מבוא לראייה ממוחשבת – 22928
2017א

מנחה: אמיר אגוזי
egozi5@gmail.com

מפגש מס' 2

מה היה בפעם שעברה?

- צבע
- פילטרים – linear filters
- קונבולוציה - convolution
- איתור ספים – edge detection

מה היום?

- פירמידות – עיבוד תמונה ברזולוציות שונות.
- זיהוי נקודות בתמונה –
  - Harris corner detector
  - מציאת נקודות בסקאלות שונות.
- תיאור נקודות עניין – feature description
  - תיאור עמיד בפני שינויי סקאלה, ומאפיינים גאומטריים
  - SIFT
- מציאת התאמות בין נקודות – matching problem
- הערכת ביצועי מערכת - ROC / Precision-Recall

מה היום ? (2)

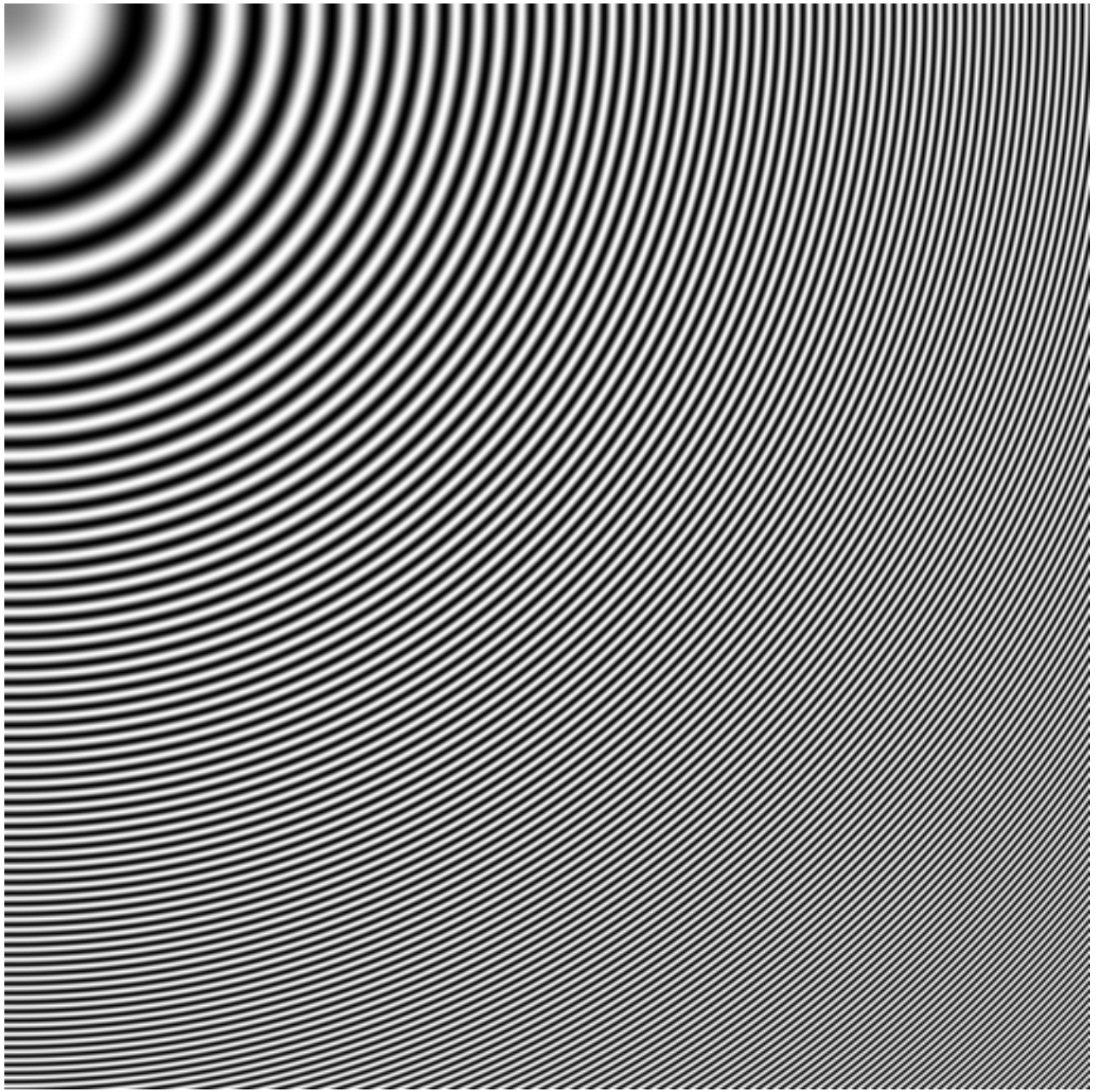
- התאמה למודל
- Line fitting
  - Least square/ Total LS
  - RANSAC
  - Hough transform
- Generalized Hough transform
- K-means

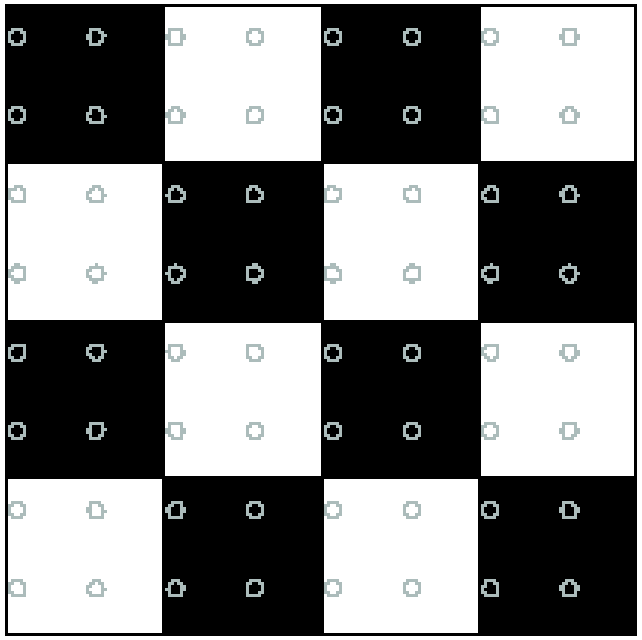
פירמידות

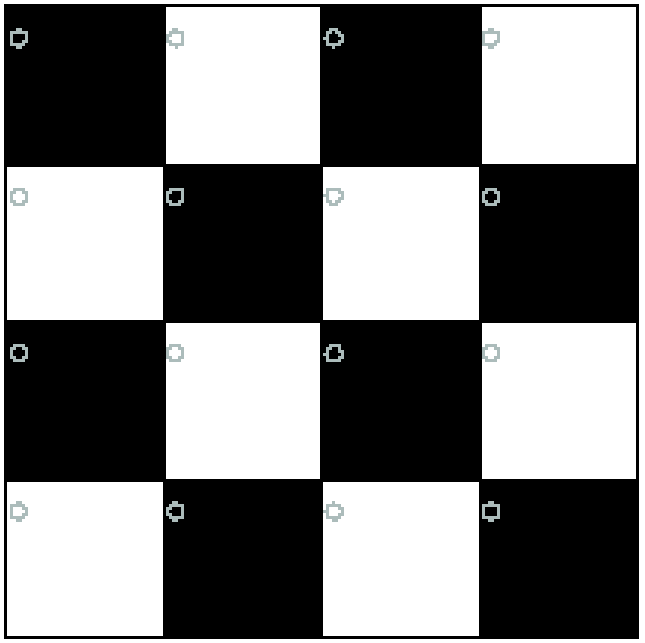
פירמידה גאוסיאנית

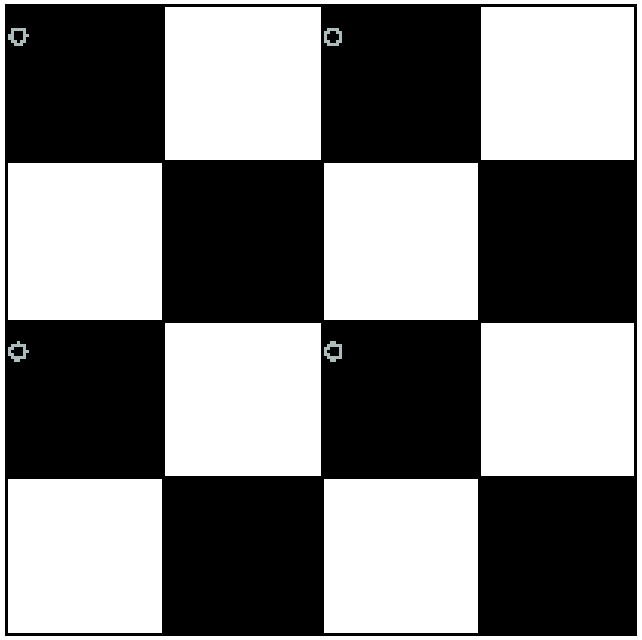
512 256 128 64 32 16 8

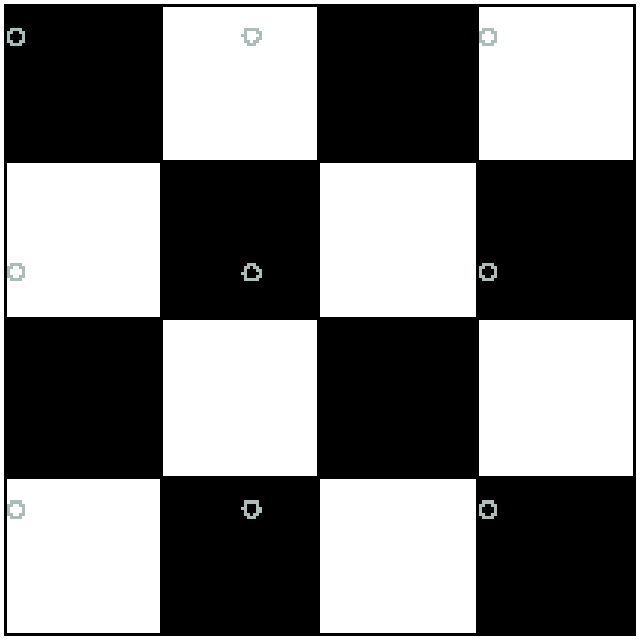
Aliasing







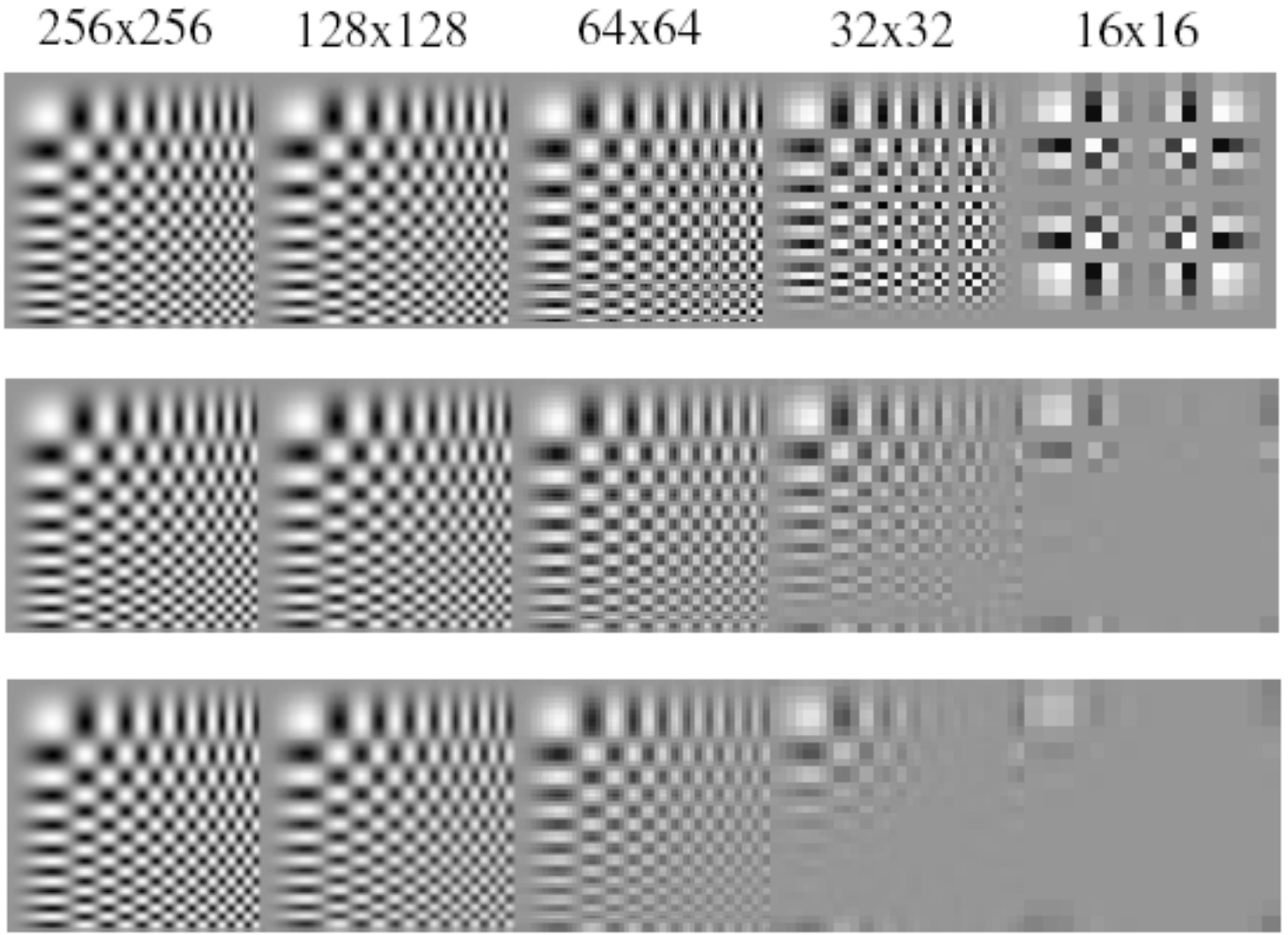




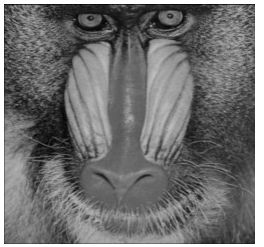
Aliasing

Resample the checkerboard by taking one sample at each circle. In the case of the top left board, new representation is reasonable.
Top right also yields a reasonable representation. Bottom left is all black (dubious) and bottom right has checks that are too big.

Sampling with smoothing = anti-aliasing



בניית פירמידה לפלסיאנית



Reduce

דגימה
Sampling
↓ 2

פילטר קירוב
Approximation
filter

רמה *i*

רמה *i-1*
(תמונת
קלט)

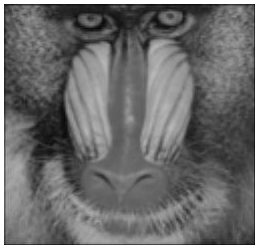
דגימה
Sampling
↑ 2

פילטר
אינטרפולציה
Interpolation
filter

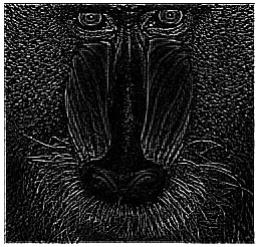
Expand

הפרש

קירוב
לרמה
i-1



הפרש
רמה *i-1*



פירמידה לפלסיאנית - הערות

- הפרדת תדרים: רמות שונות כוללות תדרים שונים
- שיחזור: מפירמידה לפלסיאנית, עם קודקוד גאוסיאני, ניתן לשחזר את התמונה המקורית!
- ב- Python/OpenCV
- בשביל מה זה טוב?
  - ניתוח רב-רזולוציות
  - דחיסת תמונות
  - ניקוי תמונות
  - שילוב תמונות (image blending)

איפה בתמונה נמצא מידע מעניין?

איתור, תיאור, והתאמת נק' עניין

Feature detection, description and matching

Why local features?

- may have a specific semantic interpretation in the context of a certain application.
- provide a limited set of well localized and individually identifiable anchor points.
- robust image representation, that allows to recognize objects or scenes without the need for segmentation.

From - Local invariant features detectors – a survey, Tuytelaars & Mikolajczyk, 2007

Uses for feature point detectors and descriptors in computer vision and graphics.

- Image alignment and building panoramas
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation
- ... other

Good features should have the following properties:

- Repeatability = invariance + robustness
- Distinctiveness/informativeness
- Locality
- Quantity
- Accuracy
- Efficiency

Type of image features

- Edges
- Corners
- Blobs / Regions

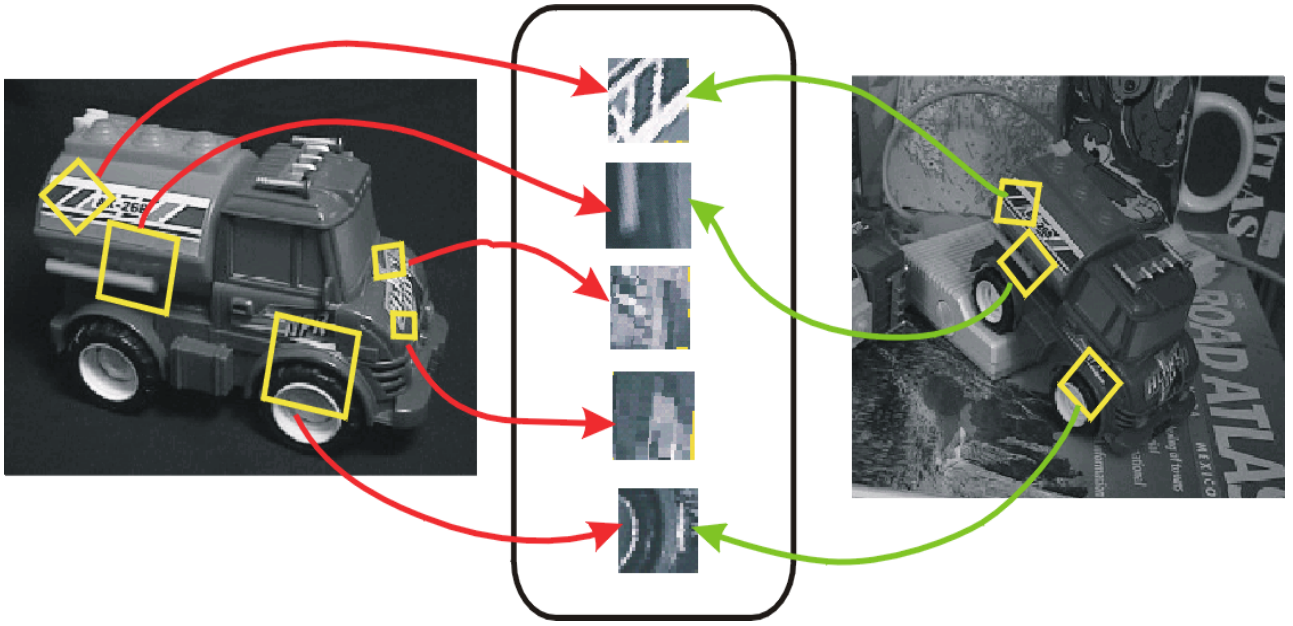
Many Existing Detectors Available

Hessian & Harris	[Beaudet ‘78], [Harris ‘88]
Laplacian, DoG	[Lindeberg ‘98], [Lowe 1999]
Harris-/Hessian-Laplace	[Mikolajczyk & Schmid ‘01]
Harris-/Hessian-Affine	[Mikolajczyk & Schmid ‘04]
EBR and IBR	[Tuytelaars & Van Gool ‘04]
MSER	[Matas ‘02]
Salient Regions	[Kadir & Brady ‘01]

Others…

Main questions

- How to define salient features?
  - Given a definition, how to (robustly) detect these feature?
- How to *describe* a local feature?
- How to find the corresponding points?



