COMPUTER VISION AND PHOTOGRAMMETRY

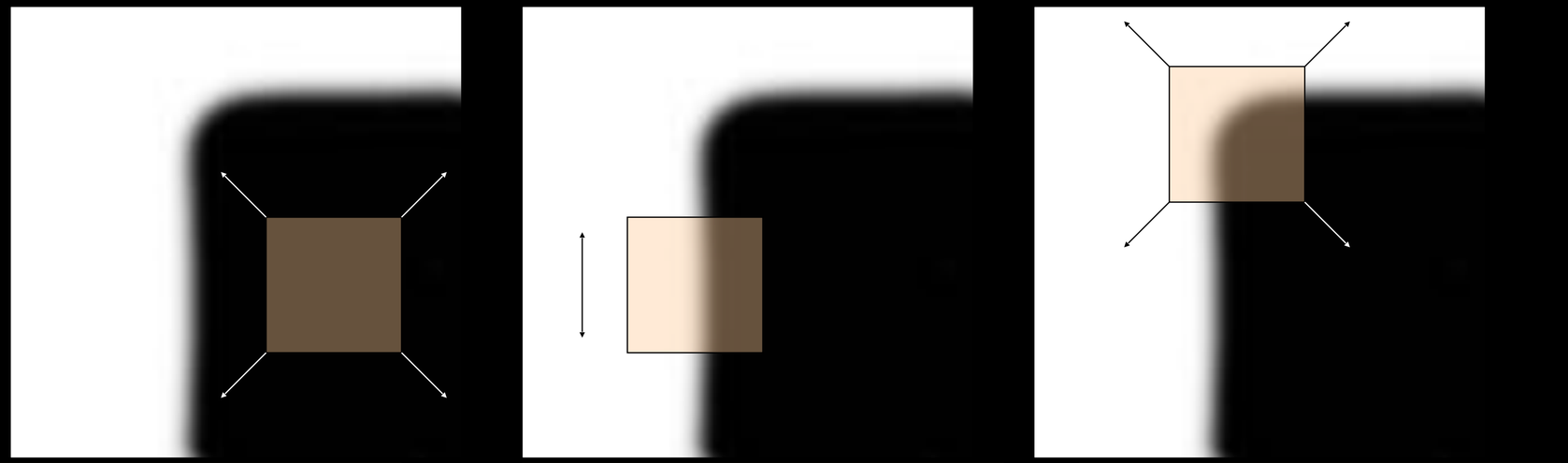
FEATURE DESCRIPTORS AND MATCHING

LAST WEEK

- General
- Keypoint detection
  - Corners (Harris)
  - Scale invariance: LoG and DoG, Harris-Laplace

CORNER DETECTION: BASIC IDEA

- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"FLAT" REGION:
NO CHANGE IN ALL DIRECTIONS

"EDGE":
NO CHANGE ALONG THE EDGE DIRECTION

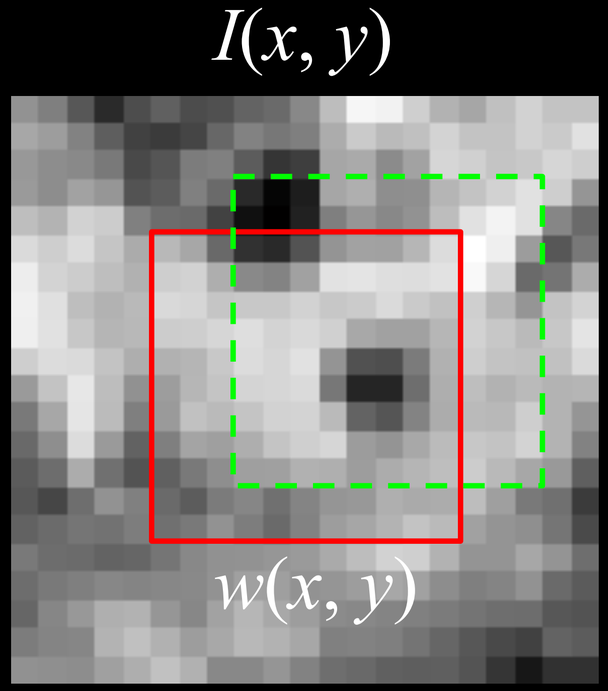
"CORNER":
SIGNIFICANT CHANGE IN ALL DIRECTIONS

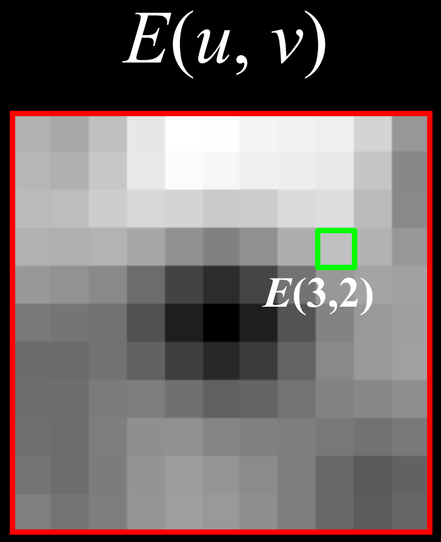
Source: A. Efros

CORNER DETECTION: MATHEMATICS

- Change in appearance of window w(x,y) for the shift [u,v]:

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

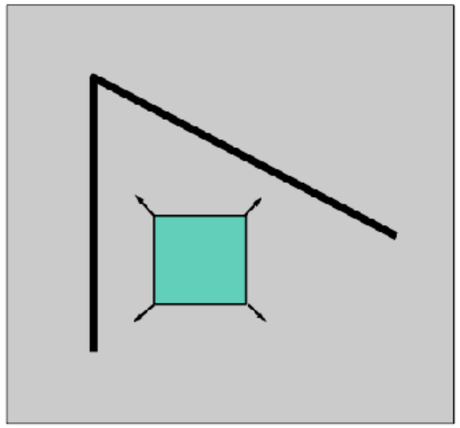




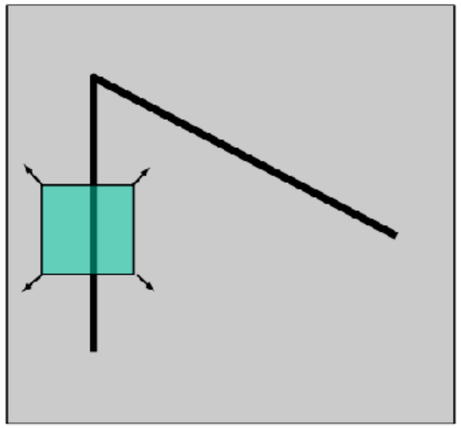
HARRIS CORNER MATH

Local measure of feature uniqueness

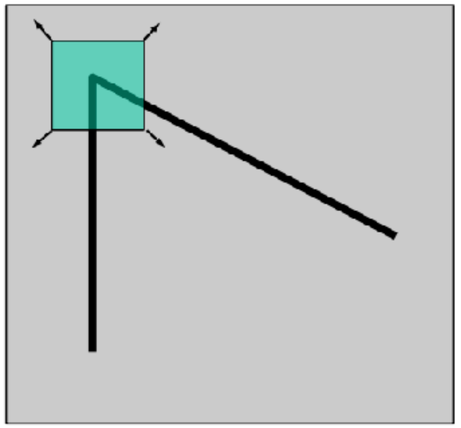
- E(u,v) = amount of change when you shift the window by (u,v)



E(u,v) is small
for **all** shifts



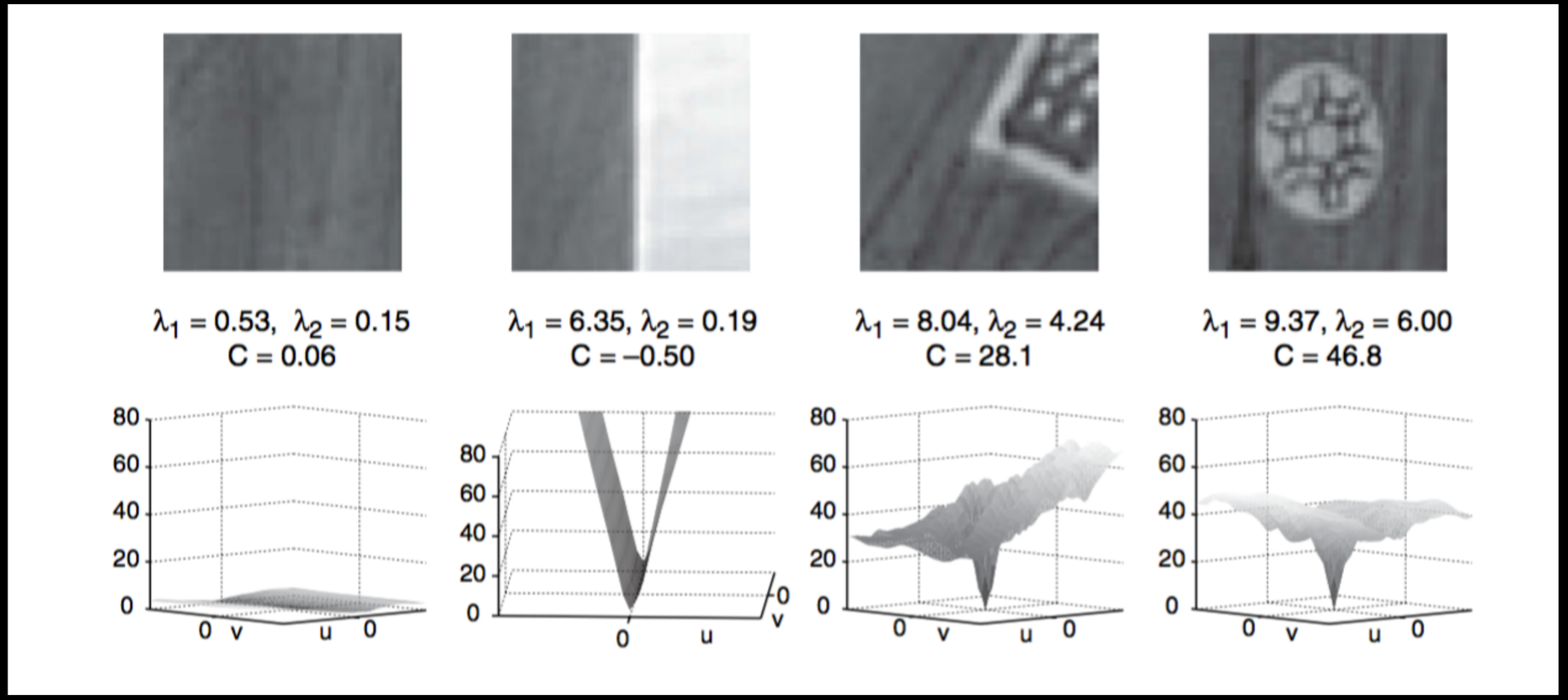
E(u,v) is small
for **some** shifts



E(u,v) is small
for **no** shifts

We want $\min_{(u,v)} E(u,v)$ to be large

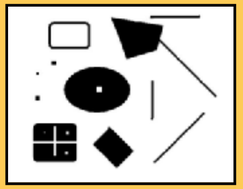
HARRIS CORNER MATH



Harris Detector [Harris88]

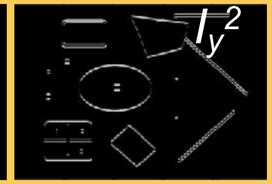
- Second moment matrix

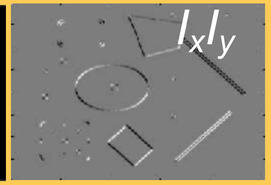
$$\mu(\sigma_I,\sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$



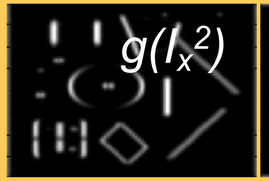
1. Image derivatives (optionally, blur first)

2. Square of derivatives





3. Gaussian filter $g(\sigma_I)$



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

4. Cornerness function – both eigenvalues are strong

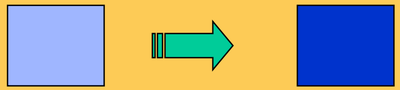
$$har = \det[\mu(\sigma_I,\sigma_D)] - \alpha[\text{trace}(\mu(\sigma_I,\sigma_D))^2] =$$
$$g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$

5. Non-maxima suppression

INVARIANCE AND COVARIANCE

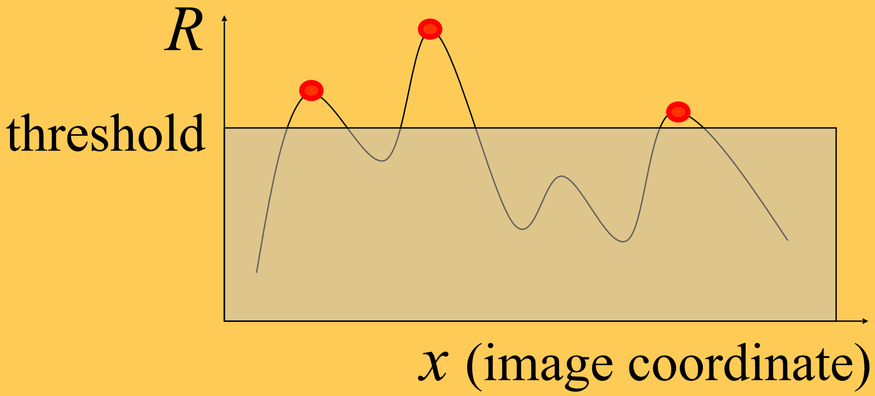
- We want corner locations to be *invariant* to photometric transformations and *covariant* to geometric transformations
  - **Invariance:** image is transformed and corner locations do not change
  - **Covariance:** if we have two transformed versions of the same image, features should be detected in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

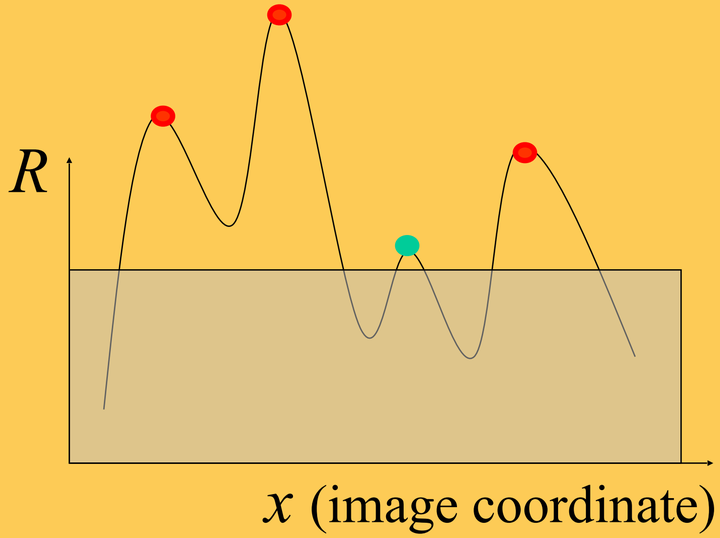
AFFINE INTENSITY CHANGE



$$I \rightarrow a\,I + b$$

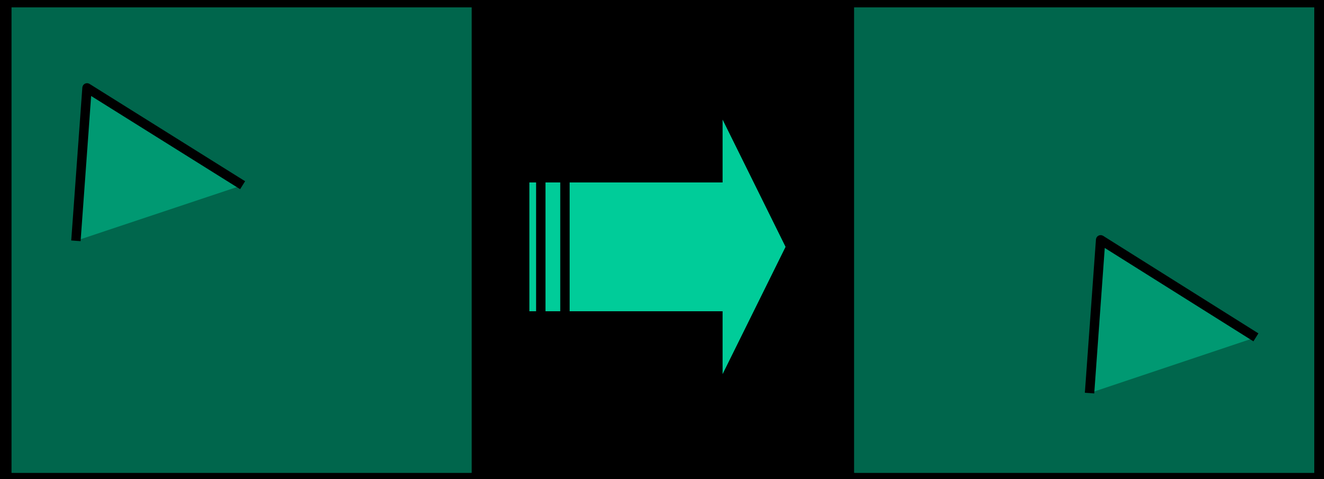
- Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$
- Intensity scaling: $I \rightarrow a\,I$





