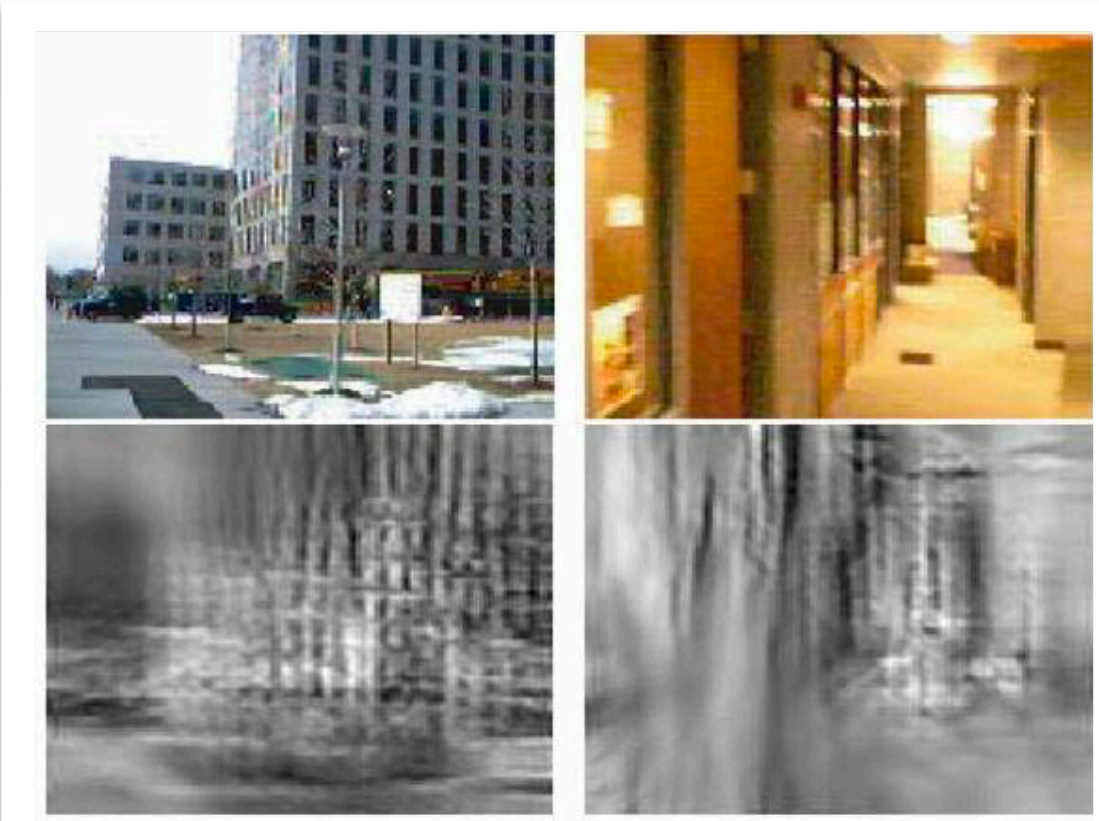
Descriptors



Outline

- Motivation
- Interest point detection
  - Cornerness
  - Optimization (first-order,second-order)
  - Scale-space
- Descriptors
  - SIFT
  - Other fast descriptors (SURF, BRIEF)
  - Dense descriptors
- **Ransac [interlude]**

Overall pipeline for finding correspondence in 2 images

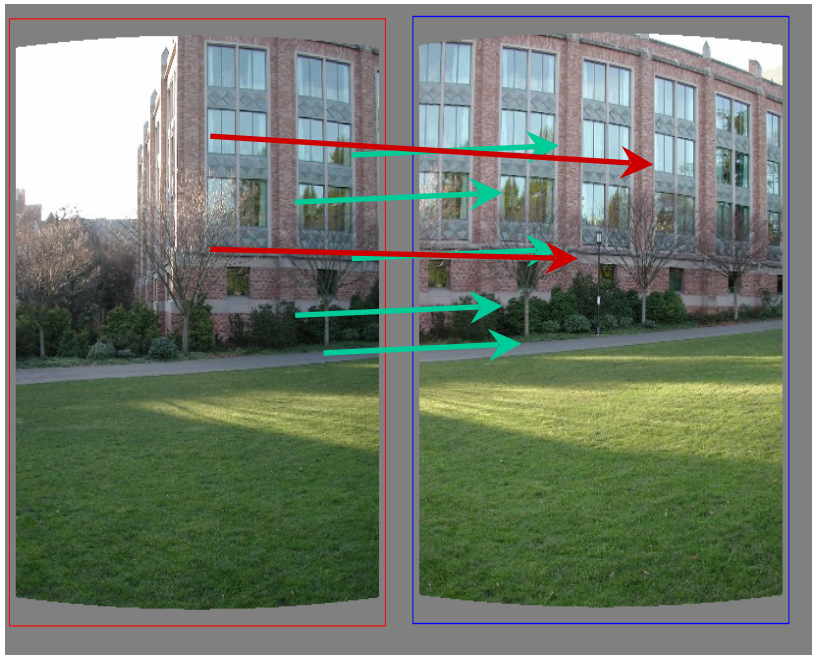
- Find N “interesting” patches in left image and M “interesting” patches in right image

- Compute O(NM) patch similarities and candidate correspondences (e.g., each patch in left image, find best match in right)

- **Extract subset of correspondences that are consistent with a model**

Feature matching

- Exhaustive search
  - for each feature in one image, look at *all* the other features in the other image(s)



How do we remove bad matches?

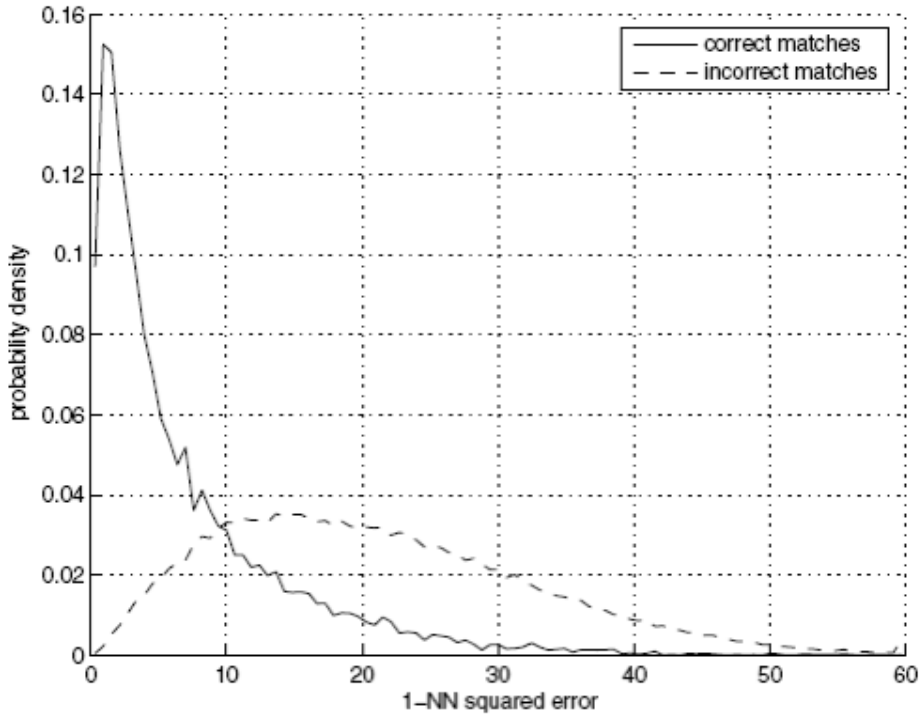
Repeated textures (like windows) are notoriously challenging!

