u \\ v \end{bmatrix}$$

λ_1, λ_2 – eigenvalues of M

Ellipse $E(u,v)$ = const

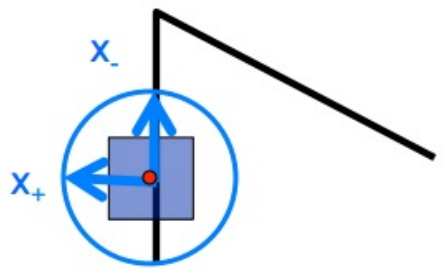
direction of the fastest change

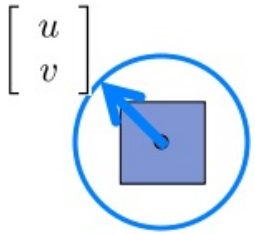
direction of the slowest change



Classification via Eigen Values

$$E(u,v) = \sum_{(x,y)\in W} [u\ v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$





Eigenvalues and eigenvectors of H

- Define shifts with the smallest and largest change (E value)
- x_+ = direction of largest increase in E.
- λ_+ = amount of increase in direction x_+
- x_- = direction of smallest increase in E.
- λ_- = amount of increase in direction x_+

$$Hx_+ = \lambda_+ x_+$$
$$Hx_- = \lambda_- x_-$$

Source: S. Seitz.

Eigen Values Computation

The **eigenvectors** of a matrix **A** are the vectors **x** that satisfy:

$$Ax = \lambda x$$

The scalar λ is the **eigenvalue** corresponding to **x**

- The eigenvalues are found by solving:

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

- In our case, ***A*** = ***H*** is a 2x2 matrix, so we have

$$det \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

- The solution:

$$\lambda_{\pm} = \frac{1}{2}\left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2}\right]$$

Once you know λ, you find **x** by solving

$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$

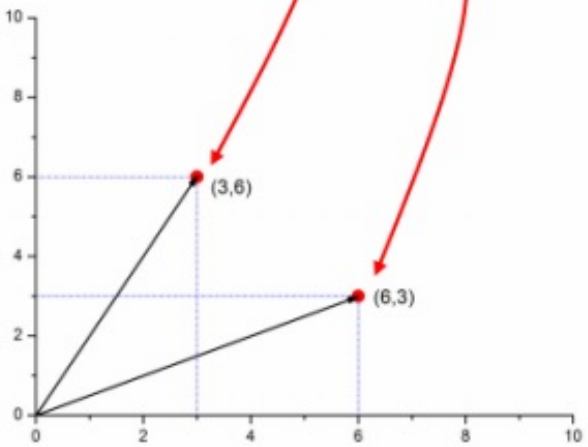
Source: S. Seitz.

Eigen Values Numerical Example

Numerical Example 1

$$[A] = \begin{bmatrix} 3 & 6 \\ 6 & 3 \end{bmatrix}$$

Let's get the eigenvalues of matrix [A]



$$\begin{vmatrix} 3-\lambda & 6 \\ 6 & 3-\lambda \end{vmatrix} = 0$$

$$(3-\lambda)^2 - 6\times 6 = 0$$

$$9 - 6\lambda + \lambda^2 - 36 = 0$$

$$\lambda^2 - 6\lambda - 27 = 0$$

$$(\lambda - 9)(\lambda + 3) = 0$$

$$\lambda_1 = 9, \lambda_2 = -3$$

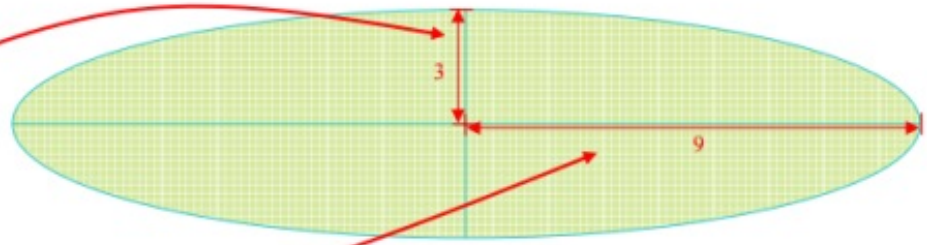
Two eigenvalues from 2 variables

The eigenvalues represent the magnitudes, or lengths, of the major and minor axes of an ellipse.
See Next Page!

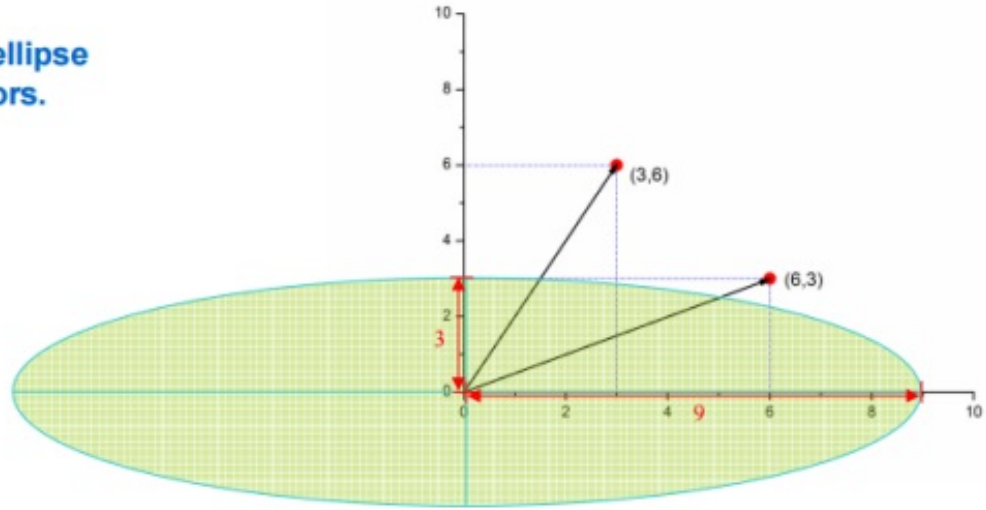
Eigen Values Numerical Example

The eigenvalues represent the magnitudes, or lengths, of the major and minor axes of an ellipse.

$\lambda_1 = 9, \lambda_2 = -3$

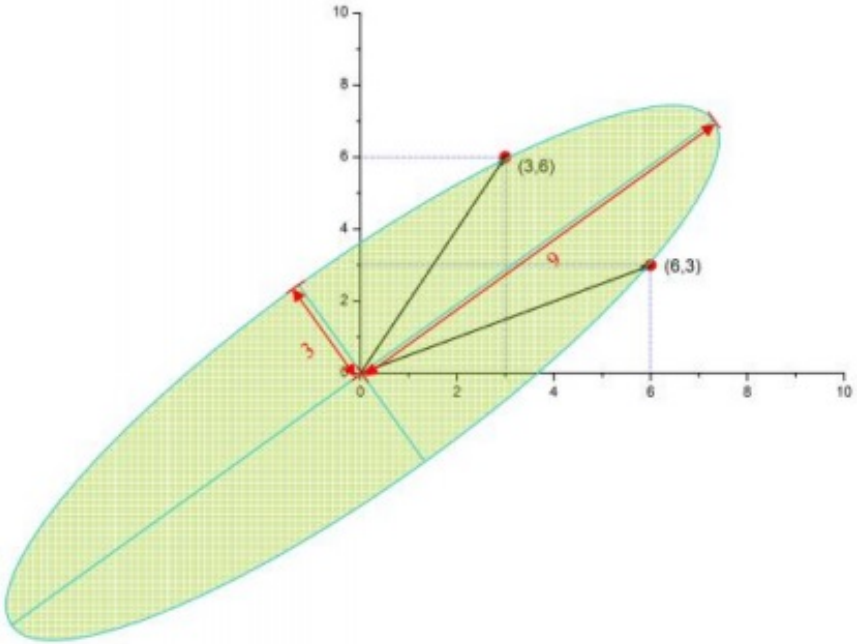


Let's overlap the center of ellipse onto center of the two vectors.



Eigen Values Numerical Example

Then rotate the ellipse to its envelope can be on the two points.



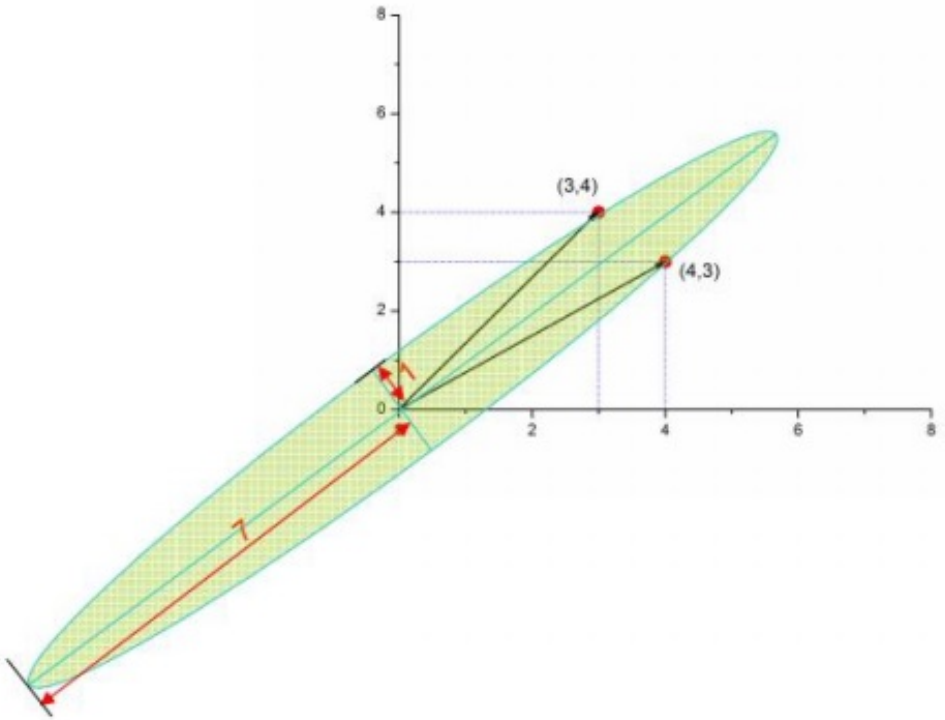
Eigen Values Numerical Example

If the two points are closer than the case of numerical example 1,

$$[A] = \begin{bmatrix} 3 & 4 \\ 4 & 3 \end{bmatrix}$$

Then the eigenvalues are

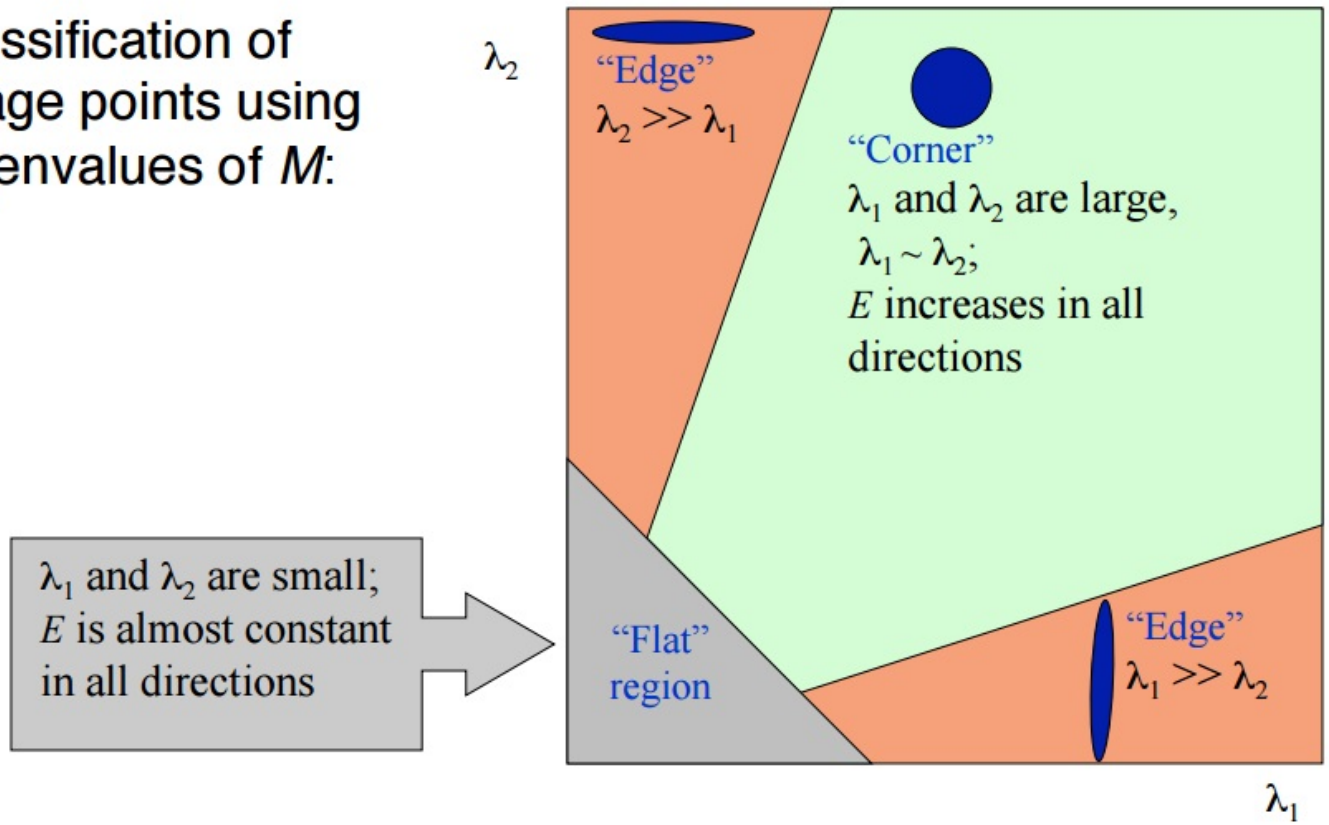
$$\lambda_1 = 7, \lambda_2 = -1$$



The sum of the eigenvalues(=λ_1+ λ_2=7-1=6) is always equal to the sum of the diagonal elements(3+3), i.e. trace of the original matrix.

Classification via Eigen Values

Classification of image points using eigenvalues of *M*:



Source: Kokkinos, Saverese.