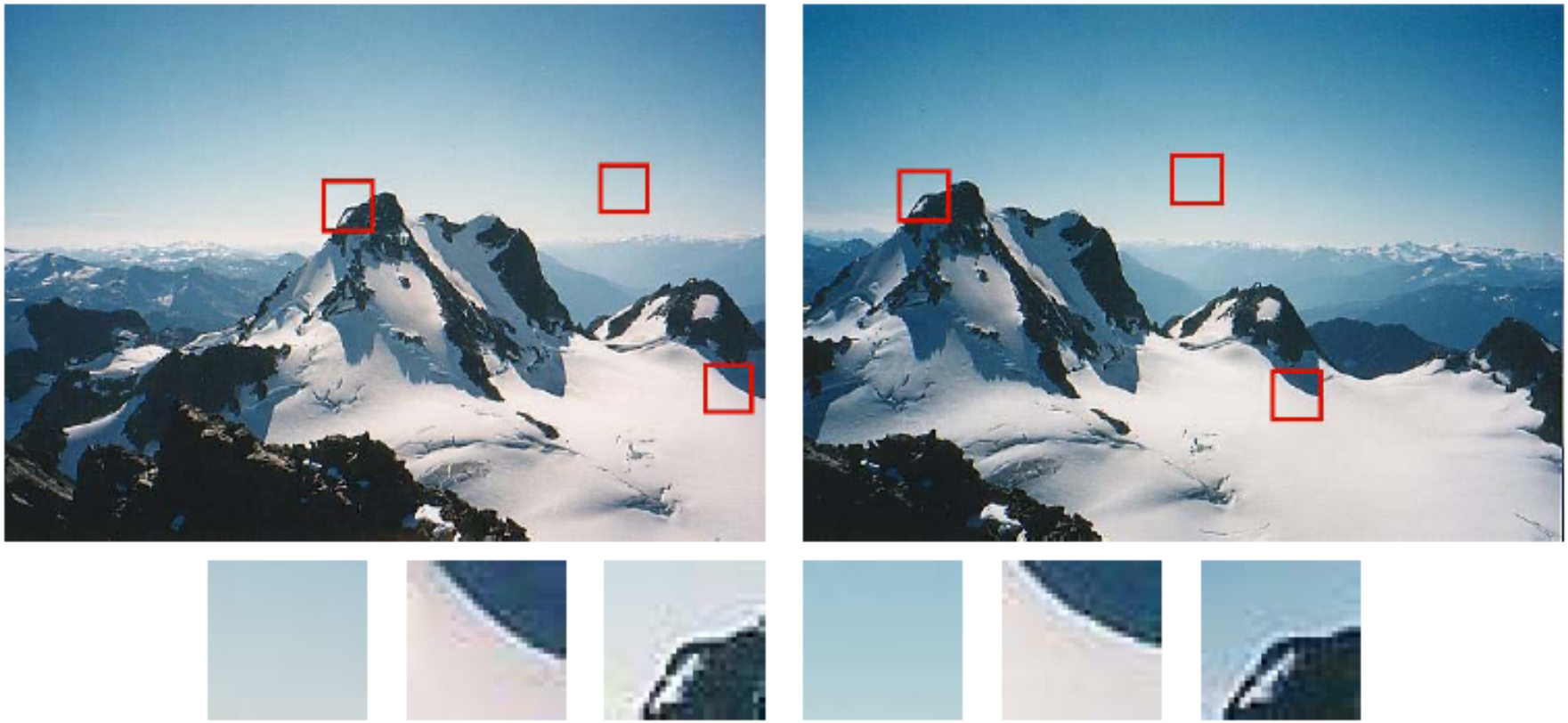


Figure 4.3: *Image pairs with extracted patches below. Notice how some patches can be localized or matched with higher accuracy than others.*

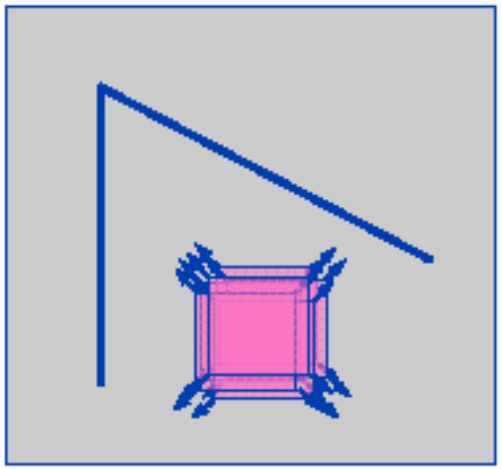
Finding Corners

- Corners are repeatable and **distinctive**
- **Key property:** in the region around a corner, image gradient has two or more dominant directions

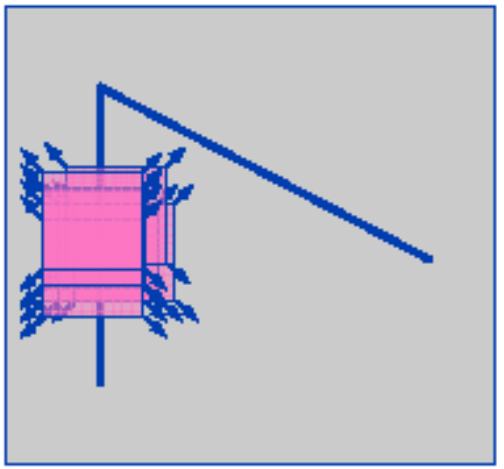
C.Harris and M.Stephens. *Proceedings of the 4th Alvey Vision Conference*: 1988, pages 147--151.

Source: Lana Lazebnik

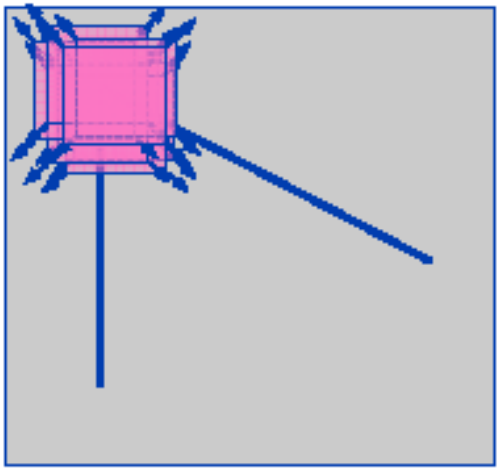
Harris corner detector: basic idea



“flat” region:
No change in
All direction



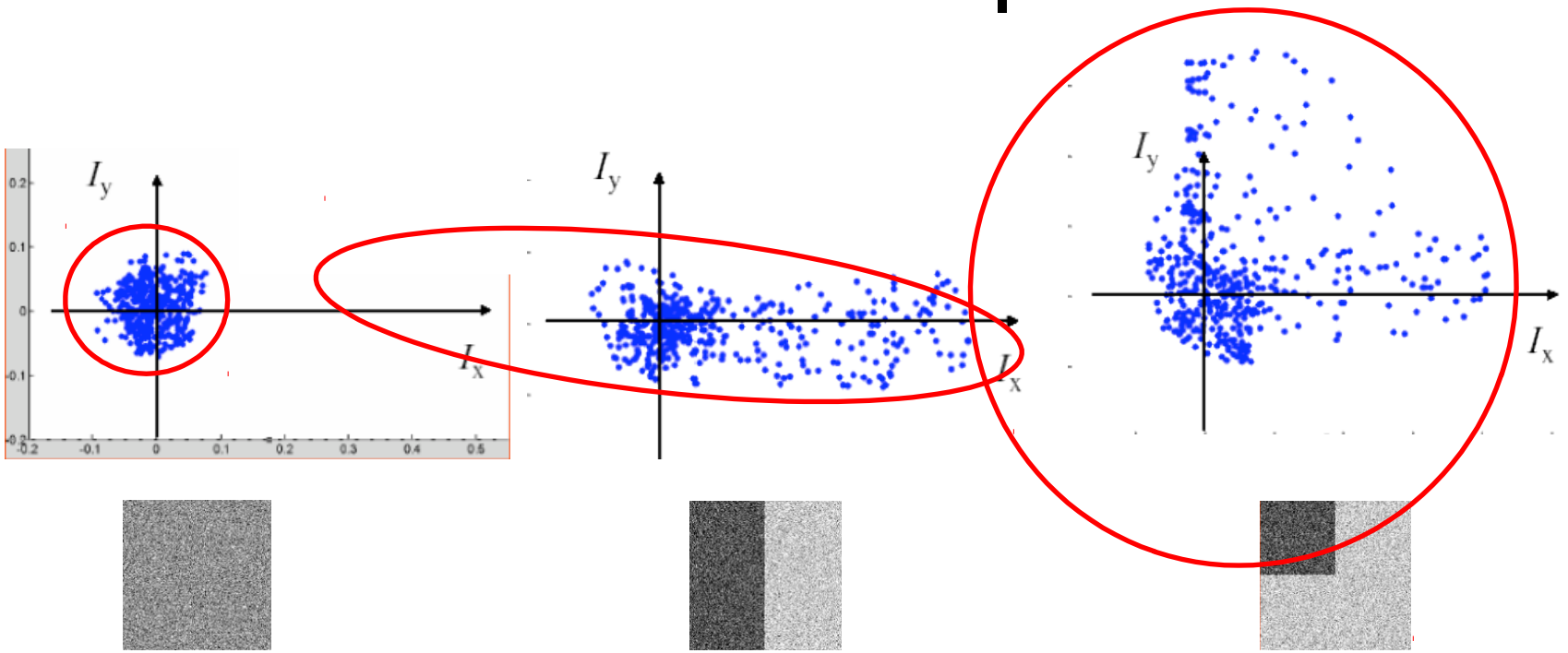
“edge”
No change along
the edge direction



“corner”:
Significant change
In all directions.

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

אפיון פיזור הגרדיאנטים



- נתאים לנקודות אליפסה (התפלגות גאוסיאנית) שמרכזה בראשית
- פינה היא: "אליפסה הדומה לעיגול גדול" במרחב הגרדיאנטים
- פינה היא: התפלגות גאוסיאנית עם שונות גבוהה בשני צירים במרחב הגרדיאנטים

Based on slides by Robert Collins, Pen State, 04

הסתברות ואלגברה לינארית - חזרה

- Data points/features $A = \begin{bmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \end{bmatrix}$
- Covariance matrix

$$\Sigma = E\left[(A - E[A])(A - E[A])^T\right] = E[AA^T] - \mu\mu^T$$

- Estimation

$$\hat{\Sigma} = \frac{1}{n-1}(A - \hat{\mu})(A - \hat{\mu})^T$$

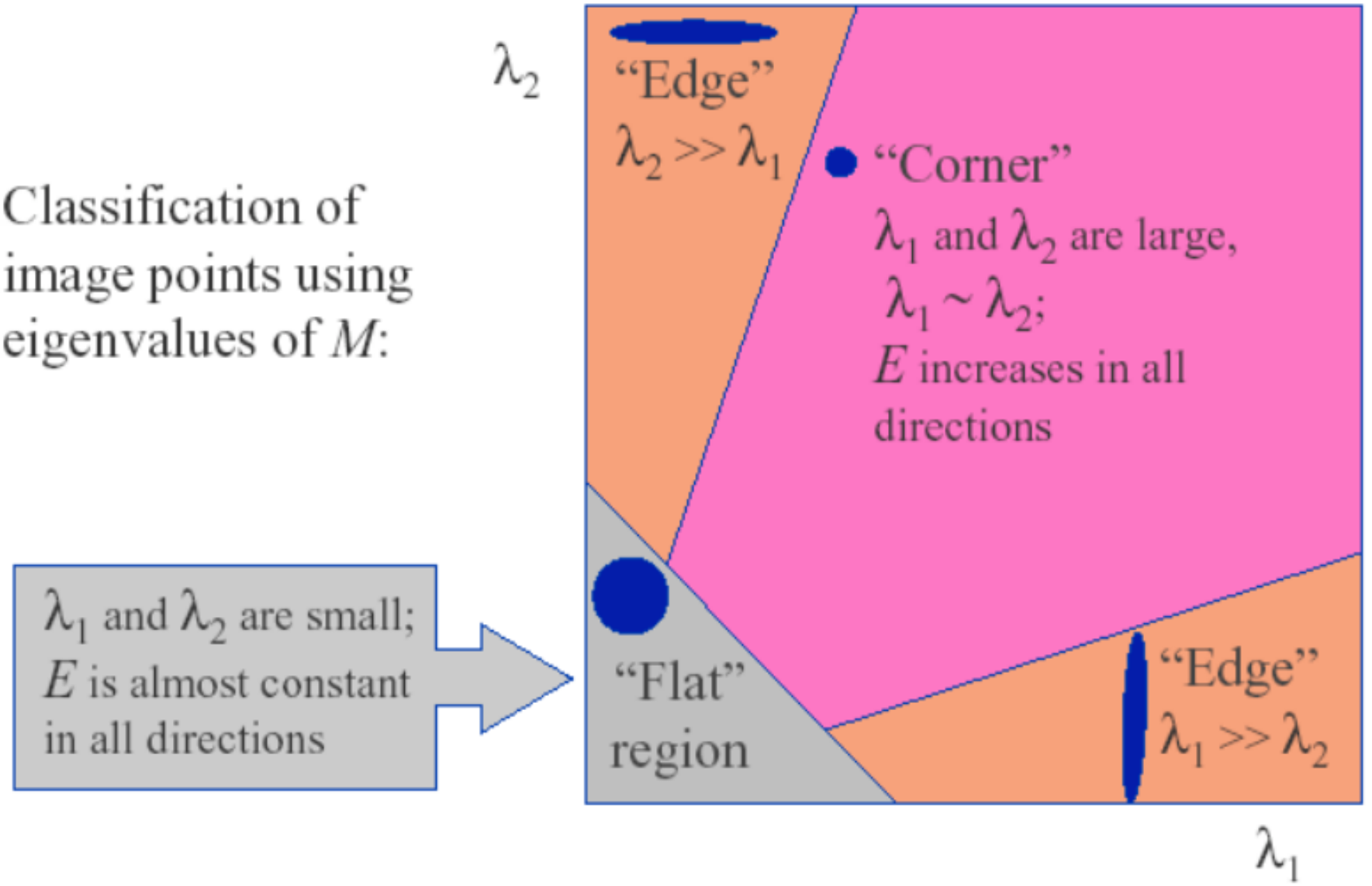
- Eigen values/vectors:

$$\Sigma\mathbf{v} = \lambda\mathbf{v}$$

Robert Collins
CSE486, Penn State

Classification via Eigenvalues

Classification of image points using eigenvalues of M:



Robert Collins
CSE486, Penn State

Corner Response Measure

Measure of corner response:

$$R = \det M - k\left(\text{trace}\, M\right)^2$$

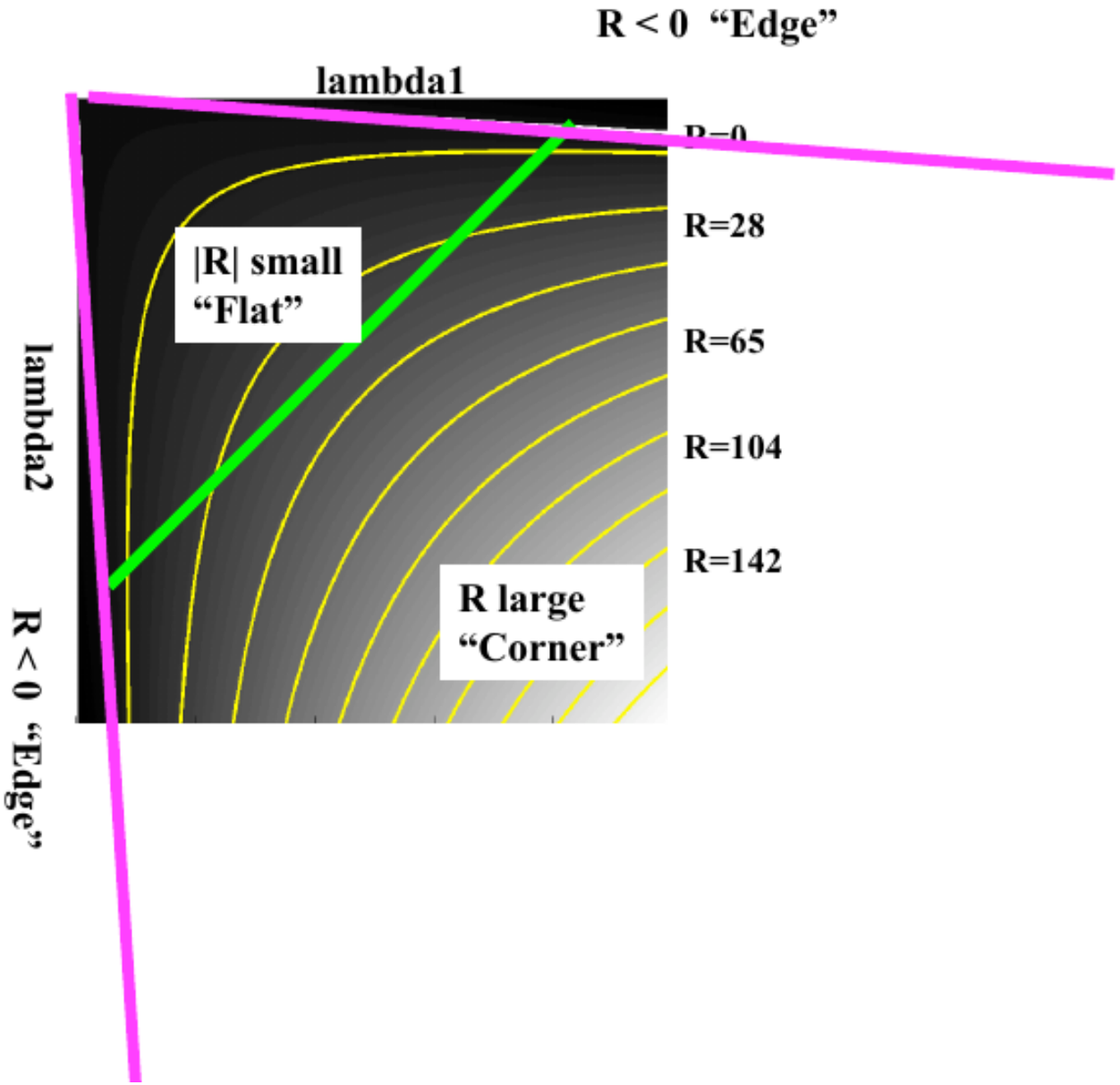
$$\det M = \lambda_1 \lambda_2$$
$$\text{trace}\, M = \lambda_1 + \lambda_2$$

(k is an empirically determined constant; $k = 0.04$ - 0.06)

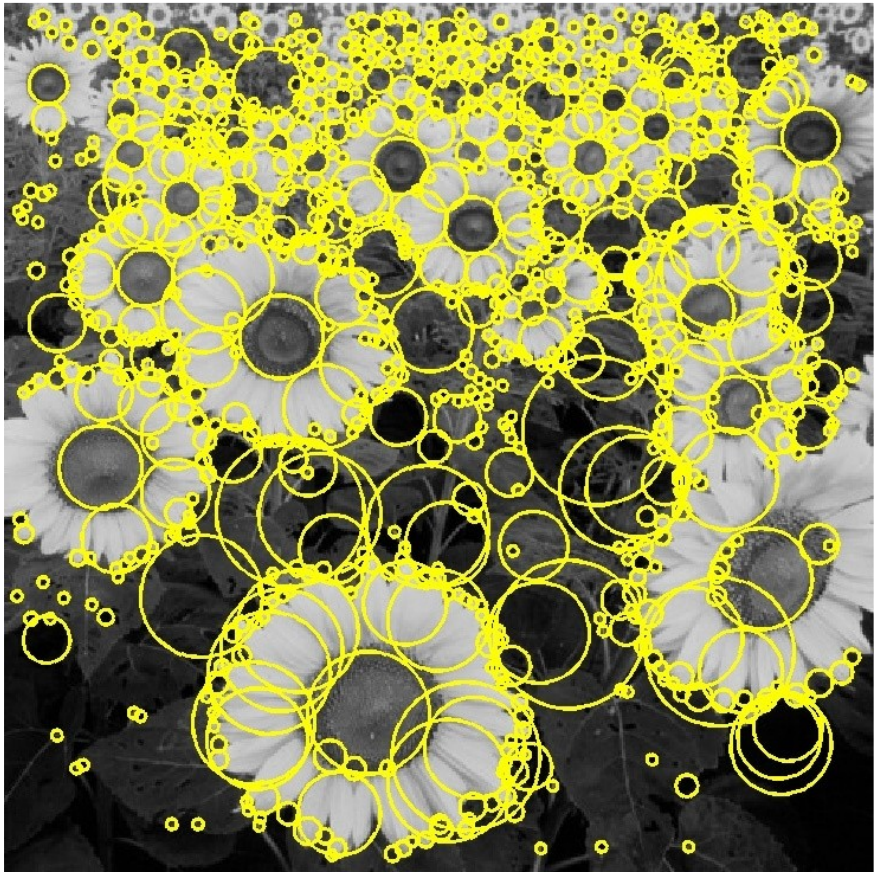
Robert Collins
CSE486, Penn State

Corner Response Map

- 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



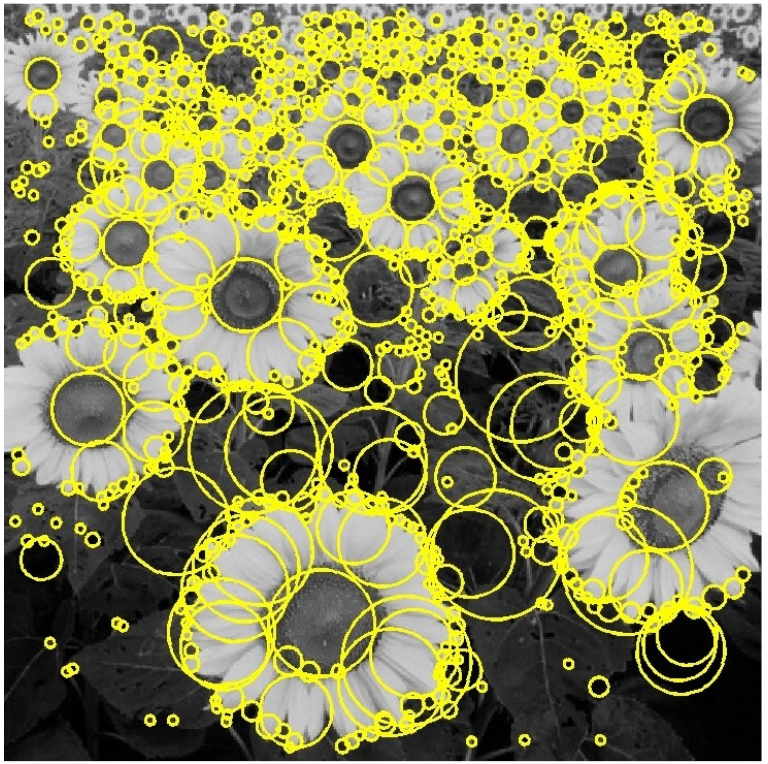
Blob detection

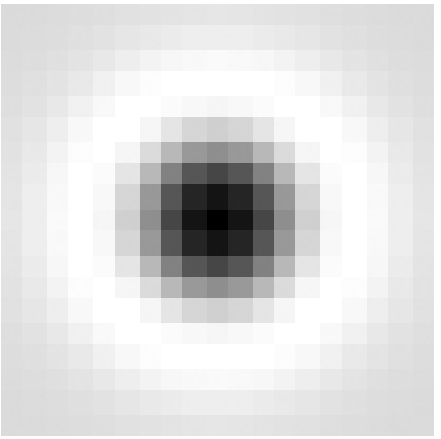


Blob detection: basic idea

To detect blobs, convolve the image with a “blob filter” at multiple scales and look for maxima of filter response in the resulting