Partially invariant to affine intensity change

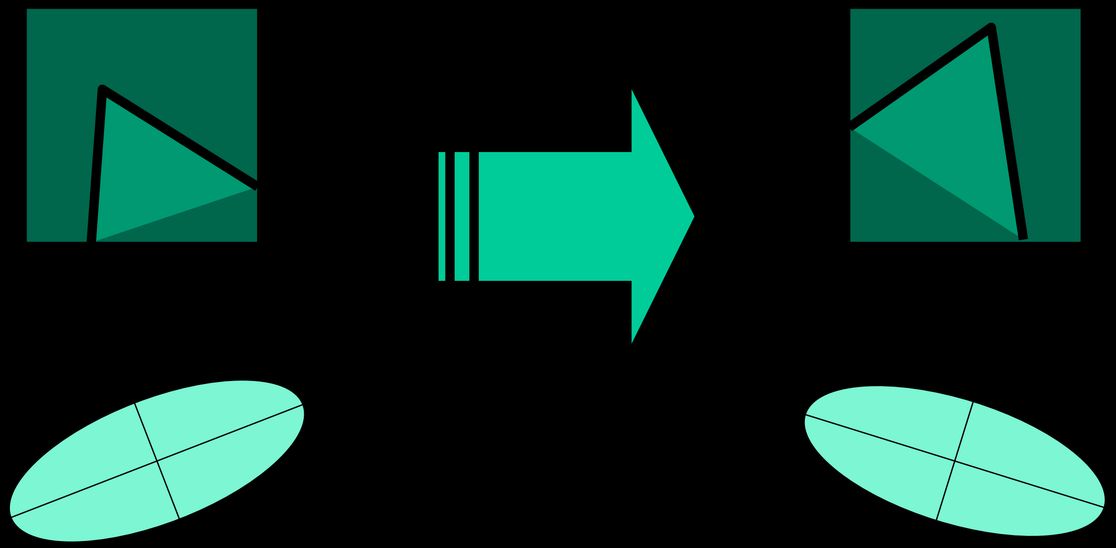
IMAGE TRANSLATION



- Derivatives and window function are shift-invariant

Corner location is covariant w.r.t. translation

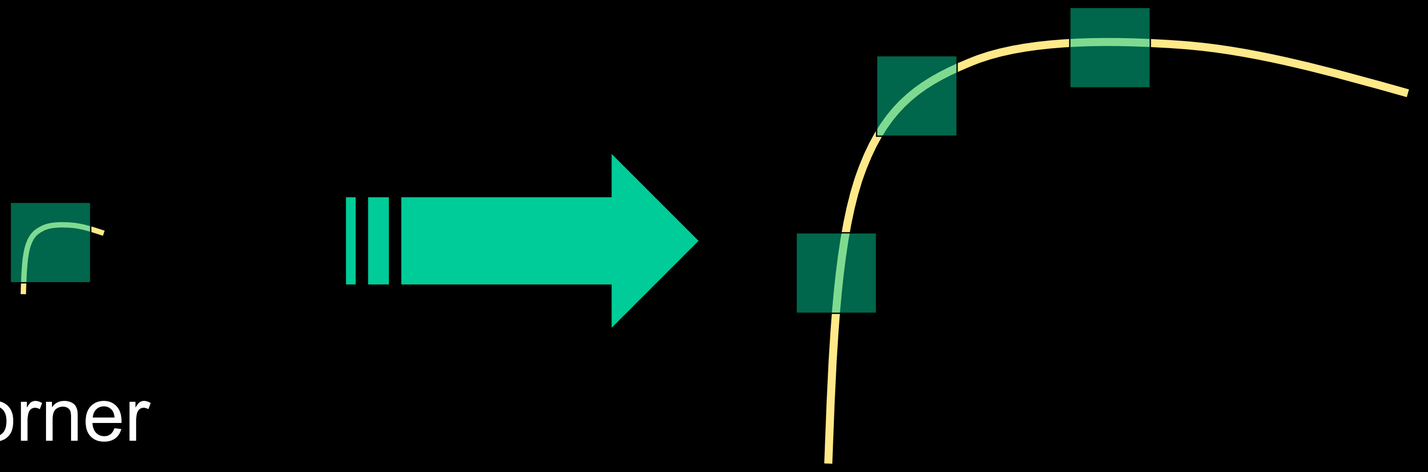
IMAGE ROTATION



Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner location is covariant w.r.t. rotation

SCALING



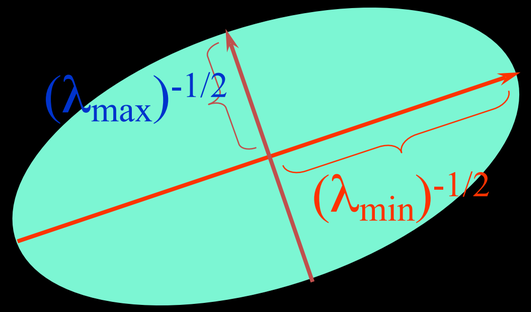
Corner

All points will be classified as edges

Corner location is not covariant to scaling!

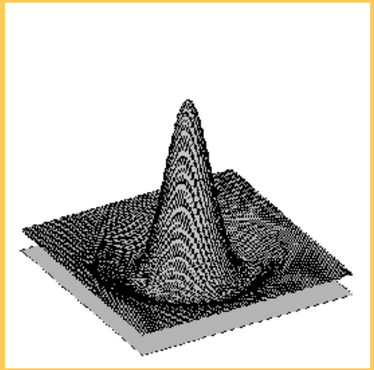
REVIEW: HARRIS CORNER DETECTOR

- Approximate distinctiveness by local auto-correlation.
- Approximate local auto-correlation by second moment matrix
- Quantify distinctiveness (or cornerness) as function of the eigenvalues of the second moment matrix.
- But we don't actually need to compute the eigenvalues by using the determinant and trace of the second moment matrix.



TEXT

FIND LOCAL MAXIMA IN POSITION–SCALE SPACE



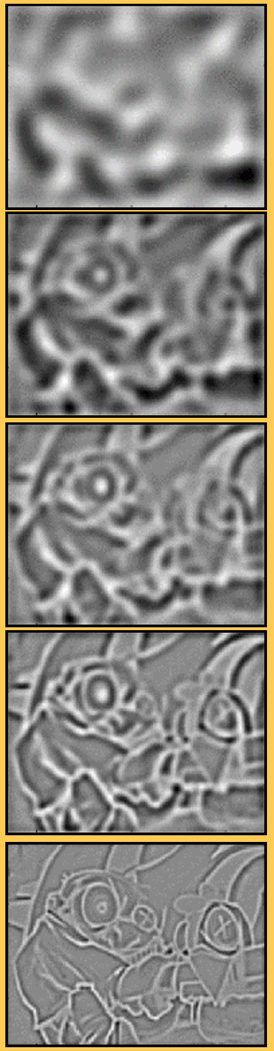
$$L_{xx}(\sigma) + L_{yy}(\sigma) \rightarrow \sigma^3$$

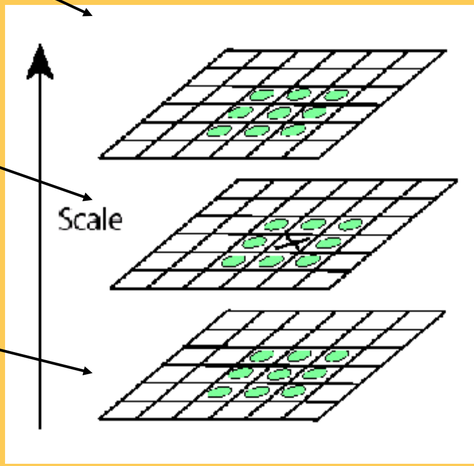
σ^5

σ^4

σ^2

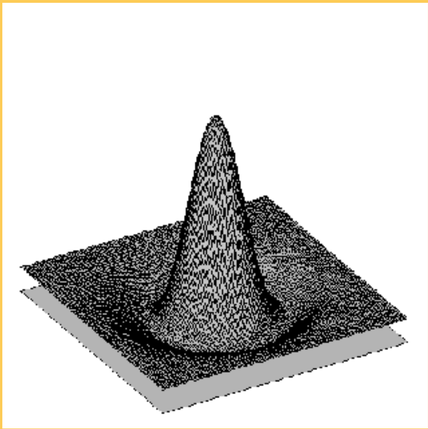
σ





⇒ **List of**
(x, y, s)

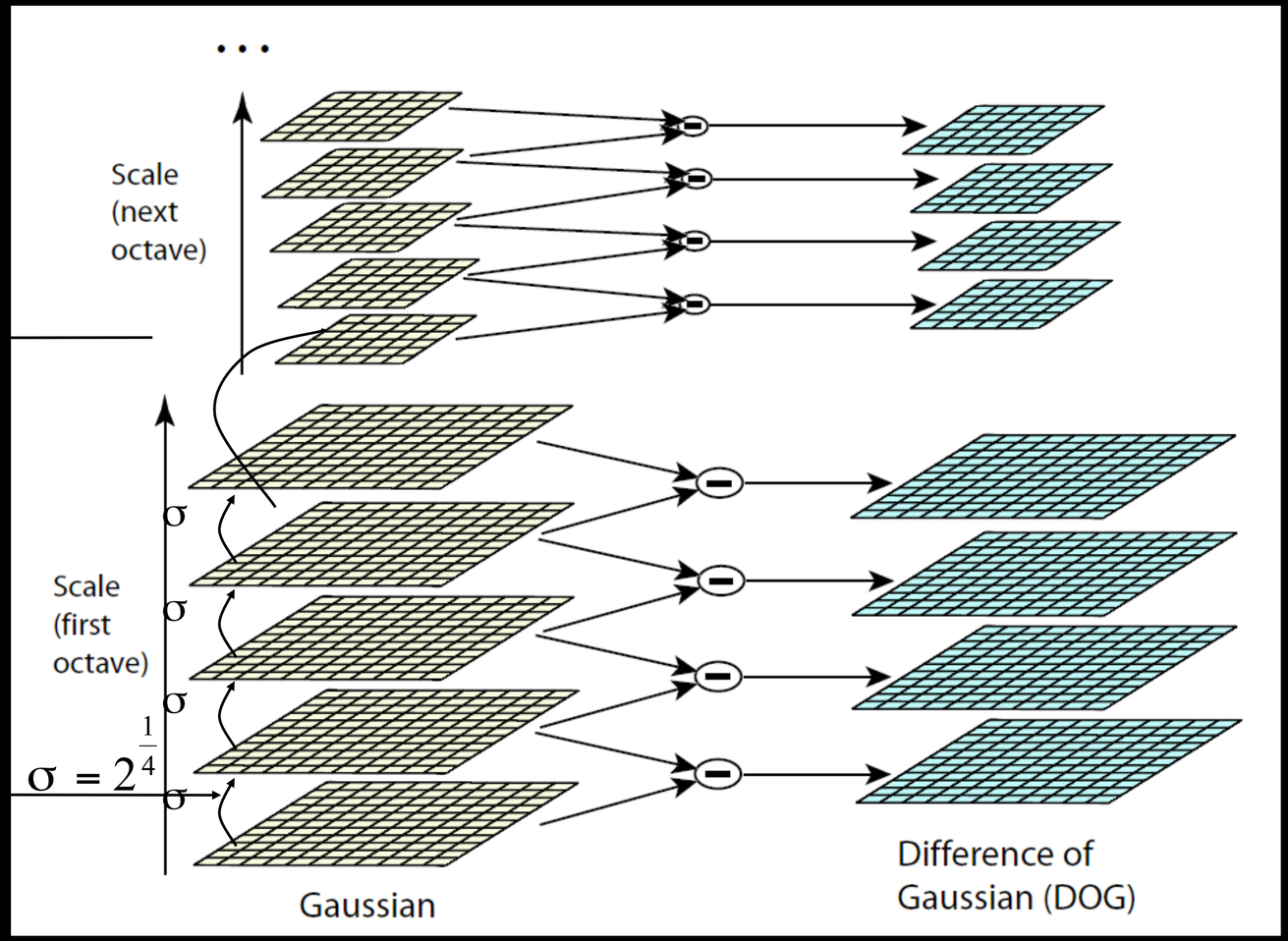
DIFFERENCE-OF-GAUSSIAN (DOG)



K. Grauman, B. Leibe

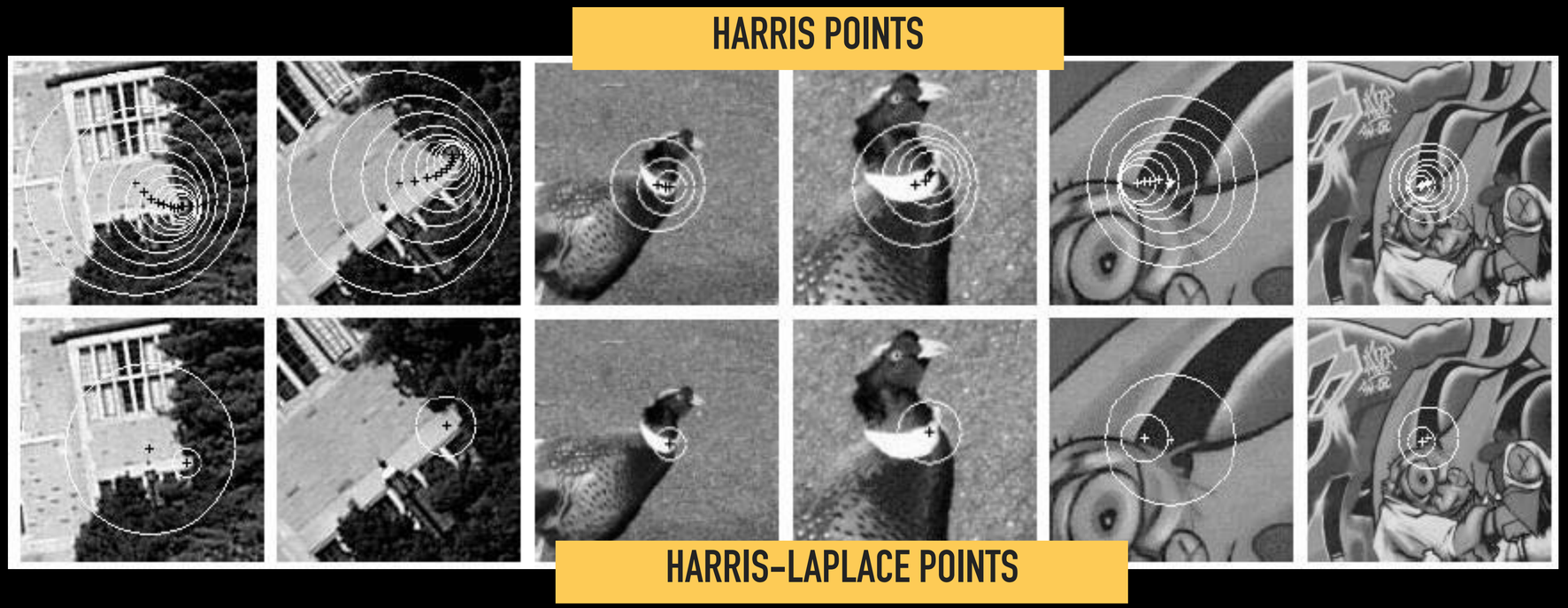
DOG – EFFICIENT COMPUTATION

- Computation in Gaussian scale pyramid



HARRIS-LAPLACE [MIKOLAJCZYK '01]

1. Initialization: Multiscale Harris corner detection
2. Scale selection based on Laplacian
   (same procedure with Hessian ⇒ Hessian-Laplace)

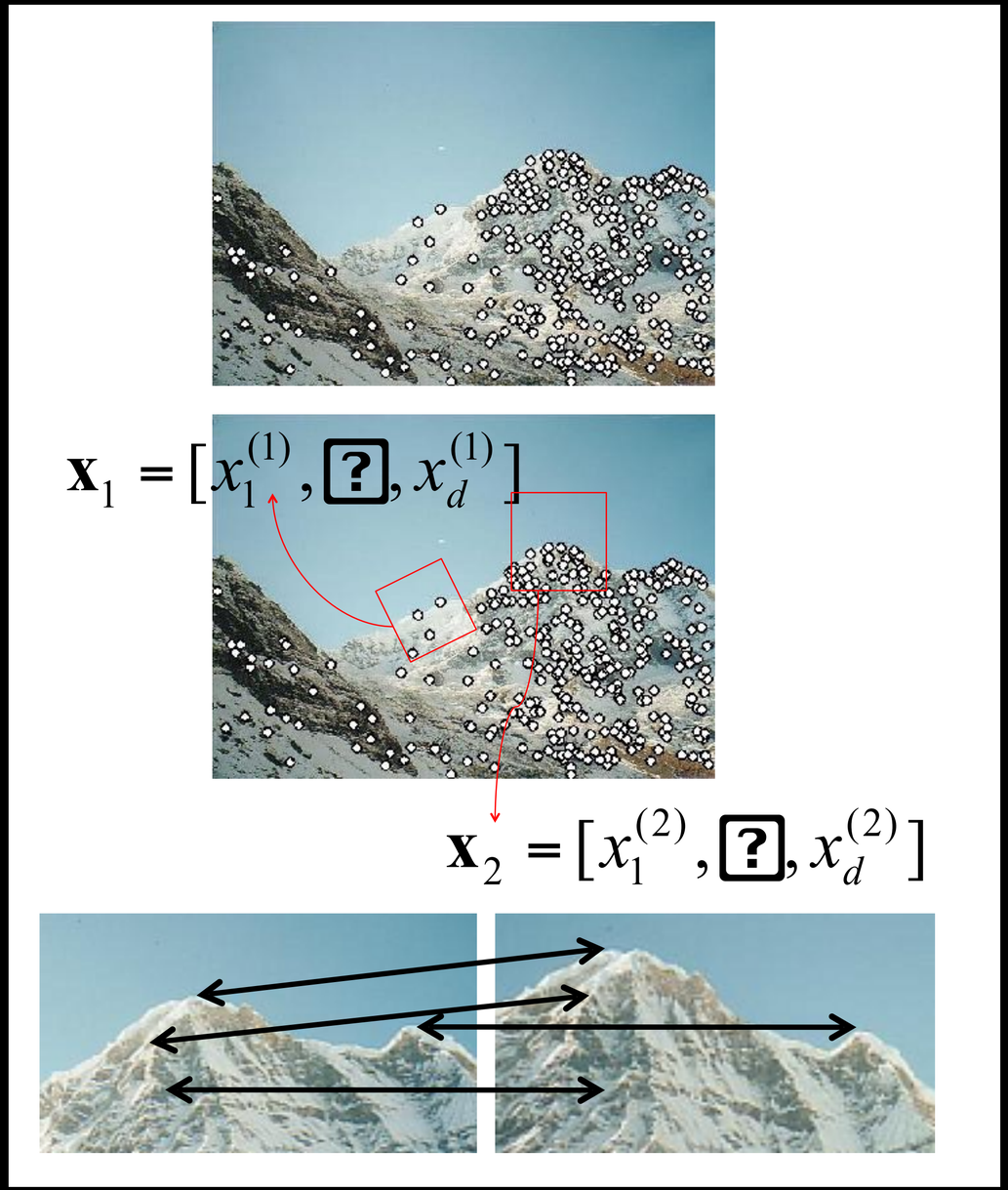


TODAY: FEATURE DESCRIPTORS

- Orientation estimation
- Feature descriptors
  - SIFT and variations
    - PCA-SIFT
    - GLOH
    - SURF
- Matching Strategy
- RANSAC