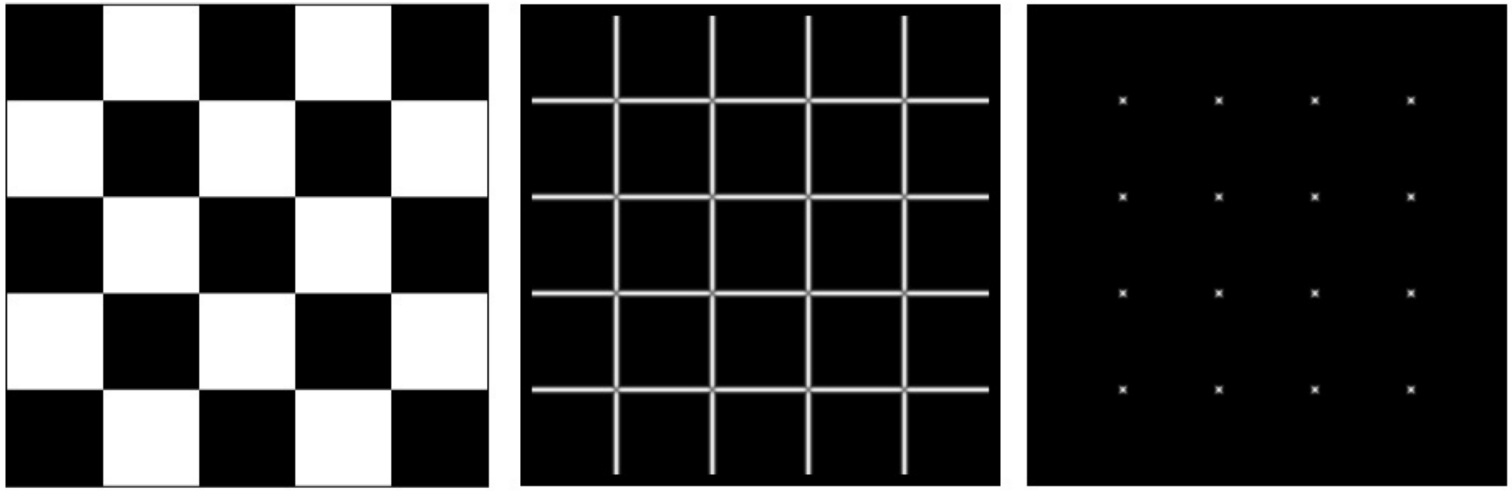
Classification via Eigen Values

How are λ_+, x_+, λ_-, and x_+ relevant for feature detection?

- What's our feature scoring function?

Want *E(u,v)* to be *large* for small shifts in *all* directions

- the *minimum* of *E(u,v)* should be large, over all unit vectors [u v]
- this minimum is given by the smaller eigenvalue (λ_-) of *H*



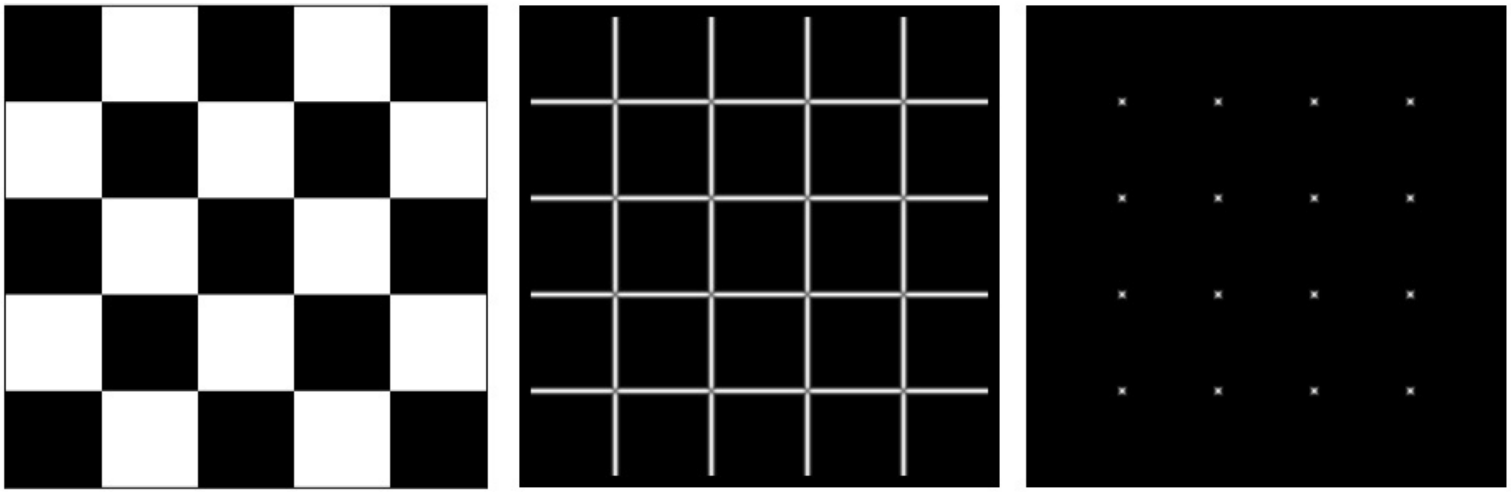
Source: S. Seitz.

I λ_+ λ_-

Corner Detection Pipeline

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



Source: S. Seitz.

I λ_+ λ_-

Harris Operator

λ- is a variant of the "Harris operator" for feature detection

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

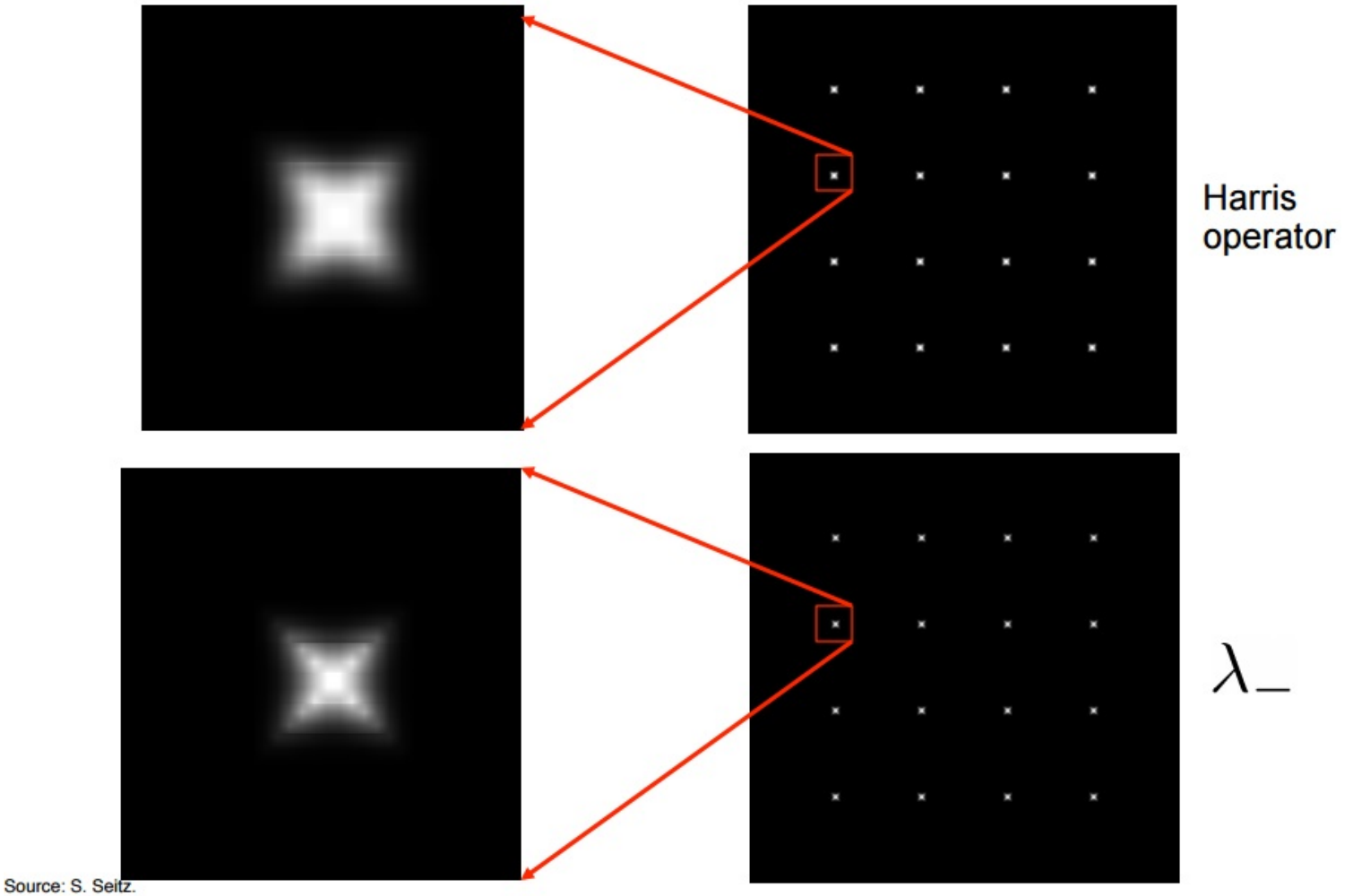
$$= \frac{determinant(H)}{trace(H)}$$

- The *trace* is the sum of the diagonals, i.e., *trace(H)* = $h_{11} + h_{22}$
- Very similar to λ- but less expensive (no square root)
- Called the "Harris Corner Detector" or "Harris Operator"
- Lots of other detectors, this is one of the most popular

C.Harris and M.Stephens. "A Combined Corner and Edge Detector."
Proceedings of the 4th Alvey Vision Conference: pages 147--151. 1988.

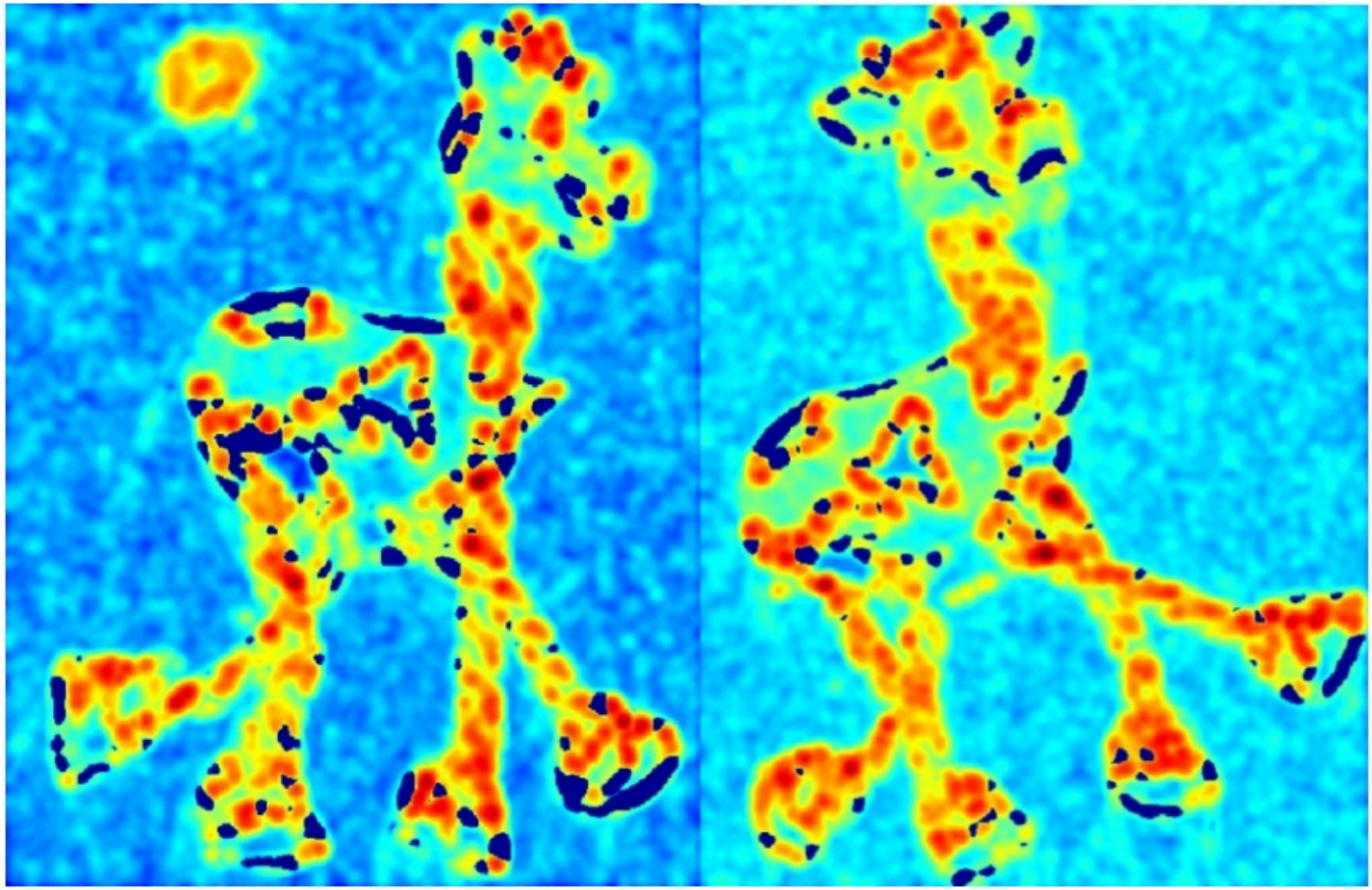
Source: S. Seitz.