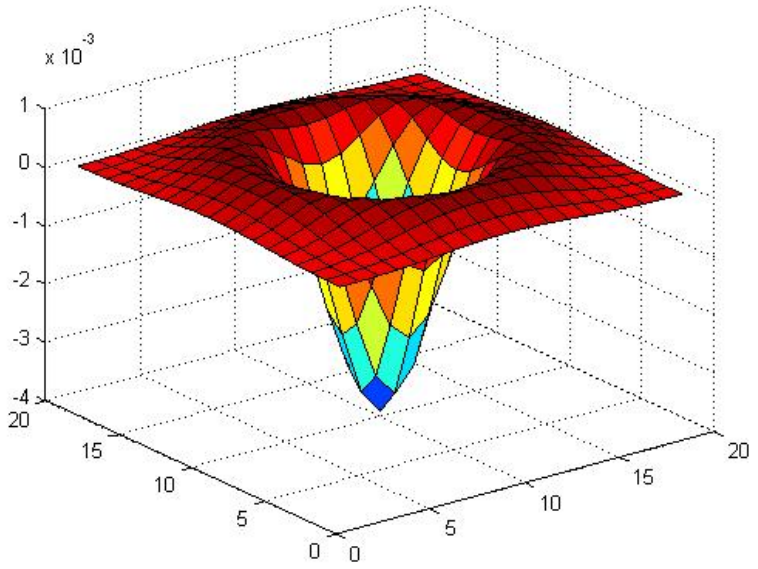
- *scale space*

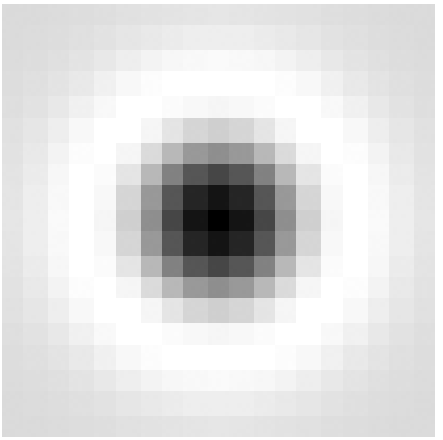




Blob filter

Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D





$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

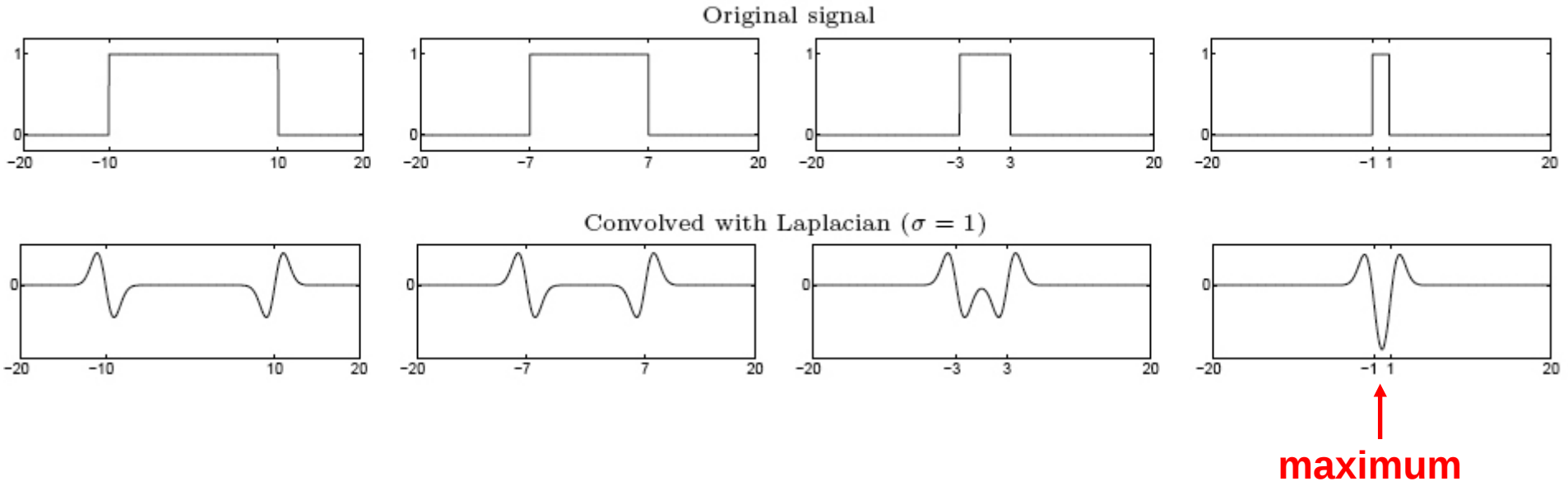
Scale invariance - motivation

- Description regions have to be adapted to scale changes

- Interest points have to be repeatable for scale changes

From edges to blobs

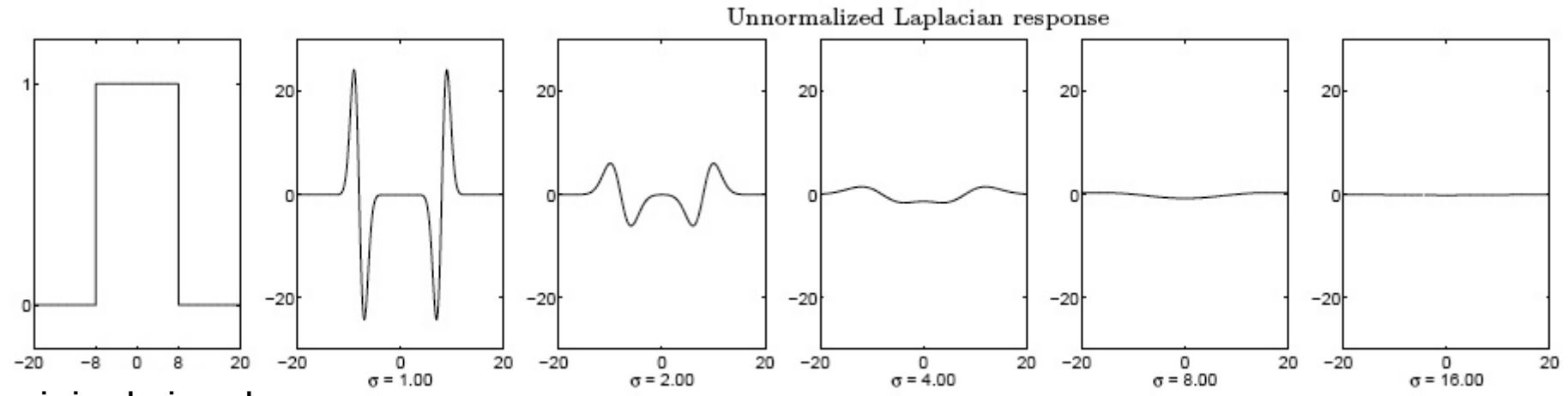
- Edge = ripple
- Blob = superposition of two ripples



Spatial selection: the magnitude of the Laplacian response will achieve a maximum at the center of the blob, provided the scale of the Laplacian is "matched" to the scale of the blob

Scale selection

- We want to find the characteristic scale of the blob by convolving it with Laplacians at several scales and looking for the maximum response
- However, Laplacian response decays as scale increases:



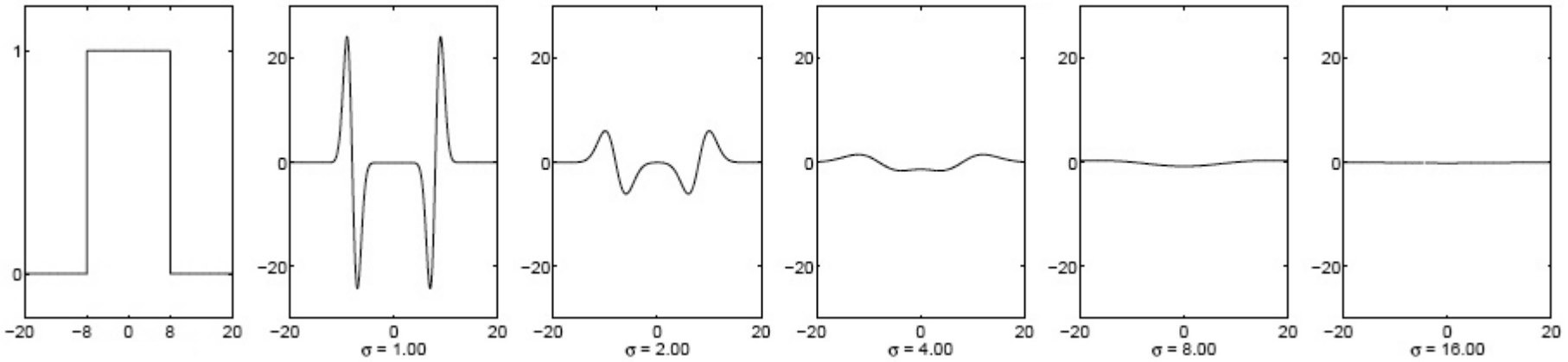
Scale normalization

- The response of a derivative of Gaussian filter to a perfect step edge decreases as σ increases
- To keep response the same (scale-invariant), must multiply Gaussian derivative by σ .
- Laplacian is the second Gaussian derivative, so it must be multiplied by σ^2.

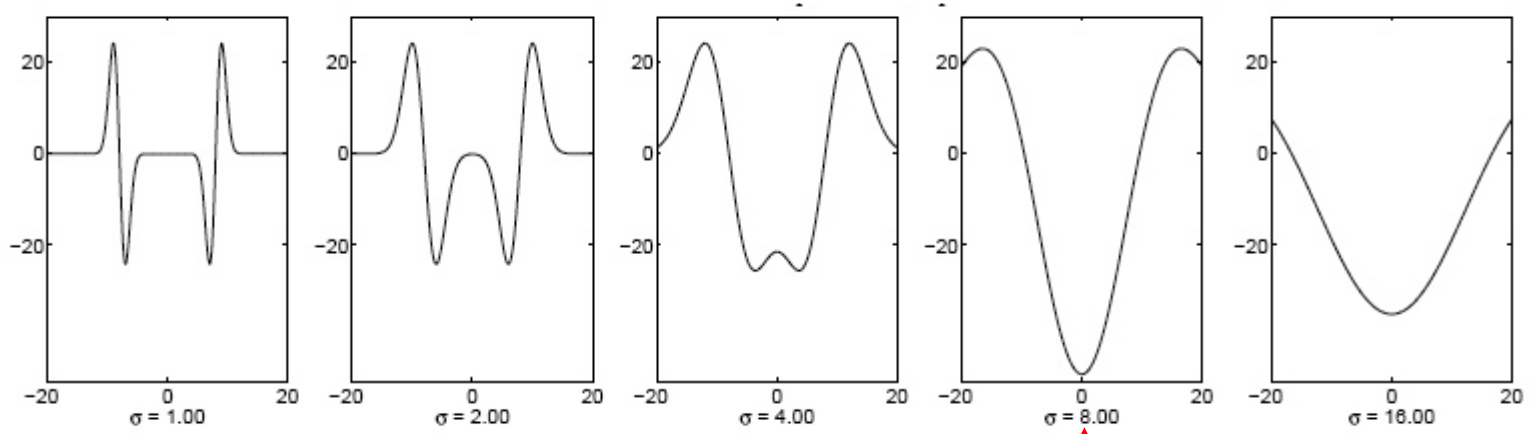
Effect of scale normalization

Original signal

Unnormalized Laplacian response

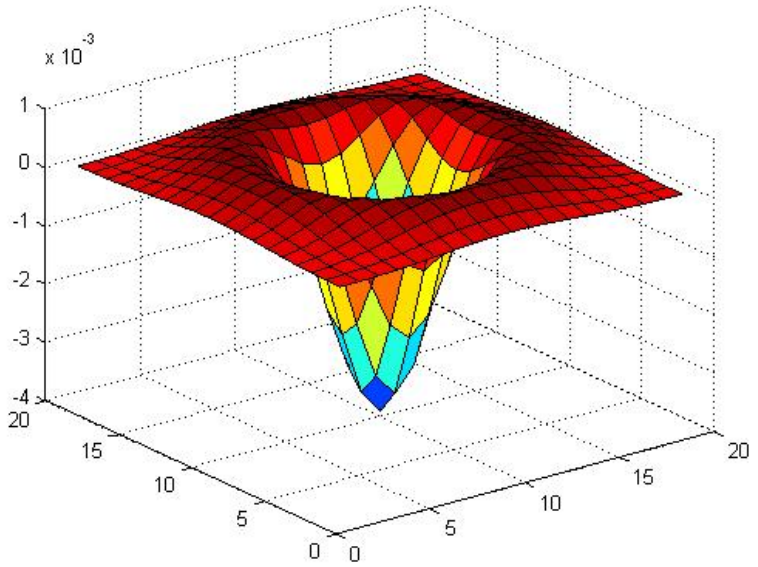


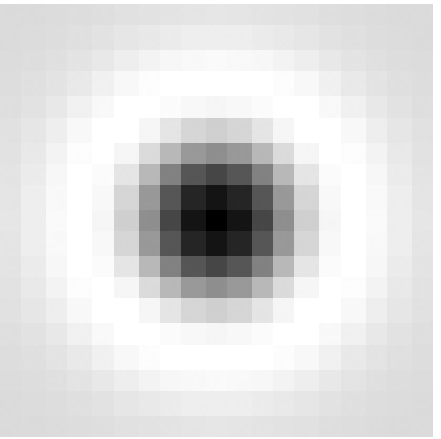
Scale-normalized Laplacian response



Blob detection in 2D

Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

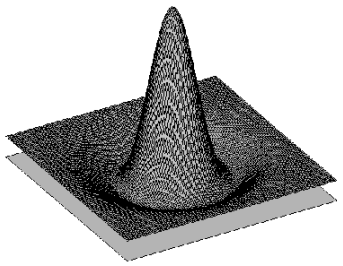


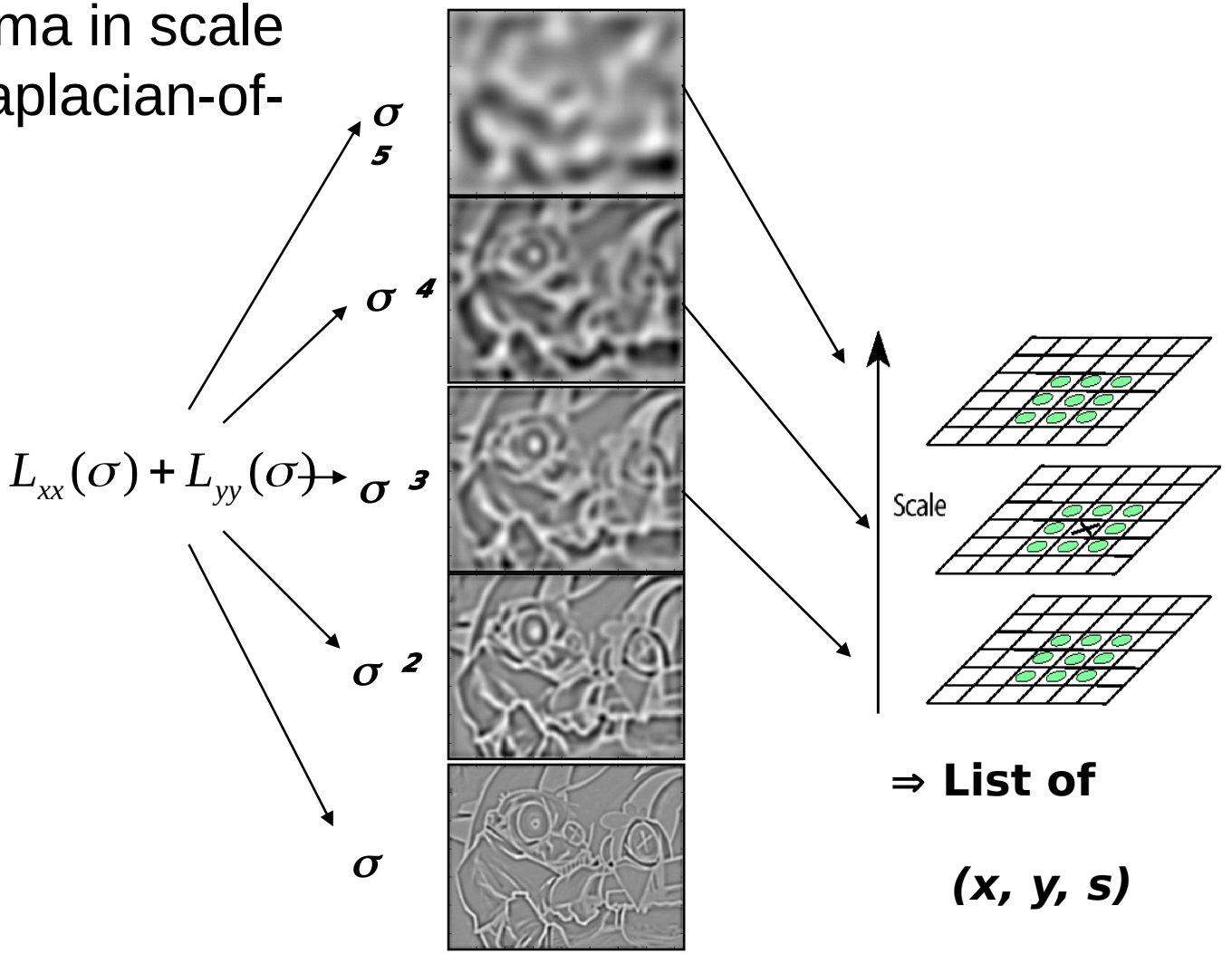


Scale-normalized: $$\nabla^2_{norm} g = \sigma^2 \left(\frac{\partial^2 g}{\partial^2 x} + \frac{\partial^2 g}{\partial^2 y} \right)$$

Laplacian-of-Gaussian (LoG)

- Local maxima in scale space of Laplacian-of-Gaussian





Efficient implementation

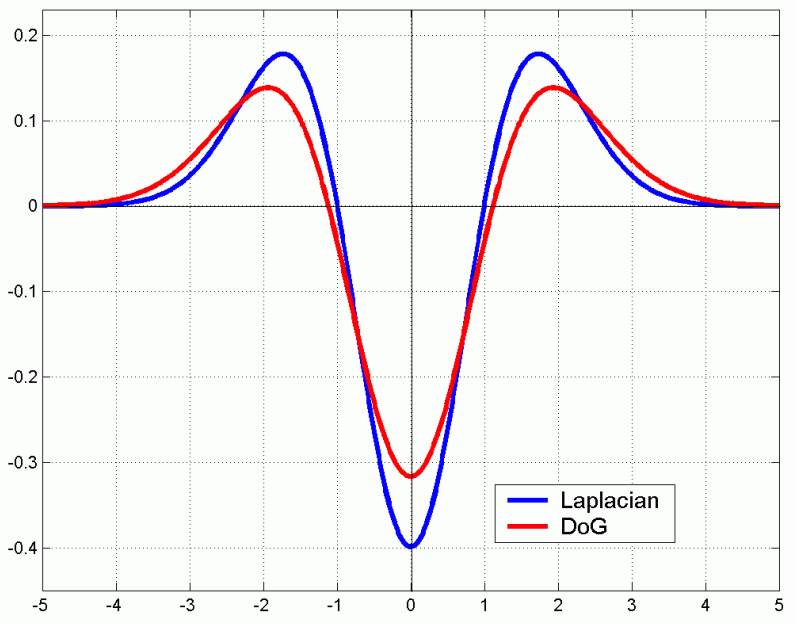
- Approximating the Laplacian with a difference of Gaussians:

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

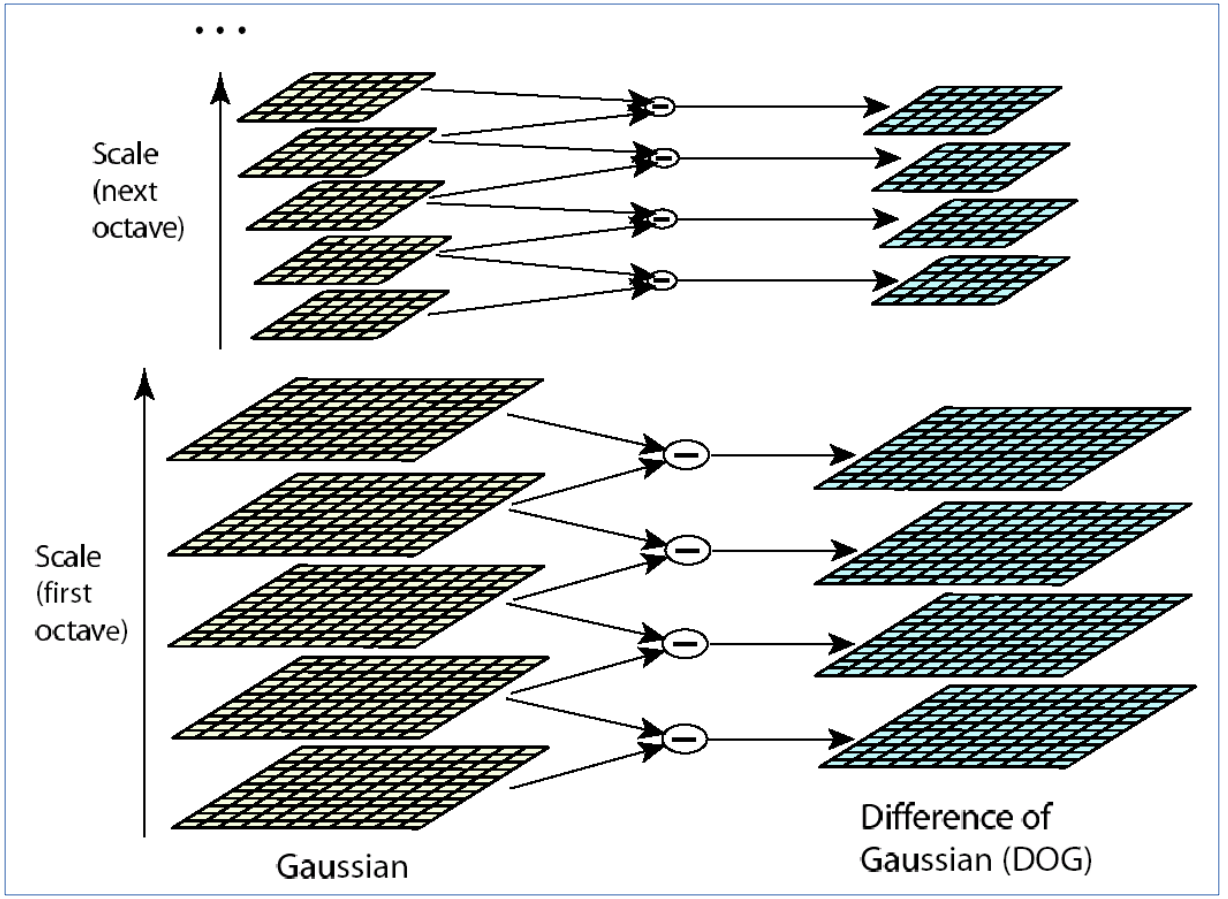
(Laplacian)

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

(Difference of Gaussians)



Efficient implementation



David G. Lowe. *IJCV* 60 (2), pp. 91-110, 2004.