LOCAL FEATURES: MAIN COMPONENTS

- Detection: Identify the interest points
- Description: Extract vector feature descriptor surrounding each interest point.
- Matching: Determine correspondence between descriptors in two views



DESCRIPTORS

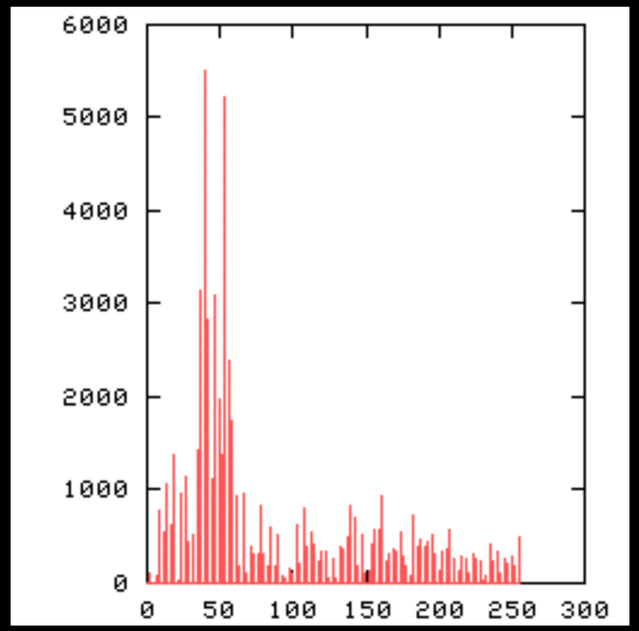
- Stereo: Patches around keypoints
  - After rectification, it’s only translations, so it’s fine
- What about changes in orientation, scale, affine transformations?
  - Rectification for those things… patches will still vary significantly.
  - Descriptors ***invariant*** but keeping ***discriminability***

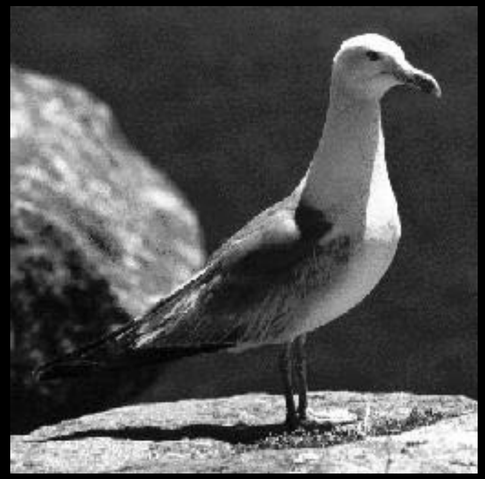
IMAGE REPRESENTATIONS

- Templates
  - Intensity, gradients, etc.
  - SDD, NCC
- Histograms
  - Color, texture, SIFT descriptors, etc.
  - Orientation, scale, affine transformations

IMAGE REPRESENTATIONS: HISTOGRAMS

- Global histogram
- Represent distribution of features
  - Color, texture, depth, …





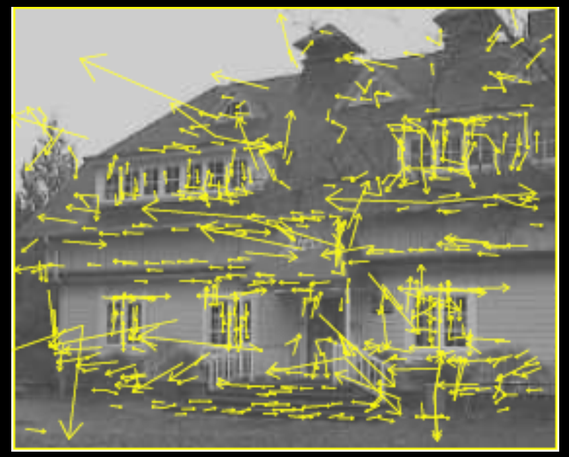
Images from Dave Kauchak

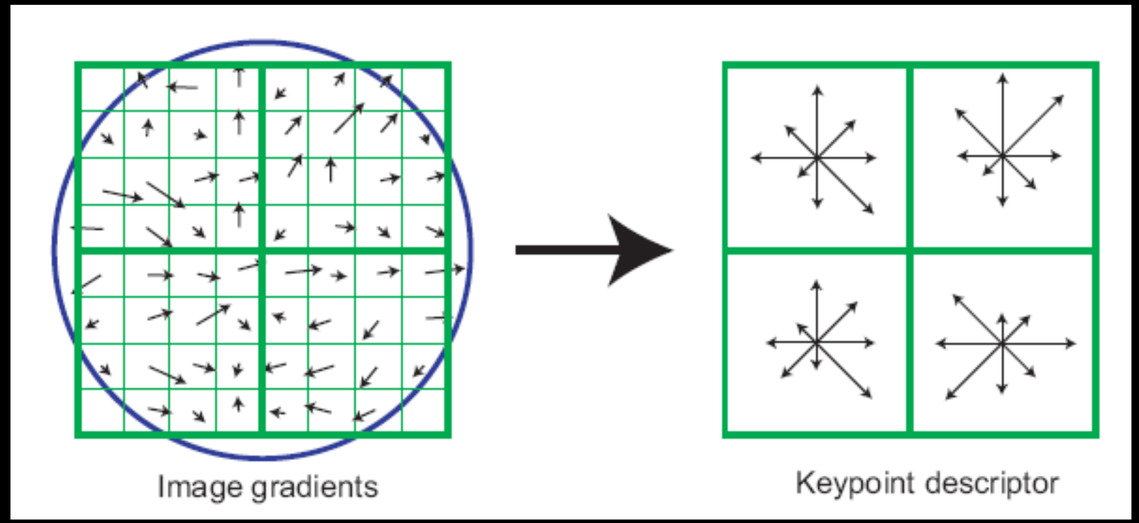
Space Shuttle Cargo Bay

SIFT

SIFT

- Histograms of oriented gradients

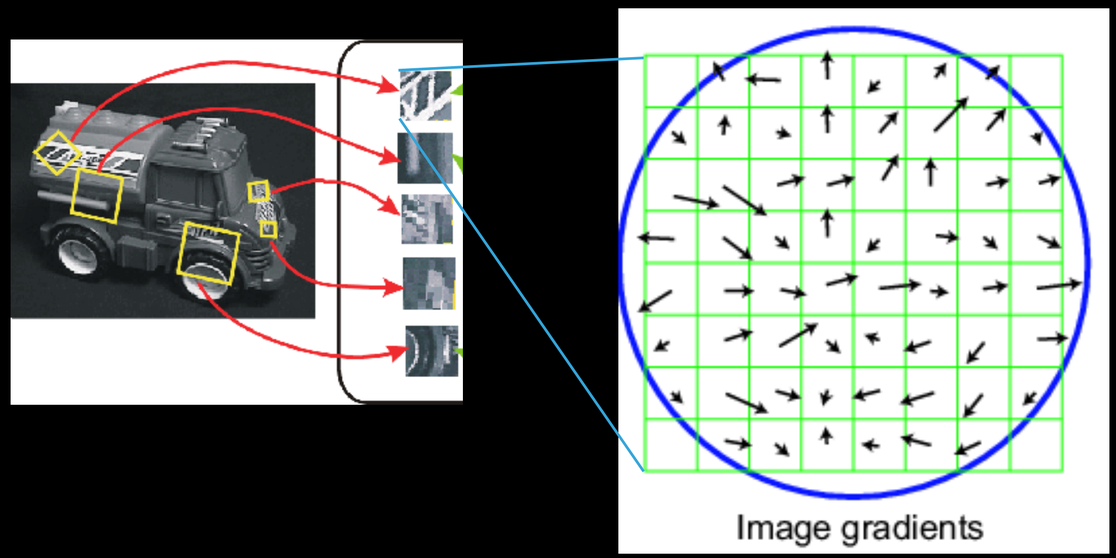




SIFT – Lowe IJCV 2004

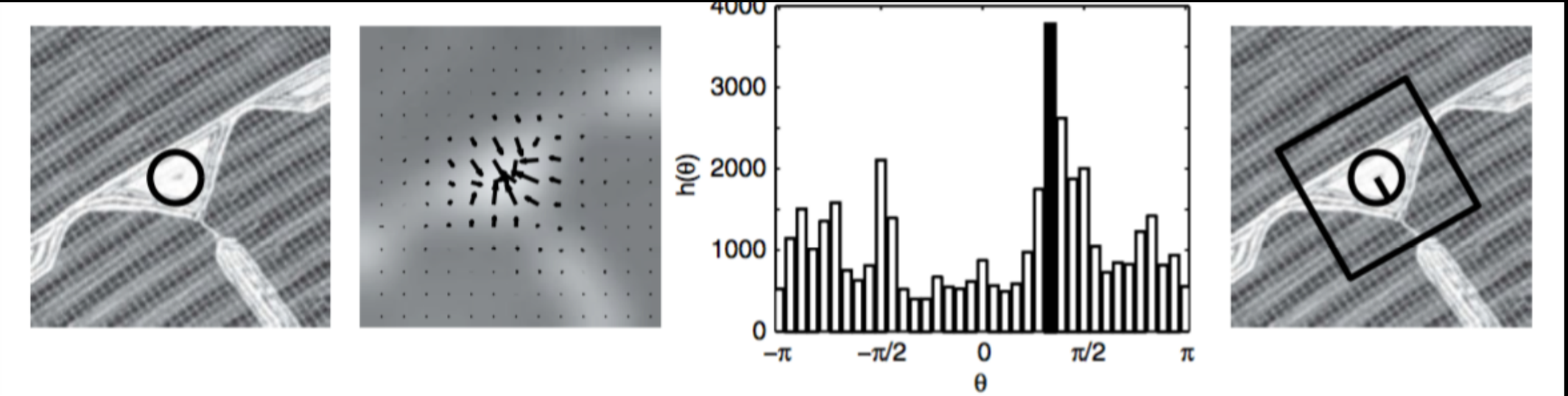
SIFT VECTOR FORMATION

- Computed on rotated and scaled version of window according to computed orientation & scale
  - resample the window
- Based on gradients weighted by a Gaussian of variance half the window (for smooth falloff)



SIFT VECTOR FORMATION

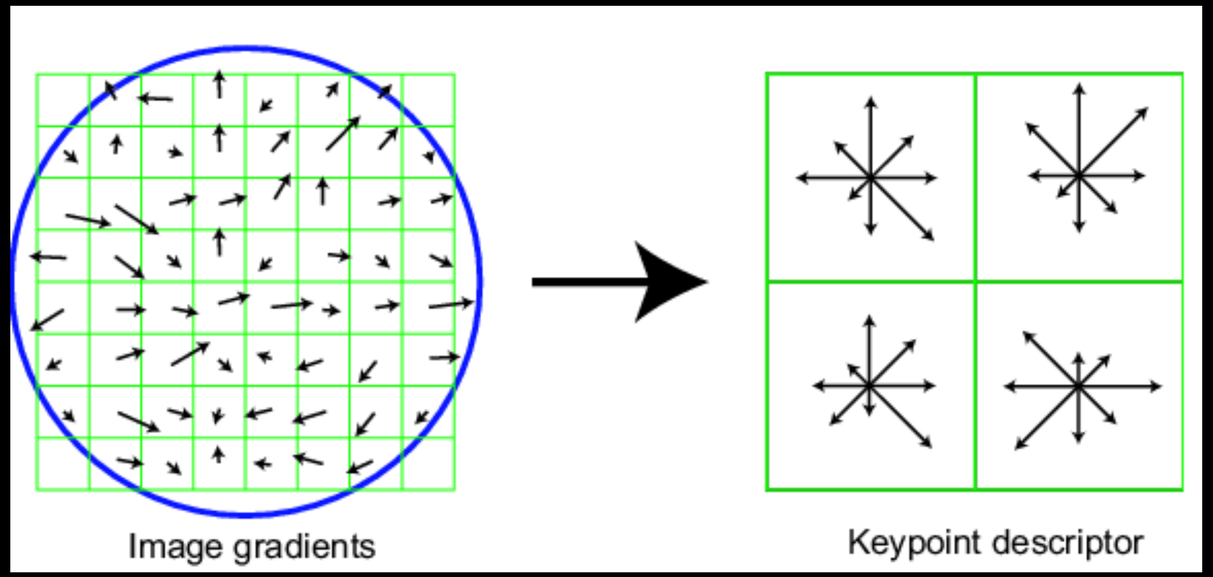
- Computed on rotated and scaled version of window according to computed orientation & scale



- 36 bins. Take maximum. If more than one peak, two keypoints are set with different orientations.

SIFT VECTOR FORMATION

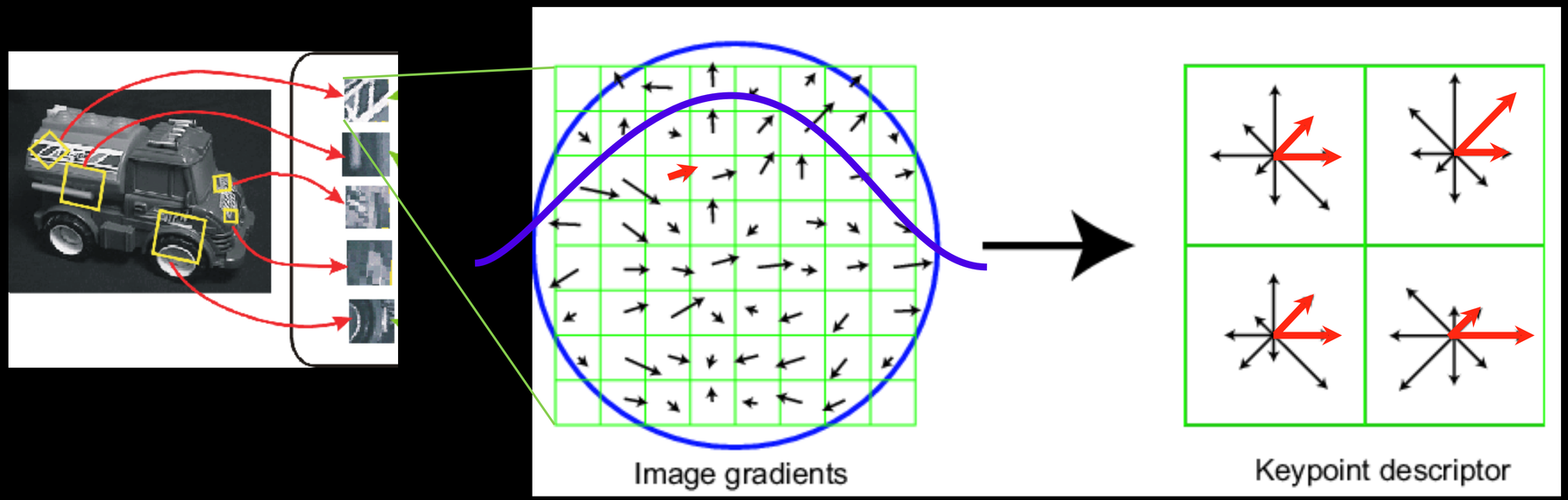
- 16 by 16 window
- 4x4 array of gradient orientation histogram weighted by magnitude
- 8 orientations x 4x4 array = 128 dimensions
- Motivation: some sensitivity to spatial layout, but not too much.



showing only 2x2 here but is 4x4

ENSURE SMOOTHNESS

- Gaussian weight
- Trilinear interpolation
  - a given gradient contributes to 8 bins: 4 in space times 2 in orientation



REDUCE EFFECT OF ILLUMINATION

- 128-dim vector normalized to 1
- Threshold gradient magnitudes to avoid excessive influence of high gradients
  - after normalization, clamp gradients >0.2
  - renormalize

SUMMARY SIFT

- Use DoG to get keypoints and scale
- Take a region around the point and compute dominant orientation, (mode of histogram of directions)
- Sample 16x16 pixels at the detected scale and orientation
- Compute 4x4 grid of histograms of gradient directions using 8 bins = 128 dimensional feature descriptor
  - Sum of magnitudes.
  - Scaled by a gaussian
- Normalize, clip, re-normalize