Feature-space outlier rejection

One philosophy: heuristically score the *certainty* of a match (and only keep certain matches).

Version 0: Distance(SIFT(patch1),SIFT(patch2)) < threshold

Find threshold for classifying correct-vs-incorrect matches from *training dataset*



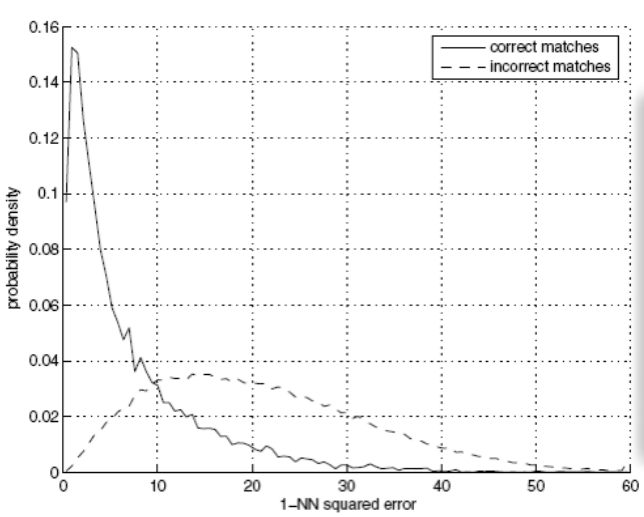
Question: for a given threshold, can we visualize the ~~number~~ fraction of true correct matches? False correct matches? How should we set the threshold?

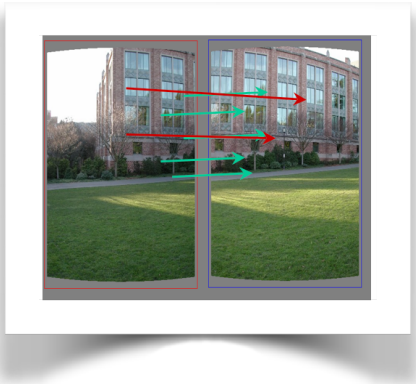
Version1: reject unstable matches

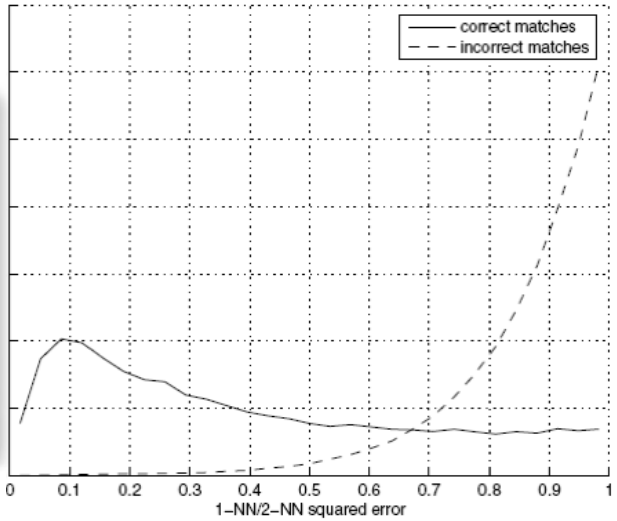
"Look for patches that have a good match. If there's a good *alternative* match, don't trust it!"

[Lowe, 1999]:

- 1-NN: SSD of the closest match
- 2-NN: SSD of the second-closest match
- Look at how much better 1-NN is than 2-NN, e.g. 1-NN/2-NN