Comparison of Keypoint Detectors

Table 7.1 Overview of feature detectors.

Feature Detector	Corner	Blob	Region	Rotation invariant	Scale invariant	Affine invariant	Repeatability	Localization accuracy	Robustness	Efficiency
Harris	√			√			+++	+++	+++	++
Hessian		√		√			++	++	++	+
SUSAN	√			√			++	++	++	+++
Harris-Laplace	√	(√)		√	√		+++	+++	++	+
Hessian-Laplace	(√)	√		√	√		+++	+++	+++	+
DoG	(√)	√		√	√		++	++	++	++
SURF	(√)	√		√	√		++	++	++	+++
Harris-Affine	√	(√)		√	√	√	+++	+++	++	++
Hessian-Affine	(√)	√		√	√	√	+++	+++	+++	++
Salient Regions	(√)	√		√	√	(√)	+	+	++	+
Edge-based	√			√	√	√	+++	+++	+	+
MSER			√	√	√	√	+++	+++	++	+++
Intensity-based			√	√	√	√	++	++	++	++
Superpixels			√	√	(√)	(√)	+	+	+	+

Tuytelaars Mikolajczyk 2008

Invariance

Suppose we are comparing two images I_1 and I_2

- I_2 may be a transformed version of I_1
- What kinds of transformations are we likely to encounter in practice?

Invariance

Suppose we are comparing two images I_1 and I_2

- I_2 may be a transformed version of I_1
- What kinds of transformations are we likely to encounter in practice?

We'd like to find the same features regardless of the transformation

- This is called transformational ***invariance***
- Most feature methods are designed to be invariant to
  - Translation, 2D rotation, scale
- They can usually also handle
  - Limited 3D rotations (SIFT works up to about 60 degrees)
  - Limited affine transformations (some are fully affine invariant)
  - Limited illumination/contrast changes

How to achieve invariance

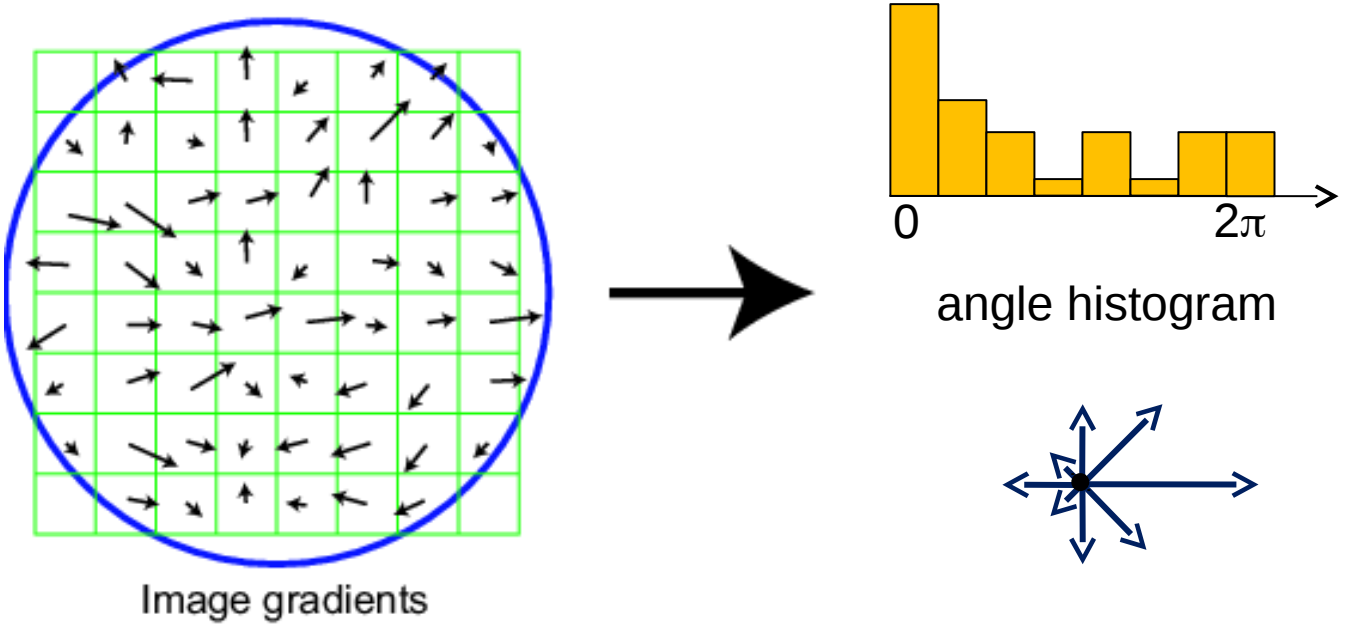
Need both of the following:

1. Make sure your 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 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…

Scale Invariant Feature Transform

Basic idea:

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90˚) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations



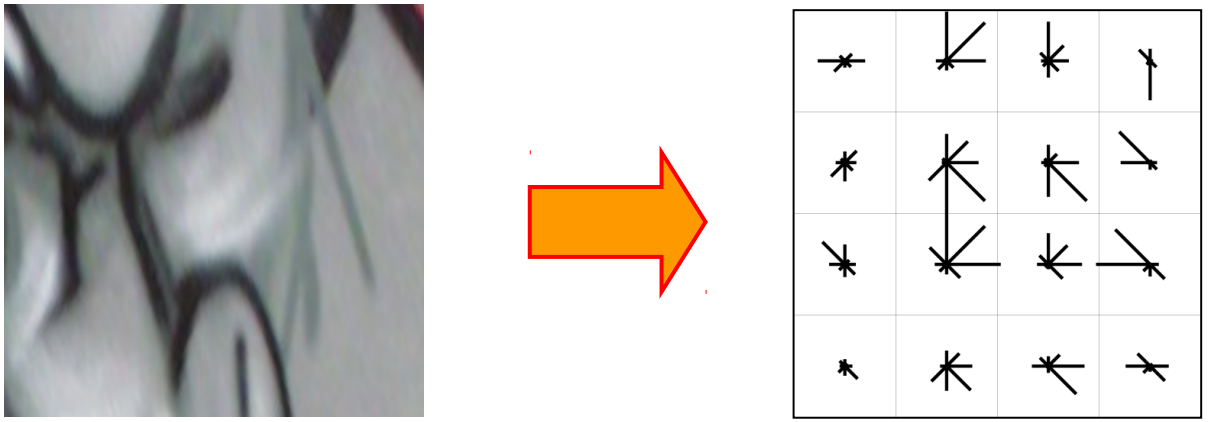
Adapted from slide by David Lowe

Feature descriptors: SIFT

Scale **I**nvariant **F**eature **T**ransform

Descriptor computation:

- Divide patch into 4x4 sub-patches: 16 cells
- Compute histogram of gradient orientations (8 reference angles) for all pixels inside each sub-patch
- Resulting descriptor: 4x4x8 = 128 dimensions



David G. Lowe. *IJCV* 60 (2), pp. 91-110, 2004.

Source: Lana Lazebnik

Feature detection/descriptors links

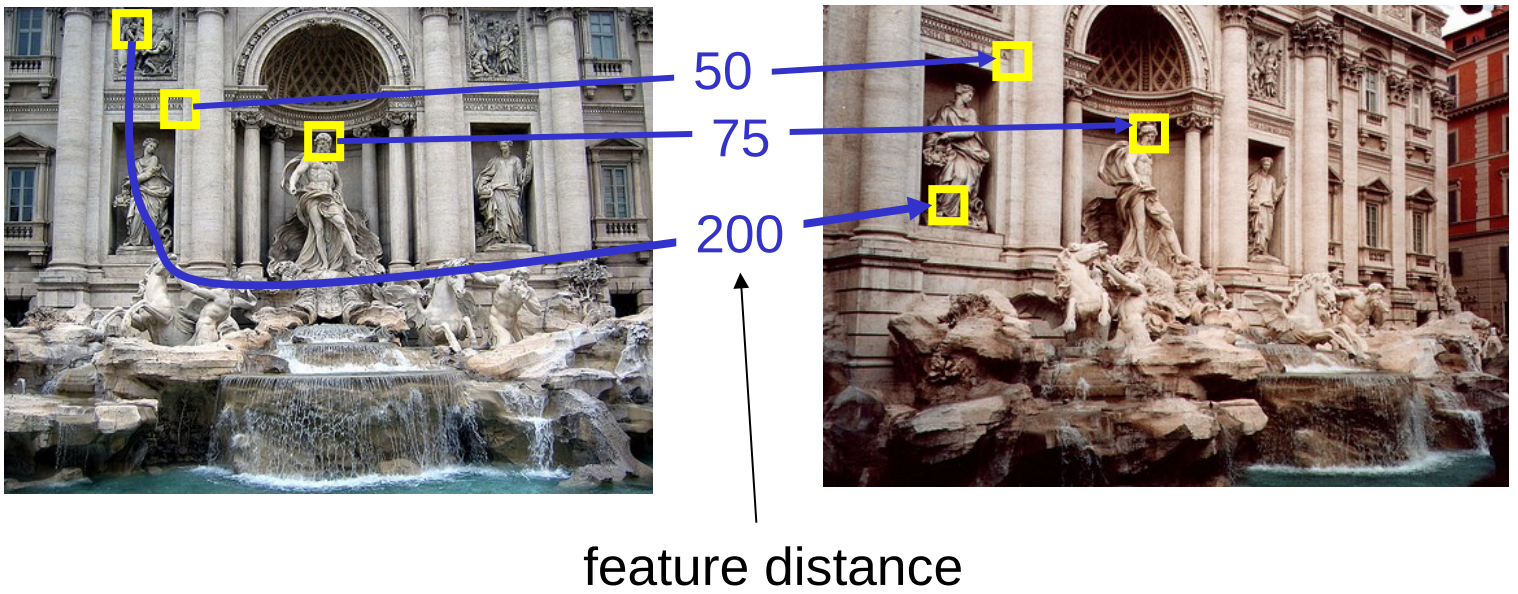
- http://www.robots.ox.ac.uk/~vgg/research/affine/index.html
  - Carefully designed test setup
    - Dataset with transformations
    - Evaluation code in matlab
    - Benchmark for new detectors and descriptors
- http://www.cs.ubc.ca/~lowe/keypoints/
- http://www.vision.ee.ethz.ch/~surf
- http://lear.inrialpes.fr/software

Evaluate the results

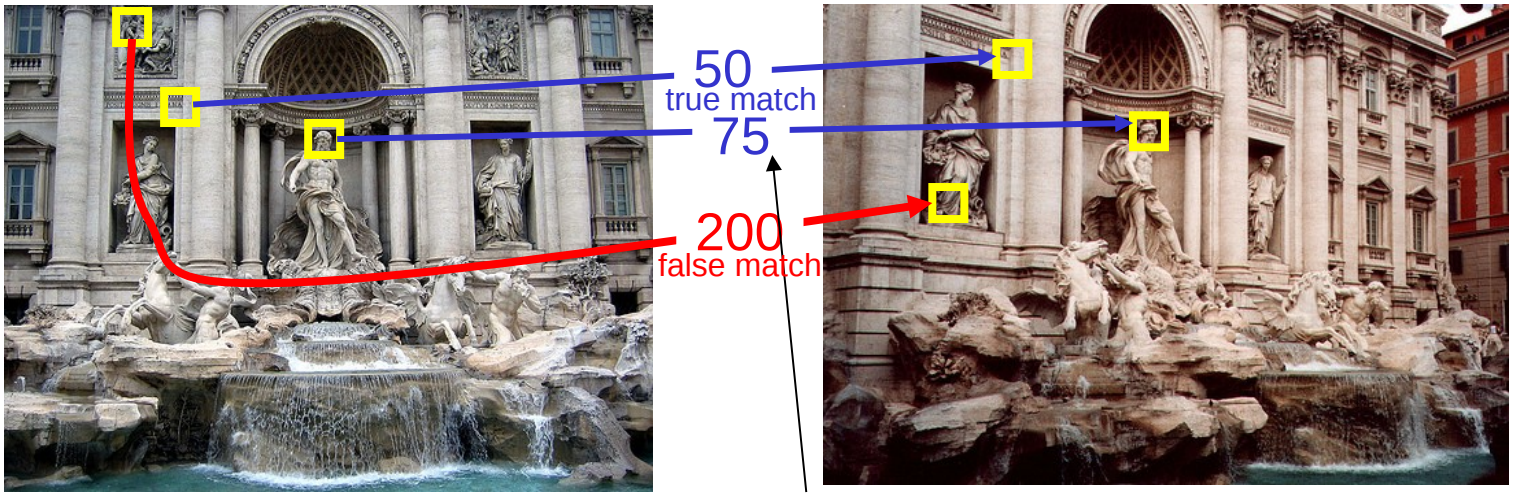
- ROC curve
- Precision/Recall

Evaluating the results

How can we measure the performance of a feature matcher?



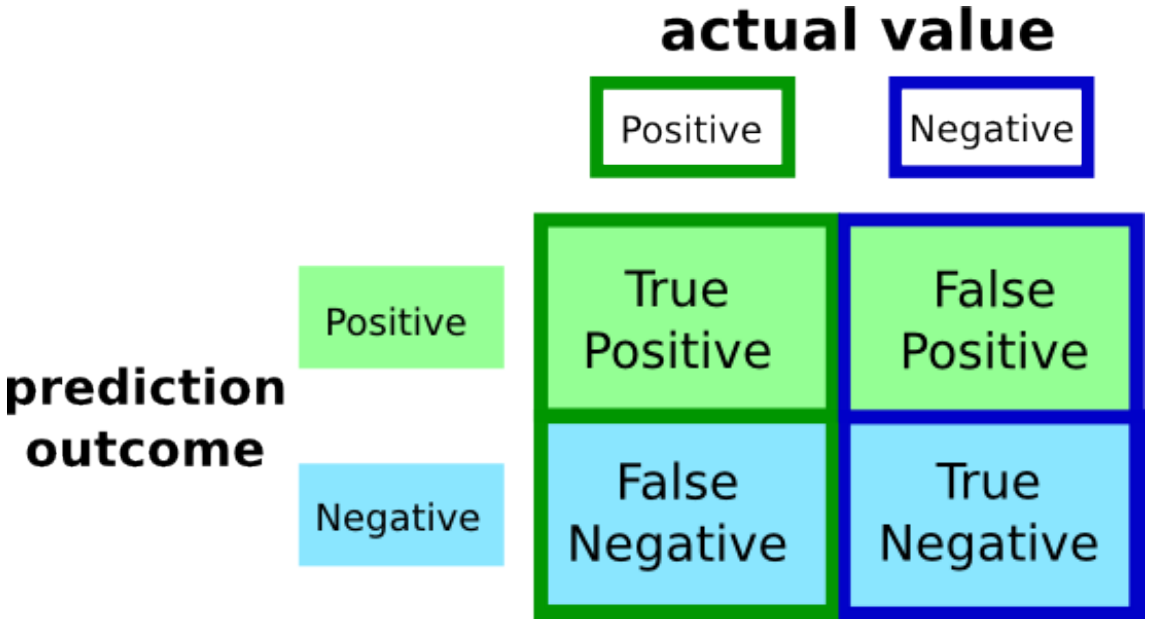
True/false positives



The distance threshold affects performance

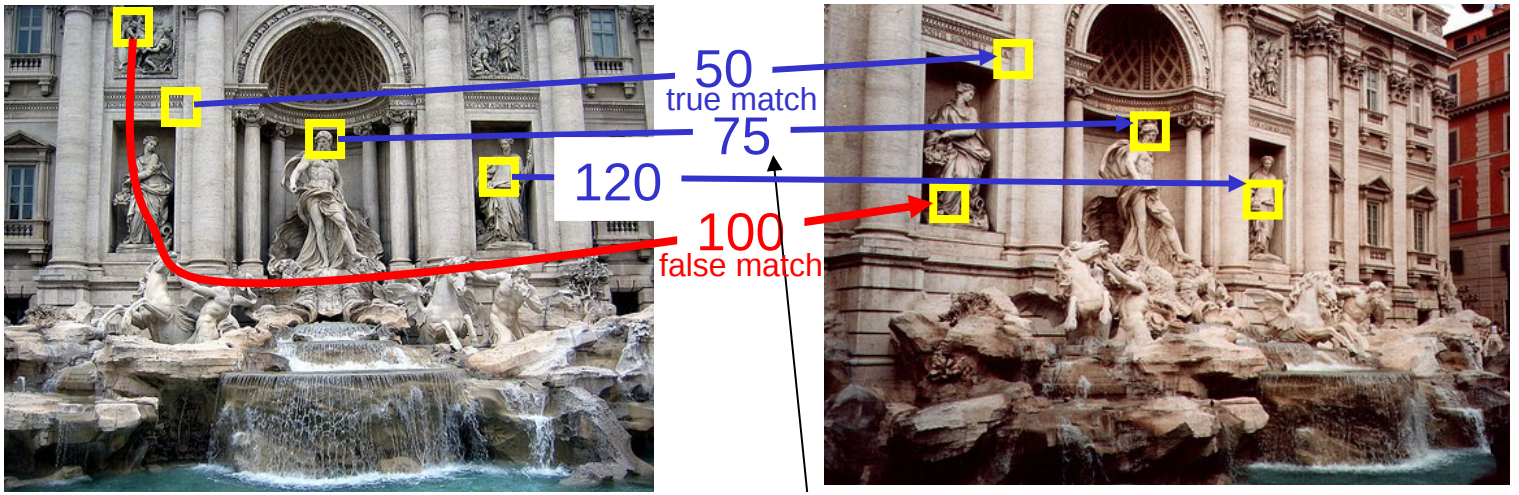
- **True positives** = # of detected matches that are correct
  - Suppose we want to maximize these—how to choose threshold?
- **False positives** = # of detected matches that are incorrect
  - Suppose we want to minimize these—how to choose threshold?