Harris Operator



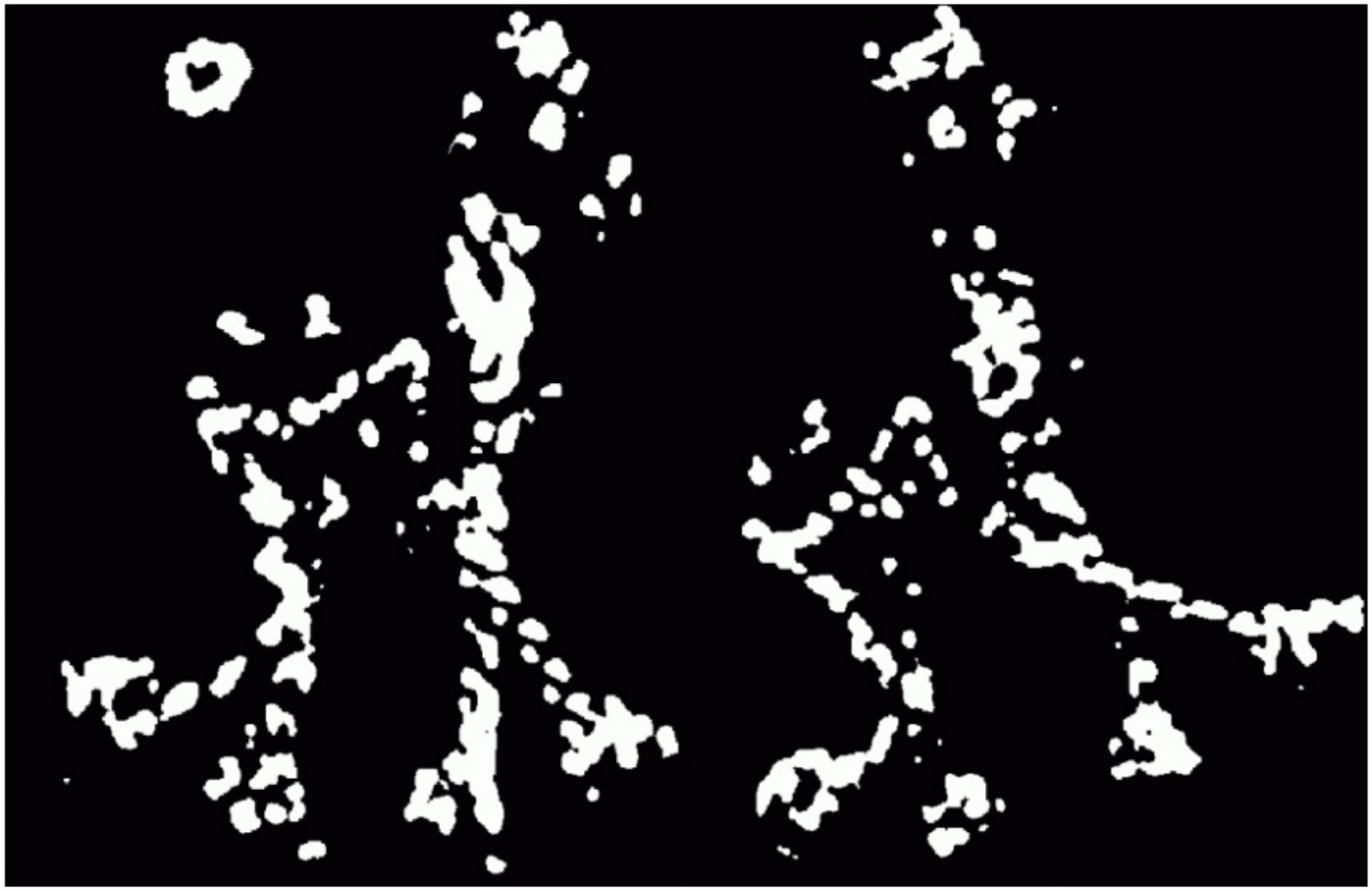
Corner Detection Example

Corner Detection Example



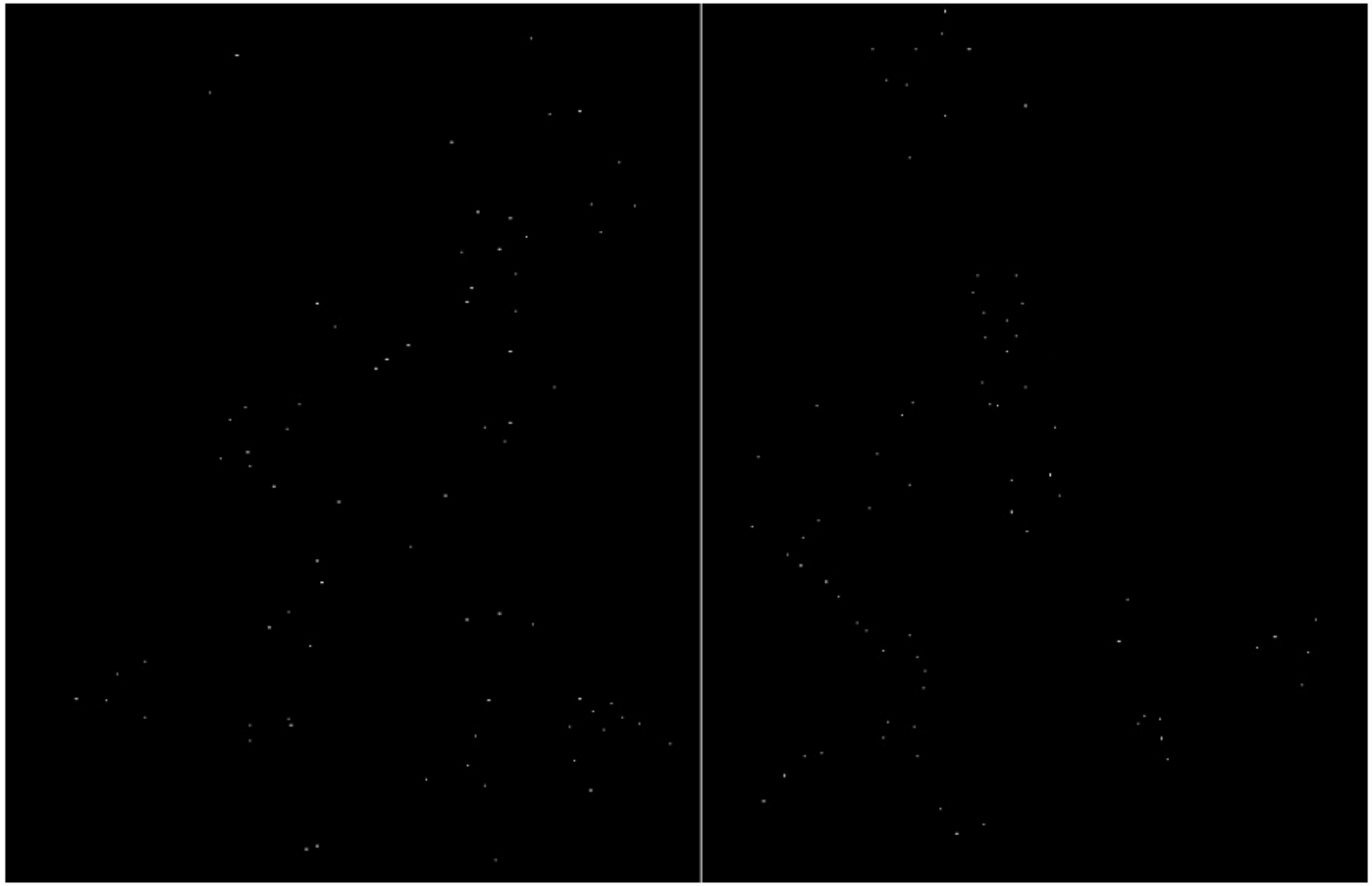
Source: S. Seitz.

Corner Detection Example



Source: S. Seitz.

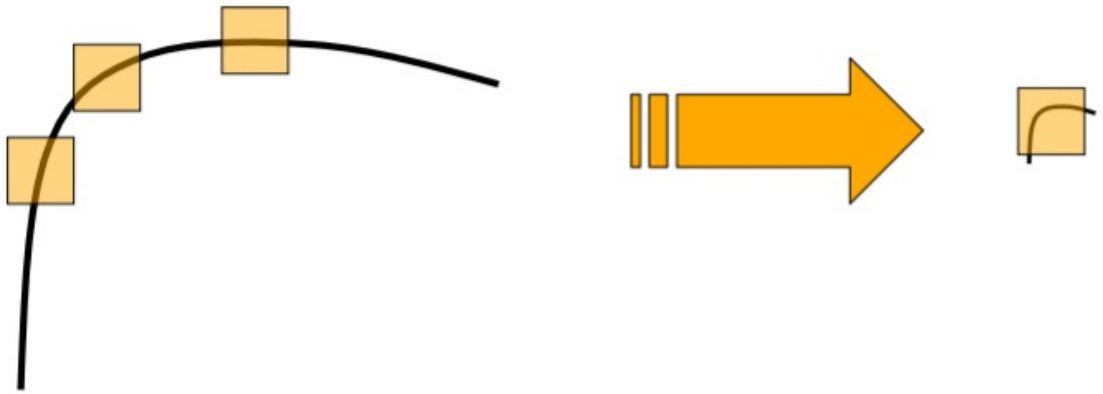
Corner Detection Example



Source: S. Seitz.

Corner Detection Example

Are Corners Scale Invariant?

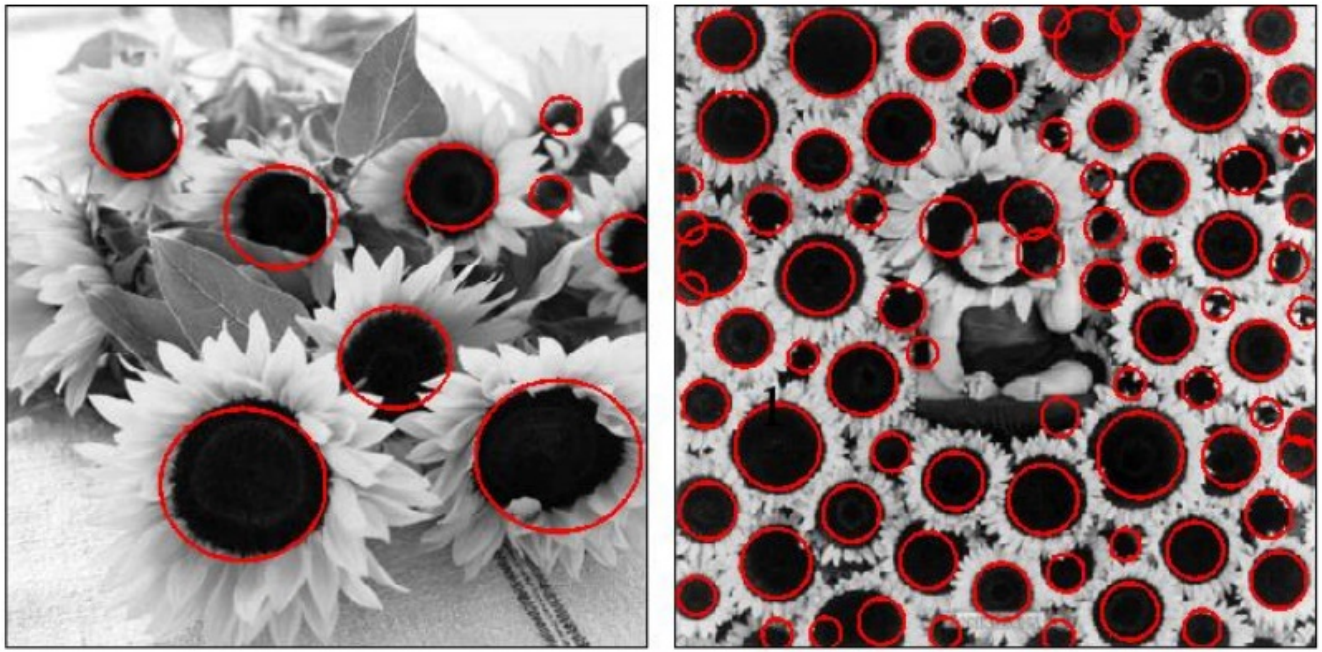


All points will be classified as edges

Corner !

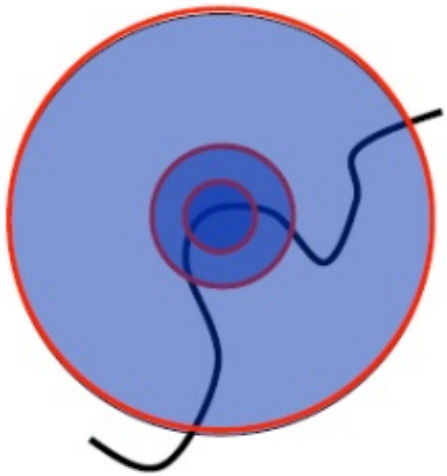
Scale Invariant Features - Blobs

Simple technique to detect features that are scale invariant



Scale Invariant Detection

Suppose you're looking for corners



Key idea: find scale that gives local maximum of f

- f is a local maximum in both position and scale
- Common definition of f: Laplacian
  (or difference between two Gaussian filtered images with different sigmas)

Efficiency - Use of DoG vs Laplacian

Computing Difference of Gaussians is simpler!

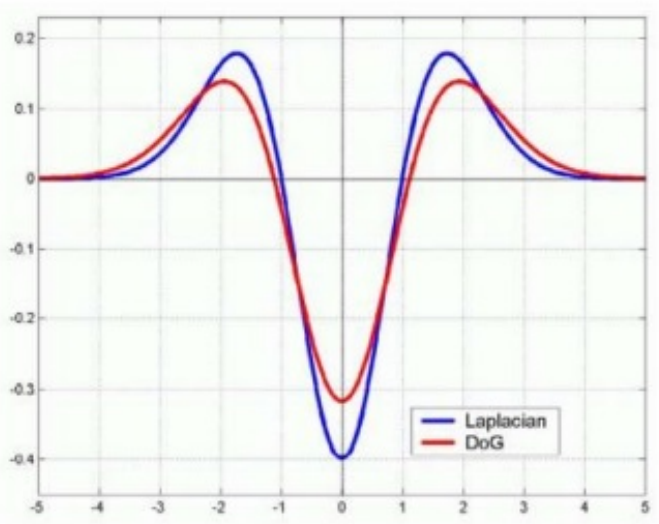
Approximating the Laplacian with a difference of Gaussians:

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

(Laplacian)

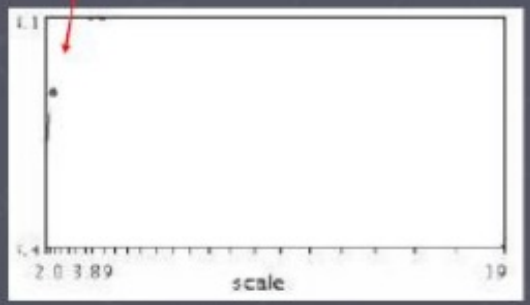
$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)