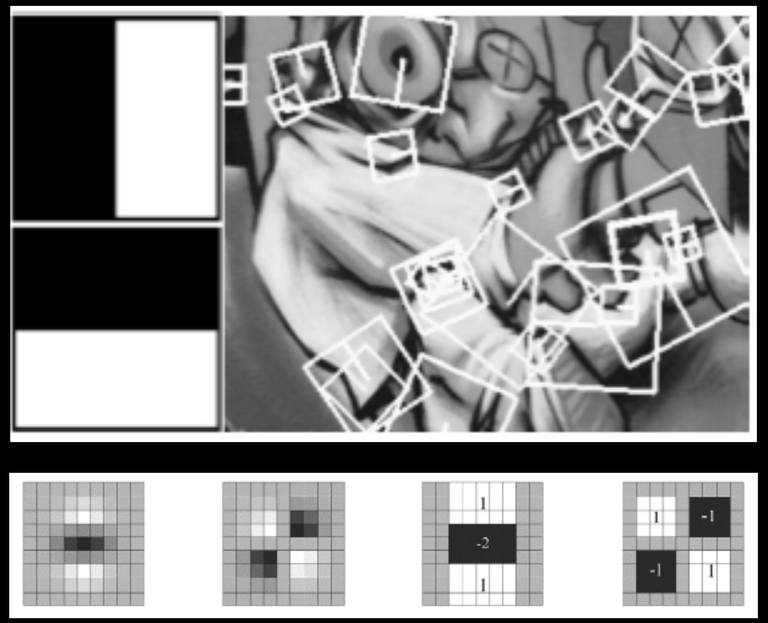
VARIATIONS OF SIFT

PCA-SIFT

- Takes 39x39 patch of gradients around the keypoint.
- Uses PCA learned from a large databased to reduce the 3042-dimensions to 36

LOCAL DESCRIPTORS: SURF

- Fast approximation of SIFT idea
- Efficient computation by 2D box filters & integral images
  ⇒ 6 times faster than SIFT
- Equivalent quality for object identification



[Bay, ECCV'06], [Cornelis, CVGPU'08]

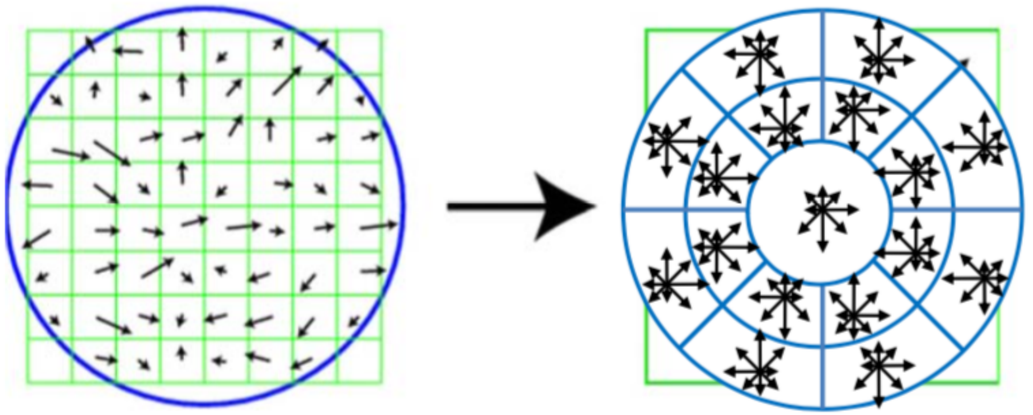
GPU implementation available

Feature extraction @ 200Hz
(detector + descriptor, 640×480 img)

http://www.vision.ee.ethz.ch/~surf

GLOH

- 3 circles, 8 regions per circle, 16 direction bins = 272
- reduced to 128 using PCA from a large descriptor database



(a) image gradients (b) keypoint descriptor

Figure 4.19 The gradient location-orientation histogram (GLOH) descriptor uses log-polar bins instead of square bins to compute orientation histograms (Mikolajczyk and Schmid 2005).

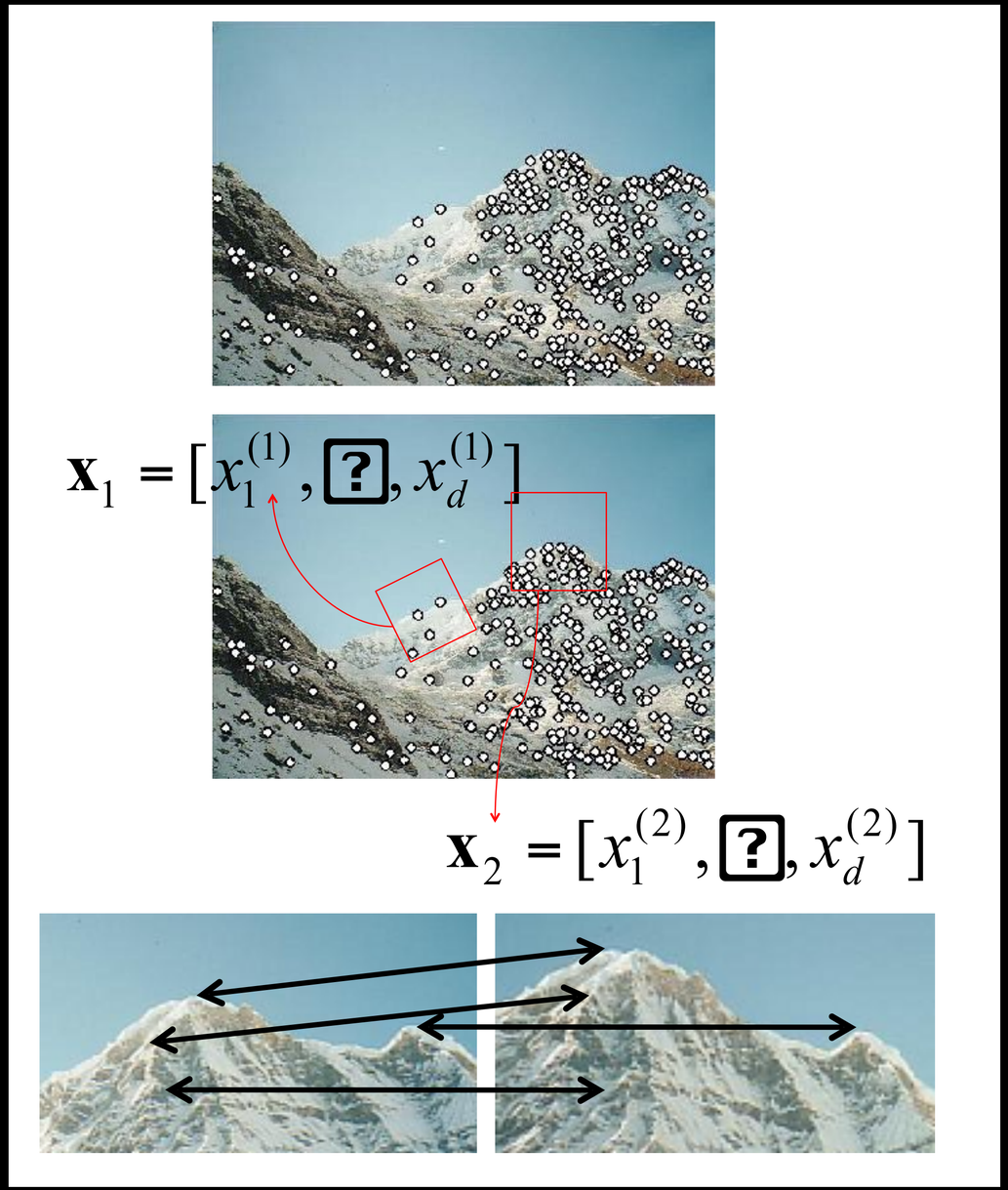
OTHER DESCRIPTORS

LOCAL DESCRIPTORS

- Most features can be thought of as templates, histograms (counts), or combinations
- The ideal descriptor should be
  - Robust
  - Distinctive
  - Compact
  - Efficient
- Most available descriptors focus on edge/gradient information
  - Capture texture information
  - Color rarely used

LOCAL FEATURES: MAIN COMPONENTS

- Detection: Identify the interest points
- Description: Extract vector feature descriptor surrounding each interest point.
- Matching: Determine correspondence between descriptors in two views



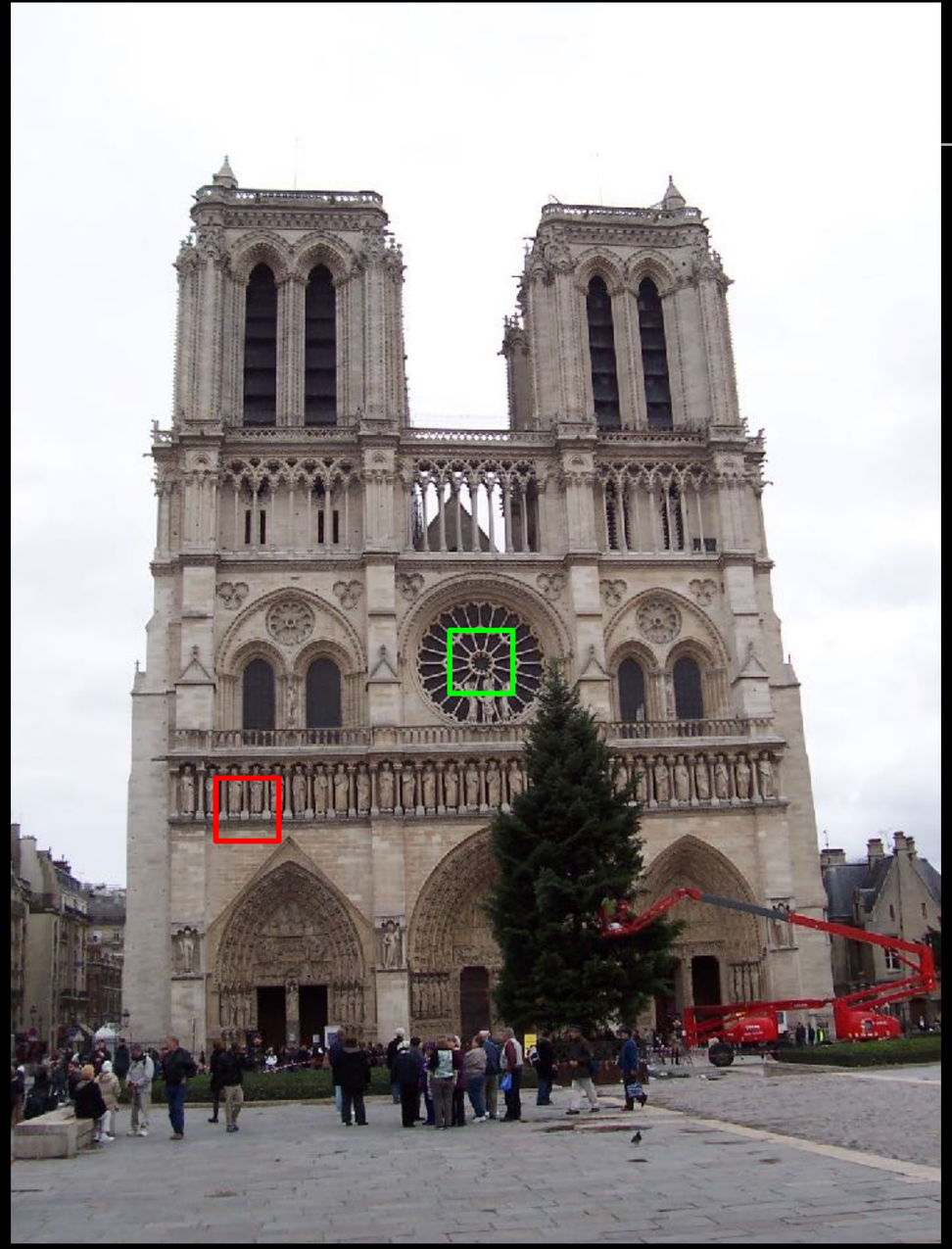
MATCHING

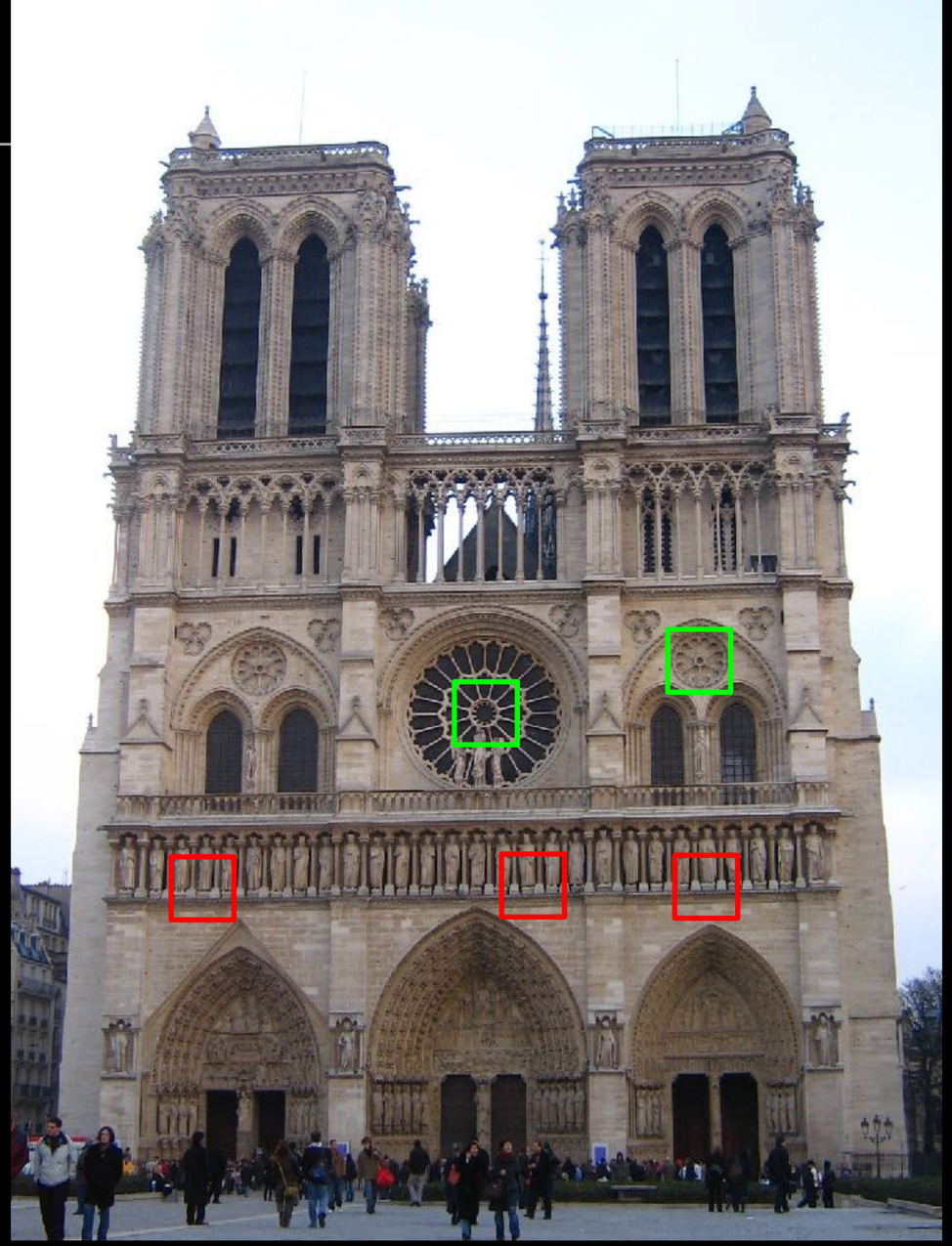
FEATURE MATCHING

- For a feature in Image1 how to find the best match in Image2?
  - Define distance function that compares two descriptors
  - Test all the features in Image2, find the one with min distance

FEATURE MATCHING

- distances in feature space can be directly used for ranking
- Euclidean distance should be meaningful
- If some features have more importance, they should be scaled first





Distance: 0.34, 0.30, 0.40

Distance: 0.61

Distance: 1.22

FEATURE MATCHING

- Let’s assumes SDD = |f1 - f2|^2

FEATURE MATCHING

- Fails in ambiguous cases
- ideas?

FEATURE MATCHING

- Ratio distance = SSD(f1, f2) / SSD(f1, f2’)
  - f2 is best SSD match to f1 in Image2
  - f2’ is 2nd best SSD match to f1 in Image2

The value is in the range [0, 1.0]

NEAREST NEIGHBOR DISTANCE RATIO

- NN1/NN2
- where NN1 is the distance to the first nearest neighbor and NN2 is the distance to the second nearest neighbor
- Sorting by this ration puts matches in order of confidence

MATCHING

- Simplest approach: Pick the nearest neighbor. Threshold on absolute distance
- Problem: Lots of self similarity in many photos
- Use ratio for confidence

EVALUATING MATCHING PERFORMANCE

- Throw out features with distance > threshold
- The 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 MATCHING PERFORMANCE

	We find a match	We did not find a match
Correct match	True positive (TP)	False negative (FN)
Incorrect match	False positive (FP)	True negative (TN)

- what is TP + FN?
- what is FP + TN?