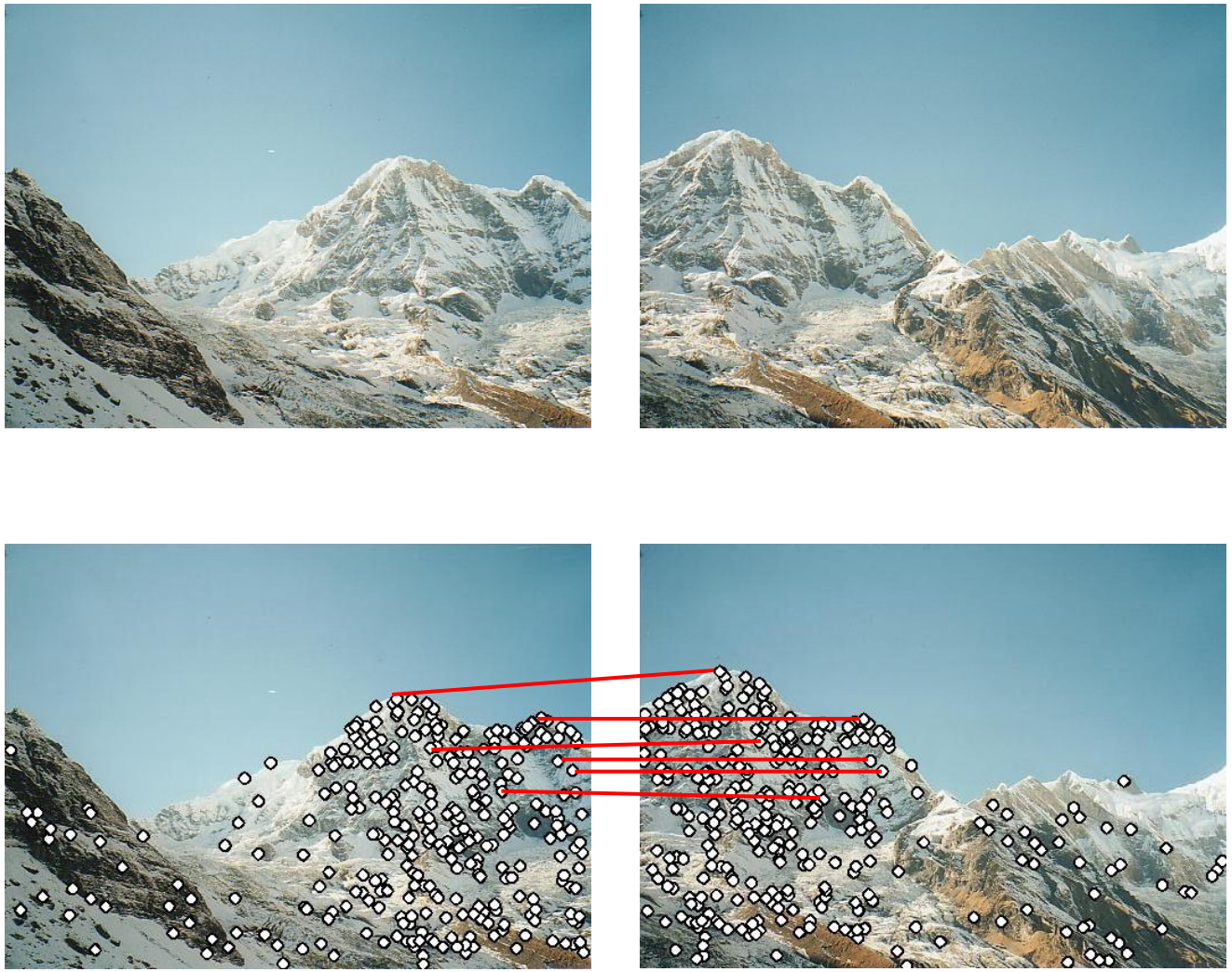


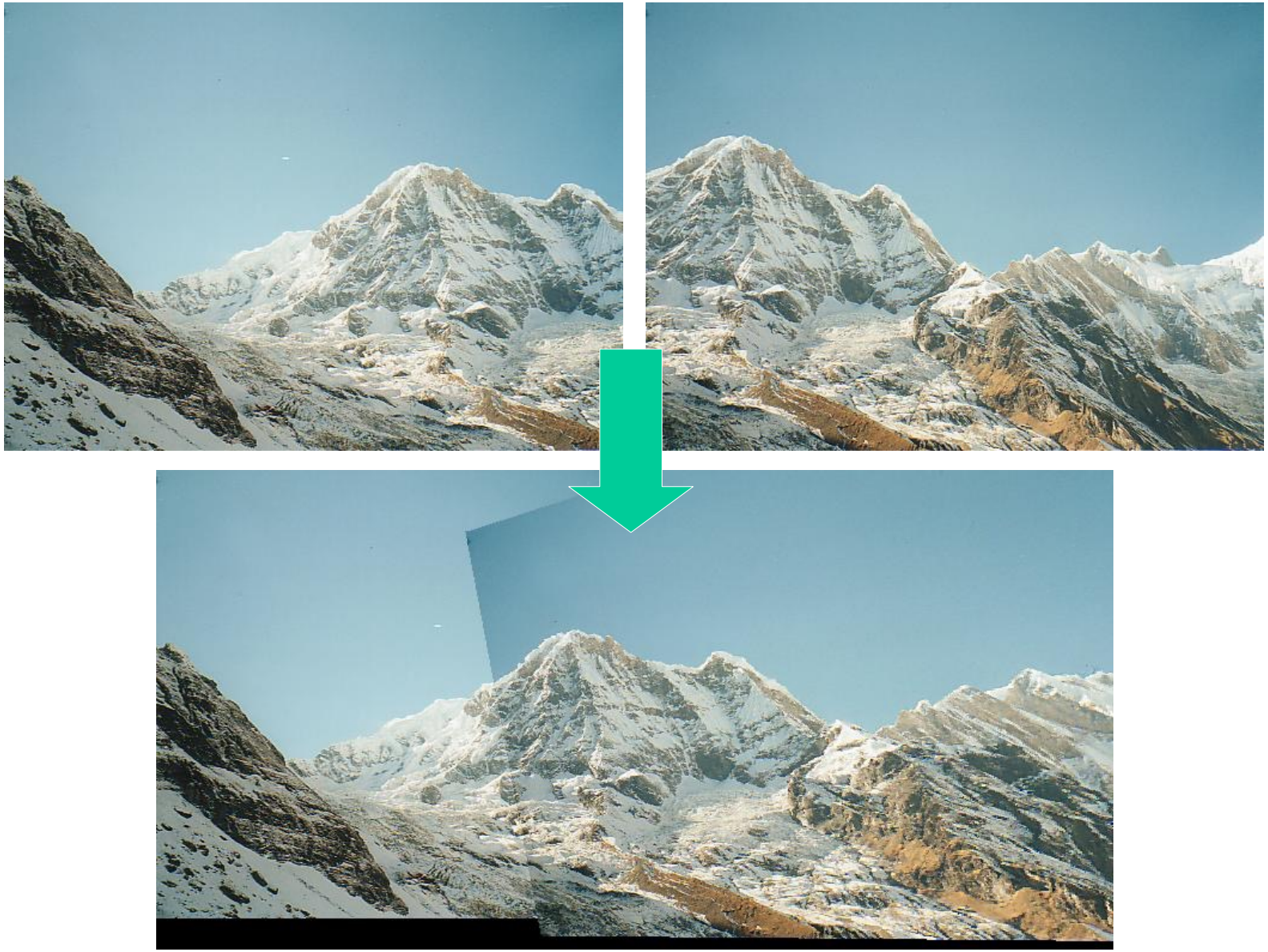




What's the underlying subtext here? We are really designing features for *binary classifiers* that identify good-vs-bad matches

Much more effective philosophy: look for matches consistent with a global *motion* model





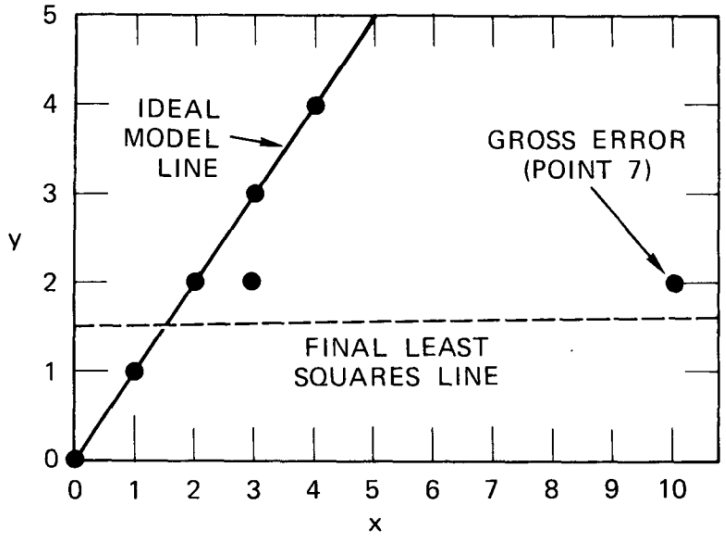
RANSAC: go-to approach for fitting models under noise… not sure why its not more widely known!