Automatic Scale Selection

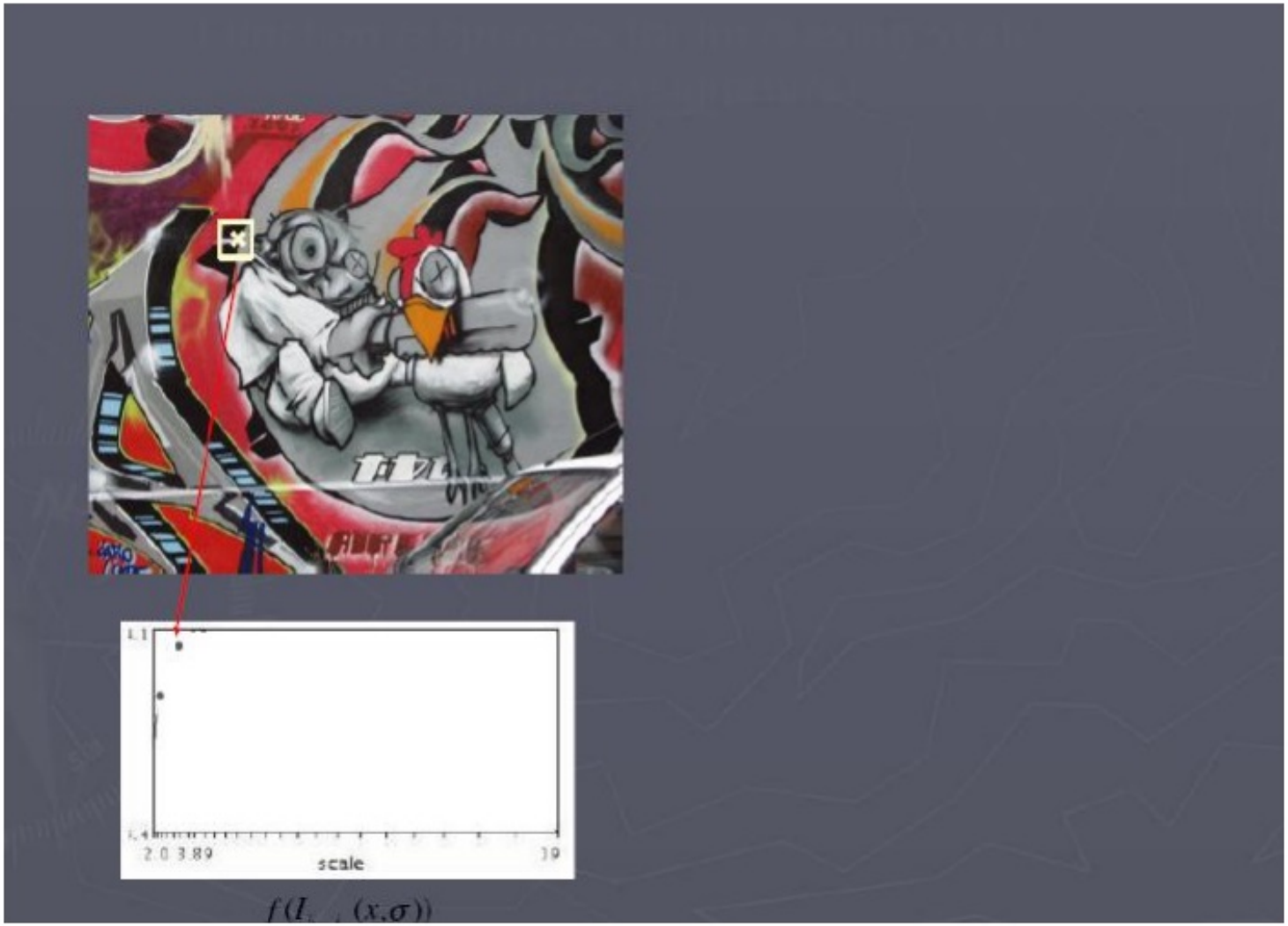
Lindeberg et al., 1996



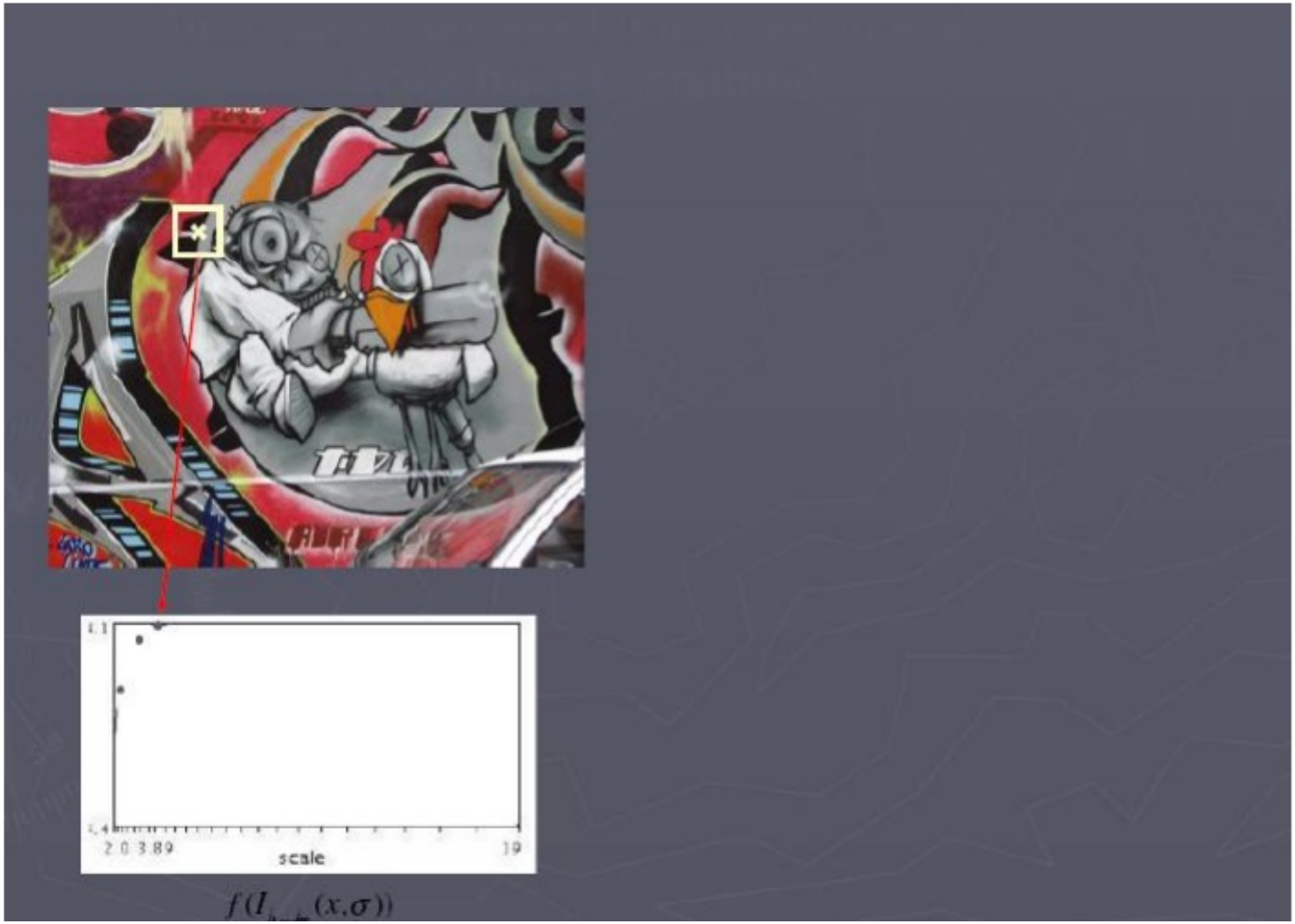
$f(I_{i_1 \ldots i_m}(x,\sigma))$

Slide from Tinne Tuytelaars

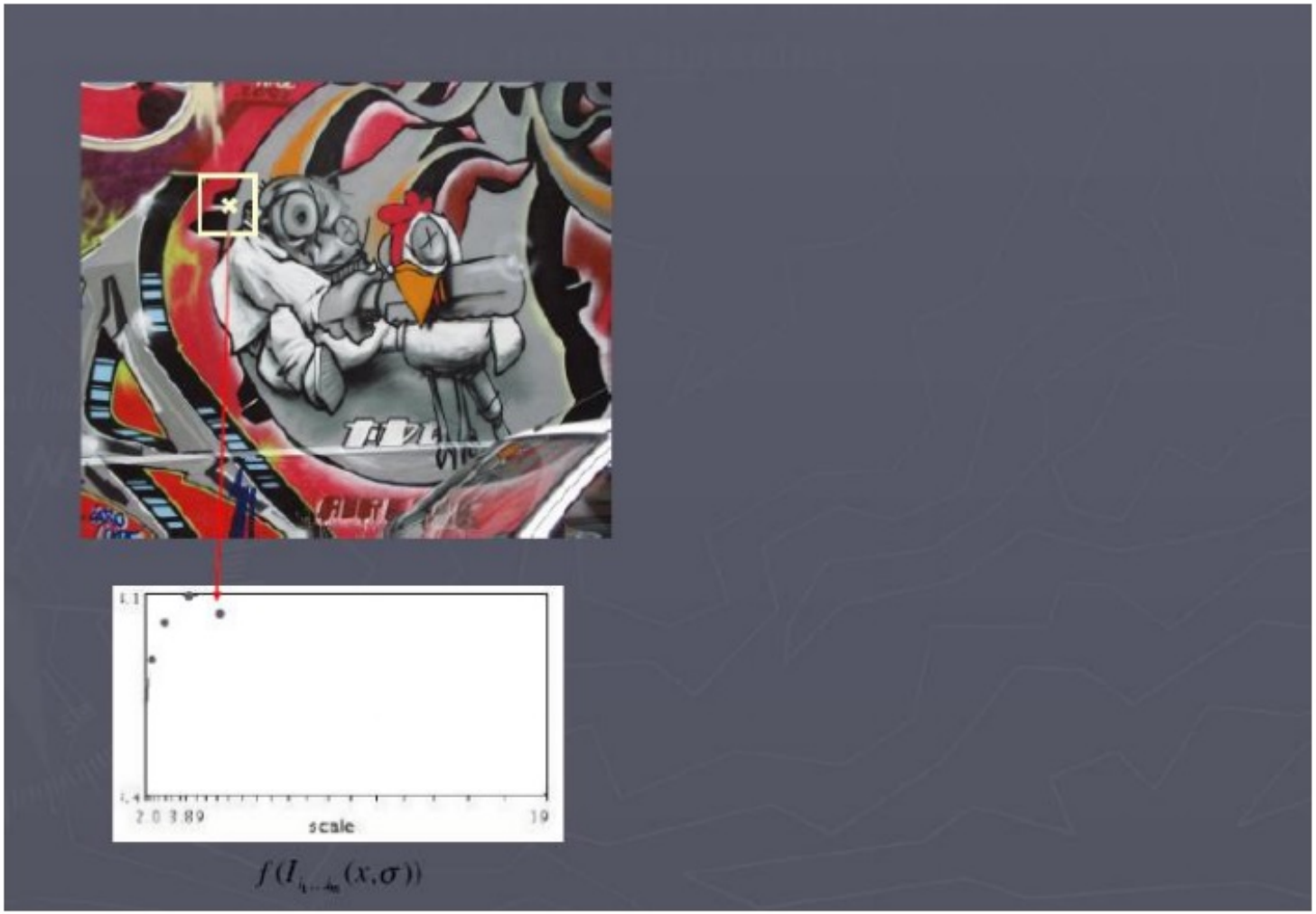
Automatic Scale Selection



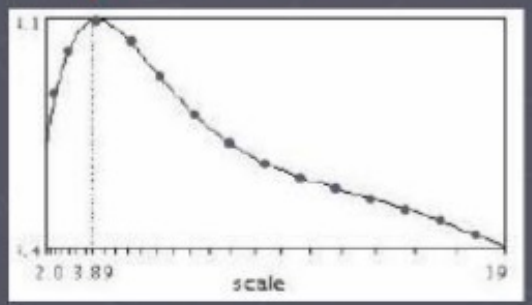
Automatic Scale Selection



Automatic Scale Selection

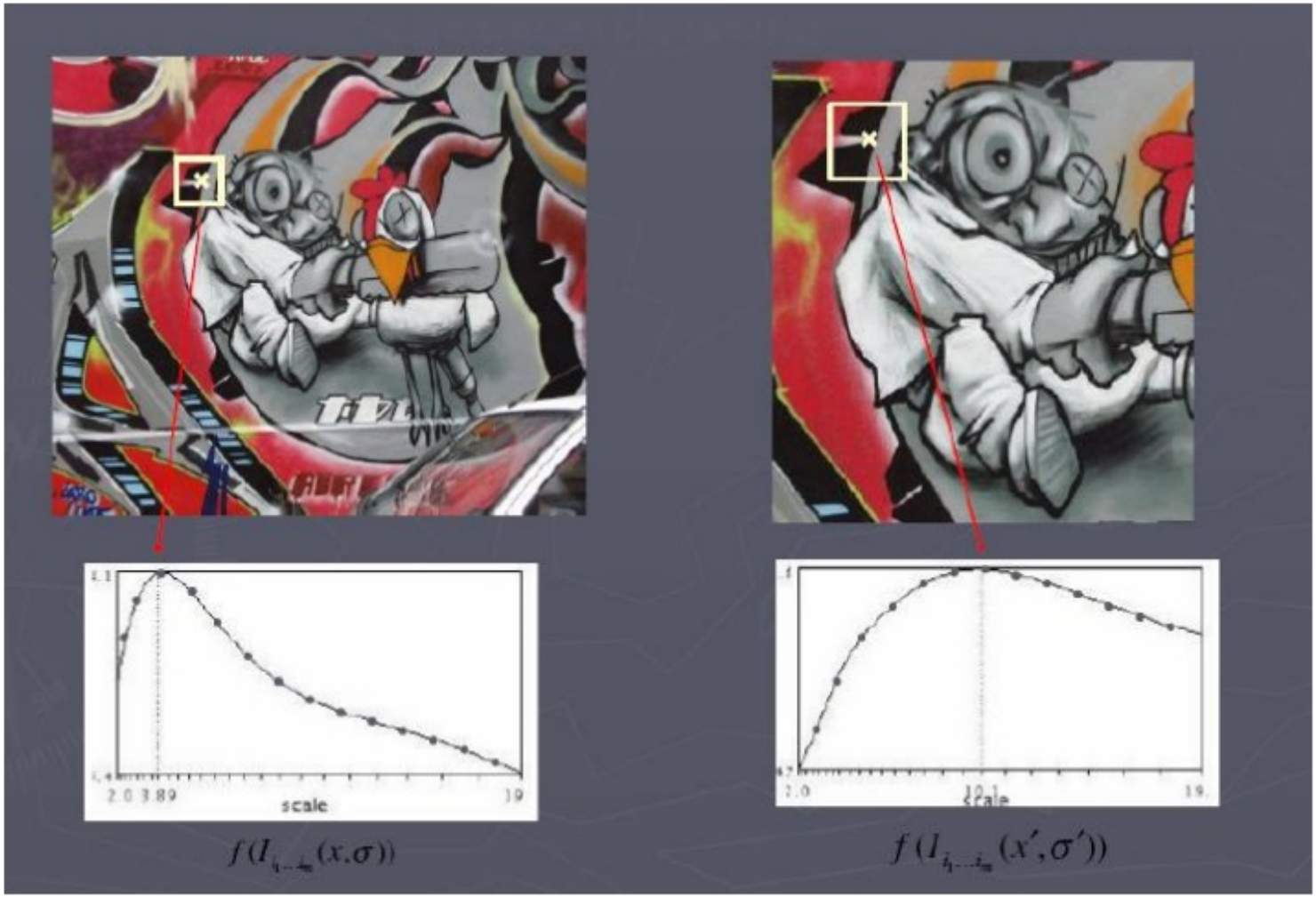


Automatic Scale Selection



$f(I_{i_1 \ldots i_m}(x,\sigma))$

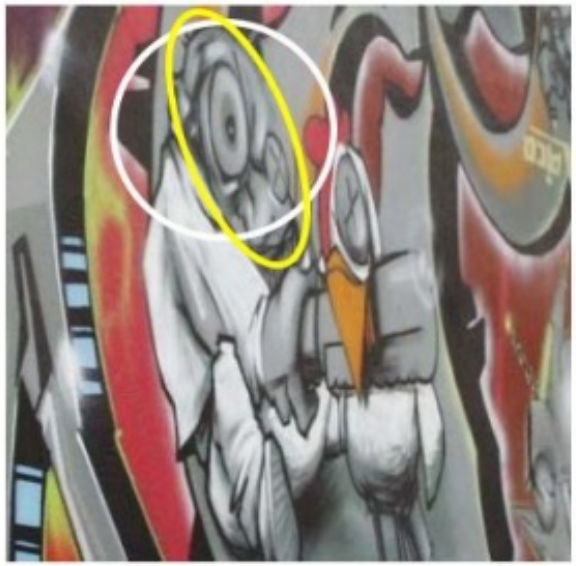
Automatic Scale Selection



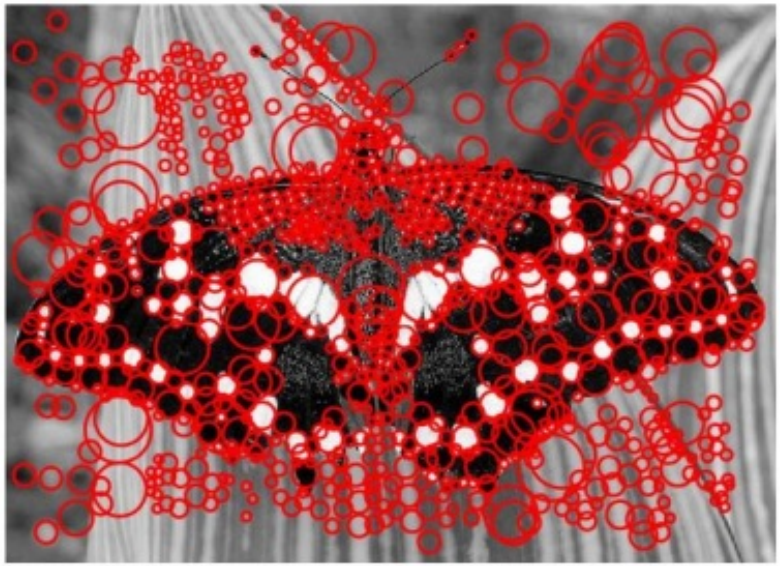
Affine Transformation

Perform Eigen decomposition on the blob window

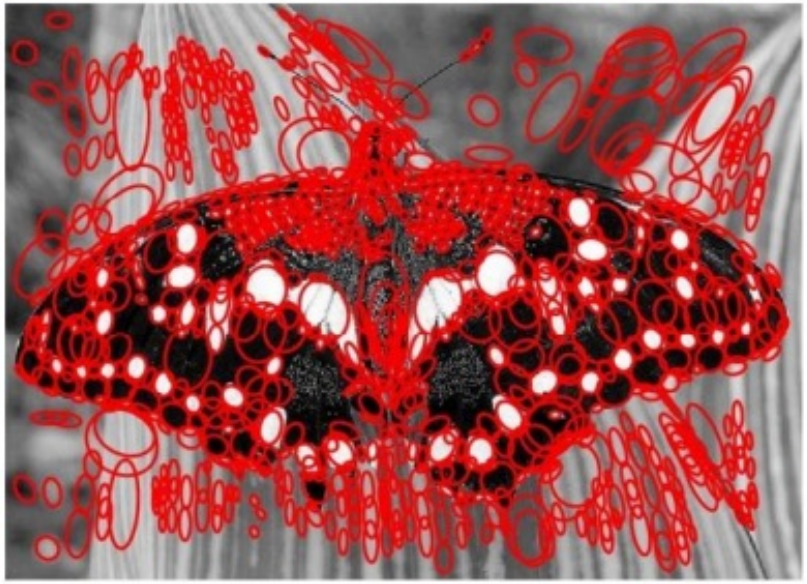
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$



After Affine Adaptation



Scale-invariant regions (blobs)



Affine-adapted blobs

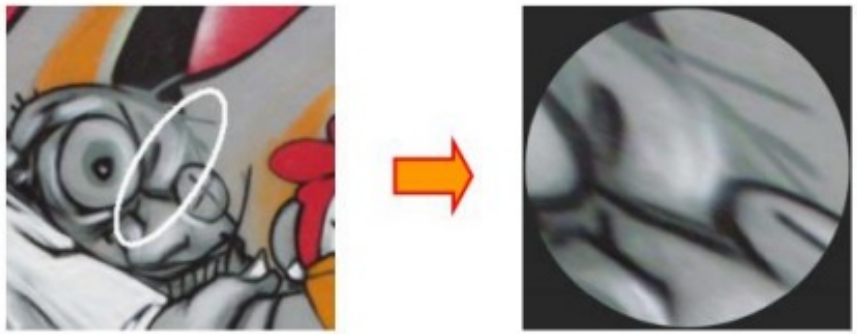
Towards Invariant Description