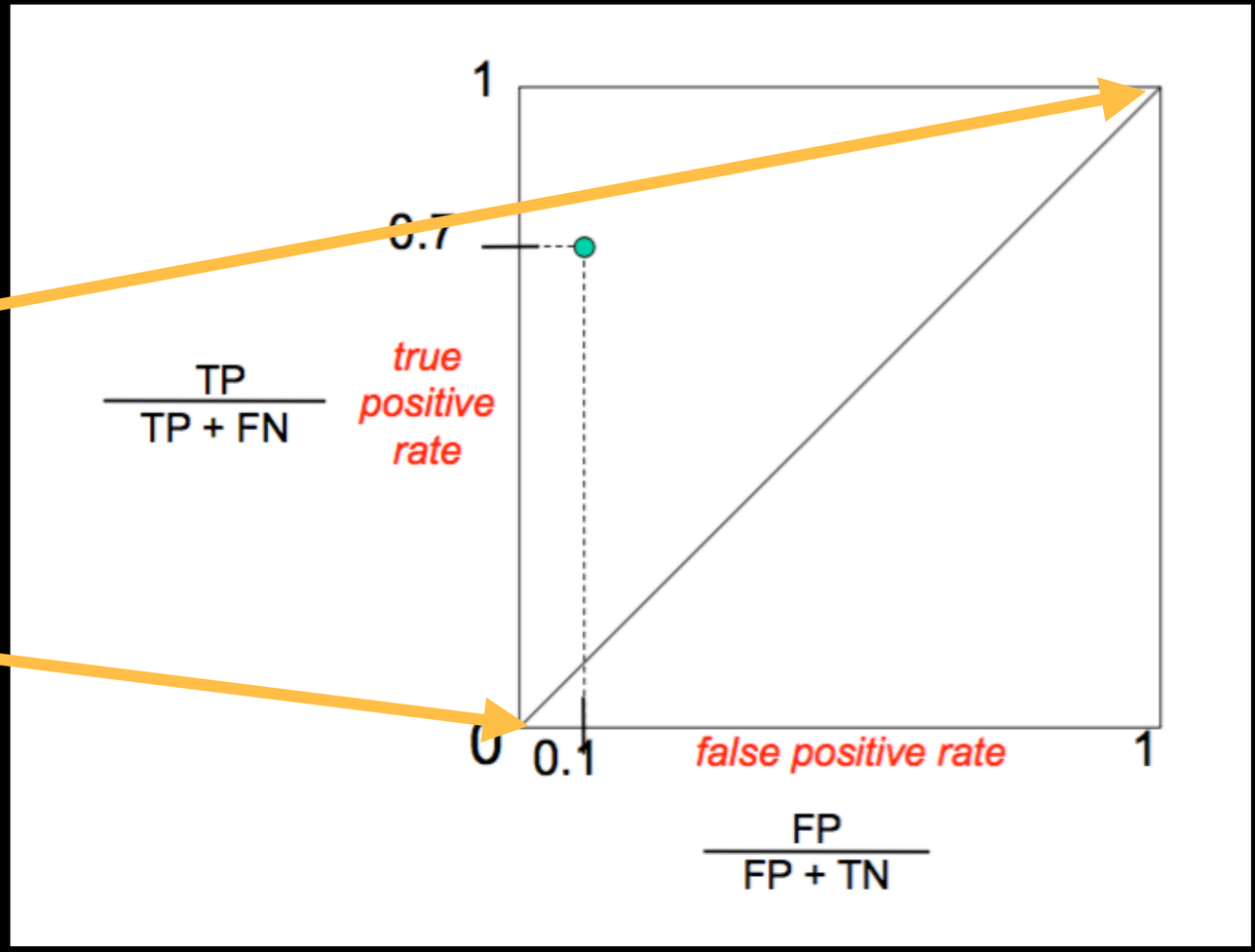
EVALUATING MATCHING PERFORMANCE

- True positive rate: RECALL
  - TPR = TP / TP + FN = TP / P
- False positive rate
  - FPR = FP / FP + TN = TP / N
- positive predictive value: PRECISION
  - PPV = TP / TP + FP = TP / P’
- Accuracy
  - TRP = TP + TN / P+N

EVALUATING MATCHING PERFORMANCE

Threshold = 1: Accept Everything

Threshold = 0: Reject Everything



TEXT

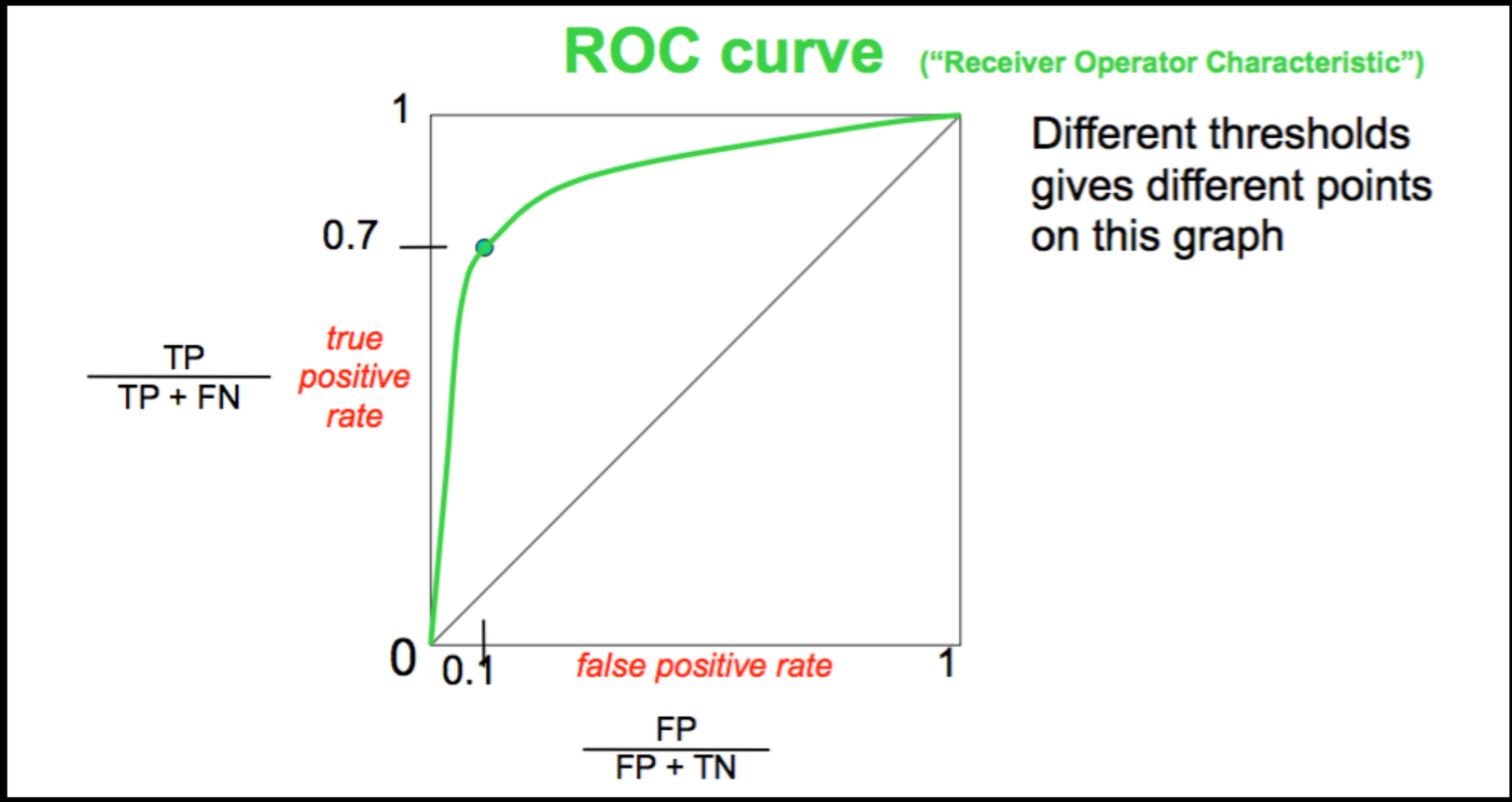
EVALUATING MATCHING PERFORMANCE

ROC Curves

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



Recall close to 1

FEATURE MATCHING

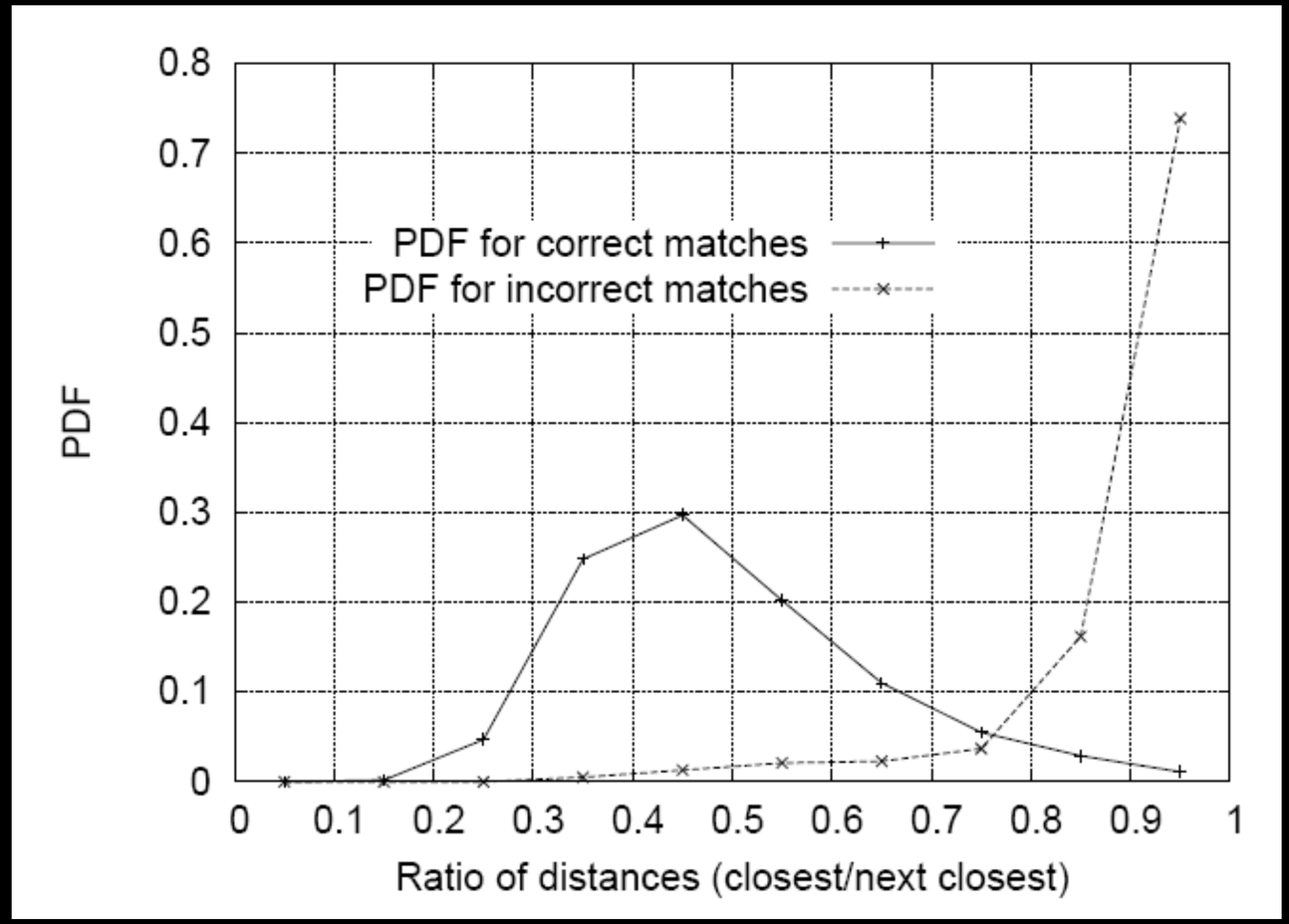
- Simple criteria: One feature matches to another if those features are nearest neighbours and their distance is below some threshold.
- Problems:
  - Ideally, threshold adapts to different regions of the feature space using training data
  - If not possible, the ratio with the second NN is a good heuristic

FEATURE MATCHING

- Matching Strategy
  - determines which correspondences are passed to next stage
    - threshold
    - nearest neighbor
    - NN distance ratio
- Data Structures and Algorithms
  - perform the matching as quickly as possible

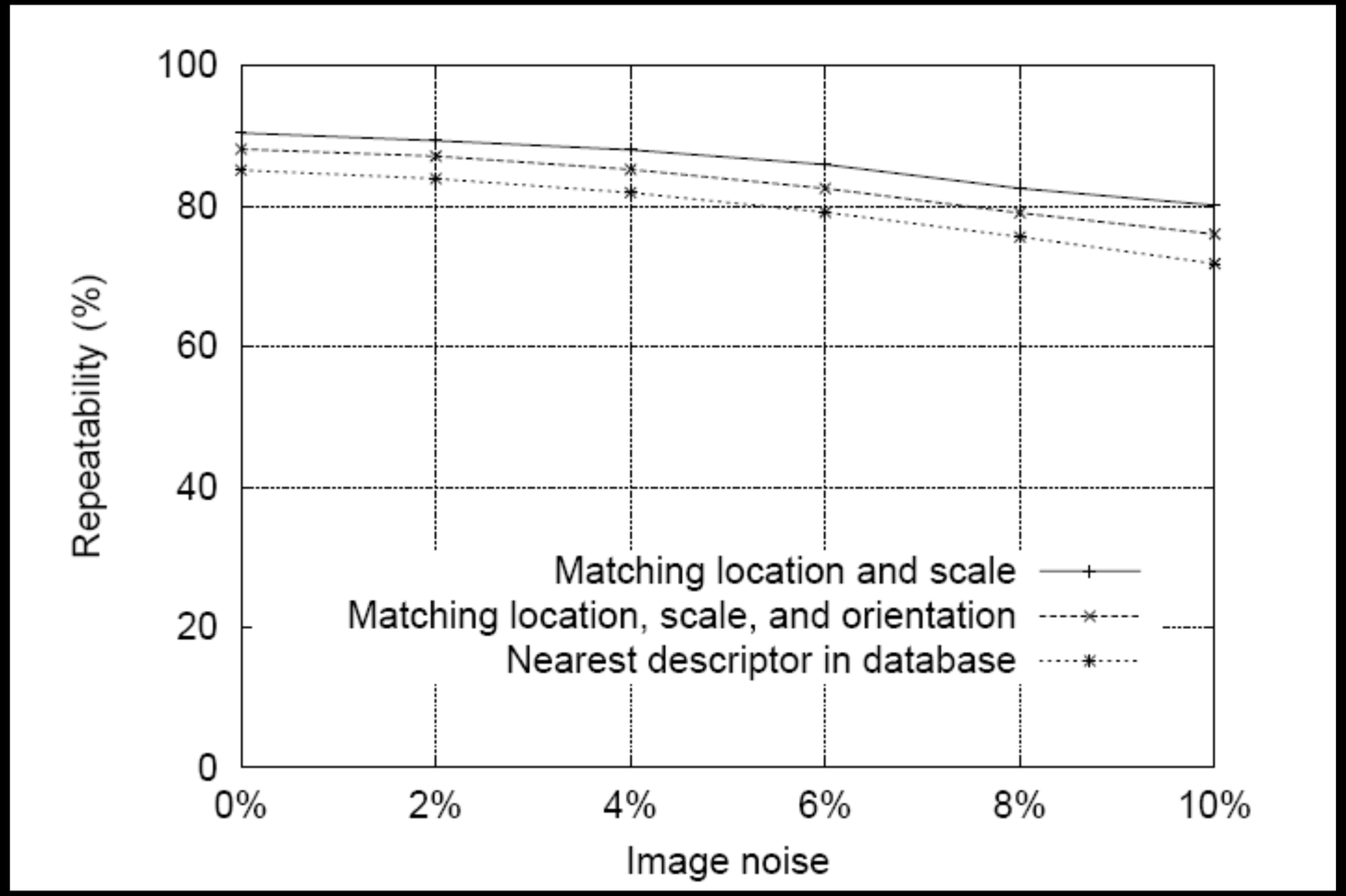
MATCHING LOCAL FEATURES

- Nearest neighbor (Euclidean distance)
- Threshold ratio of nearest to 2nd nearest descriptor
- 0.8 rejects 90% of false matches