Graphics and Image Processing
J. D. Foley Editor

Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography

Martin A. Fischler and Robert C. Bolles
SRI International

PROBLEM: Given the set of seven (x,y) pairs shown in the plot, find a best fit line, assuming that no valid datum deviates from this line by more than 0.8 units.



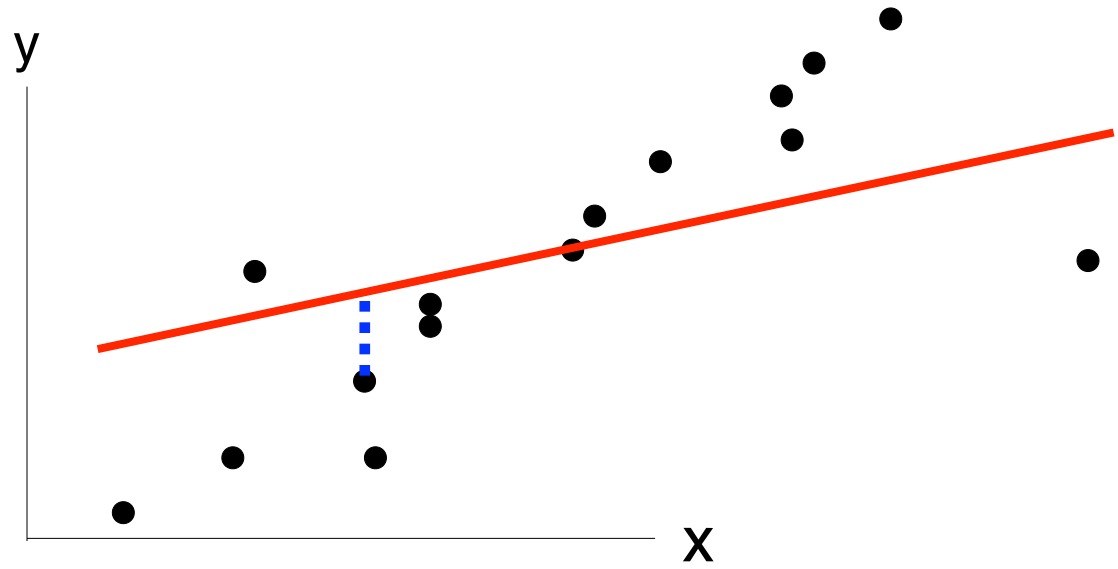
POINT	x	y
1	0	0
2	1	1
3	2	2
4	3	2
5	3	3
6	4	4
7	10	2