Evaluating the results



- TP – detected match is correct
- FP – detected match is incorrect
- FN – missing match
- TN – non-match correctly rejected

Evaluating results - example



Threshold = 60

	P	N
P'	1	0
N'	2	1

Threshold = 80

	P	N
P'	2	0
N'	1	1

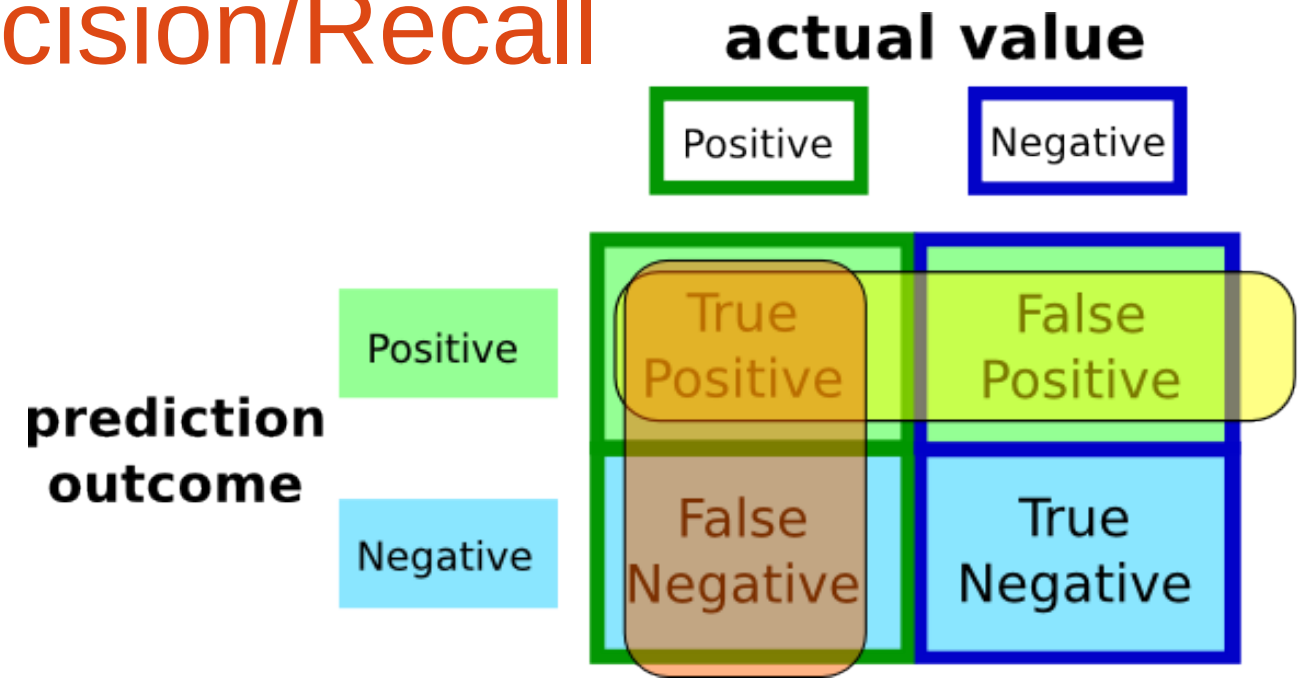
Threshold = 110

	P	N
P'	2	1
N'	1	0

Threshold = 150 (inf)

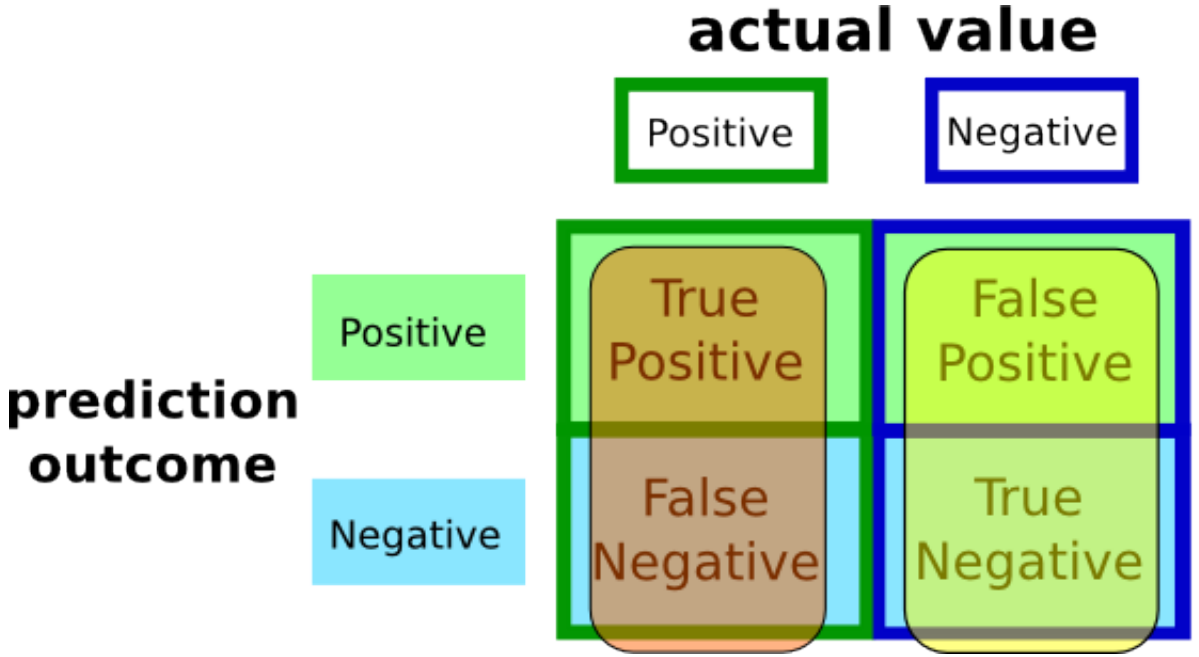
	P	N
P'	3	1
N'	0	0

Evaluating the results – Precision/Recall



- Precision – TP/(TP + FP)
- Recall – TP/(TP + FN)

Evaluating the results –
Receiver operating characteristic - ROC

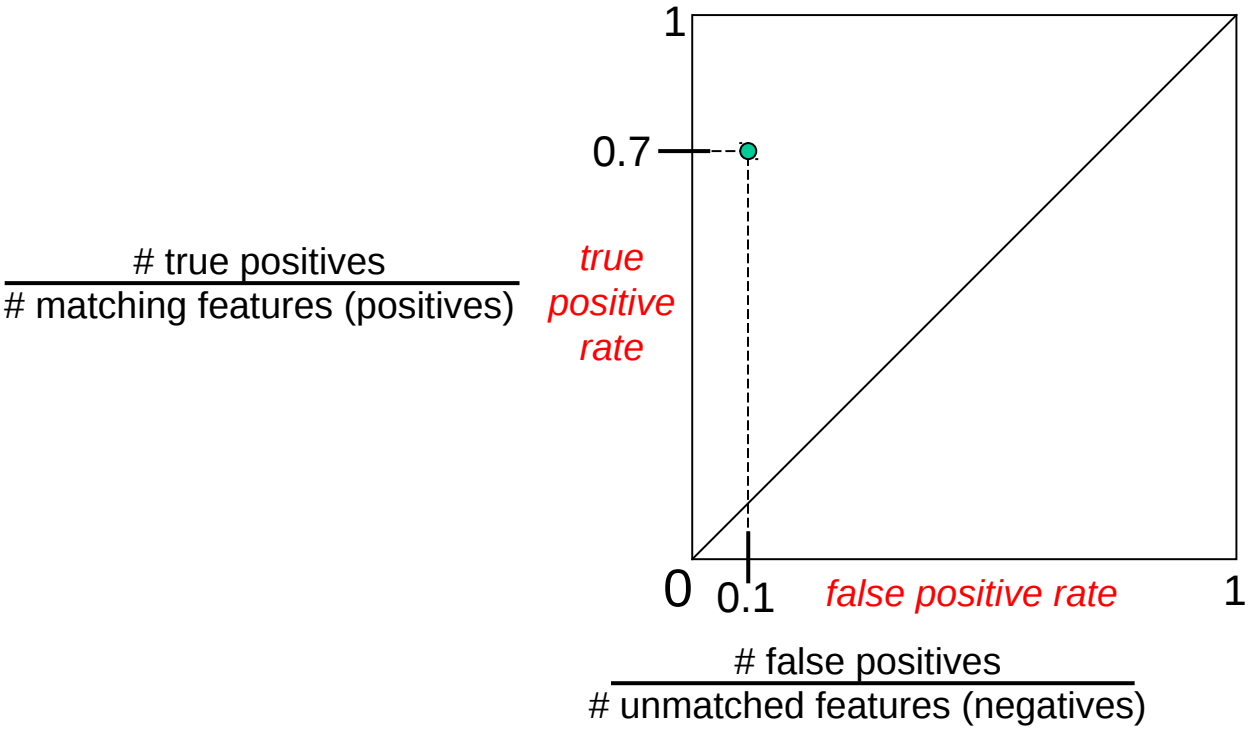


- TPR/sensitivity (Recall) – TP/(TP + FN)
- FPR – FP/(FP + TN)

AUC - Area under the curve

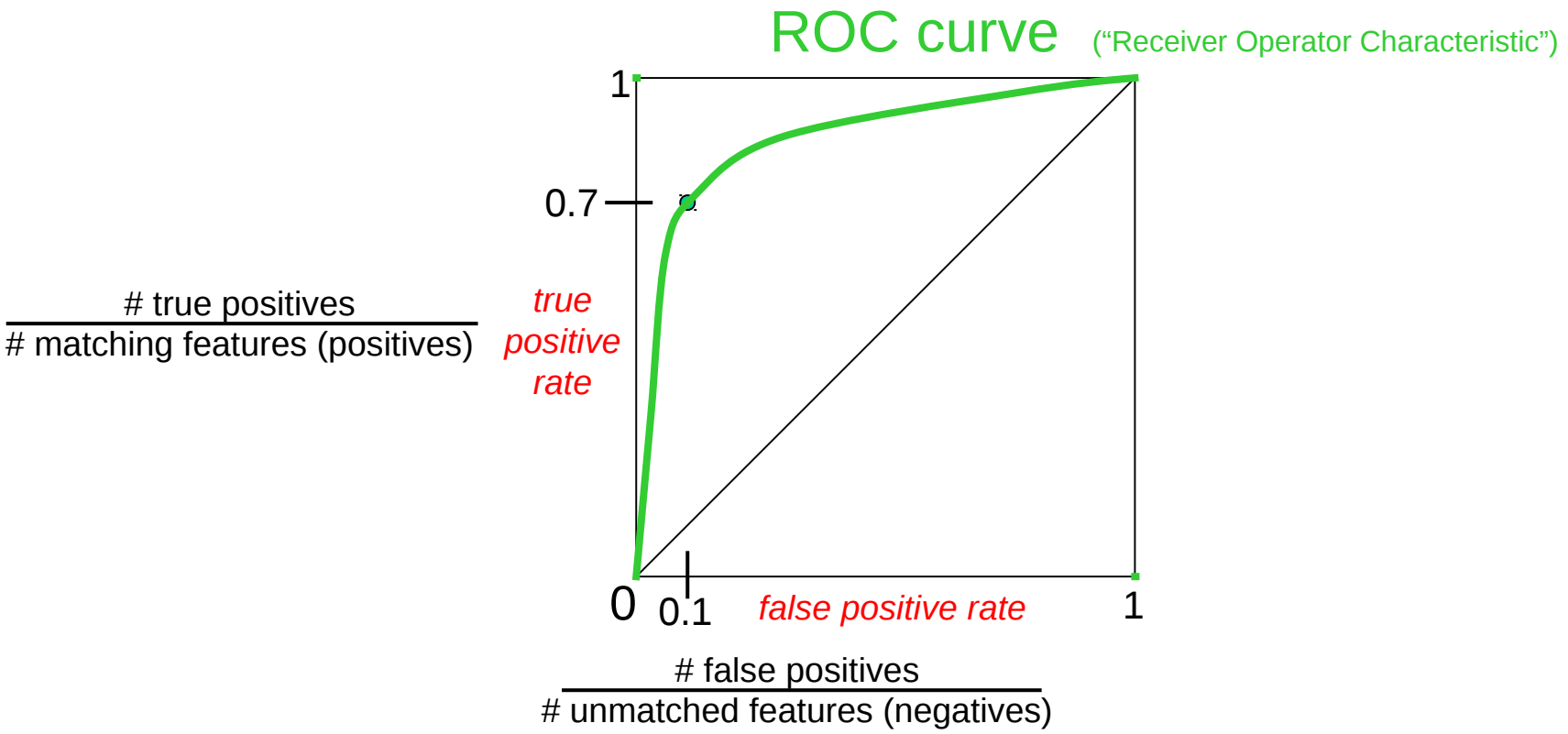
Evaluating the results

How can we measure the performance of a feature matcher?



Evaluating the results

How can we measure the performance of a feature matcher?



ROC Curves

- Generated by counting # current/incorrect matches, for different threholds
- Want to maximize area under the curve (AUC)
- Useful for comparing different feature matching methods
- For more info: http://en.wikipedia.org/wiki/Receiver_operating_characteristic
- http://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html

Model fitting

Model fitting

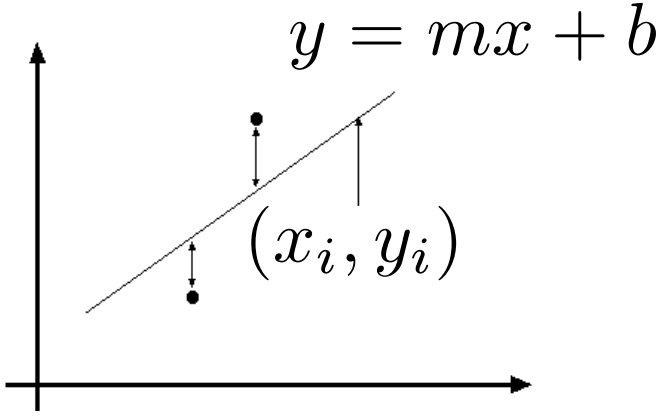
- Choose parametric model
  - Line, ellipse, Gaussian, etc.

- Three main questions:
  - What model represents the feature set?
  - Which of several model instances gets which feature?
  - How many model instances? (how many lines?)

Least Squares line fitting

- Data $(x_1, y_1), \ldots, (x_n, y_n)$
- The Model:

$$y_i = f(x_i; \theta) + \epsilon_i \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2)$$



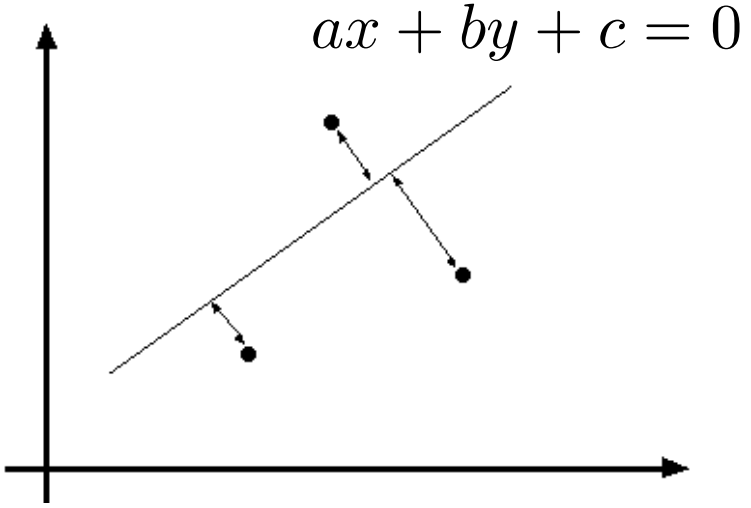
- In line fitting (linear regression):

$$f(x_i; \theta) = mx_i + b, \qquad \theta = \begin{bmatrix} m \\ b \end{bmatrix}$$

- The objective:

$$\min_{\theta} \sum_i ||y_i - f(x_i; \theta)||$$

Total Least Squares



- Data $(x_1, y_1), \ldots, (x_n, y_n)$
- Uncertainty (noise) in both coordinates.
- Minimize the distance to line:

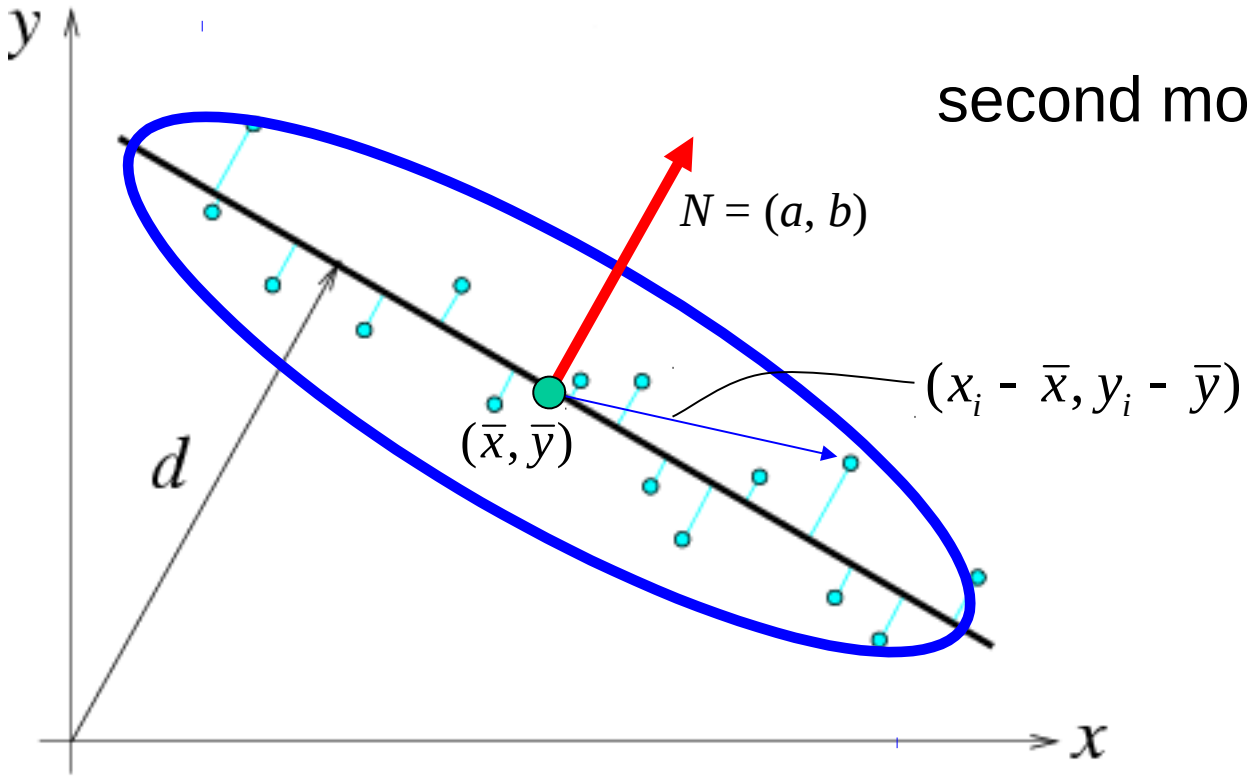
$$\min_{a,b,c} \sum_i \left(\begin{bmatrix} x_i & y_i & 1 \end{bmatrix}^T \begin{bmatrix} a & b & c \end{bmatrix} \right)^2 = \min_x ||Ax||$$

- Continue according to sec. A.2.1.