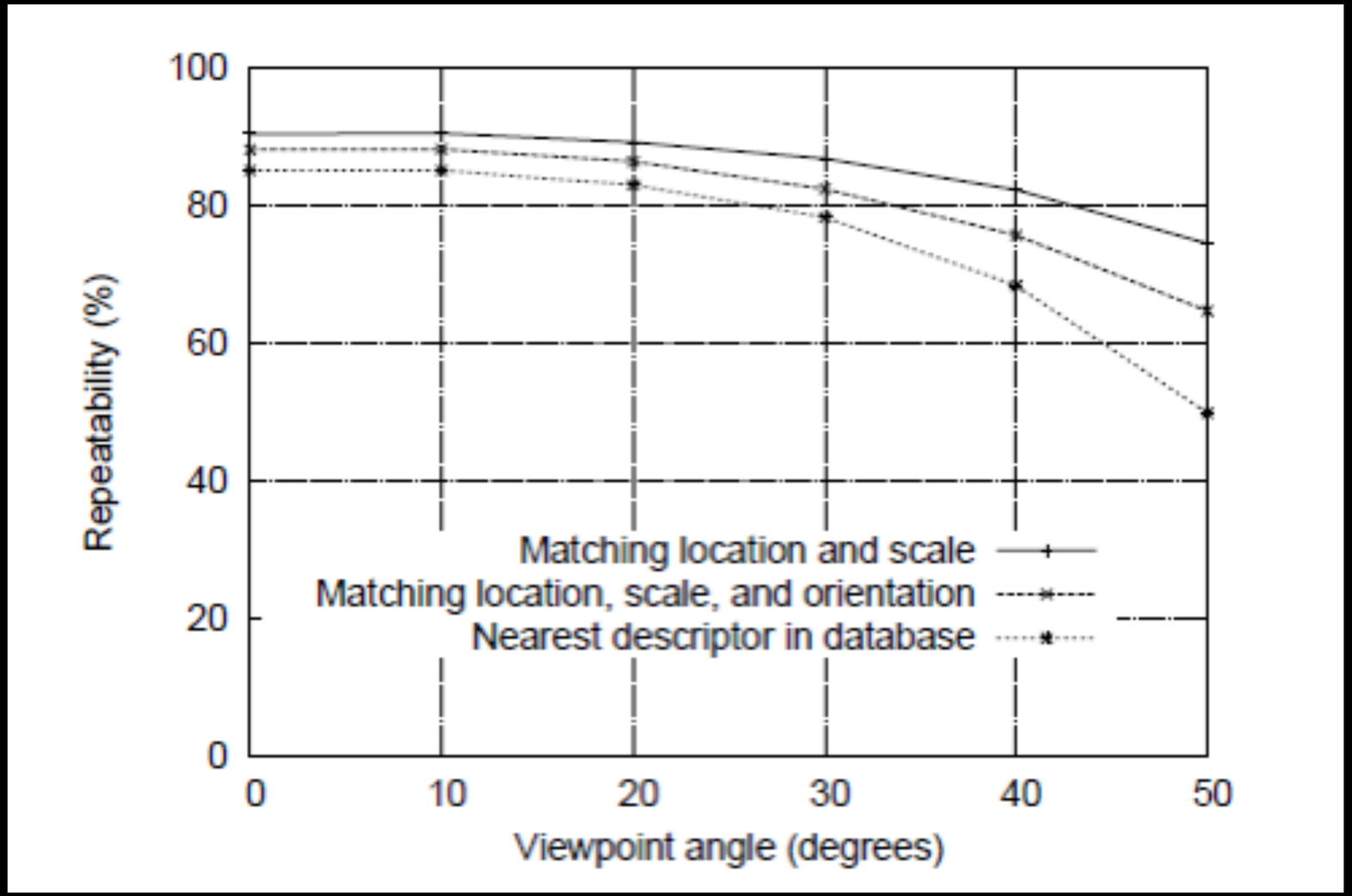
LOWE IJCV 2004

SIFT REPEATABILITY

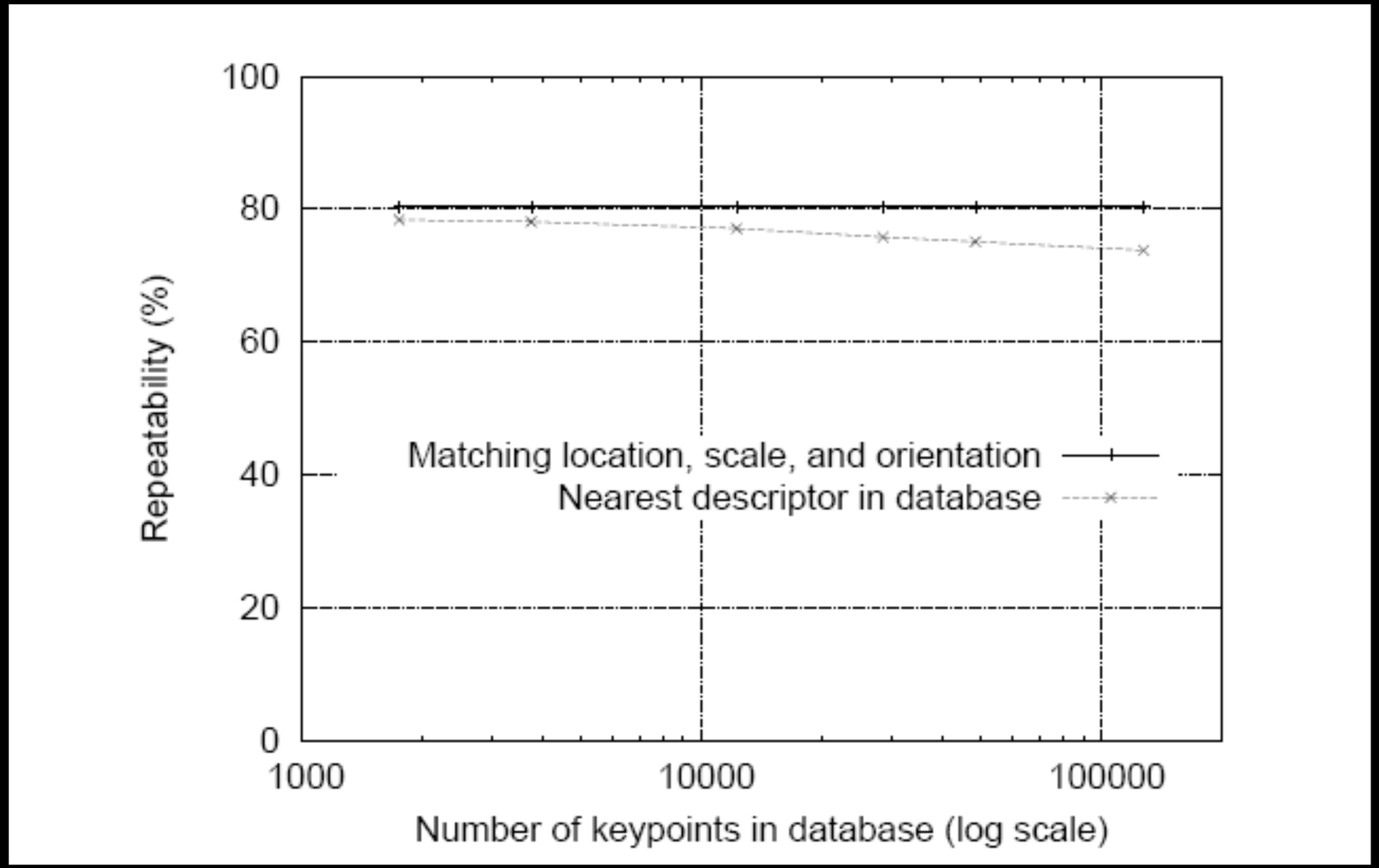


LOWE IJCV 2004

SIFT REPEATABILITY

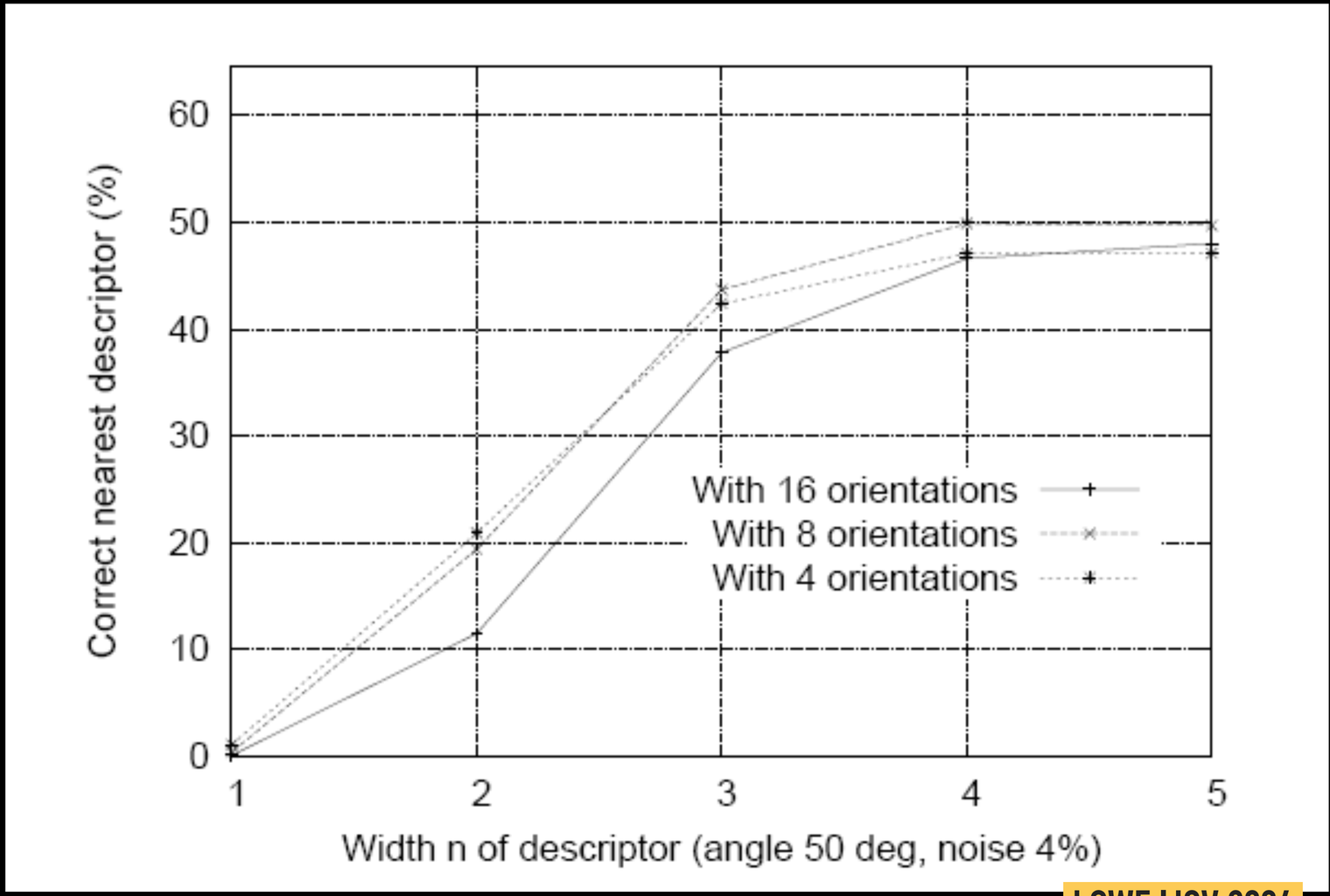


SIFT REPEATABILITY



LOWE IJCV 2004

SIFT REPEATABILITY



LOWE IJCV 2004

CHOOSING A DETECTOR

- What do you want it for?
  - Precise localization in x-y: Harris
  - Good localization in scale: Difference of Gaussian
  - Flexible region shape: MSER
- Best choice often application dependent
  - Harris-/Hessian-Laplace/DoG work well for many natural categories
  - MSER works well for buildings and printed things
- Why choose?
  - Get more points with more detectors
- There have been extensive evaluations/comparisons
  - [Mikolajczyk et al., IJCV'05, PAMI'05]
  - All detectors/descriptors shown here work well

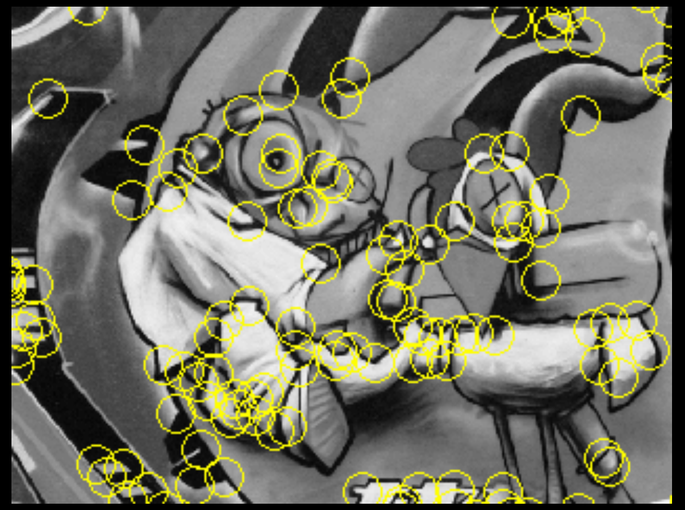
COMPARISON OF KEYPOINT DETECTORS

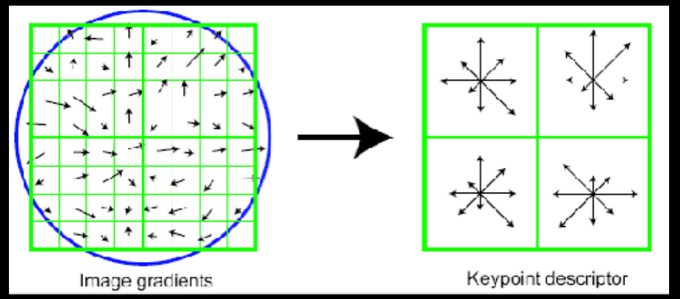
Table 7.1 Overview of feature detectors.

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			√	√	(√)	(√)	+	+	+	+

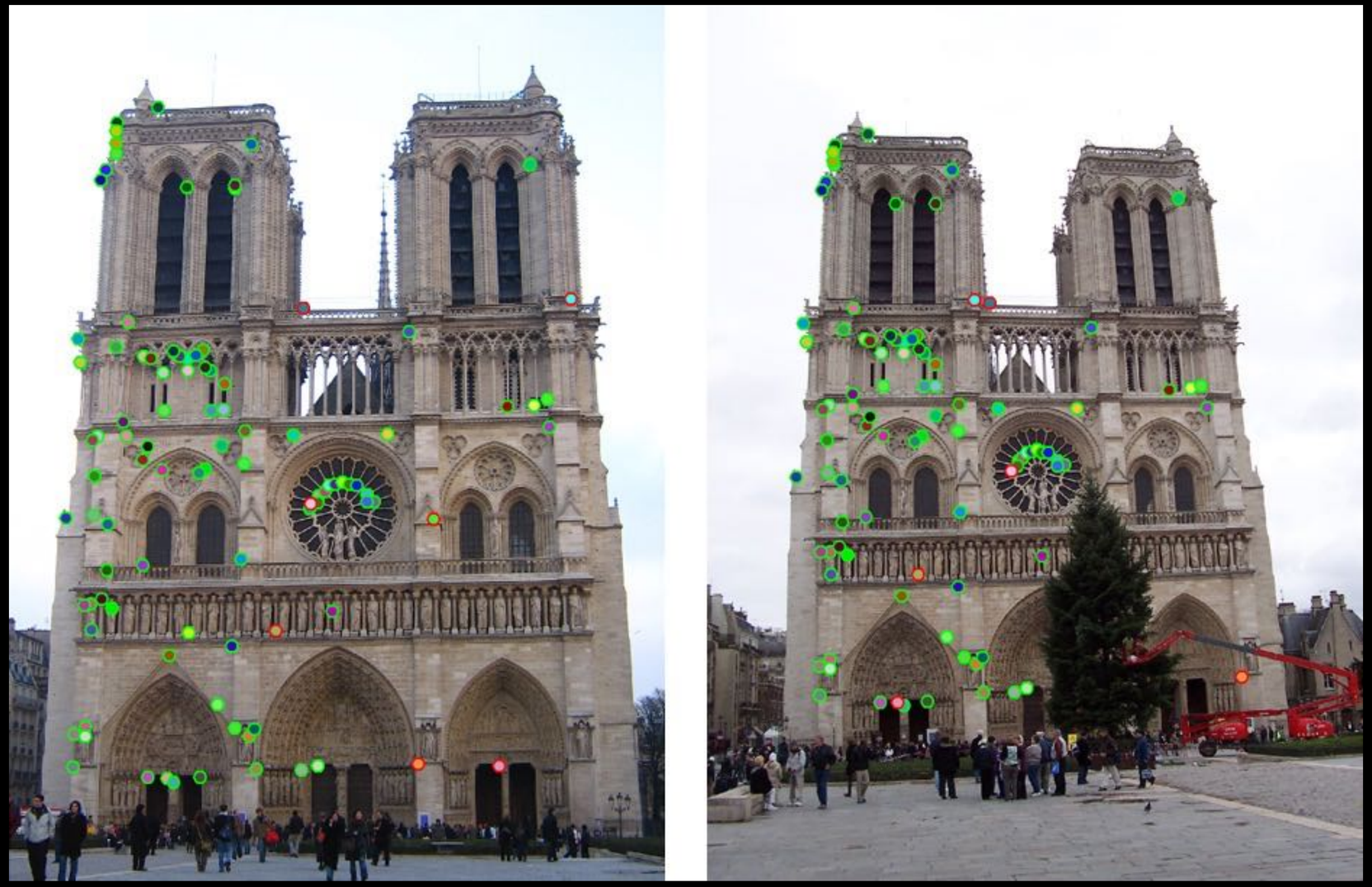
THINGS TO REMEMBER

- Keypoint detection: repeatable and distinctive
  - Corners, blobs, stable regions
  - Harris, DoG
- Descriptors: robust and selective
  - spatial histograms of orientation
  - SIFT





HOW DO WE DECIDE WHICH FEATURES MATCH?



HOW DO WE DECIDE WHICH FEATURES MATCH?