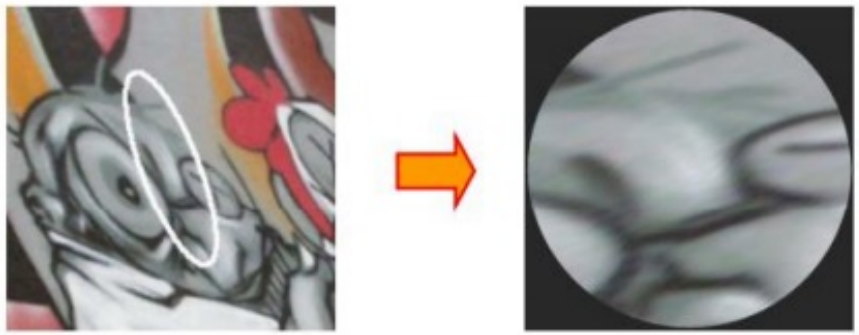
- Geometrically transformed versions of the same neighborhood will give rise to regions that are related by the same transformation
- What to do if we want to compare the appearance of these image regions?
  - *Normalization*: transform these regions into same-size circles

Affine Normalization

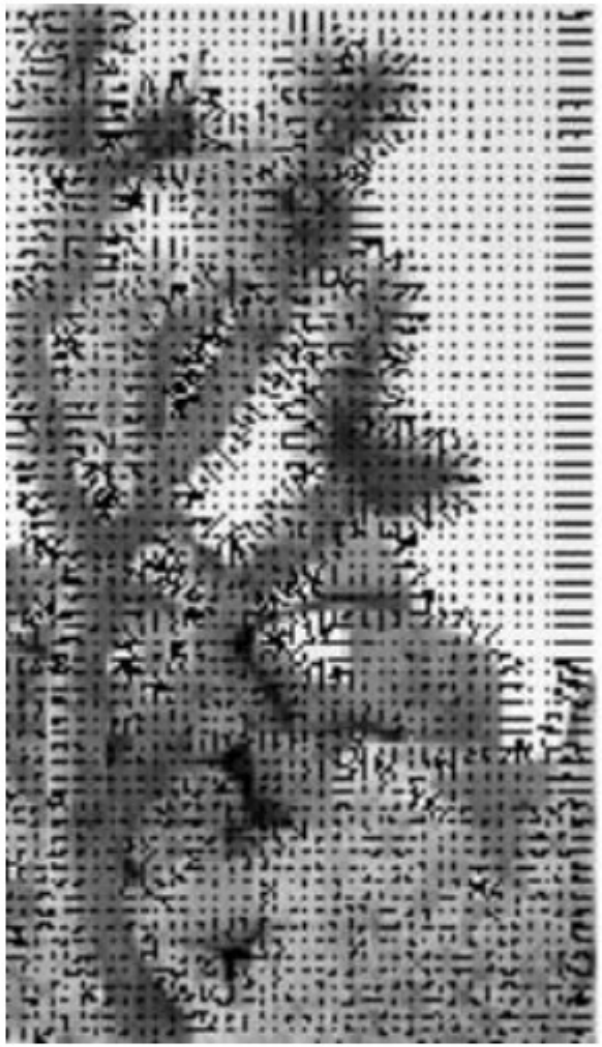
Problem: There is no unique transformation from an ellipse to a unit circle

- We can rotate or flip a unit circle, and it still stays a unit circle



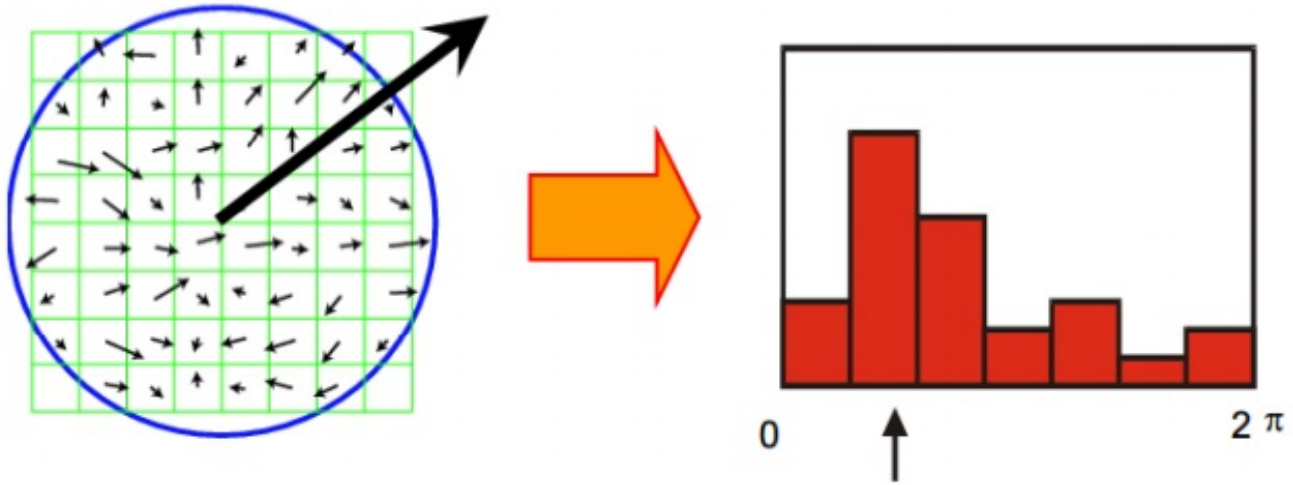


Solution - Gradient Orientations

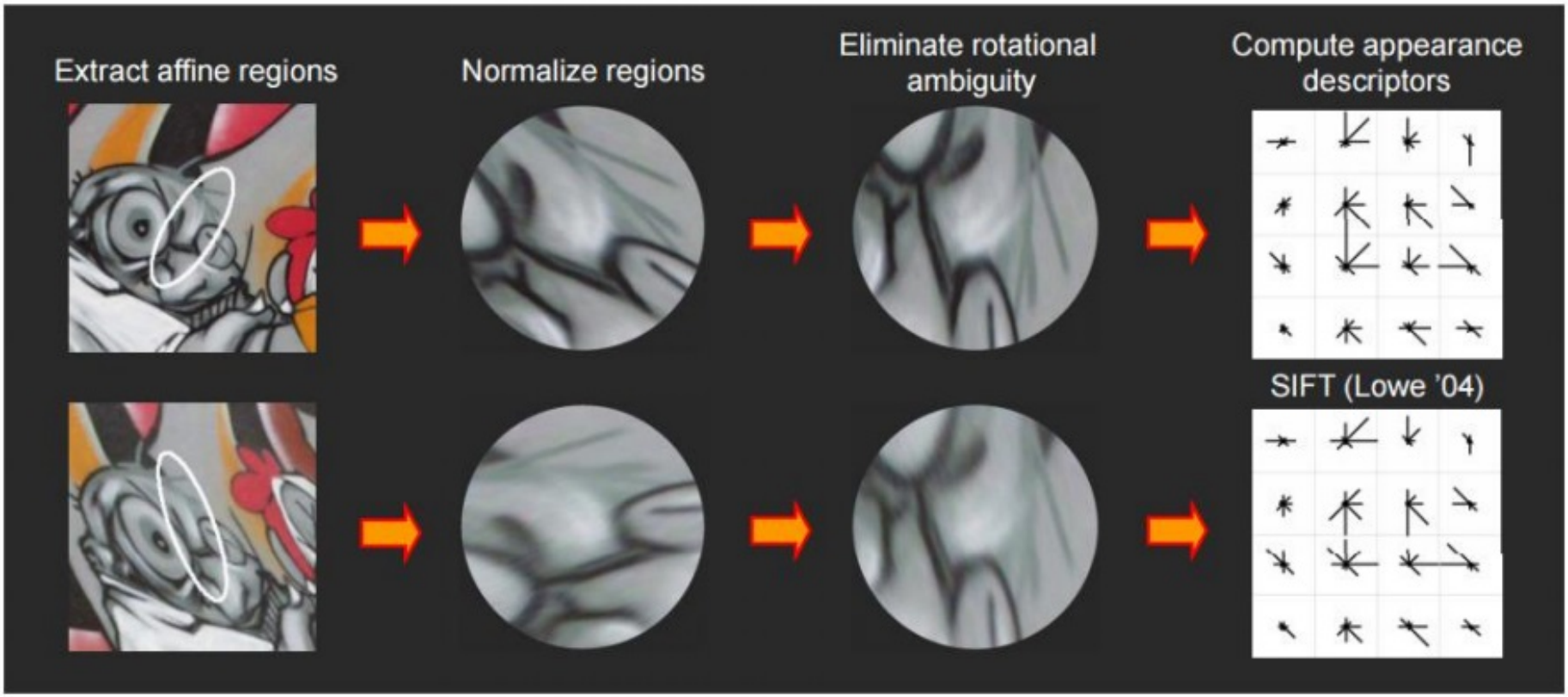


Eliminating Rotational Ambiguity

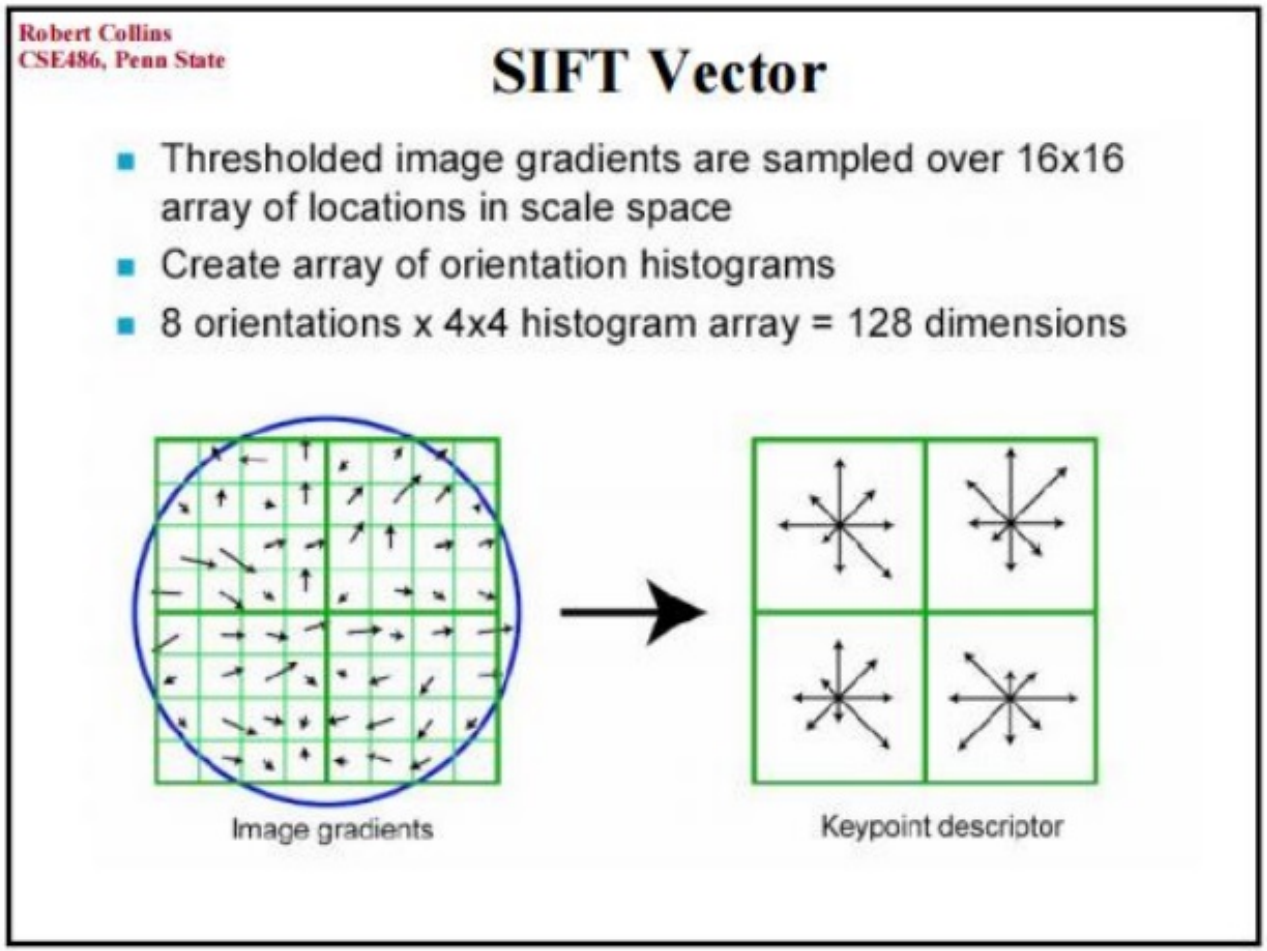
- To assign a unique orientation to circular image windows:
  - Create histogram of local gradient directions in the patch
  - Assign canonical orientation at peak of smoothed histogram