Total least squares

$$\min_{a,b,c} \sum_i \left(\begin{bmatrix} x_i & y_i & 1 \end{bmatrix}^T \begin{bmatrix} a & b & c \end{bmatrix} \right)^2 = \min_x ||Ax||$$

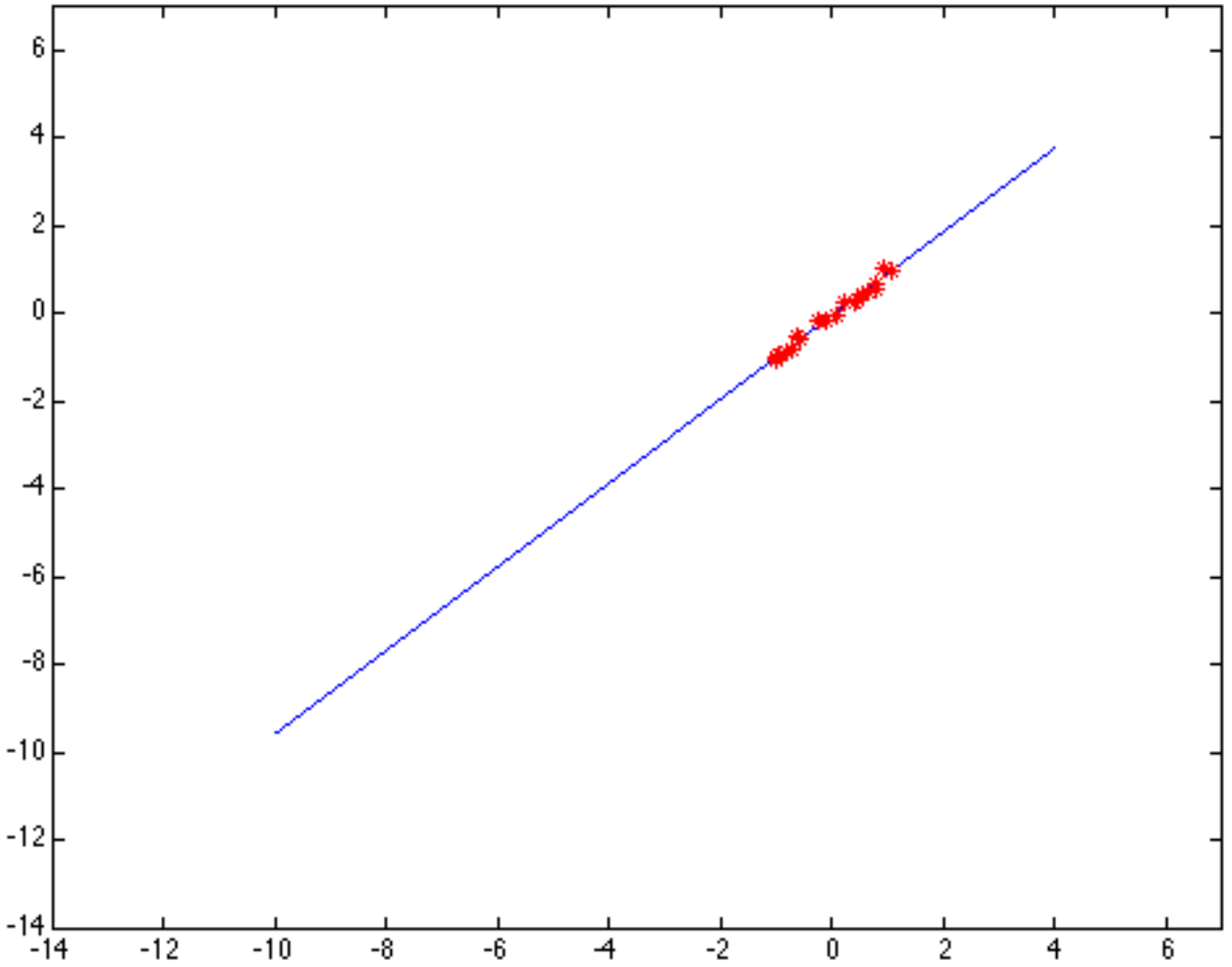
$$\hat{x} = \arg\min_x x^T (A^T A) x, \quad s.t. ||x|| = 1$$



second moment matrix

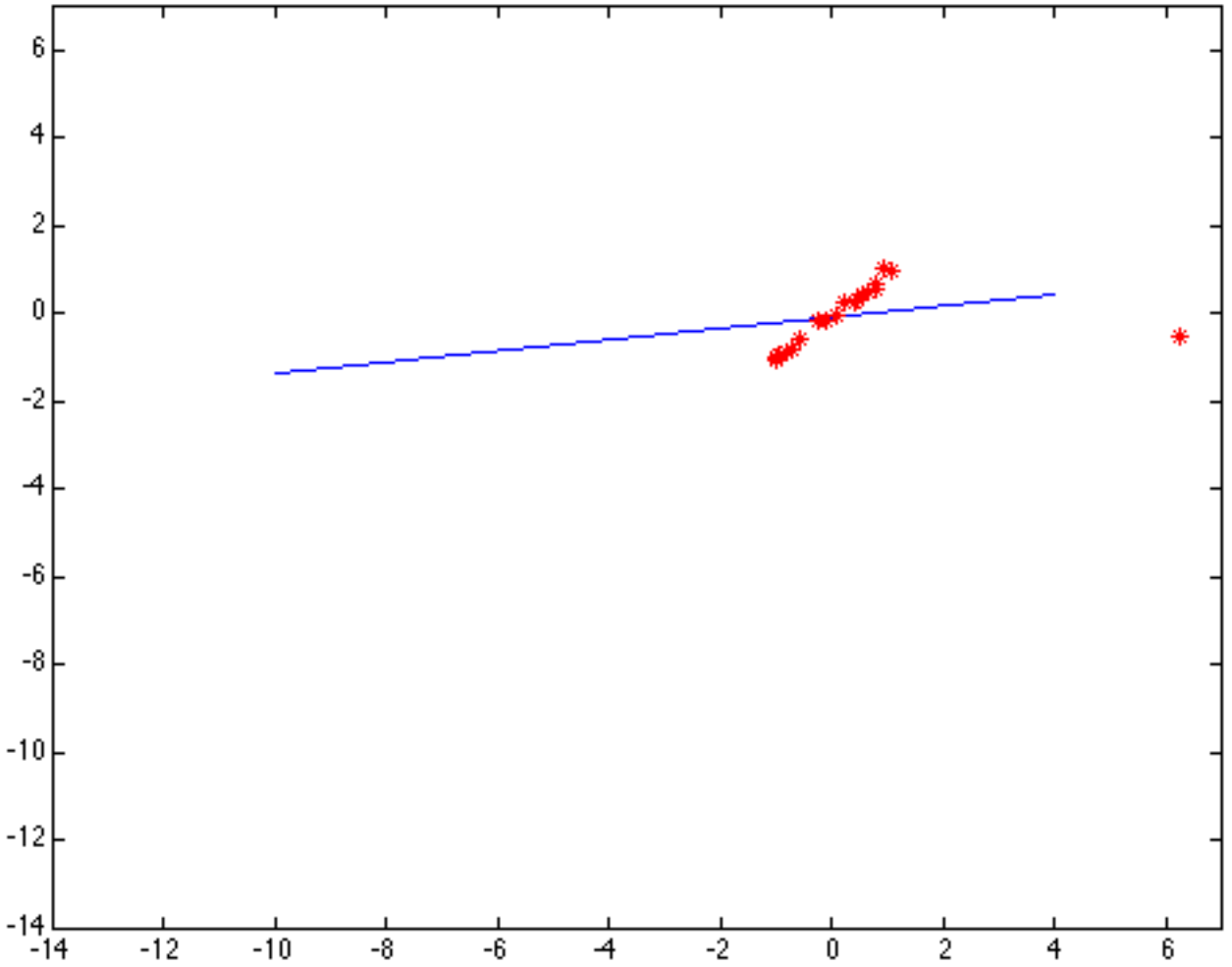
Least squares: Robustness to noise

Least squares fit to the red points:



Least squares: Robustness to noise

Least squares fit with an outlier:



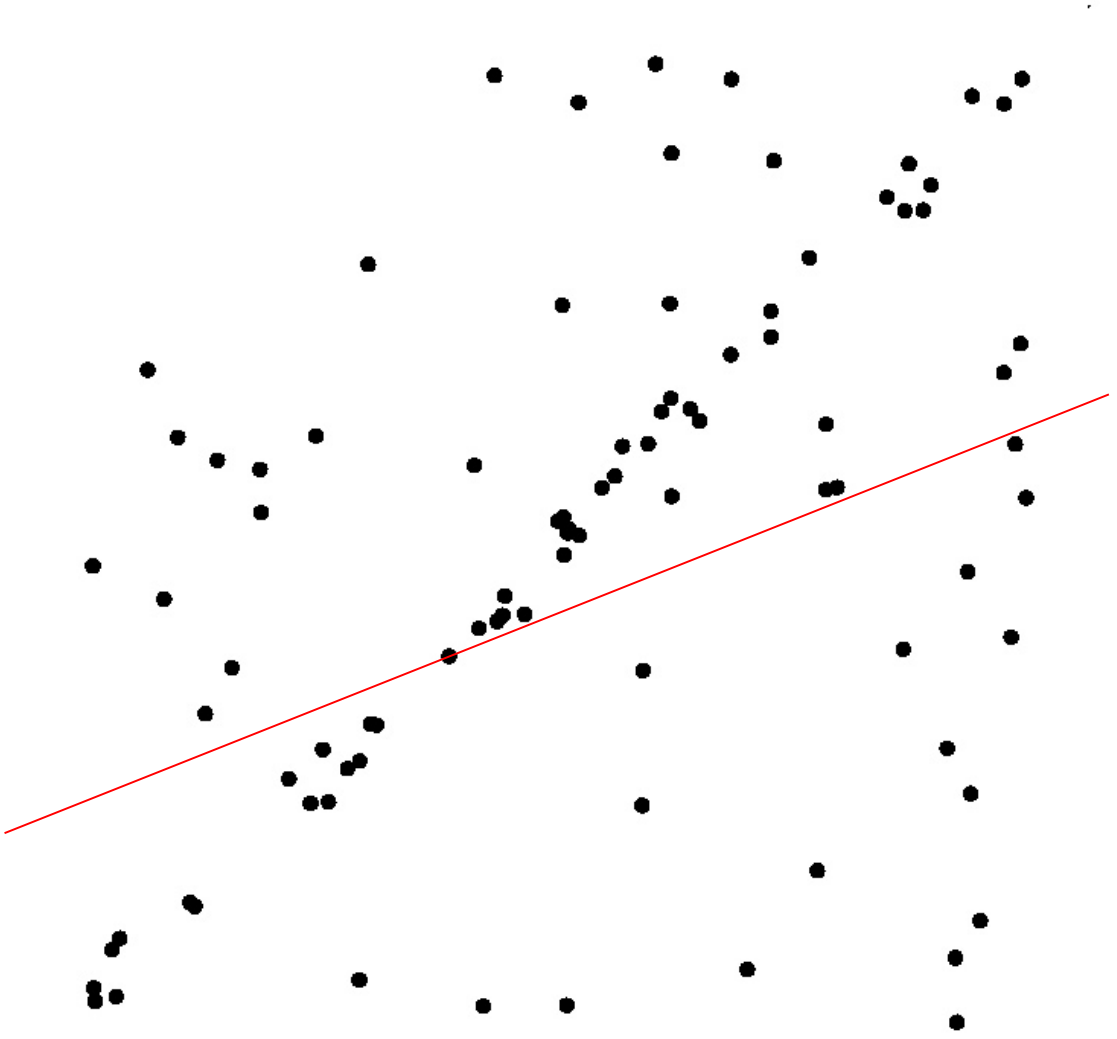
Problem: squared error heavily penalizes outliers

RANSAC

- Random sample consensus (RANSAC): Very general framework for model fitting in the presence of outliers

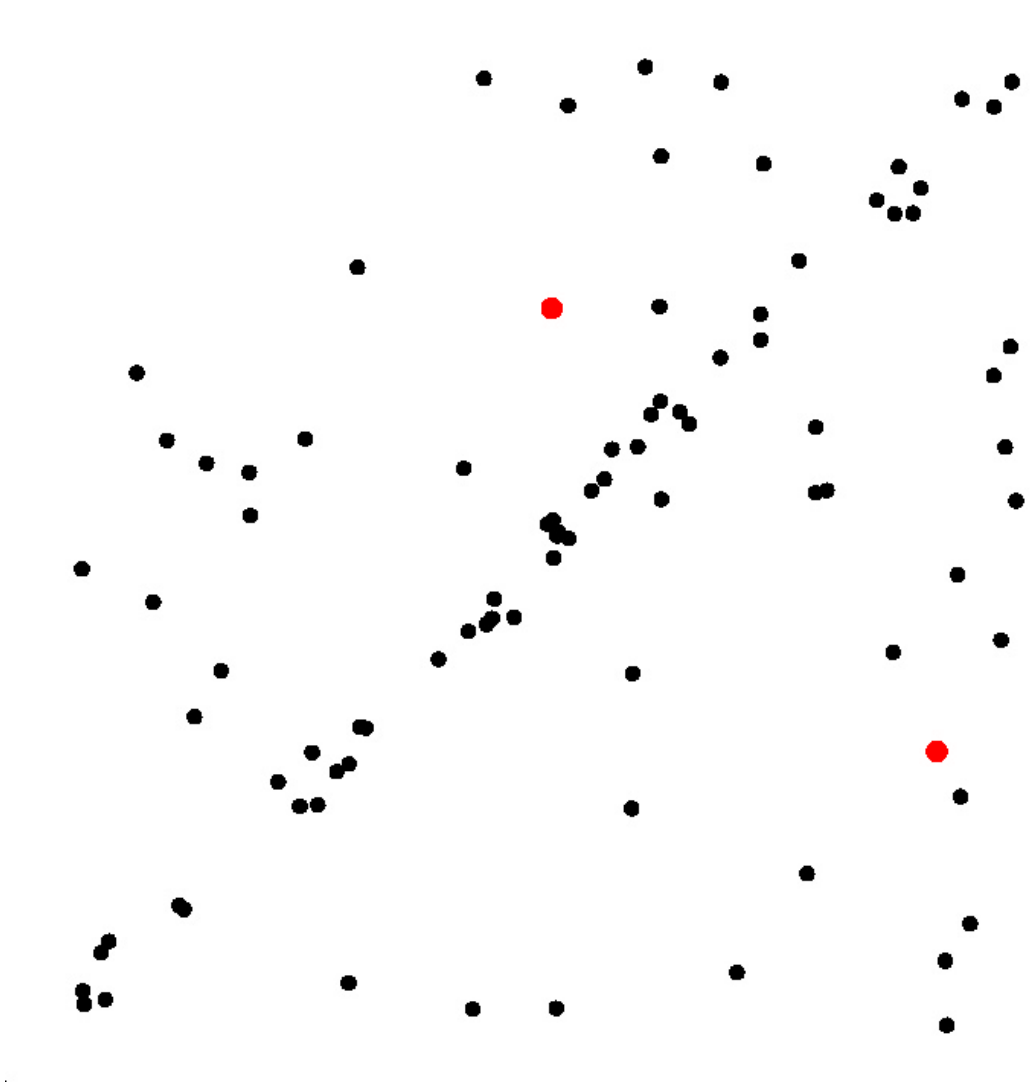
M. A. Fischler, R. C. Bolles.
Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography
. Comm. of the ACM, Vol 24, pp 381-395, 1981.