Given data $\{x_i, y_i\}$ fit line (w,b) such that s.t.

$$\min_{w,b} \sum_i (y_i - f_{w,b}(x_i))^2$$

$$f_{w,b}(x_i) = wx_i + b$$

Crucial limitation of least squares: outliers are common in real-world

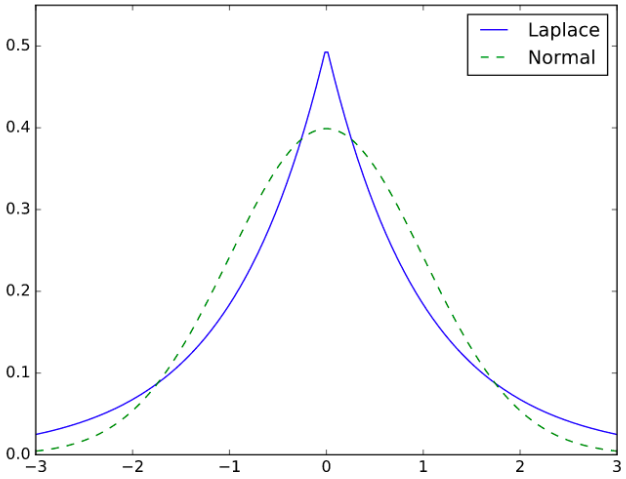


What's really going on: robust statistics

https://en.wikipedia.org/wiki/Robust_statistics

Gaussian distribution:

$$p(x) \propto e^{-\frac{1}{2\sigma^2}||x||^2}$$



Laplace distribution:

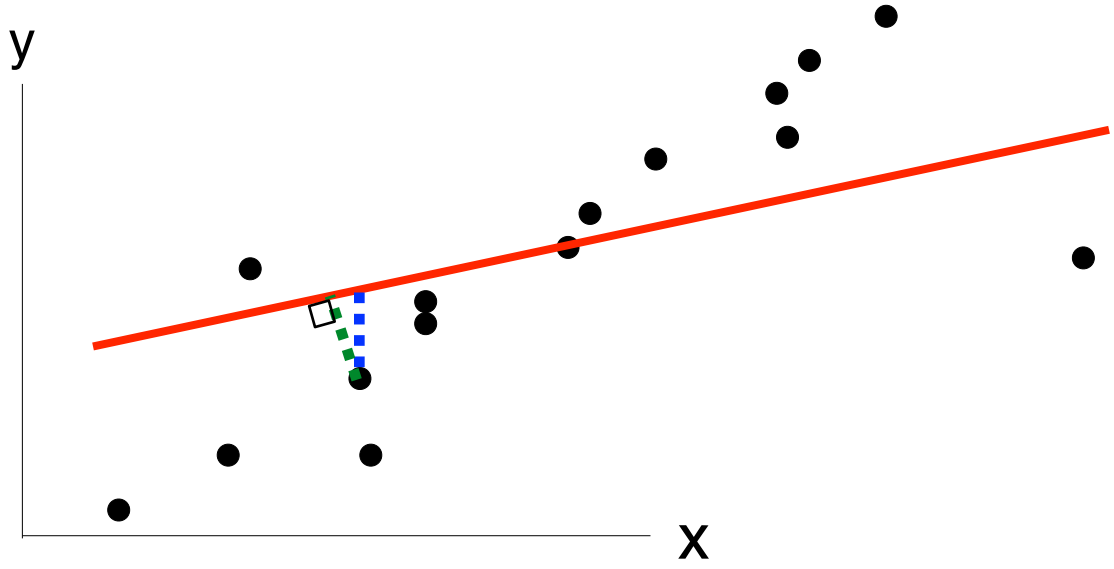
$$p(x) = \frac{1}{2s}e^{-|x|/s}$$

Minimizing squared error can be reduced to maximizing the (log) probability of a Gaussian model (that suggests 3sigma+ outliers are very rare)

We'll look at *heavy-tailed* noise models later in class

One can view RANSAC as an algorithm for optimizing a *robust* error model

Aside: shouldn't we be minimizing green error?

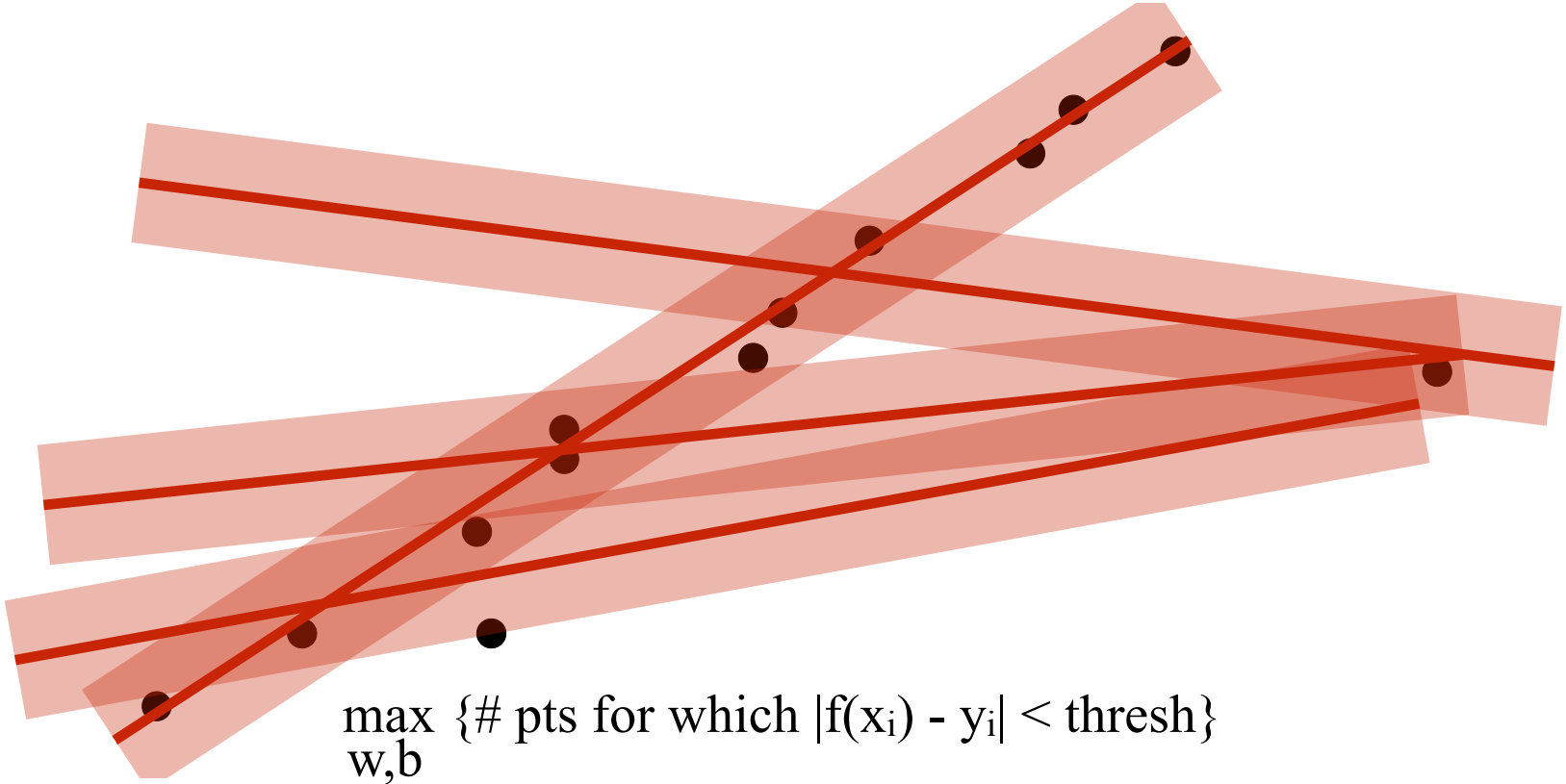


https://en.wikipedia.org/wiki/Distance_from_a_point_to_a_line

(you can work out the math offline!)