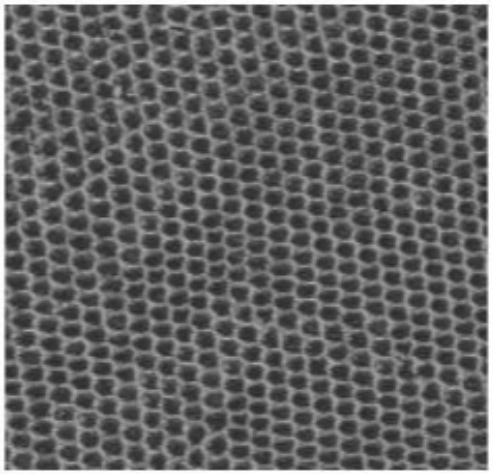
Scale and Rotational Invariant Features



Scale Invariant Feature Transform

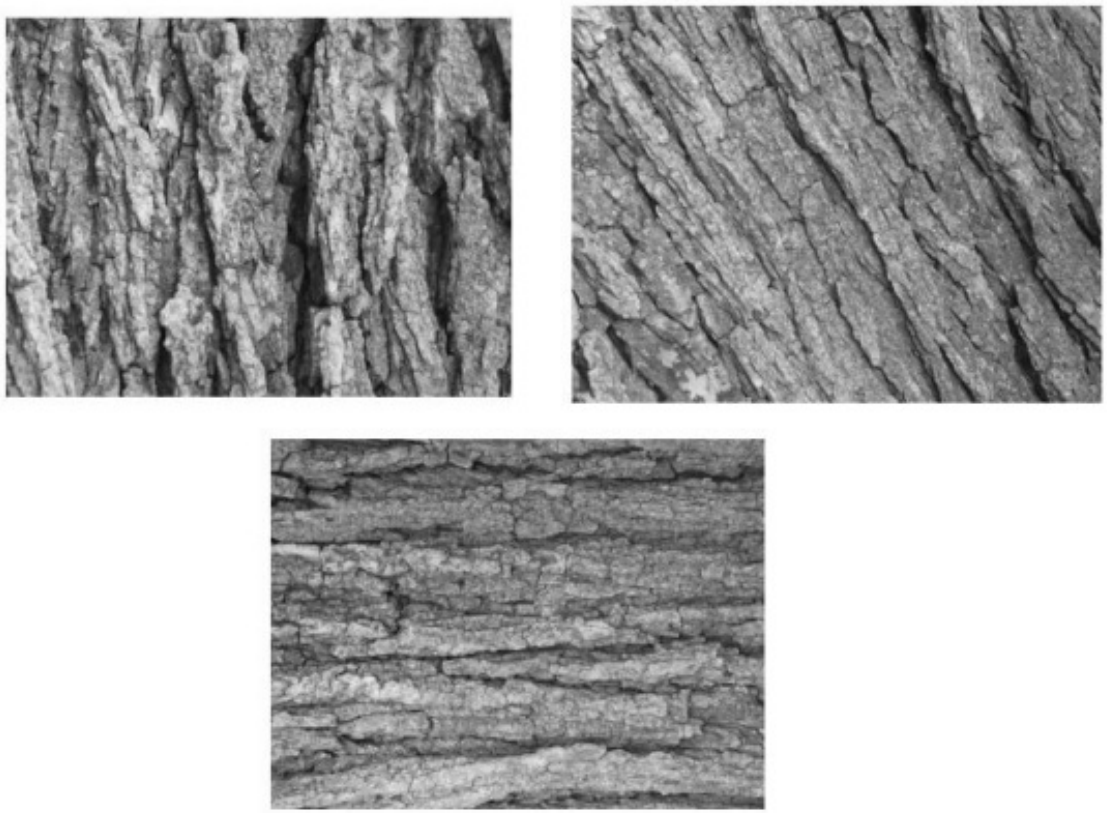


Texture

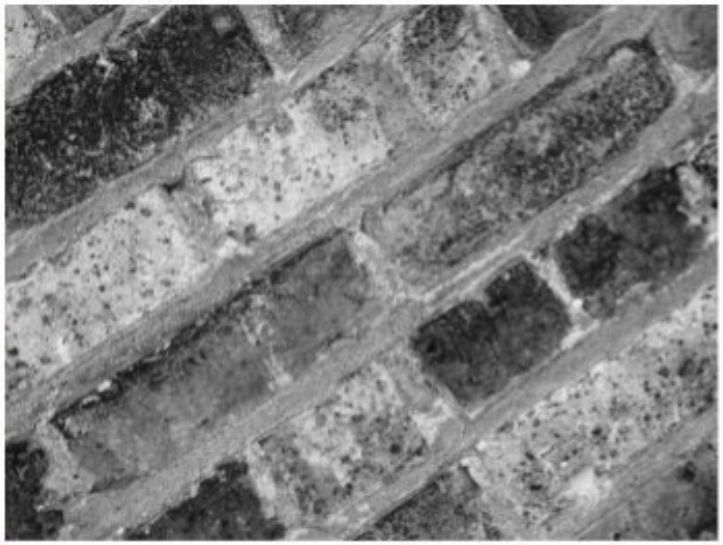




- An image obeying some statistical properties
- Similar structures repeated over and over again
- Often has some degree of randomness

Texture and Orientation



Texture and Scale



Detecting Texture using Spatial Filters

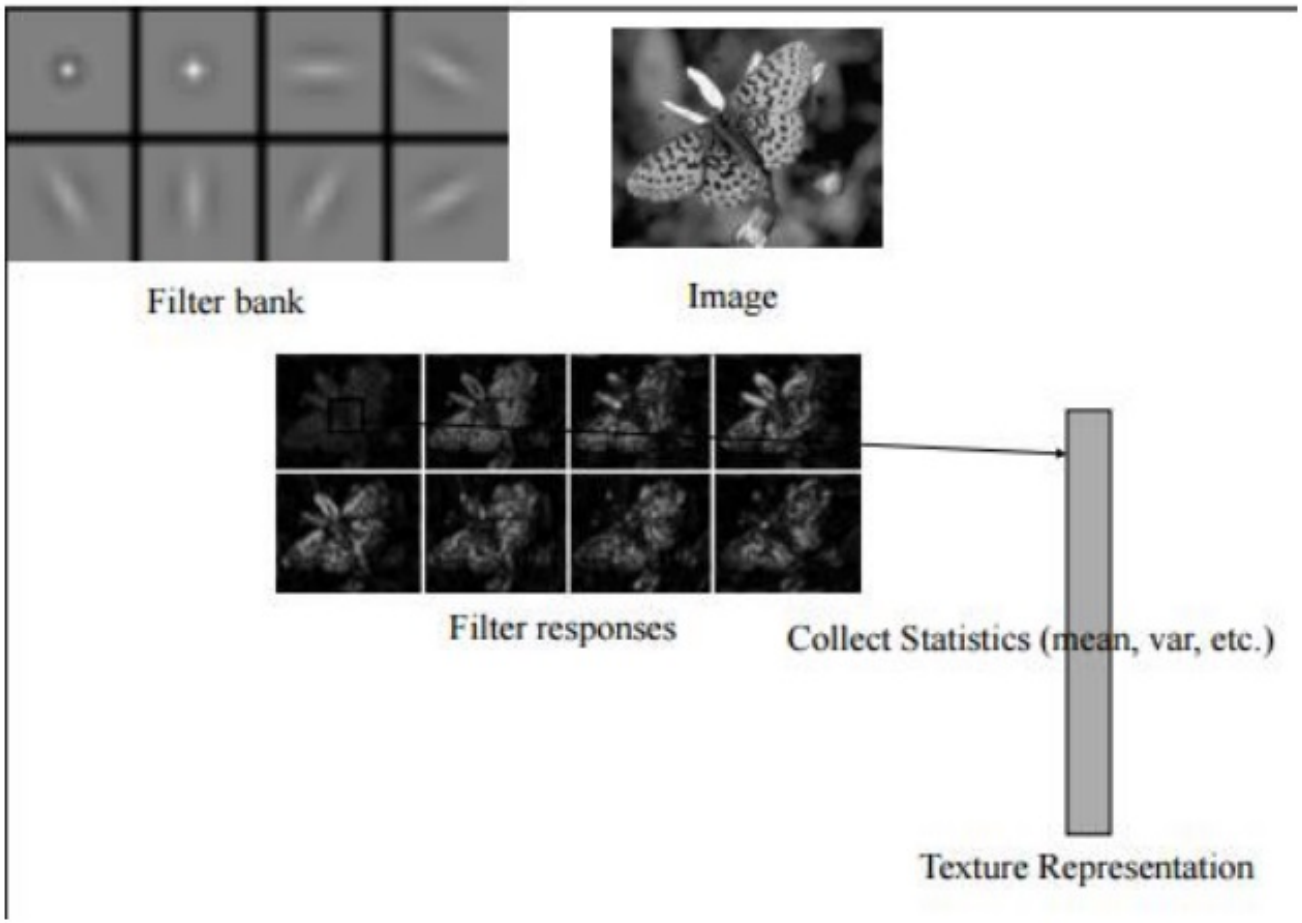
- Look for the subelements
- But what are the subelements, and how do we find them?
- Find subelements by applying filters, looking at the magnitude of the response
- Spots and bars detectors at various scales and orientations.

Typically:

- "Spot" filters are Gaussians or weighted sums of concentric Gaussians.
- "Bar" filters are differentiating oriented Gaussians

CS 534 – Texture - 13

Texture - Feature Descriptor



Feature Matching

NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

Feature Matching - Distance Measure

Feature matching

Given a feature in I_1, how to find the best match in I_2?

1. Define distance function that compares two descriptors
2. Test all the features in I_2, find the one with min distance

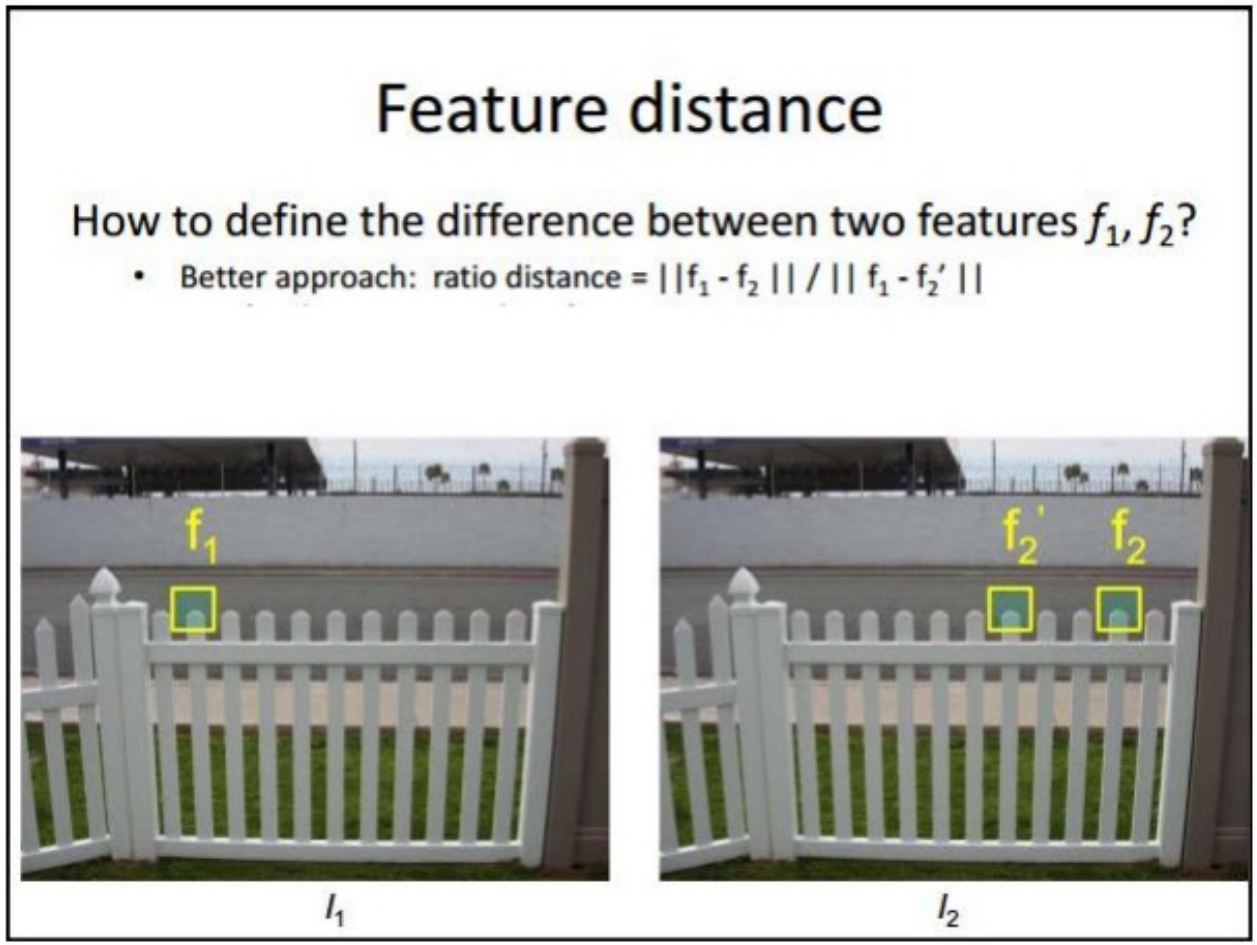
Feature Matching - L2 Norm

Feature distance

How to define the difference between two features f_1, f_2?