Fitting a Line



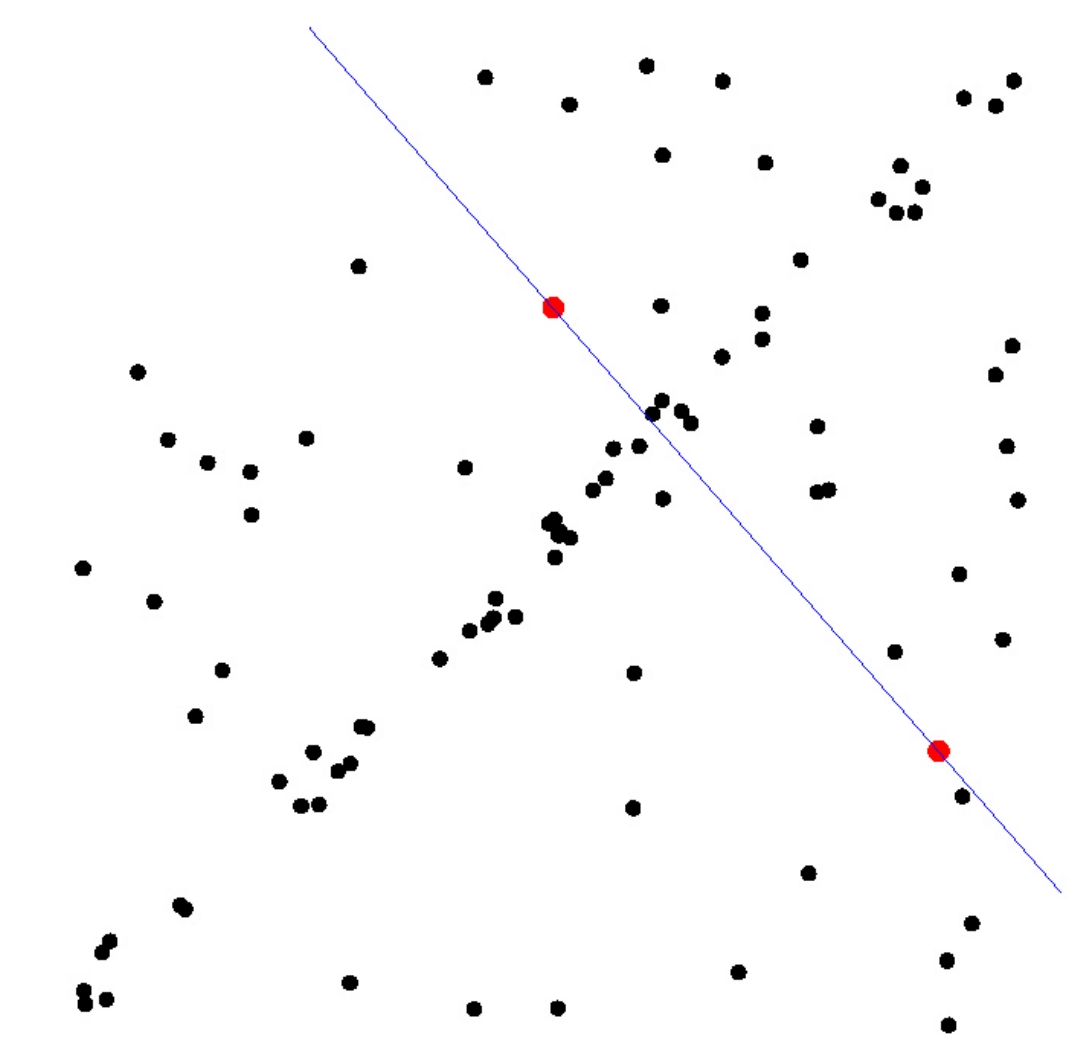
Least squares fit

RANSAC



1. Select a sample of **m** points at random

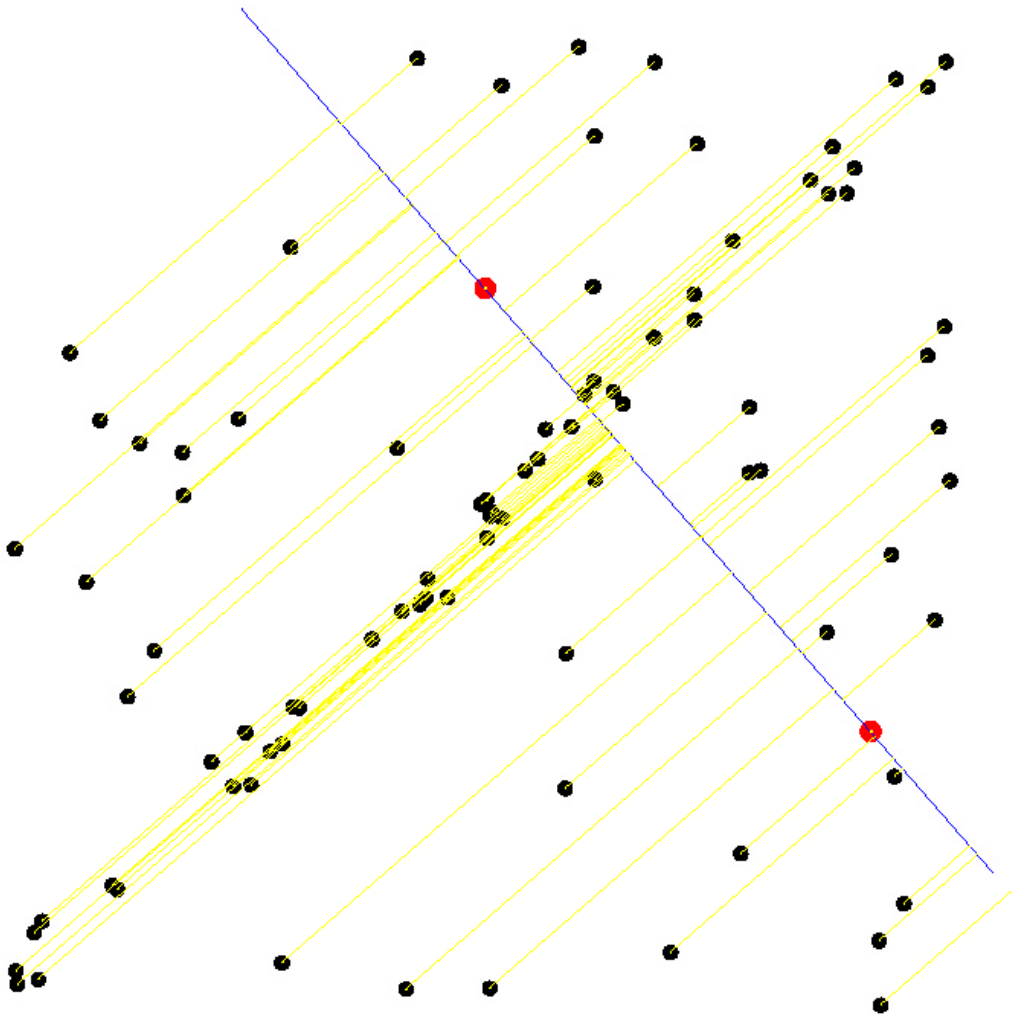
RANSAC



1. Select a sample of **m** points at random

2. **Calculate model parameters that fit the data in the sample**

RANSAC

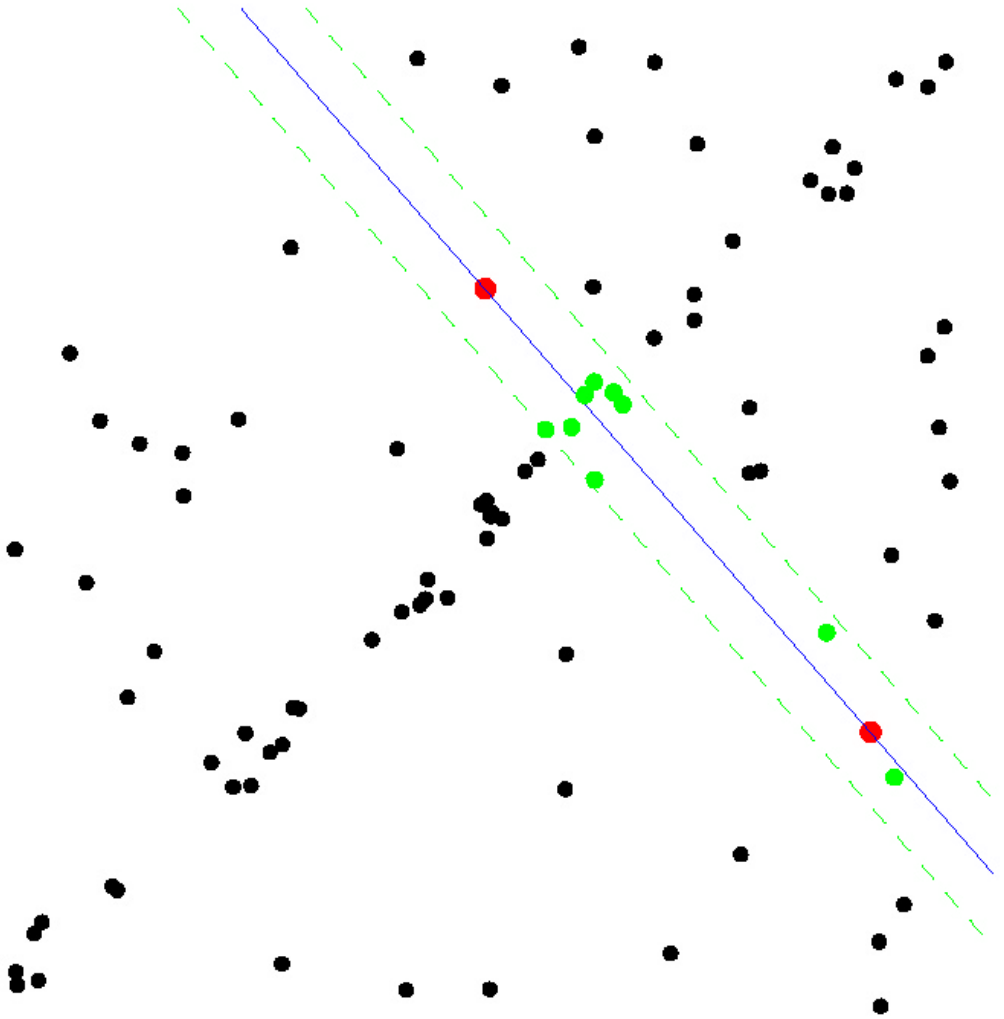


1. Select a sample of **m** points at random

2. **Calculate model parameters that fit the data in the sample**

3. **Calculate error function for each data point**

RANSAC



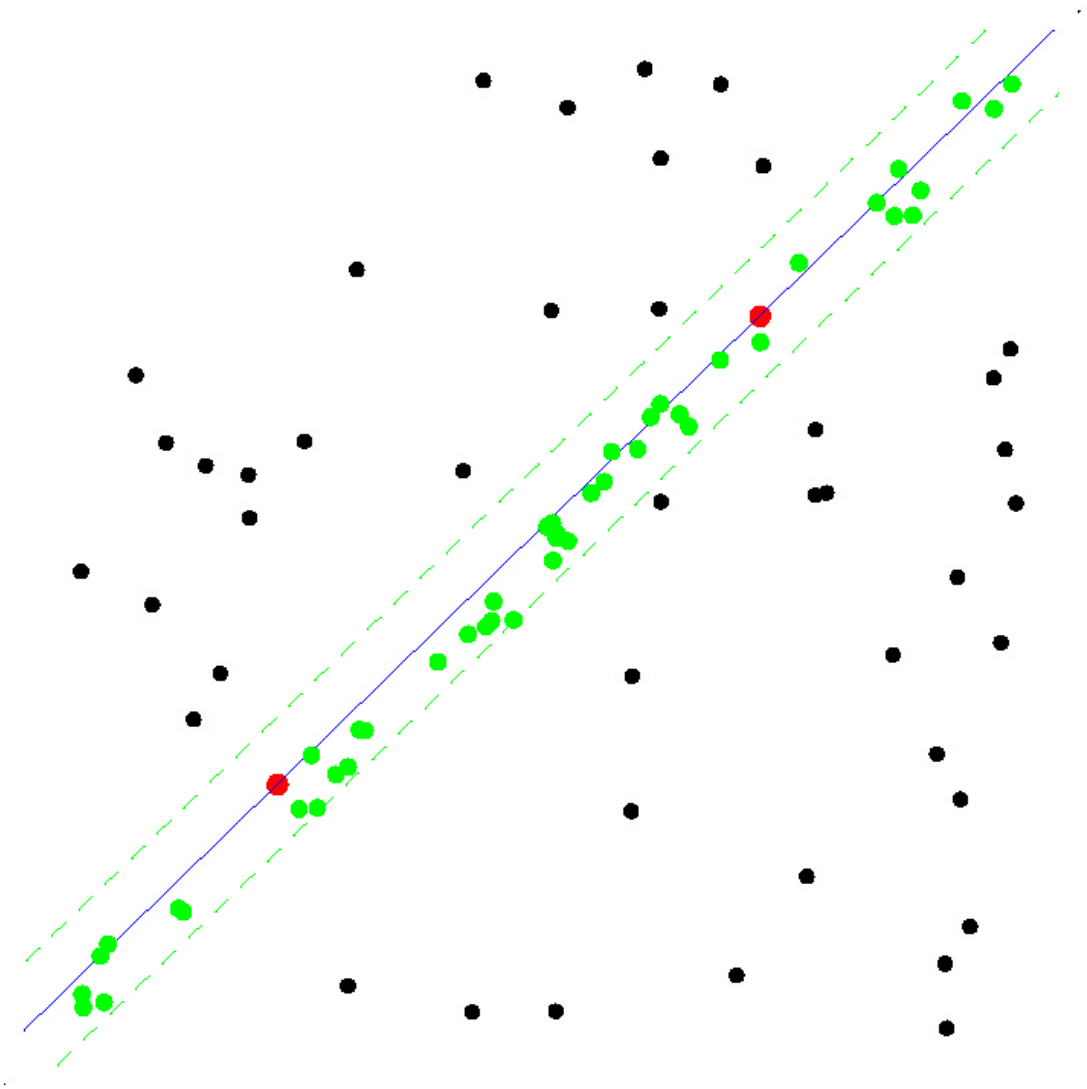
1. Select a sample of **m** points at random

2. **Calculate model parameters that fit the data in the sample**

3. **Calculate error function for each data point**

4. **Select data that support current hypothesis**

RANSAC



1. Select a sample of **m** points at random

2. **Calculate model parameters that fit the data in the sample**

3. Calculate error function for each data point

4. Select data that support current hypothesis

RANSAC for line fitting

Repeat ***N*** times:

- Draw ***s*** points uniformly at random
- Fit line to these ***s*** points
- Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than ***t***)
- If there are ***d*** or more inliers, accept the line and refit using all inliers

Choosing the parameters

- Initial number of points *s*
  - Typically minimum number needed to fit the model
- Distance threshold *t*
  - Choose *t* so probability for inlier is *p* (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$
- Number of samples *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)

$$\left(1-(1-e)^s\right)^N = 1-p$$

$$N = \log(1-p) / \log\left(1-(1-e)^s\right)$$

	proportion of outliers *e*						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Source: M. Pollefeys

RANSAC pros and cons

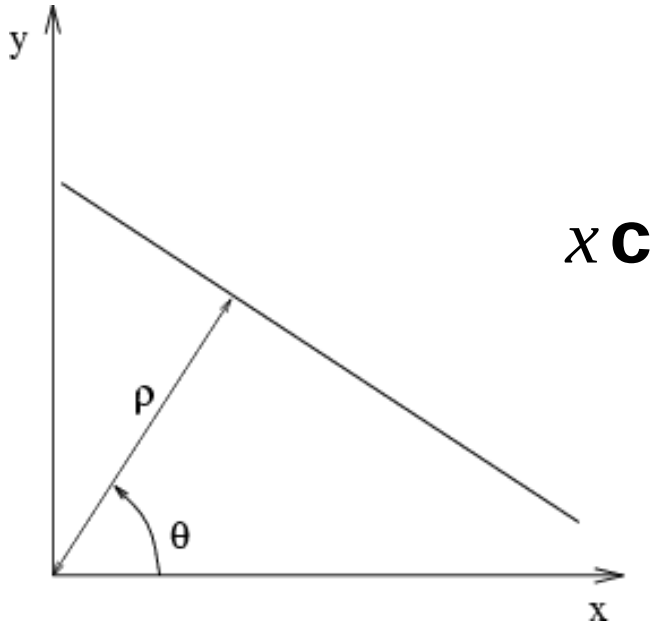
- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
- Cons
  - Lots of parameters to tune
  - Can’t always get a good initialization of the model based on the minimum number of samples
  - Sometimes too many iterations are required
  - Can fail for extremely low inlier ratios
  - We can often do better than brute-force sampling

אם יש יותר מ"קו" אחד?

Voting schemes

- Let each feature vote for all the models that are compatible with it
- Hopefully the noise features will not vote consistently for any single model
- Missing data doesn’t matter as long as there are enough features remaining to agree on a good model

Hough transform



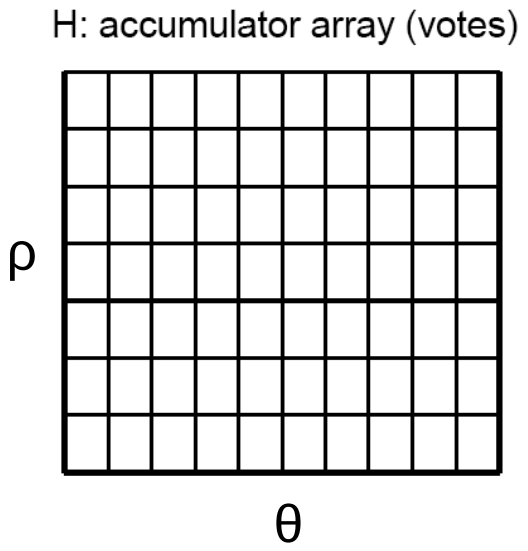
$$x\cos\theta + y\sin\theta = \rho$$

Each point will add a sinusoid in the (θ,ρ) parameter space

Algorithm outline

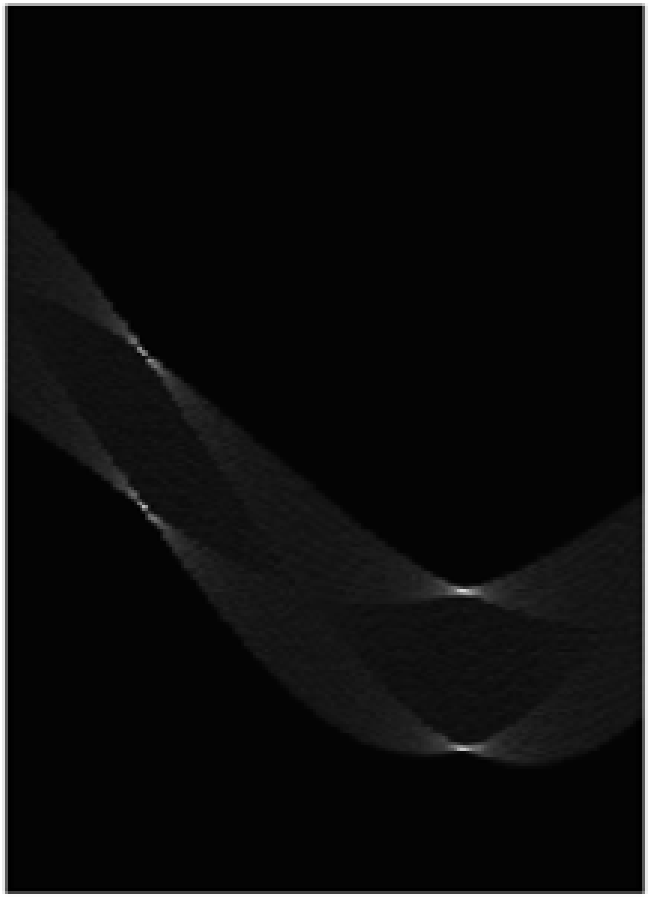
- Initialize accumulator H to all zeros

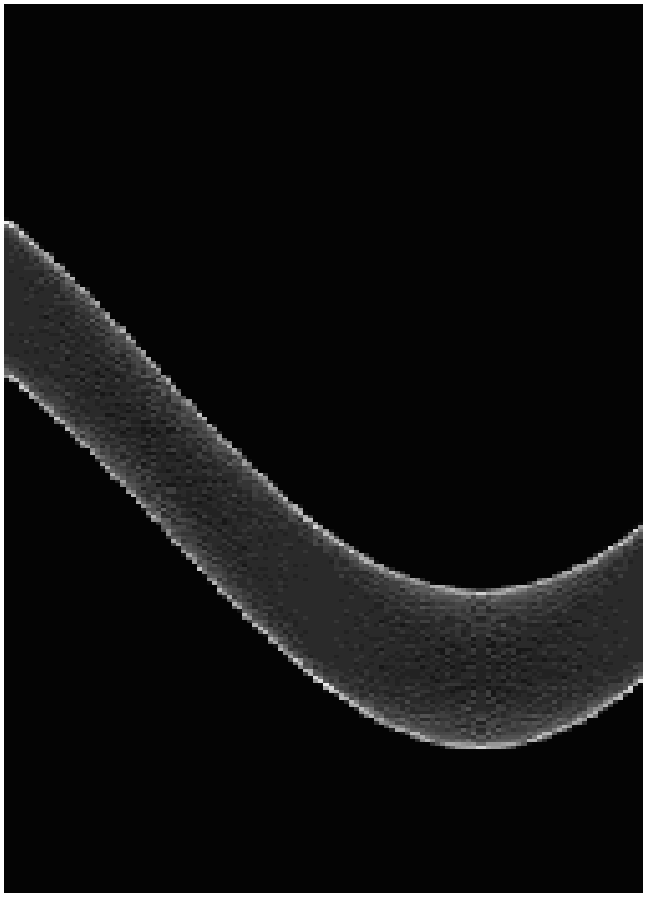
- For each edge point (x,y) in the image
  For θ = 0 to 180
  ρ = x cos θ + y sin θ
  H(θ, ρ) = H(θ, ρ) + 1
  end
  end



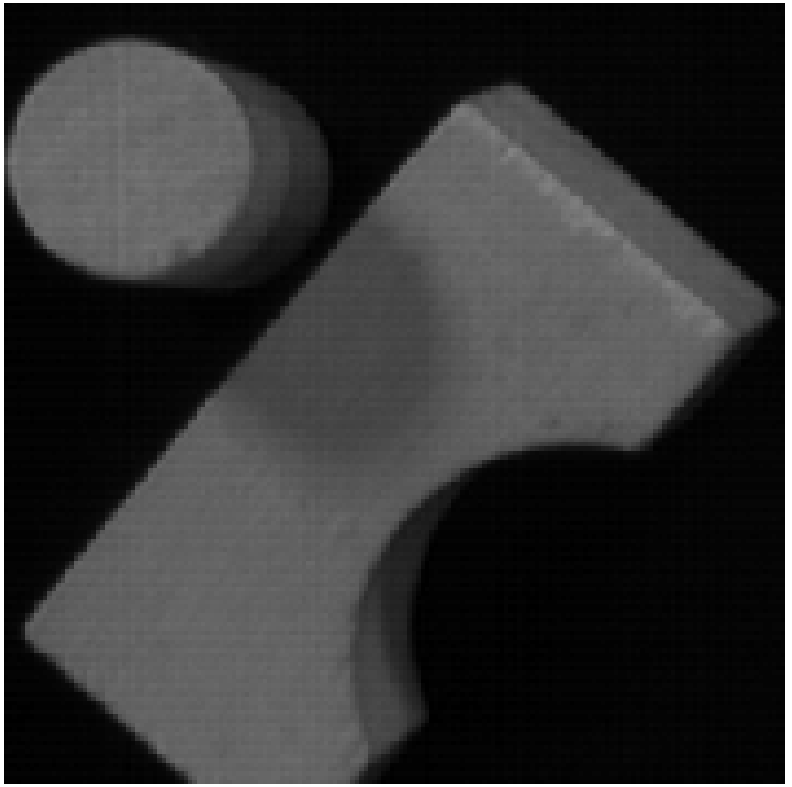
- Find the value(s) of (θ, ρ) where H(θ, ρ) is a local maximum
  - The detected line in the image is given by
    ρ = x cos θ + y sin θ