- Fitting: find the parameters of a model that best fit the data
- Alignment: find the parameters of the transformation that best align matched points

FITTING AND ALIGNMENT

- Design challenges
  - Design a suitable goodness of fit measure
    - Similarity should reflect application goals
    - Encode robustness to outliers and noise
  - Design an optimization method
    - Avoid local optima
    - Find best parameters quickly

FITTING AND ALIGNMENT: METHODS

- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Iterative closest point (ICP)
- Hypothesize and test
  - Generalized Hough transform
  - RANSAC

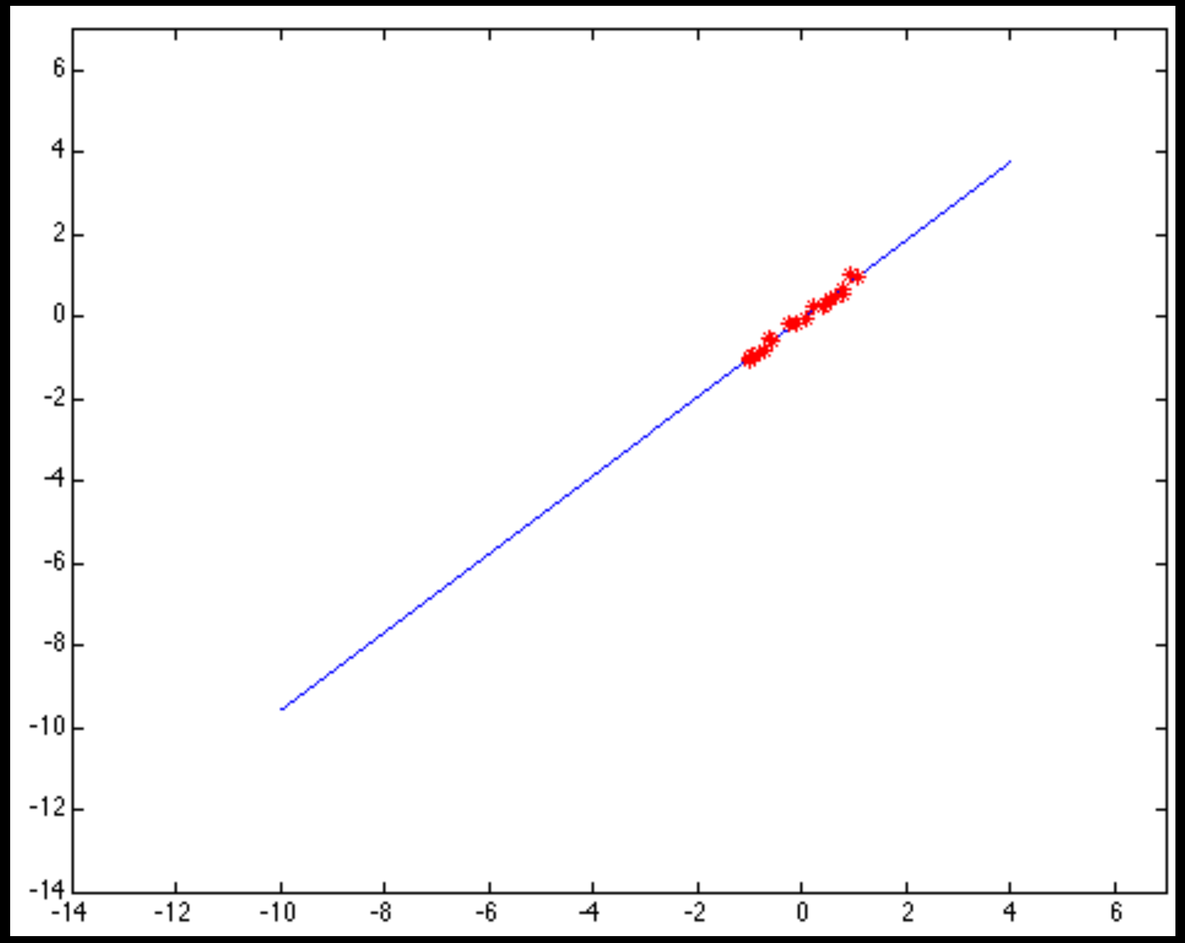
SIMPLE EXAMPLE: FITTING A LINE

LEAST SQUARES (GLOBAL) OPTIMIZATION

- Good
  - Clearly specified objective
  - Optimization is easy
- Bad
  - May not be what you want to optimize
  - Sensitive to outliers
  - Bad matches, extra points
  - Doesn’t allow you to get multiple good fits
  - Detecting multiple objects, lines, etc.

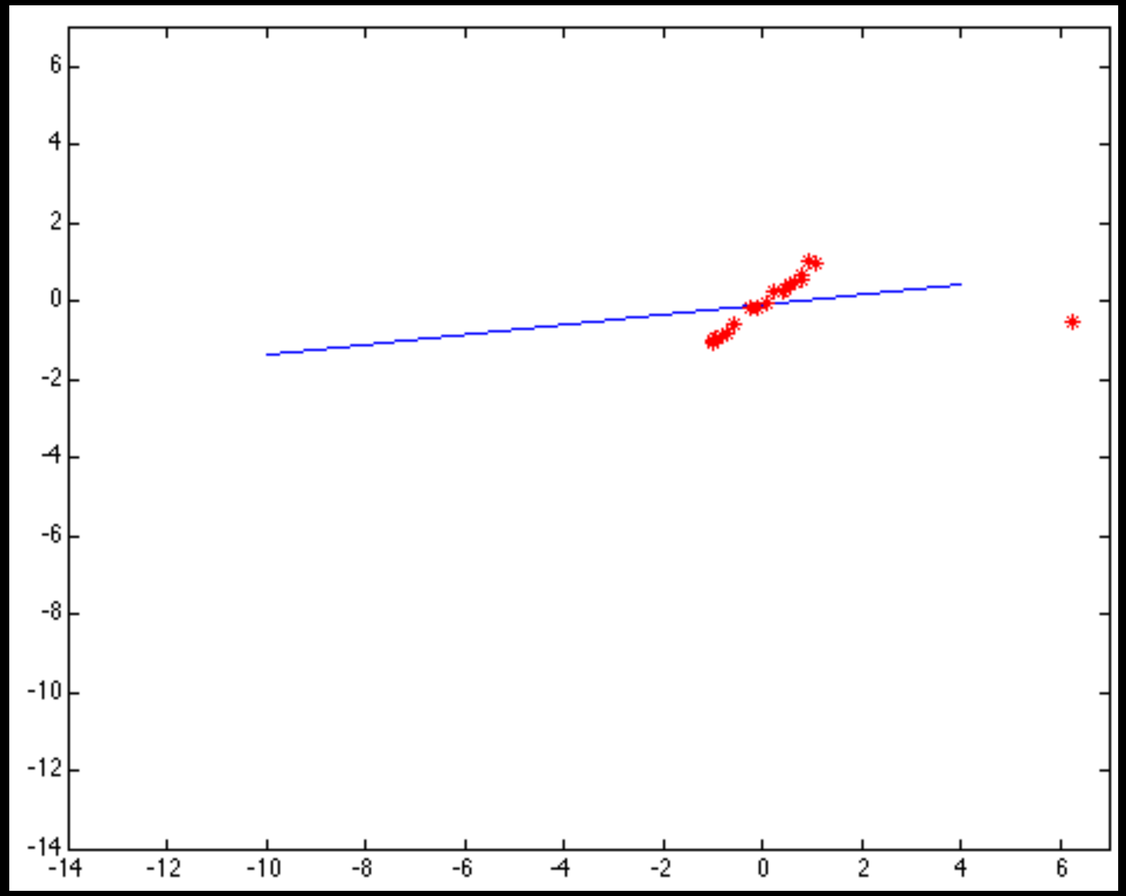
LEAST SQUARES: ROBUSTNESS TO NOISE

- Least squares fit to the red points:



LEAST SQUARES: ROBUSTNESS TO NOISE

- Least squares fit with an outlier:
- Problem: squared error heavily penalizes outliers



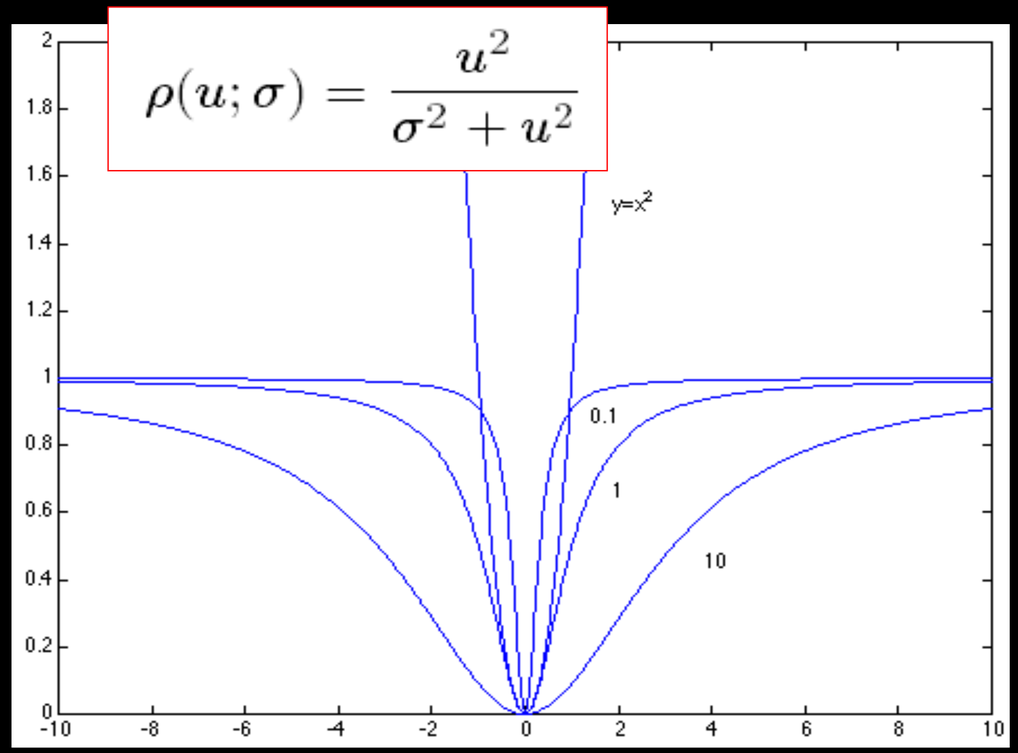
Robust least squares (to deal with outliers)

General approach:

minimize $\sum_i \rho\left(u_i\left(x_i, \theta\right), \sigma\right)$ $\qquad u^2 = \sum_{i=1}^{n} (y_i - mx_i - b)^2$

$u_i(x_i, \theta)$ – residual of ith point w.r.t. model parameters θ

ρ – robust function with scale parameter σ

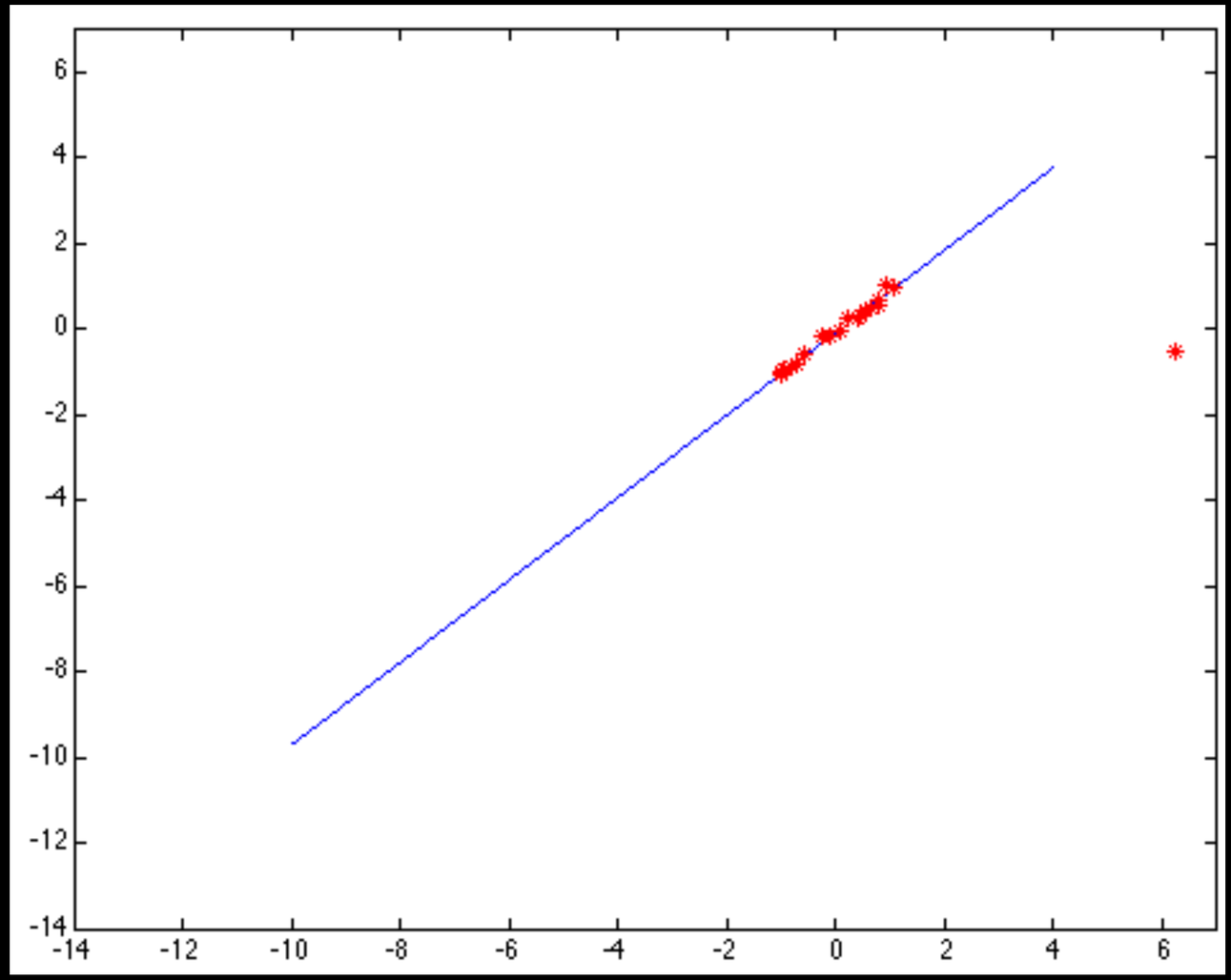


The robust function ρ

- Favors a configuration with small residuals
- Constant penalty for large residuals