- Simple approach: L_2 distance, $||f_1 - f_2||$
- can give good scores to ambiguous (incorrect) matches

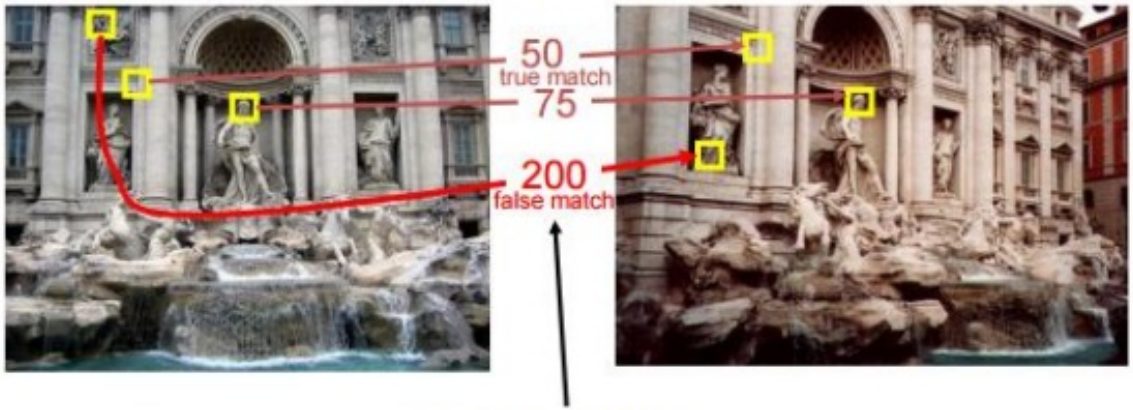
Feature Matching - Ratio of Distance



Feature Matching - Performance

True/false positives

How can we measure the performance of a feature matcher?



The distance threshold affects performance

- True positives = # of detected matches that are correct
  - Suppose we want to maximize these—how to choose threshold?
- False positives = # of detected matches that are incorrect
  - Suppose we want to minimize these—how to choose threshold?

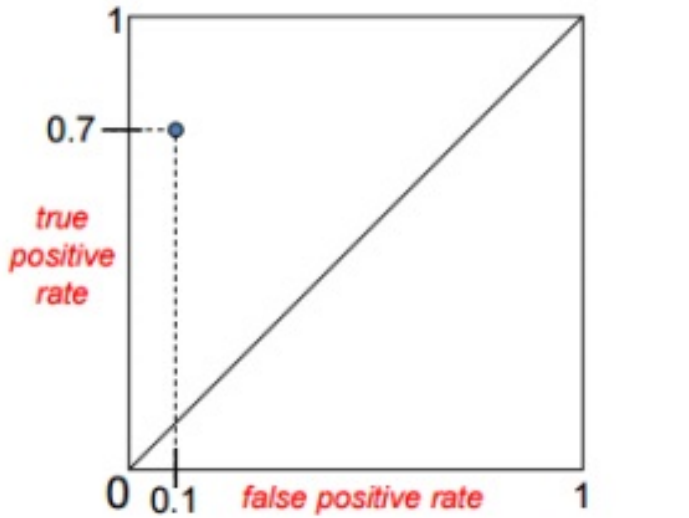
Feature Matching - Performance

Evaluating the results

How can we measure the performance of a feature matcher?

$$\frac{\text{\# true positives}}{\text{\# correctly matched features (positives)}}$$

"recall"



$$\frac{\text{\# false positives}}{\text{\# incorrectly matched features (negatives)}}$$

1 - "precision"

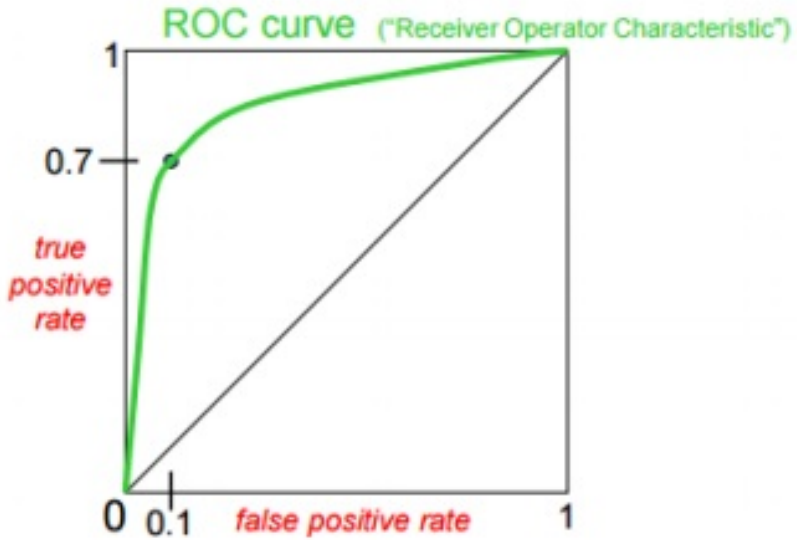
Performance Analysis - ROC Curve

Evaluating the results

How can we measure the performance of a feature matcher?

$$\frac{\text{\# true positives}}{\text{\# correctly matched features (positives)}}$$

"recall"



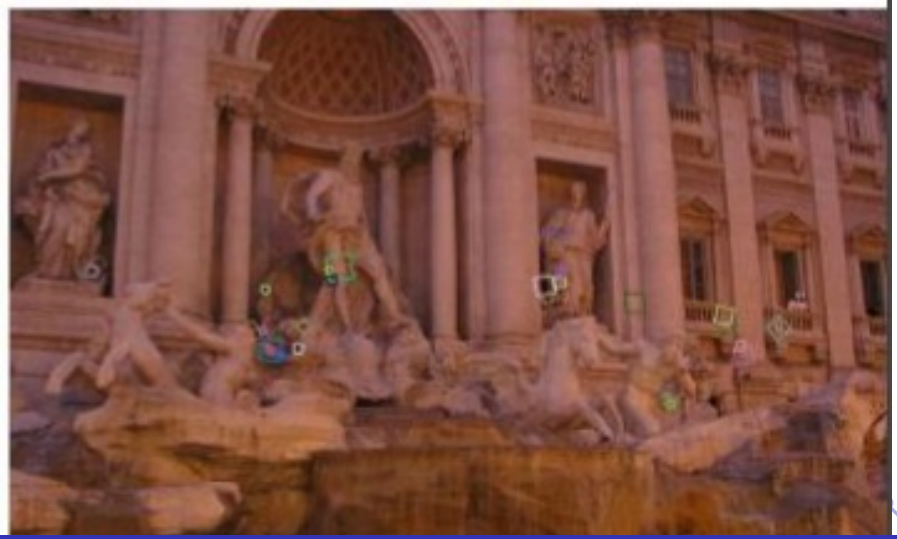
$$\frac{\text{\# false positives}}{\text{\# incorrectly matched features (negatives)}}$$

1 - "precision"

Properties of SIFT based Matching

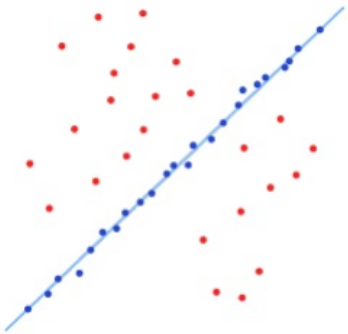
Extraordinarily robust matching technique

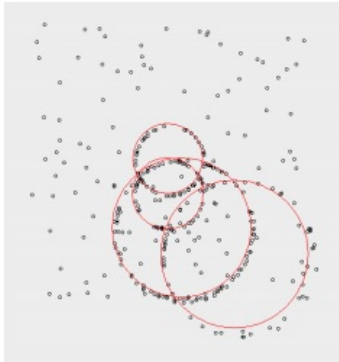
- Can handle changes in viewpoint
  - Up to about 60 degree out of plane rotation
- Can handle significant changes in illumination: Sometimes even day vs. night (below)
- Fast and efficient — can run in real time
- Lots of code available:
  http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT



RANSAC

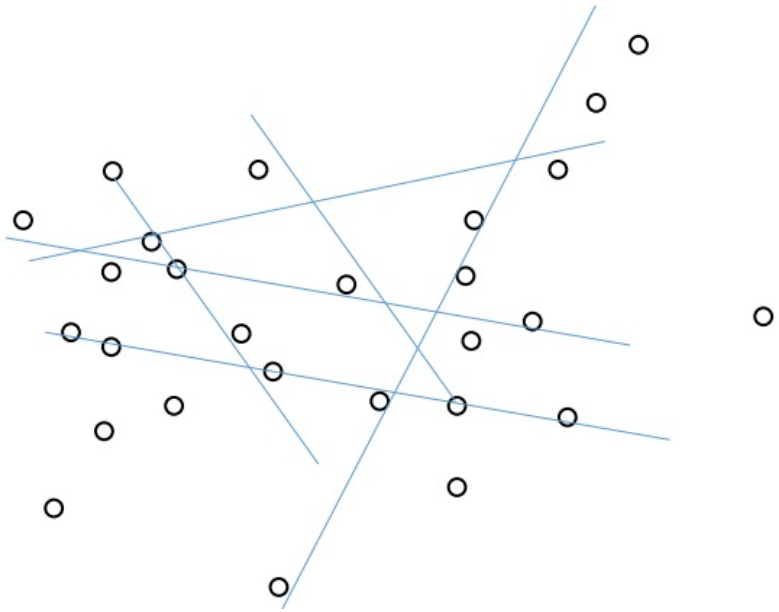
- Random Sample Consensus
- Used for Parametric Matching/Model Fitting
- Applications:





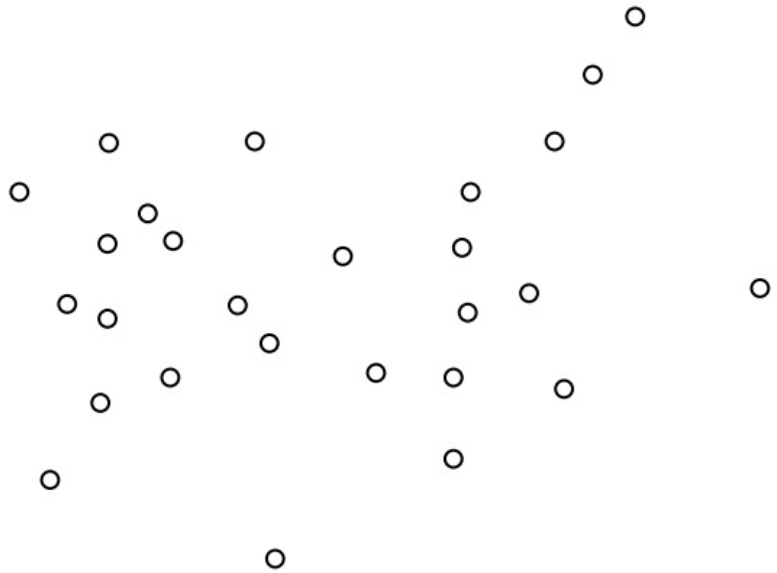
How RANSAC Works

- Fit the best possible Line to these points
- Brute Force Search ?
- Not Feasible
- Better Strategy?



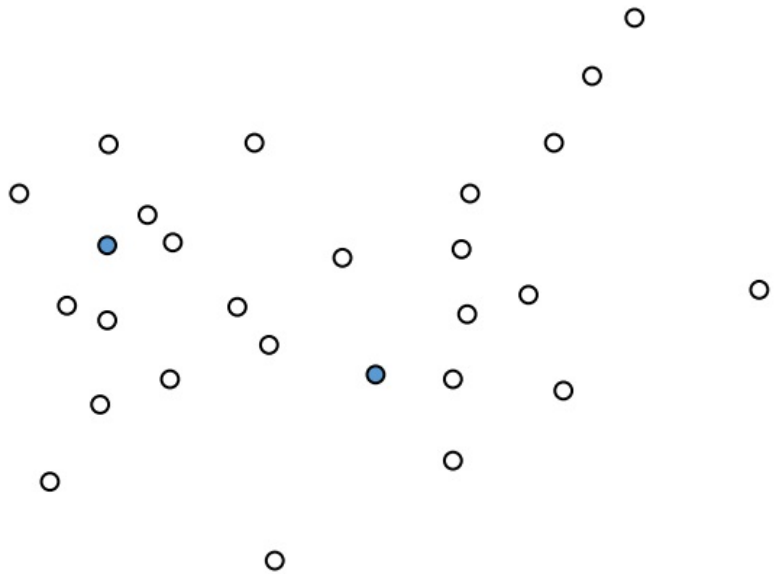
Line Fitting Using RANSAC

- Random Search – Much Faster!!!