RANSAC algorithm

https://en.wikipedia.org/wiki/Random_sample_consensus

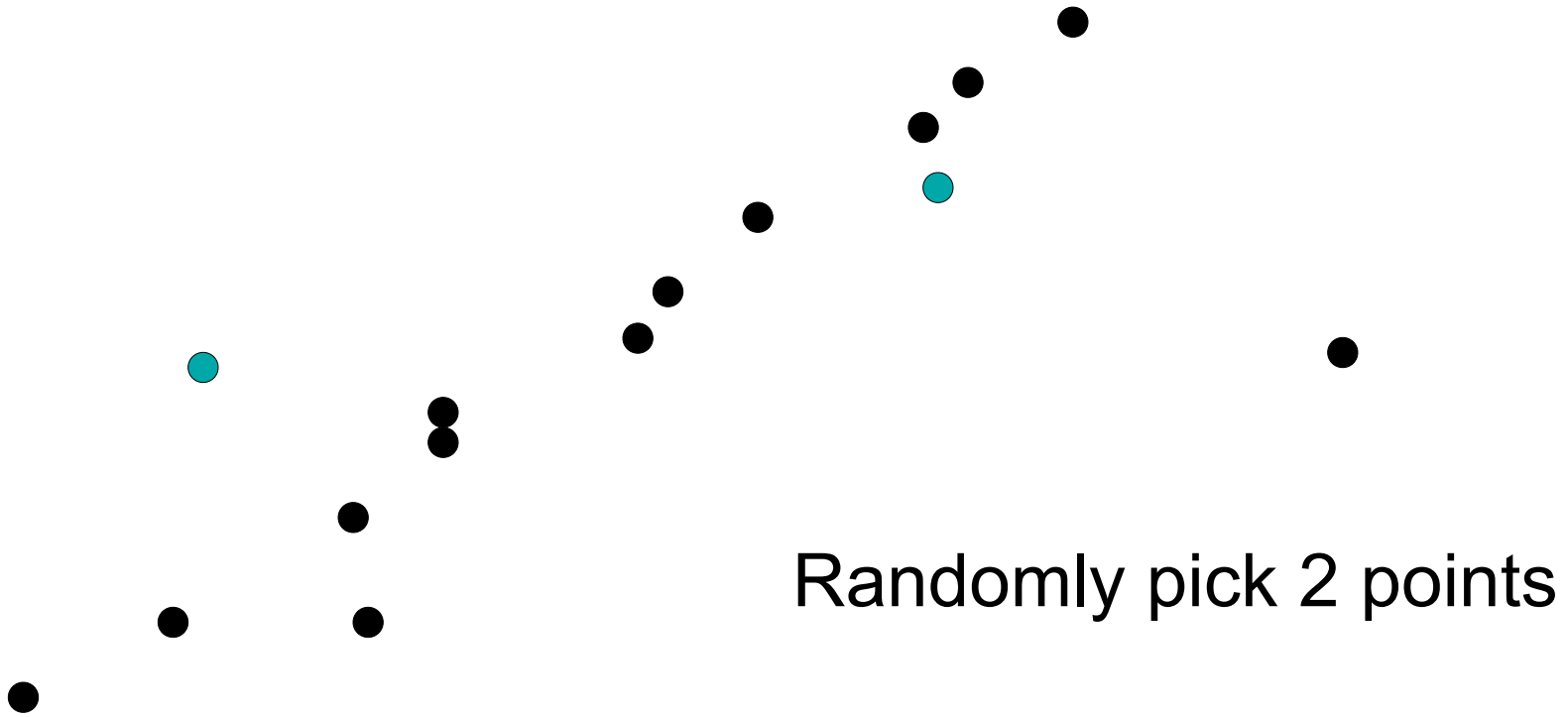


$$\max_{w,b} \{\#\text{ pts for which } |f(x_i) - y_i| < \text{thresh}\}$$

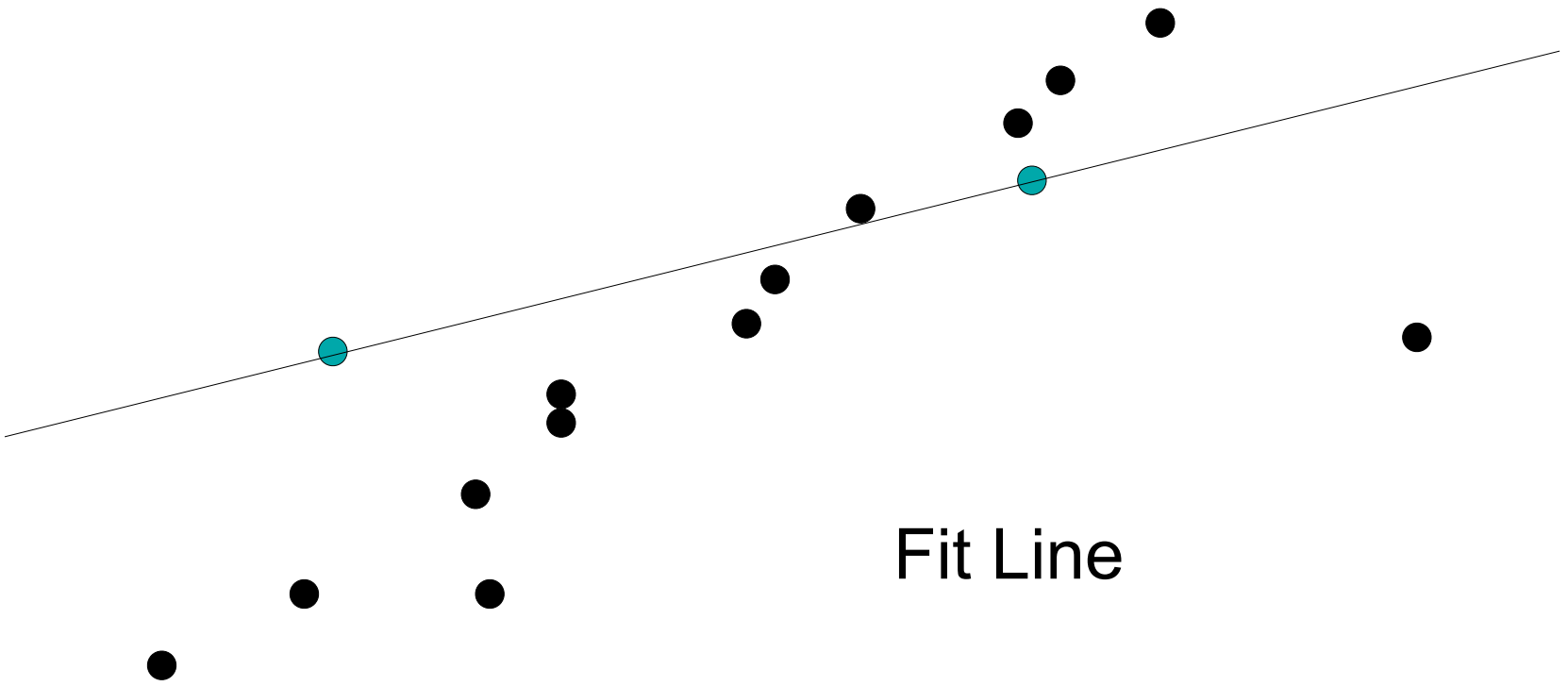
Key idea:

- (try to) Brute-force search for optimal solution with many "inliers"
- Exploit fact that inliers will all vote for same soln but outliers will vote in different ways (*stochastic* search rather than exhaustive search)

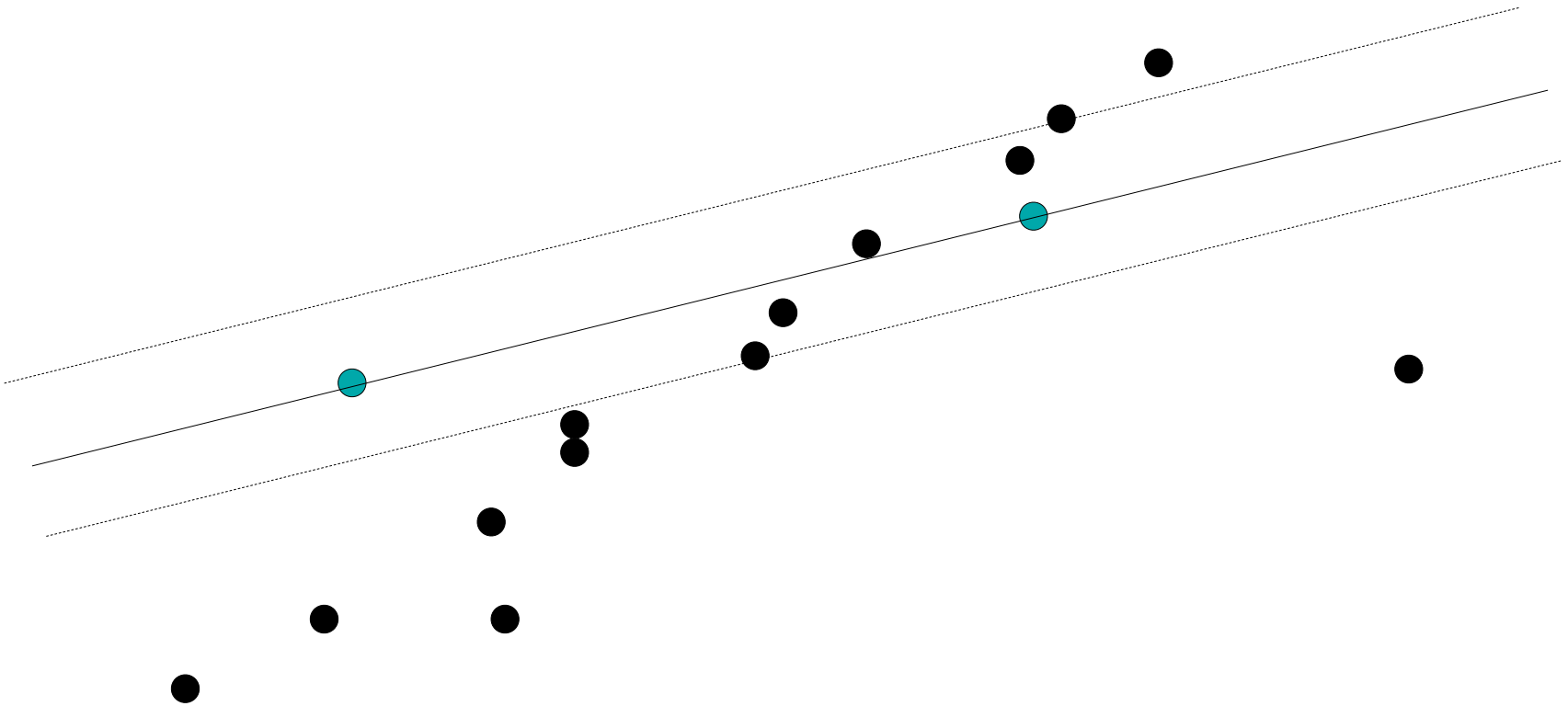
RANSAC Line Fitting Example



RANSAC Line Fitting Example

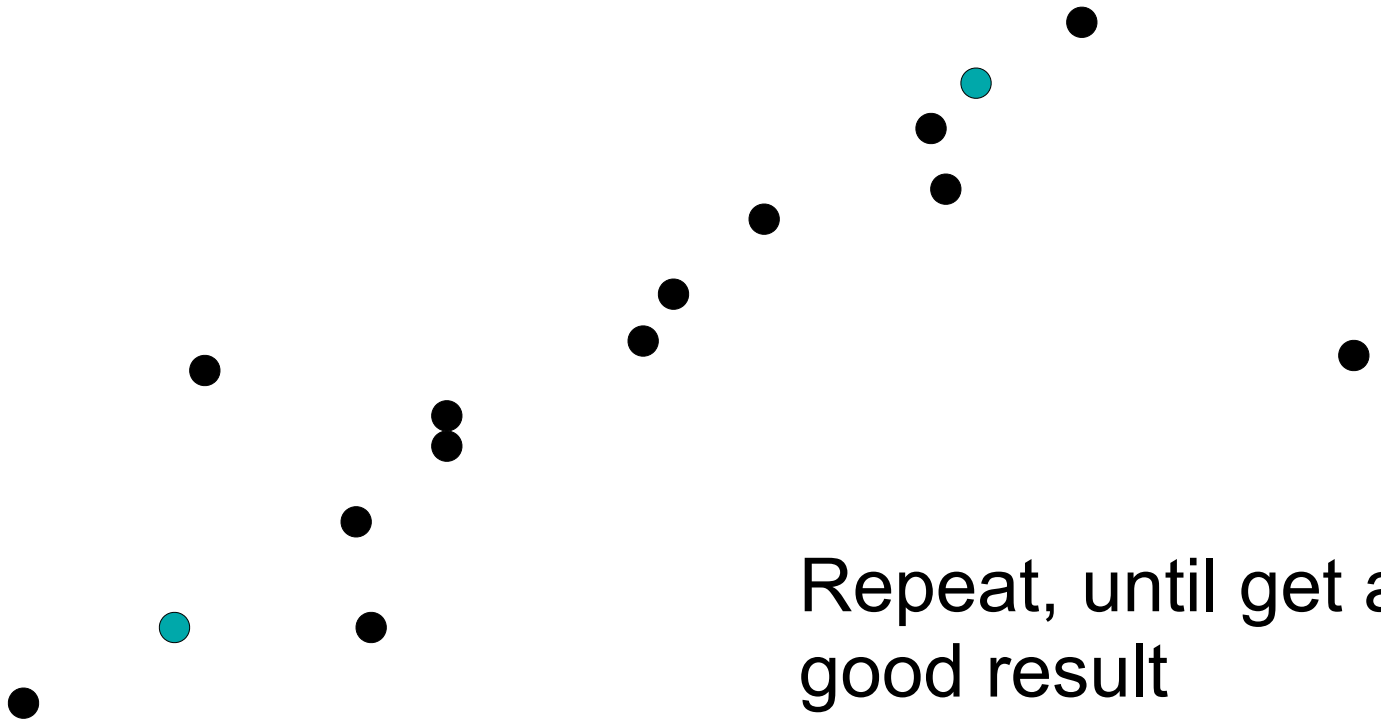


RANSAC Line Fitting Example



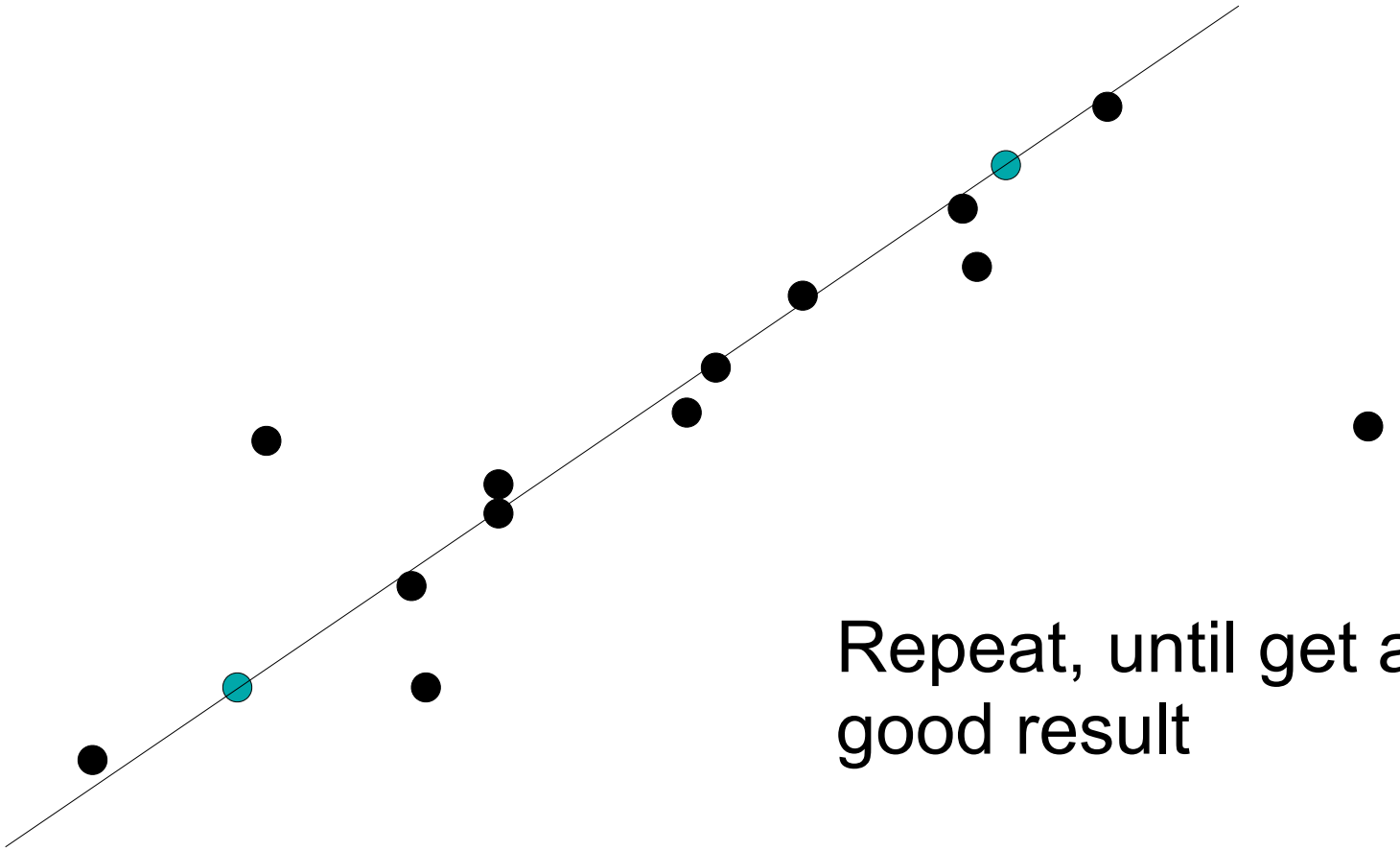
Count # of pts within a threshold of line.