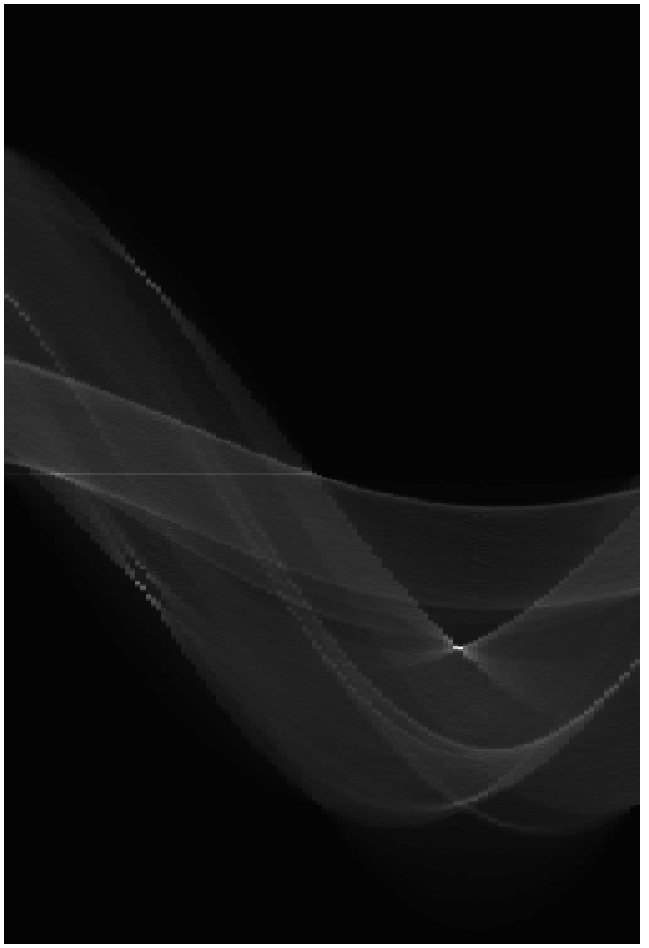
Other shapes



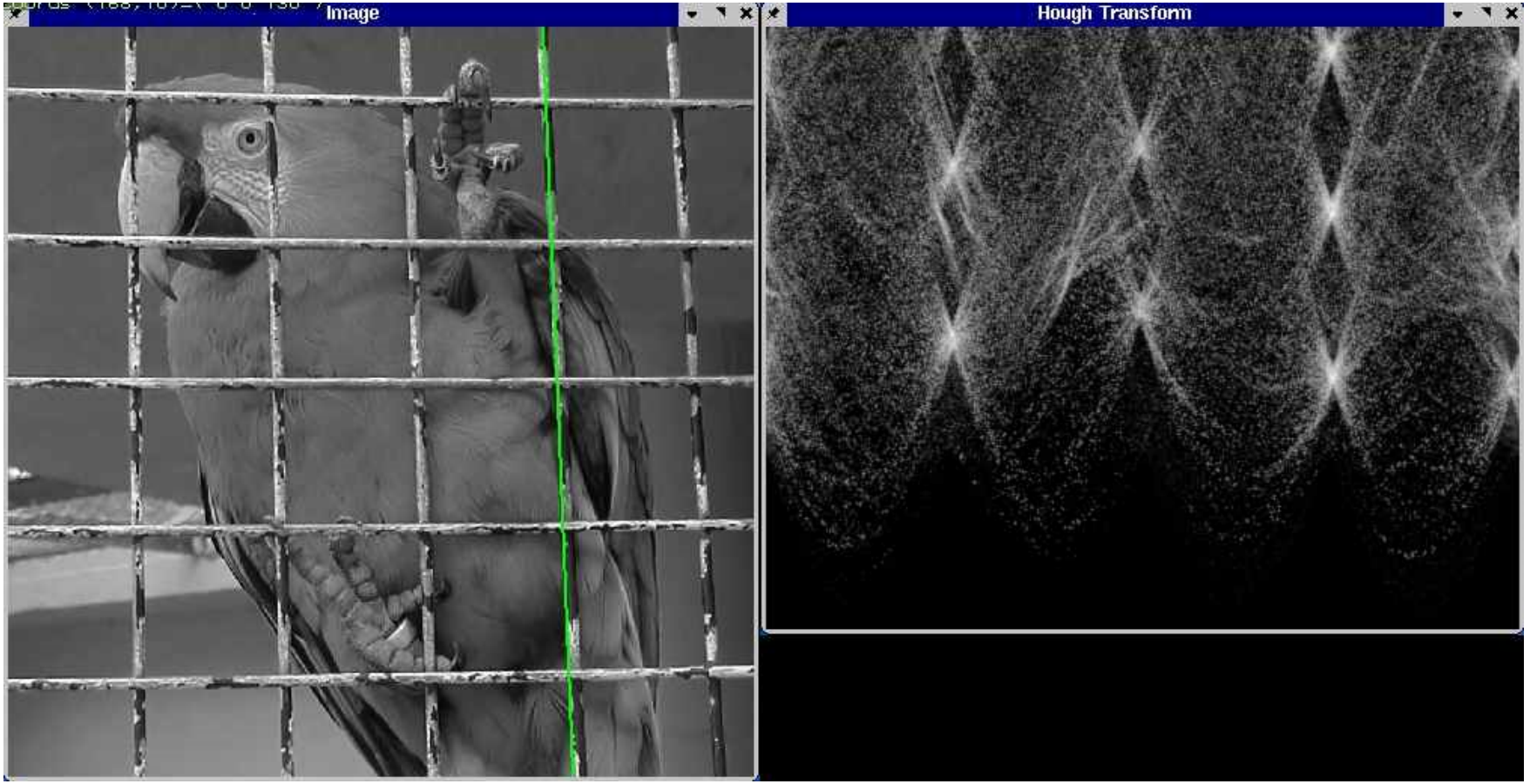


Several lines



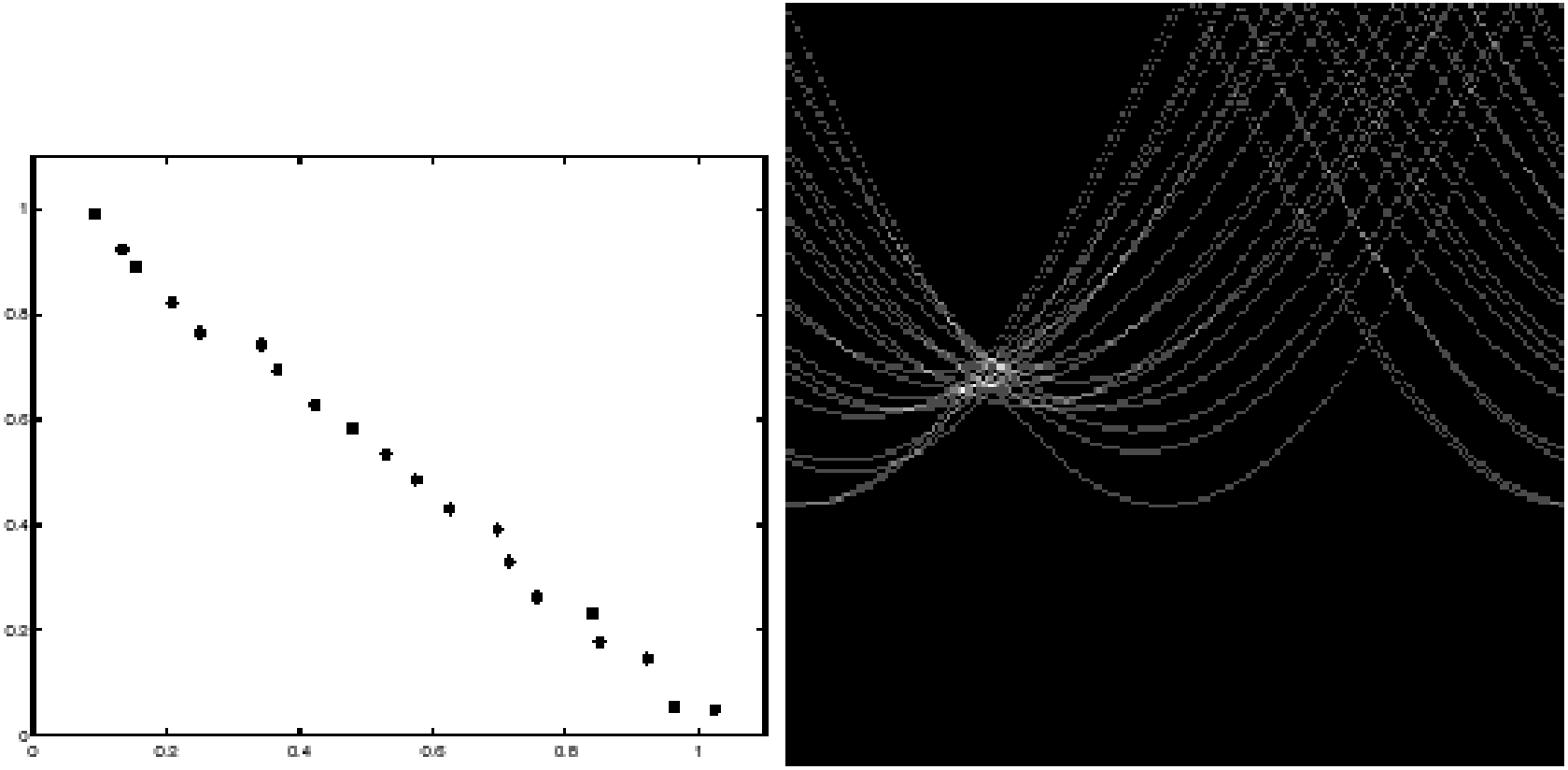


A more complicated image



http://ostatic.com/files/images/ss_hough.jpg

Effect of noise

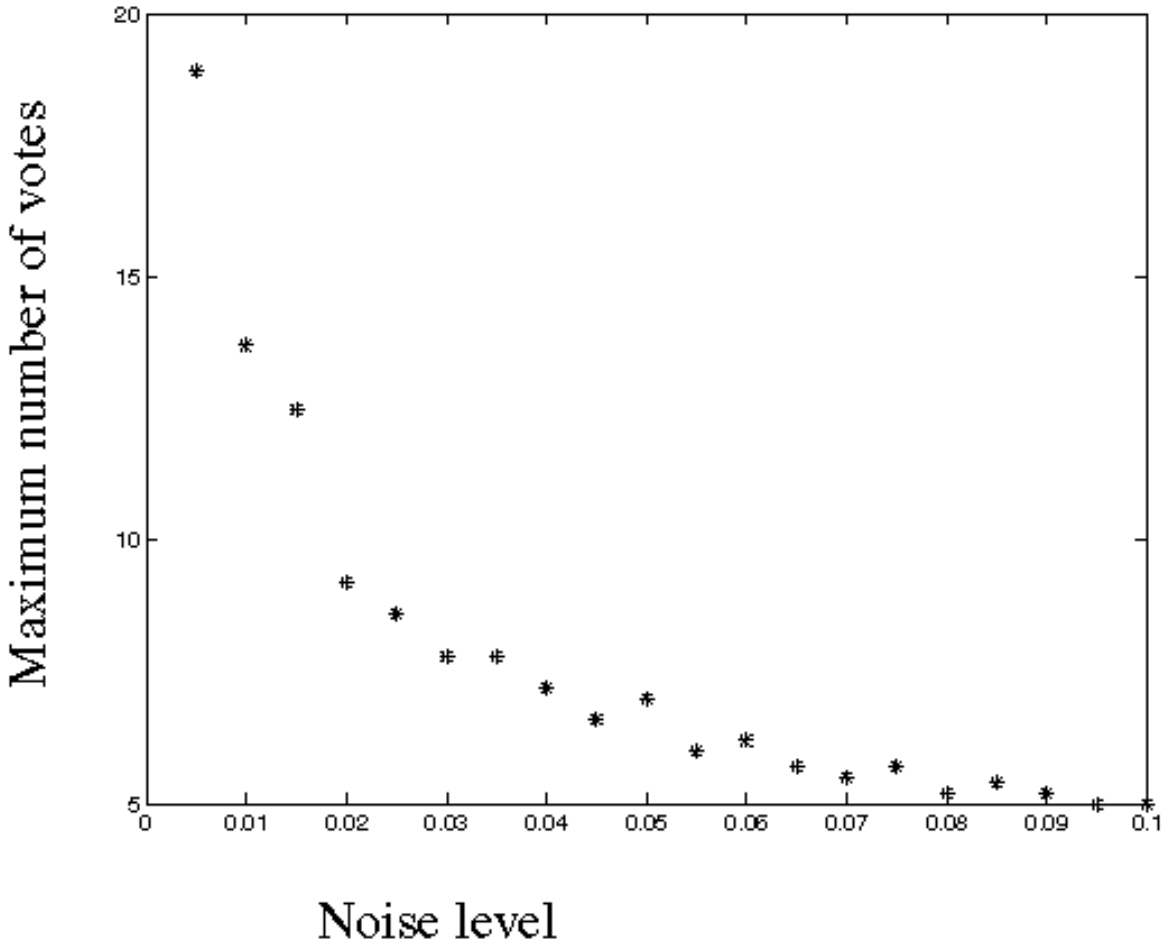


features

votes

Effect of noise

- Number of votes for a line of 20 points with increasing noise:

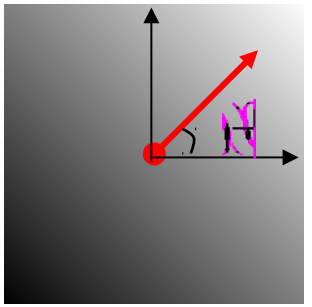


Dealing with noise

- Choose a good grid / discretization
  - Too coarse: large votes obtained when too many different lines correspond to a single bucket
  - Too fine: miss lines because some points that are not exactly collinear cast votes for different buckets
- Increment neighboring bins (smoothing in accumulator array)
- Try to get rid of irrelevant features
  - Take only edge points with significant gradient magnitude

Incorporating image gradients

- Recall: when we detect an edge point, we also know its gradient direction
- But this means that the line is uniquely determined!



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- Modified Hough transform:

For each edge point (x,y)
    θ = gradient orientation at (x,y)
    ρ = x cos θ + y sin θ
    H(θ, ρ) = H(θ, ρ) + 1
end

Hough transform for circles

- How many dimensions will the parameter space have?
- Given an oriented edge point, what are all possible bins that it can vote for?

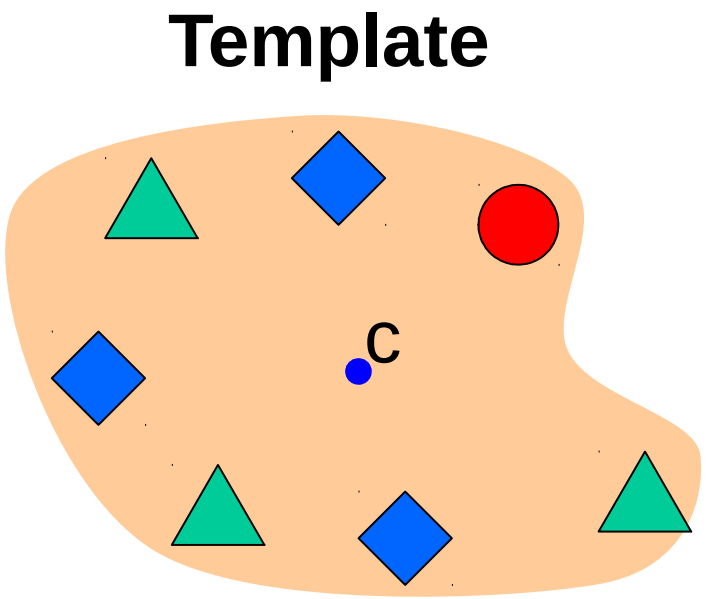
$$x = a + R\cos\theta$$

$$y = b + R\sin\theta$$

Generalized Hough transform

Generalized Hough transform

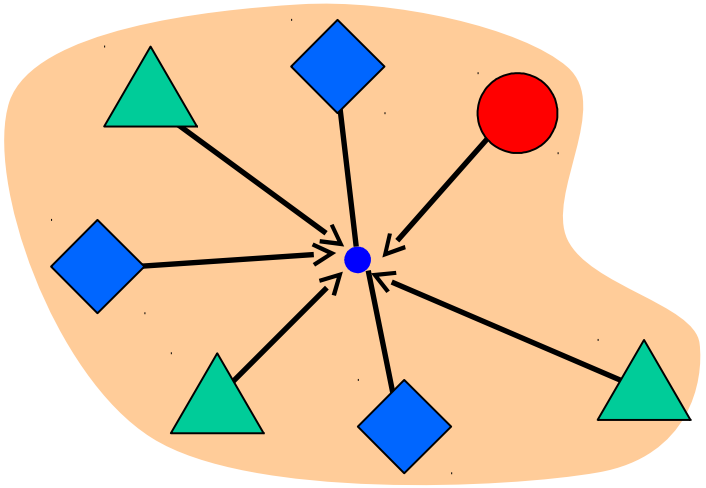
- We want to find a template defined by its reference point (center) and several distinct types of landmark points in stable spatial configuration



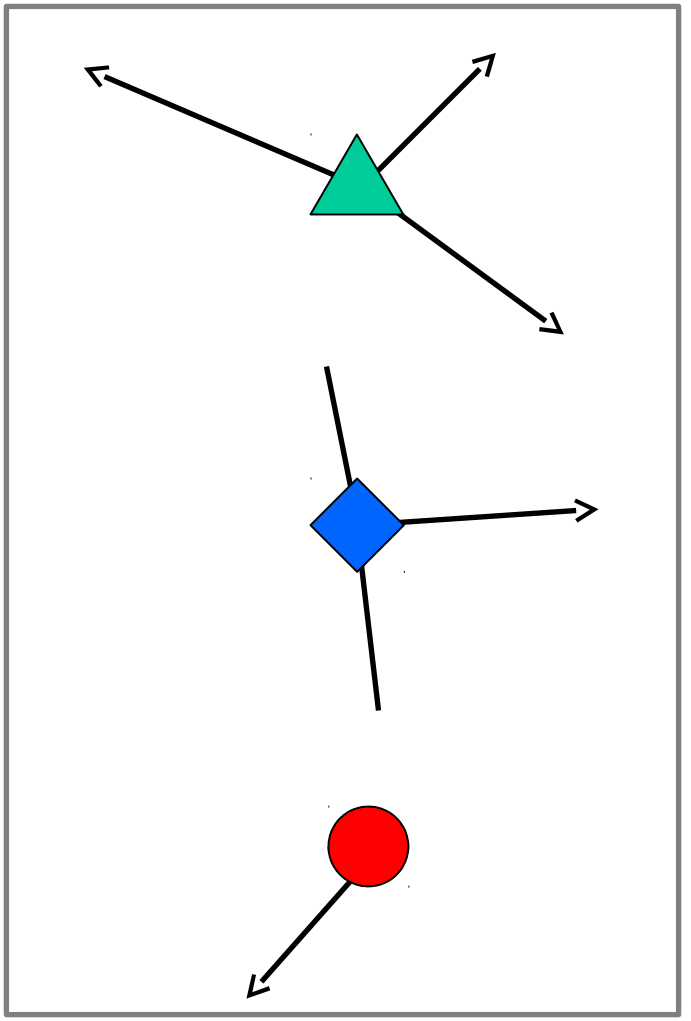
Generalized Hough transform

- Template representation: for each type of landmark point, store all possible displacement vectors towards the center

Template



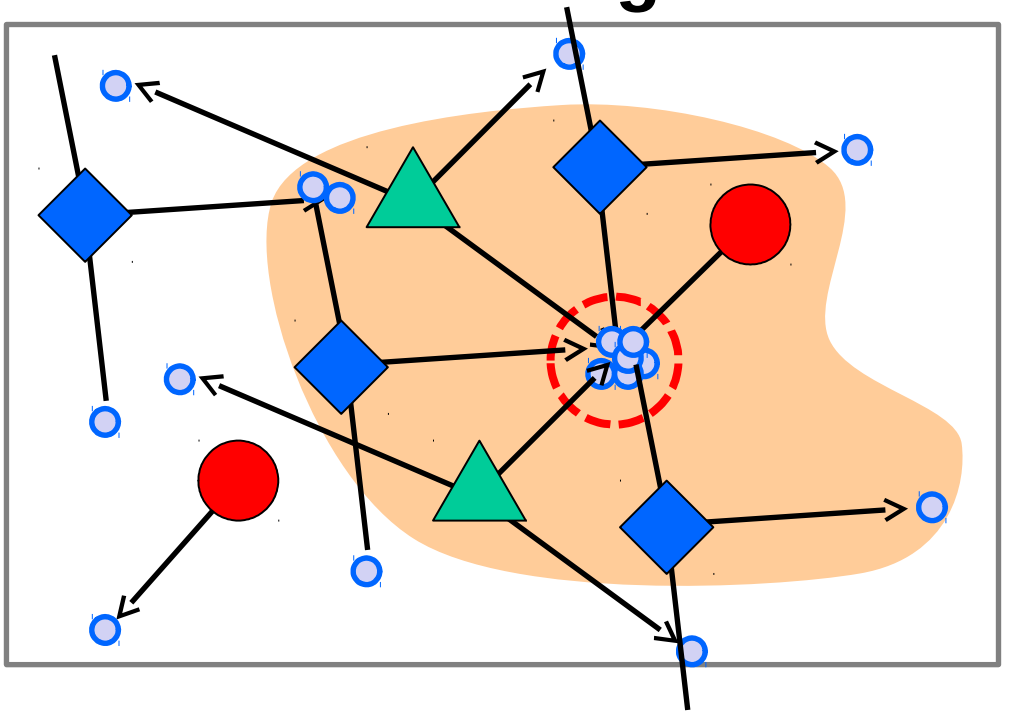
Model



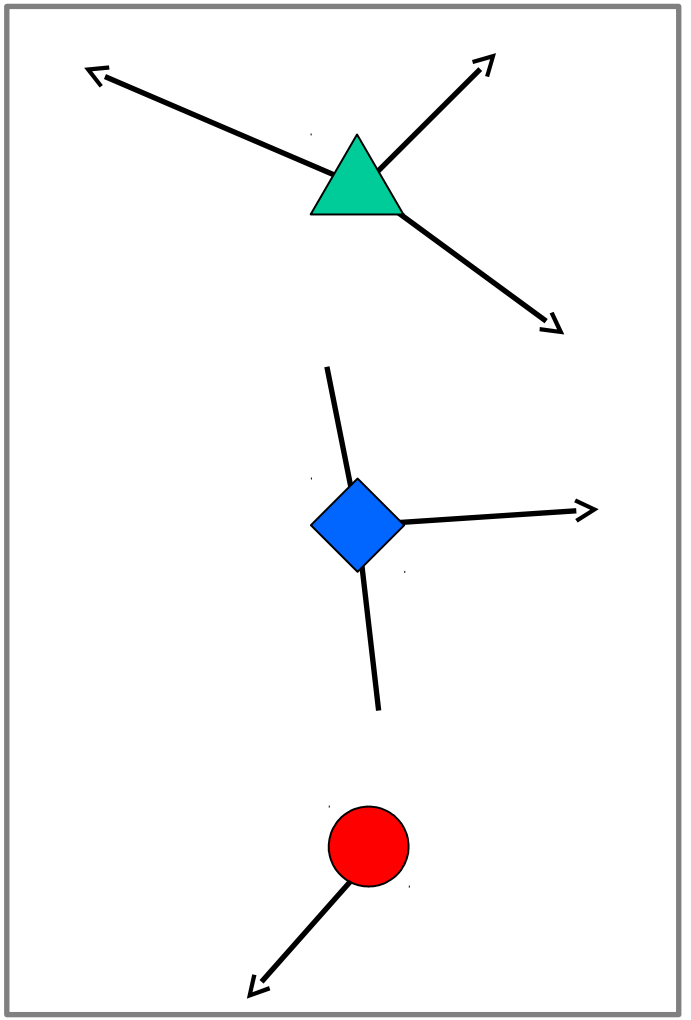
Generalized Hough transform

- Detecting the template:
  - For each feature in a new image, look up that feature type in the model and vote for the possible center locations associated with that type in the model

Test image

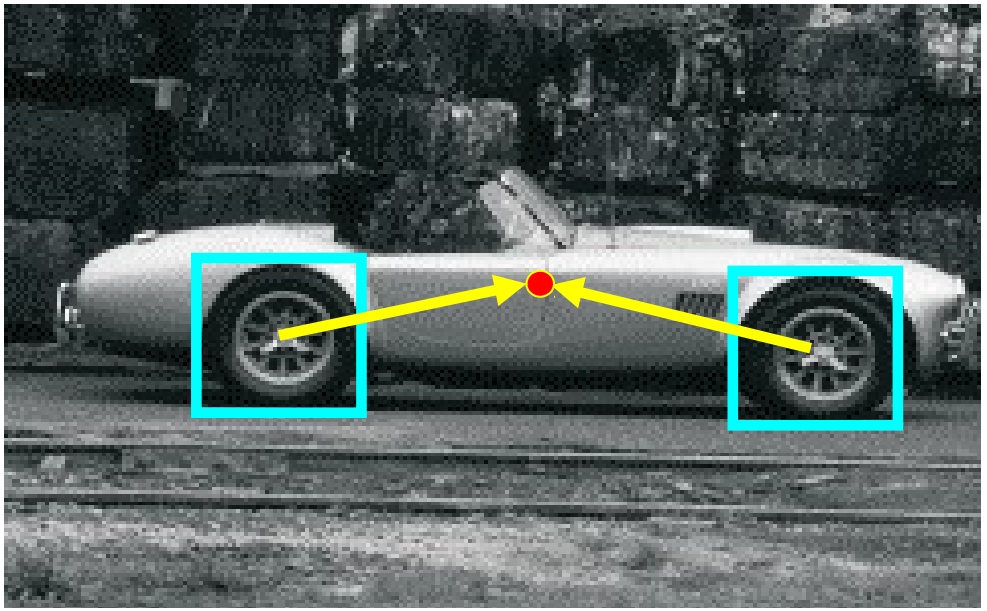


Model



Application in recognition

- Index displacements by “visual codeword”



training image

visual codeword with displacement vectors

B. Leibe, A. Leonardis, and B. Schiele,
Combined Object Categorization and Segmentation with an Implicit Shape Model,
ECCV Workshop on Statistical Learning in Computer Vision 2004

K-means clustering

- Want to minimize sum of squared Euclidean distances between points *xi* and their nearest cluster centers *mk*

$$D(X, M) = \sum_{\text{cluster } k} \sum_{\substack{\text{point } i \text{ in} \\ \text{cluster } k}} (x_i - m_k)^2$$

Algorithm:

- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each data point to the nearest center
  - Recompute each cluster center as the mean of all points assigned to it

סיכום – מה היה היום?

- פירמידות
- נקודות עניין – detection+description
- Model fitting
- RANSAC
- Hough transform
- Generalized Hough transform
- K-means

בפעם הבאה

- זיהוי פנים – face detection
- סיווג – classification
- הרצאות 5+6