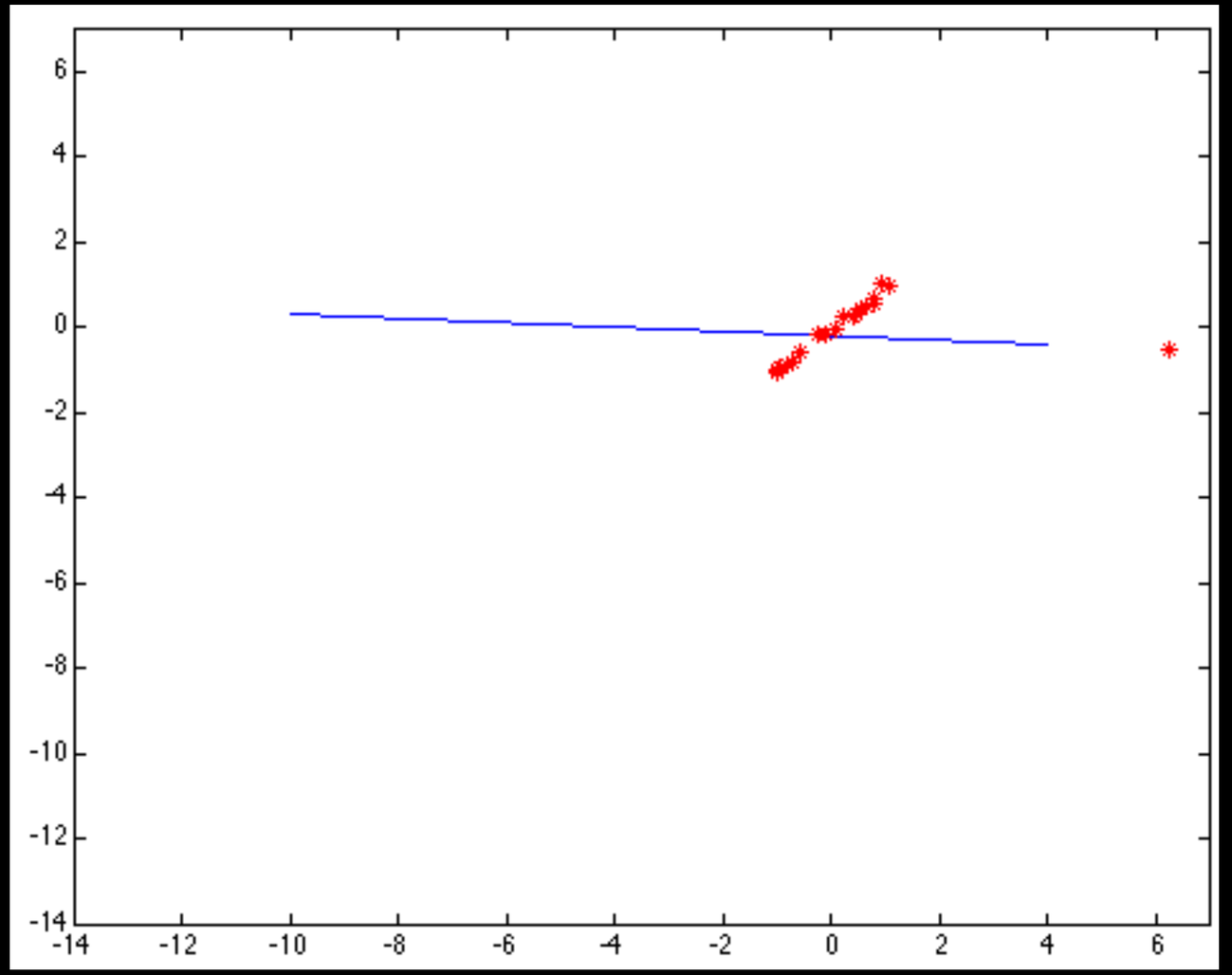
Slide from S. Savarese

CHOOSING THE SCALE: JUST RIGHT



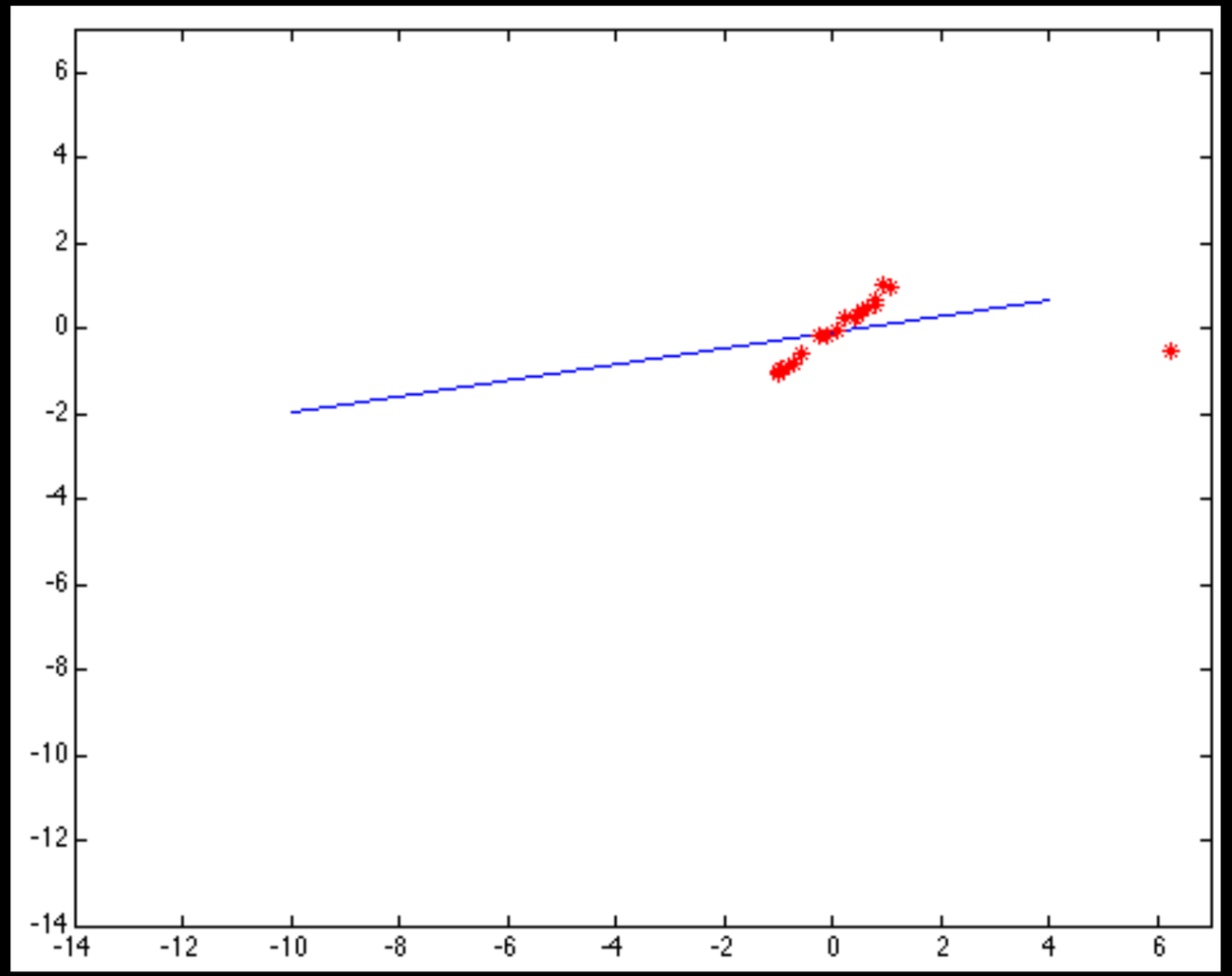
THE EFFECT OF THE OUTLIER IS MINIMIZED

CHOOSING THE SCALE: TOO SMALL



THE ERROR VALUE IS ALMOST THE SAME FOR EVERY POINT AND THE FIT IS VERY POOR

CHOOSING THE SCALE: TOO LARGE



BEHAVES MUCH THE SAME AS LEAST SQUARES

RANSAC

RANSAC (RANDOM SAMPLE CONSENSUS) :

- Algorithm:

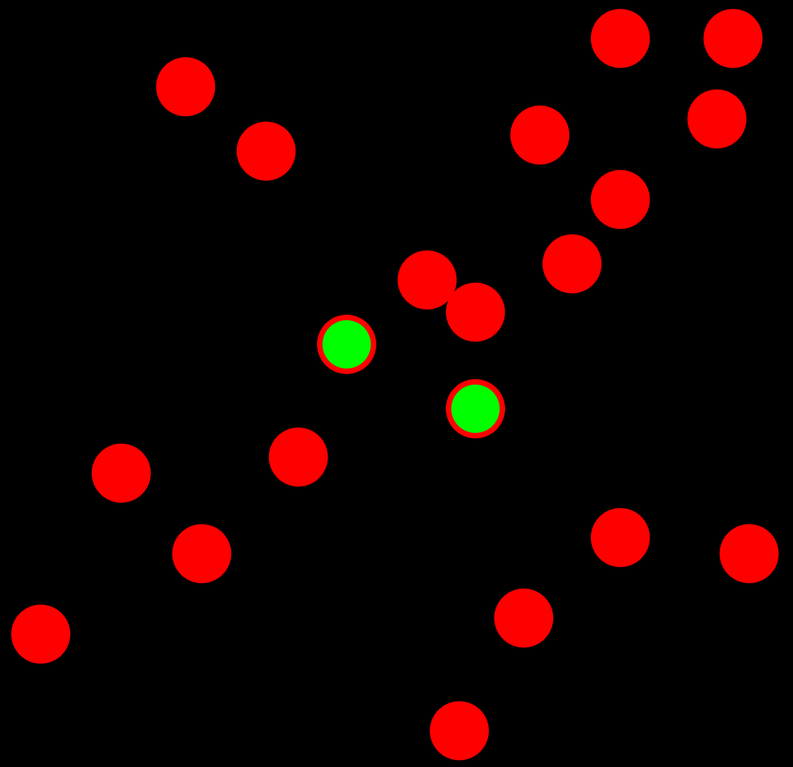
- 1. Sample (randomly) the number of points required to fit the model
- 2. Solve for model parameters using samples
- 3. Score by the fraction of inliers within a preset threshold of the model

- Repeat 1-3 until the best model is found with high confidence

Fischler & Bolles in ‘81.

RANSAC

LINE FITTING EXAMPLE



Algorithm:

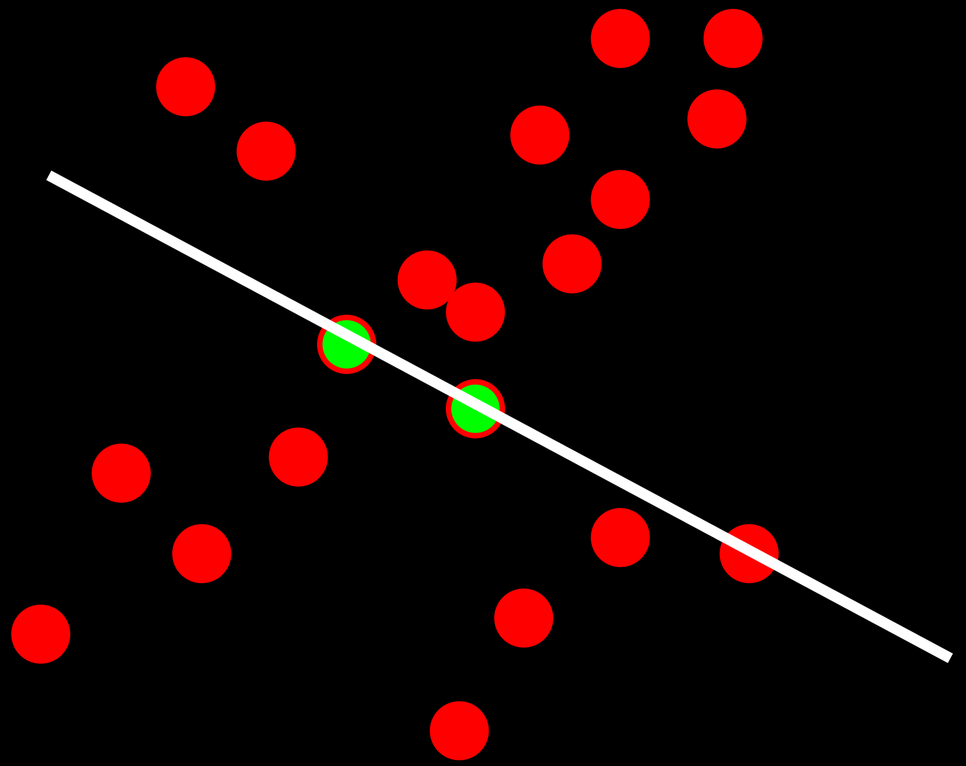
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Illustration by Savarese

RANSAC

LINE FITTING EXAMPLE



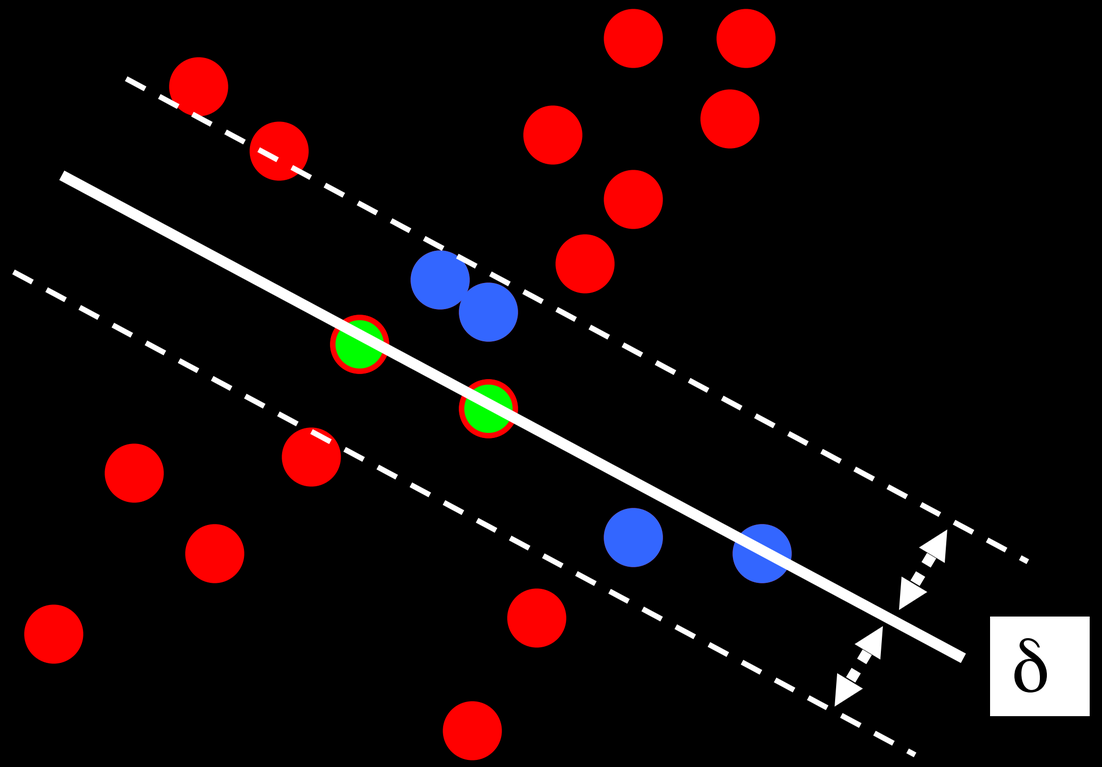
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

LINE FITTING EXAMPLE



$$N_I = 6$$

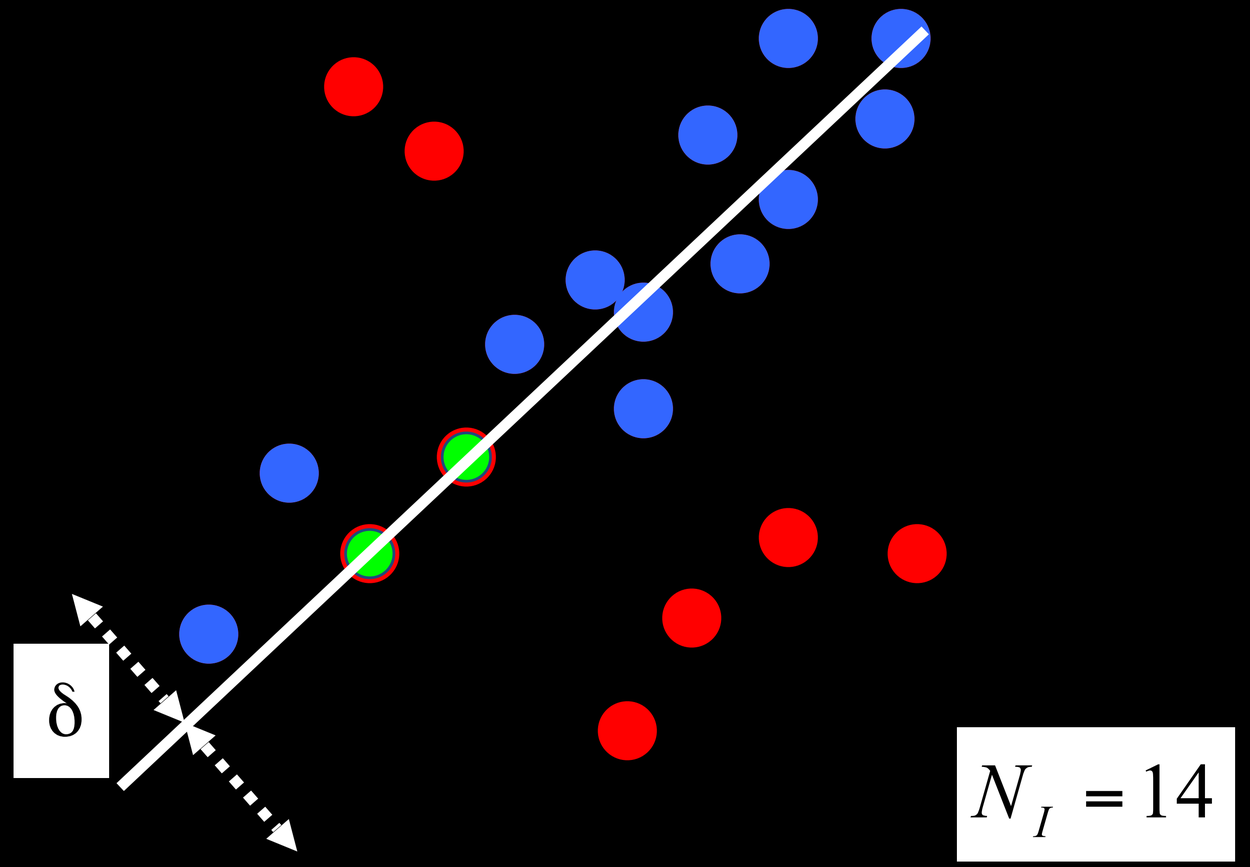
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

LINE FITTING EXAMPLE



Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

HOW TO CHOOSE PARAMETERS?

- 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*)
- Number of sampled points *s*
  - Minimum number needed to fit the model
- Distance threshold δ
  - Choose δ so that a good point with noise is likely (e.g., prob=0.95) within threshold
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$

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

	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

modified from M. Pollefeys

RANSAC CONCLUSIONS

Good

- Robust to outliers

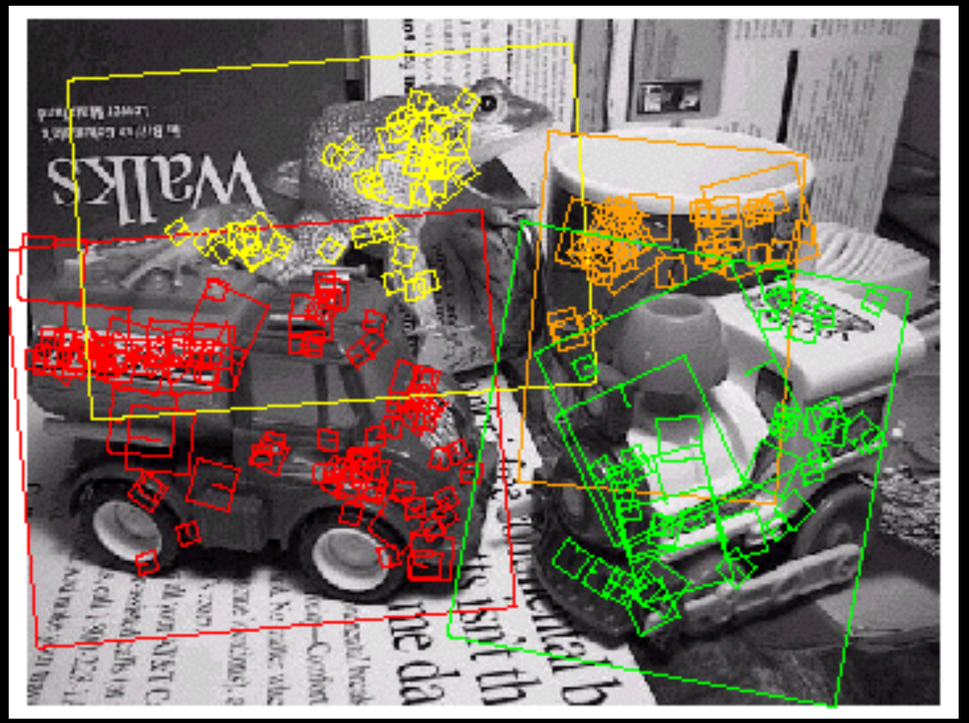
Bad

- Computational time grows quickly with fraction of outliers and number of parameters

Common applications

- Estimating fundamental matrix (relating two views)
- Computing a homography (e.g., image stitching)

EXAMPLES OF RECOGNIZED OBJECTS



TEXT

- Most slides taken from James Hays
- http://www.cc.gatech.edu/~hays/