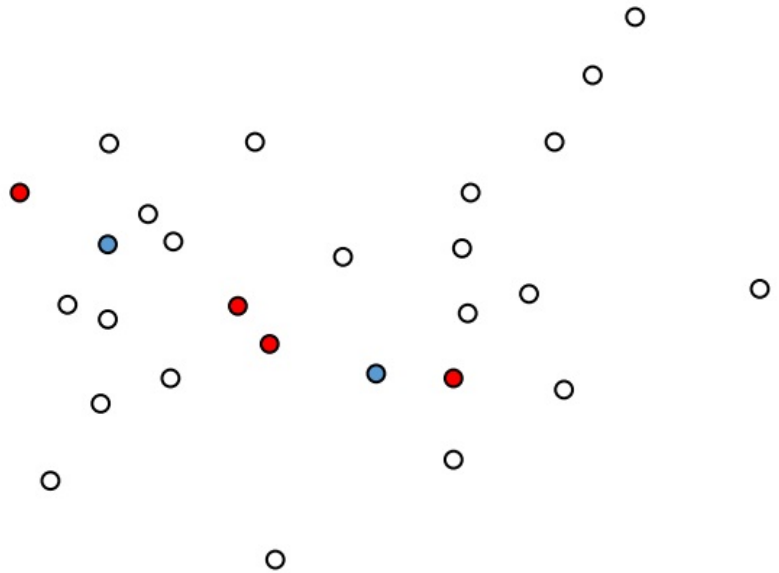
Line Fitting Using RANSAC

- Iteration 1



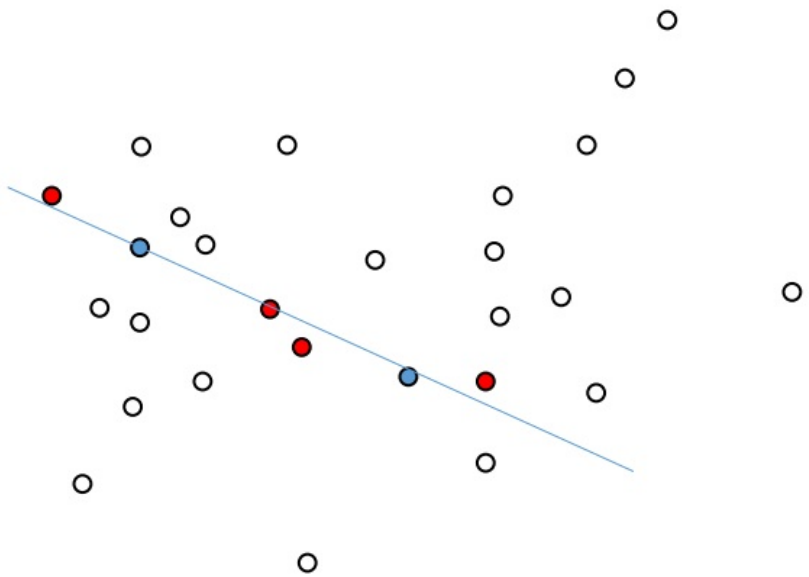
Line Fitting Using RANSAC

- Iteration 1



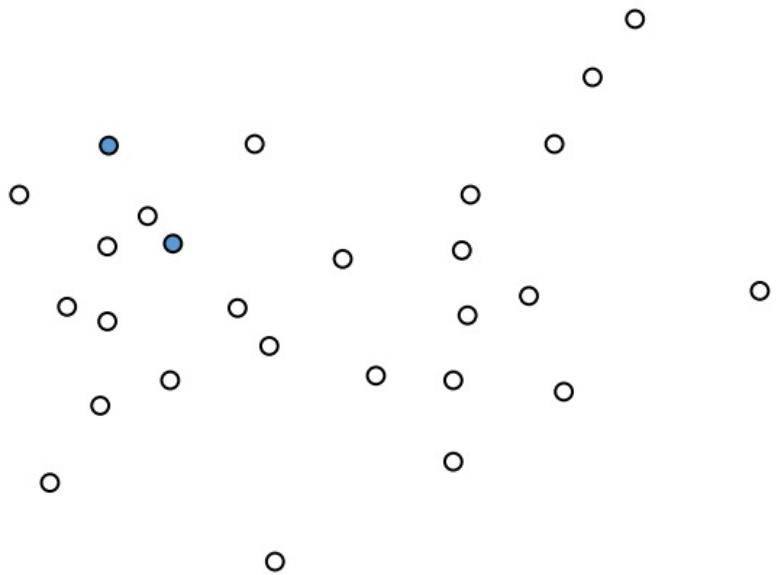
Line Fitting Using RANSAC

- Iteration 1



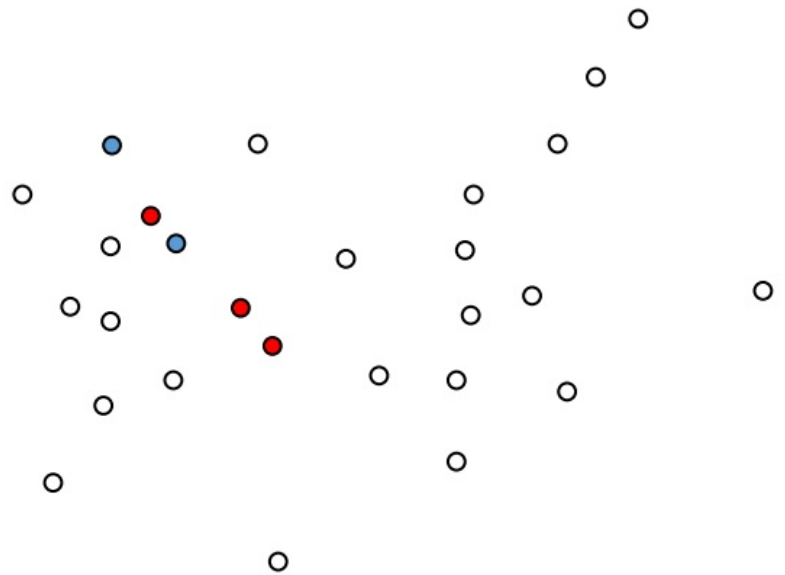
Line Fitting Using RANSAC

- Iteration 2



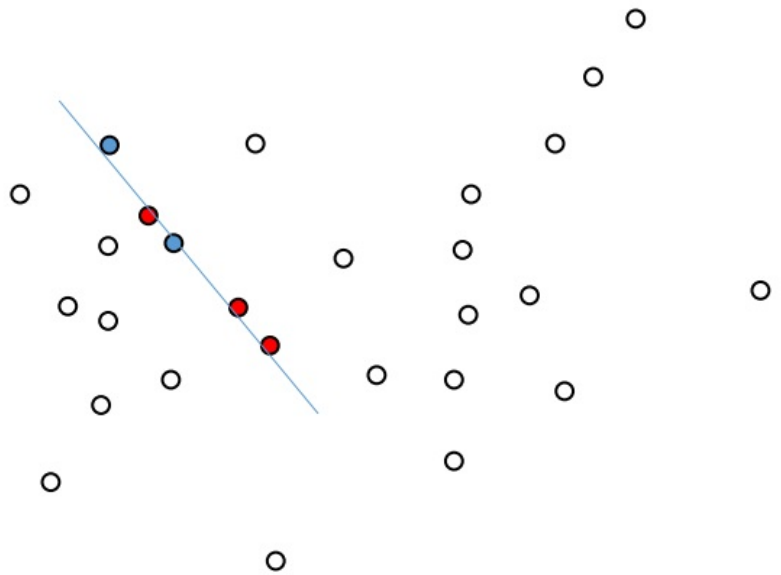
Line Fitting Using RANSAC

- Iteration 2



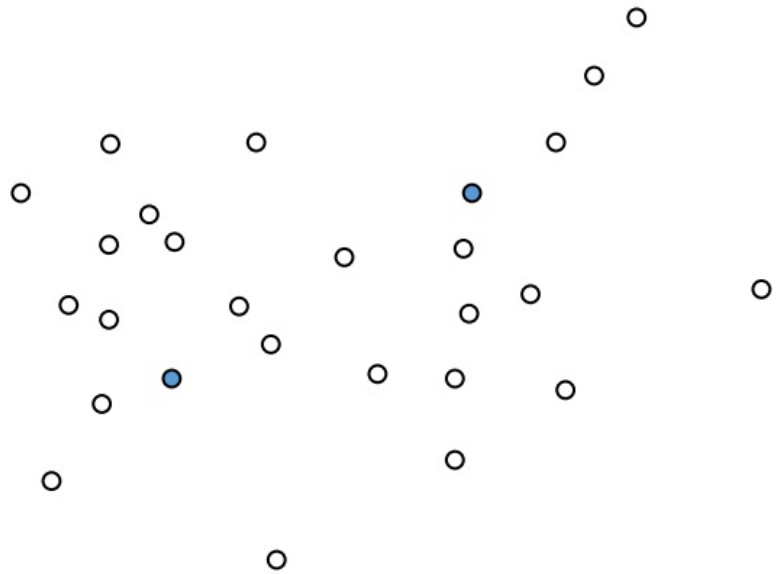
Line Fitting Using RANSAC

- Iteration 2



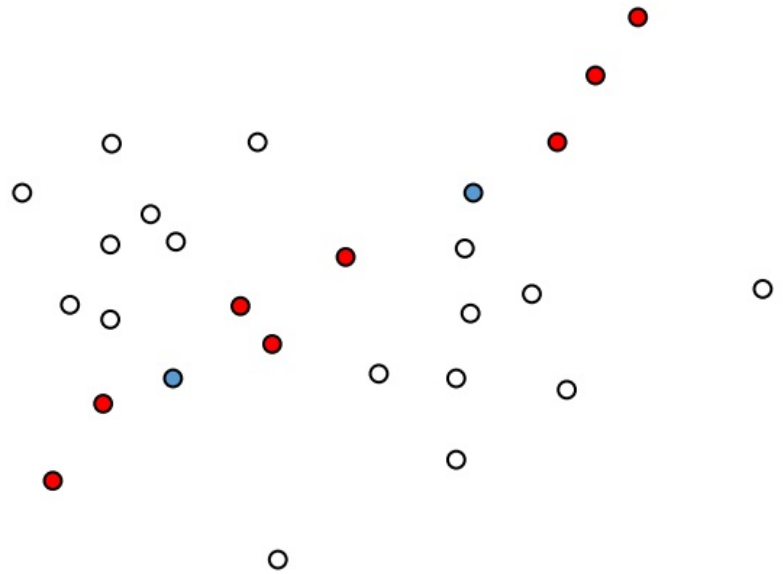
Line Fitting Using RANSAC

- ...
- Iteration 5



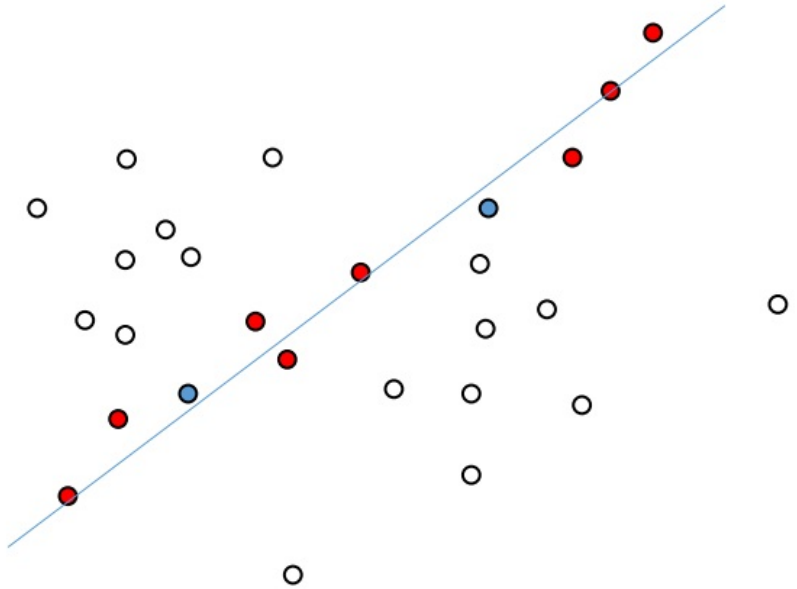
Line Fitting Using RANSAC

- Iteration 5



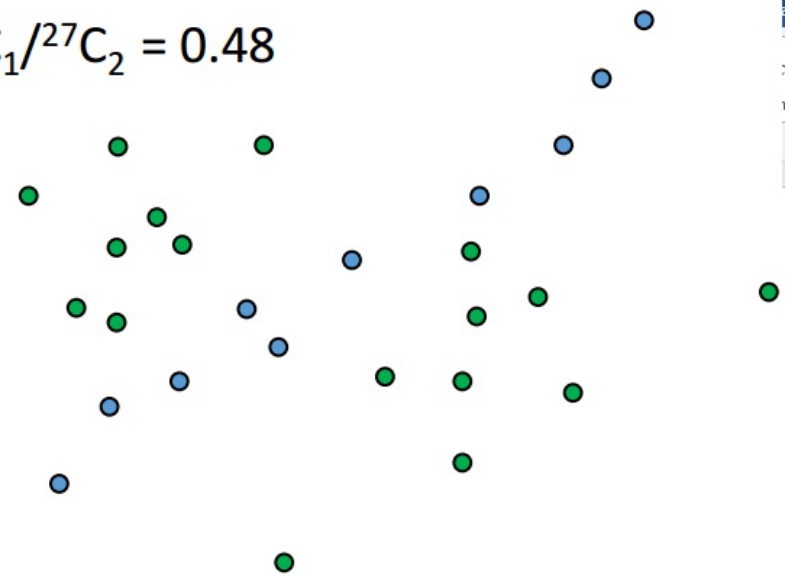
Line Fitting Using RANSAC

- Iteration 5



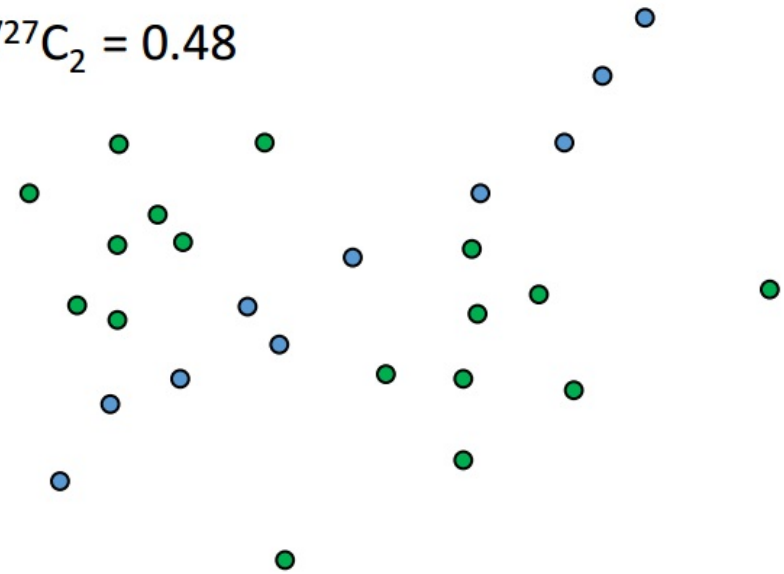
Why RANSAC Works?

- Inliers vs Outliers
- P(selecting outliers) = ${}^{17}C_2/{}^{27}C_2 + {}^{17}C_1{}^{10}C_1/{}^{27}C_2 = 0.48$



Why RANSAC Works?

- Inliers vs Outliers
- P(selecting outliers) ${}^{17}C_2/{}^{27}C_2 + {}^{17}C_1{}^{10}C_1/{}^{27}C_2 = 0.48$
- After 5 iterations...
- P(selecting outliers) = $(0.48)^5 = 0.026$



Why RANSAC Works?

- In general:

- $p = 1 - (1 - w^n)^k$

Where,
p = probability for selecting inliers
w = ratio of inliers to total #points
n = minimum #points required (for line = 2, circle =3)
k = #iterations

RANSAC Algorithm

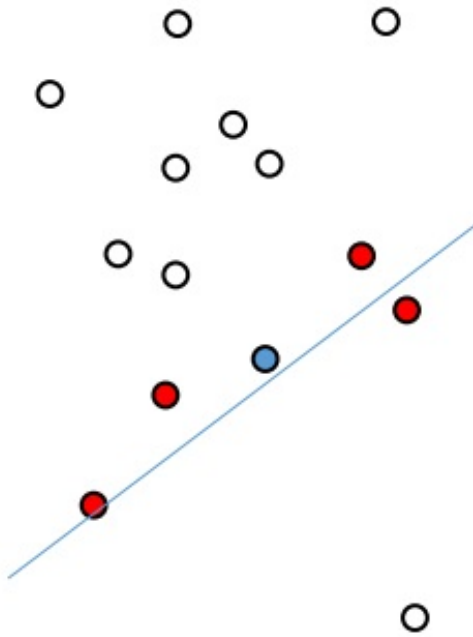
Determine:
- n—the smallest number of points required (e.g., for lines, $n = 2$, for circles, $n = 3$)
- k—the number of iterations required
- t—the threshold used to identify a point that fits well
- d—the number of nearby points required to assert a model fits well

Until k iterations have occurred
- Draw a sample of n points from the data uniformly and at random
- Fit to that set of n points
- For each data point outside the sample
  - Test the distance from the point to the structure against t; if the distance from the point to the structure is less than t, the point is close
- end
- If there are d or more points close to the structure then there is a good fit. Refit the structure using all these points. Add the result to a collection of good fits.

end

Use the best fit from this collection, using the fitting error as a criterion

Algorithm 10.4: RANSAC: Fitting Structures Using Random Sample Consensus.



Slide Credits

Radhakrishna Dasari