RANSAC Line Fitting Example



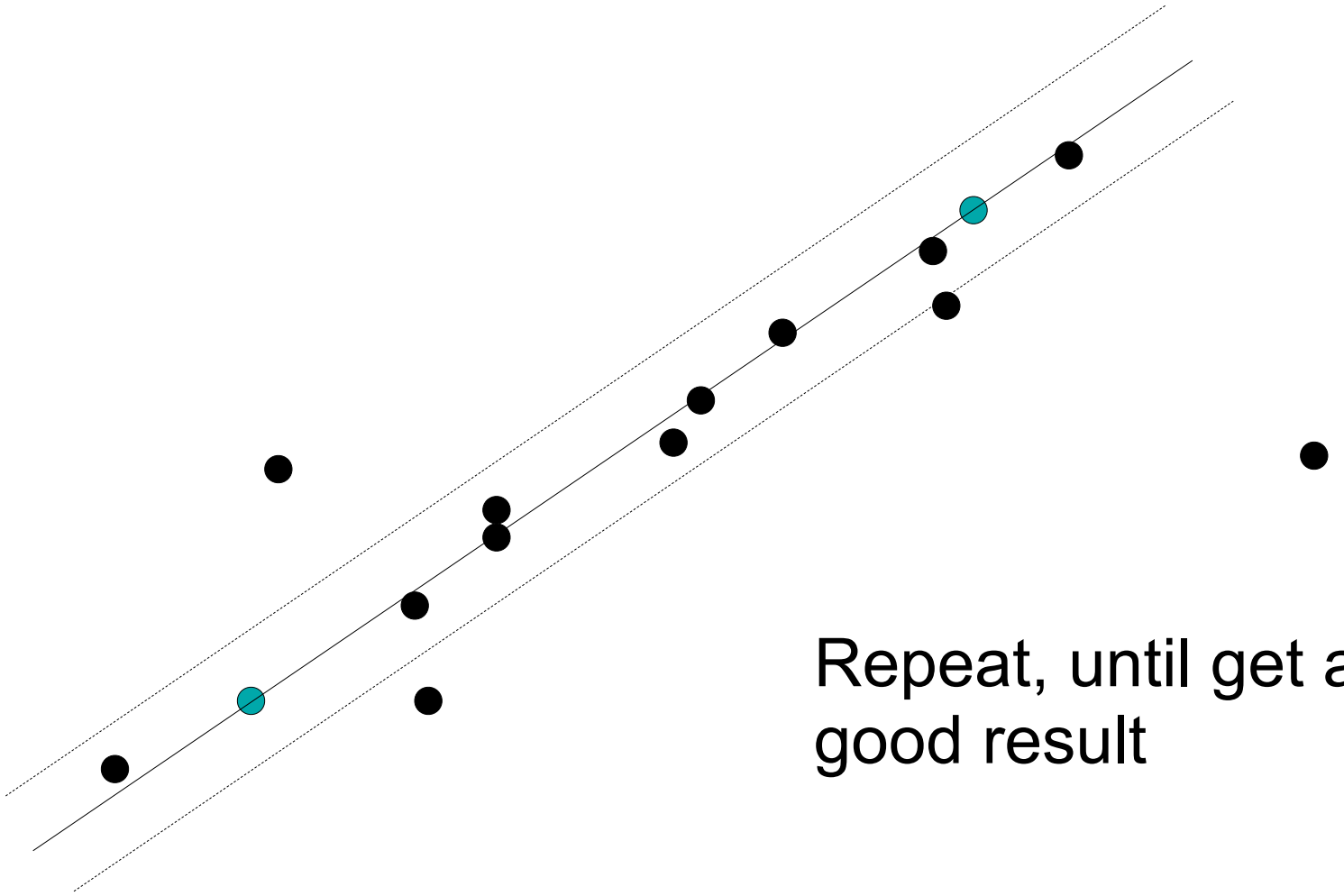
Repeat, until get a good result

RANSAC Line Fitting Example



Repeat, until get a good result

RANSAC Line Fitting Example



Repeat, until get a good result

RANSAC for line fitting

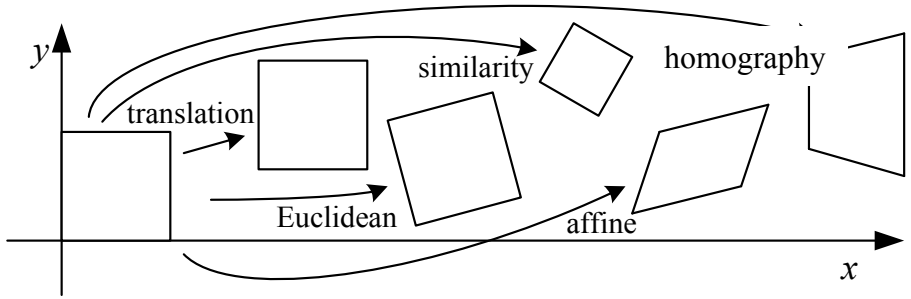
Repeat for ***k*** *trials*:

- Draw ***n*** points uniformly at random
- Fit line to these ***n*** points
- Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than threshold ***t***)
- Return line with largest inlier set

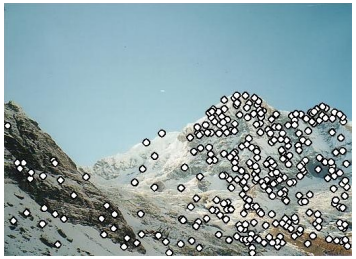
(Optionally return line with optimal least-squares inlier fit)

Source: M. Pollefeys

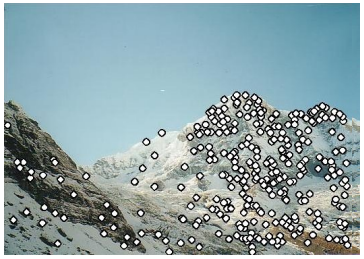
Models for aligning images

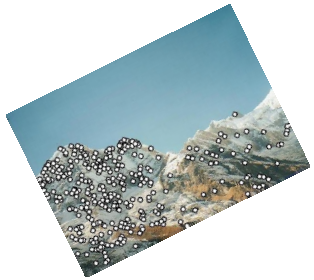


Translation



Similarity
(translation+rotation+scale)

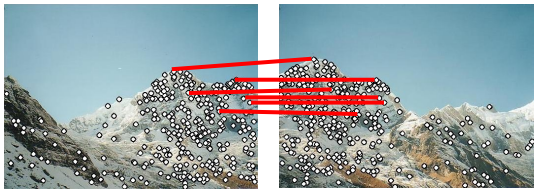




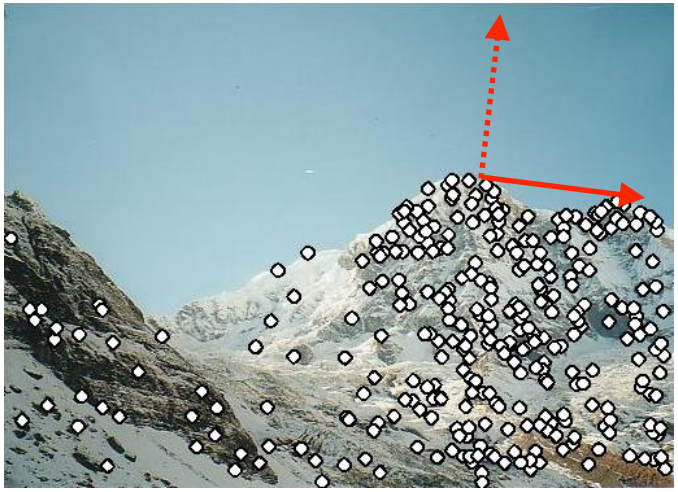
(In HW3, you'll explore homography)

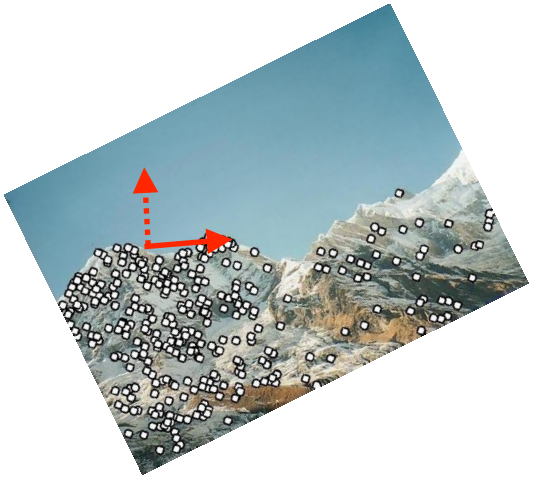
Case study: RANSAC for similarity alignment

What is the minimum number of point correspondences needed to fit a similarity (translation+rotation+scale) transform?



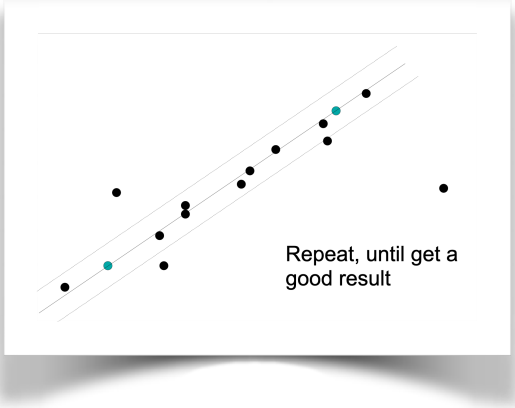
2 point correspondences





Ransac parameters

https://en.wikipedia.org/wiki/RANSAC



p: prob that RANSAC returns a good model
w: fraction of inliers in data
n: number of points used to estimate model
k: number of random *trials*

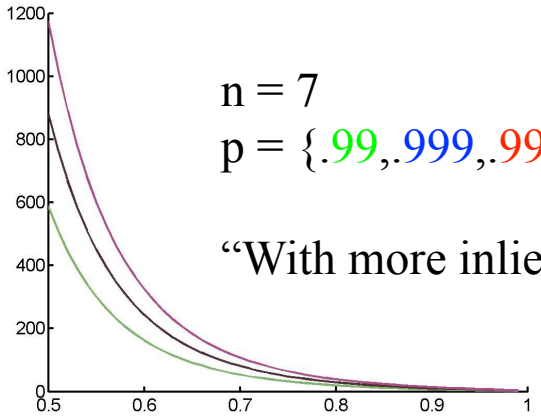
Goal: given a target ‘p’, compute number of needed trials ‘k’

$1- w^n$:prob that at least 1 out of n points is an outlier (e.g., RANSAC fails on a single trial)

$(1- w^n)^k$:prob that RANSAC fails across all k trials

$p = 1- (1- w^n)^k$:prob that RANSAC returns finds at least 1 good model

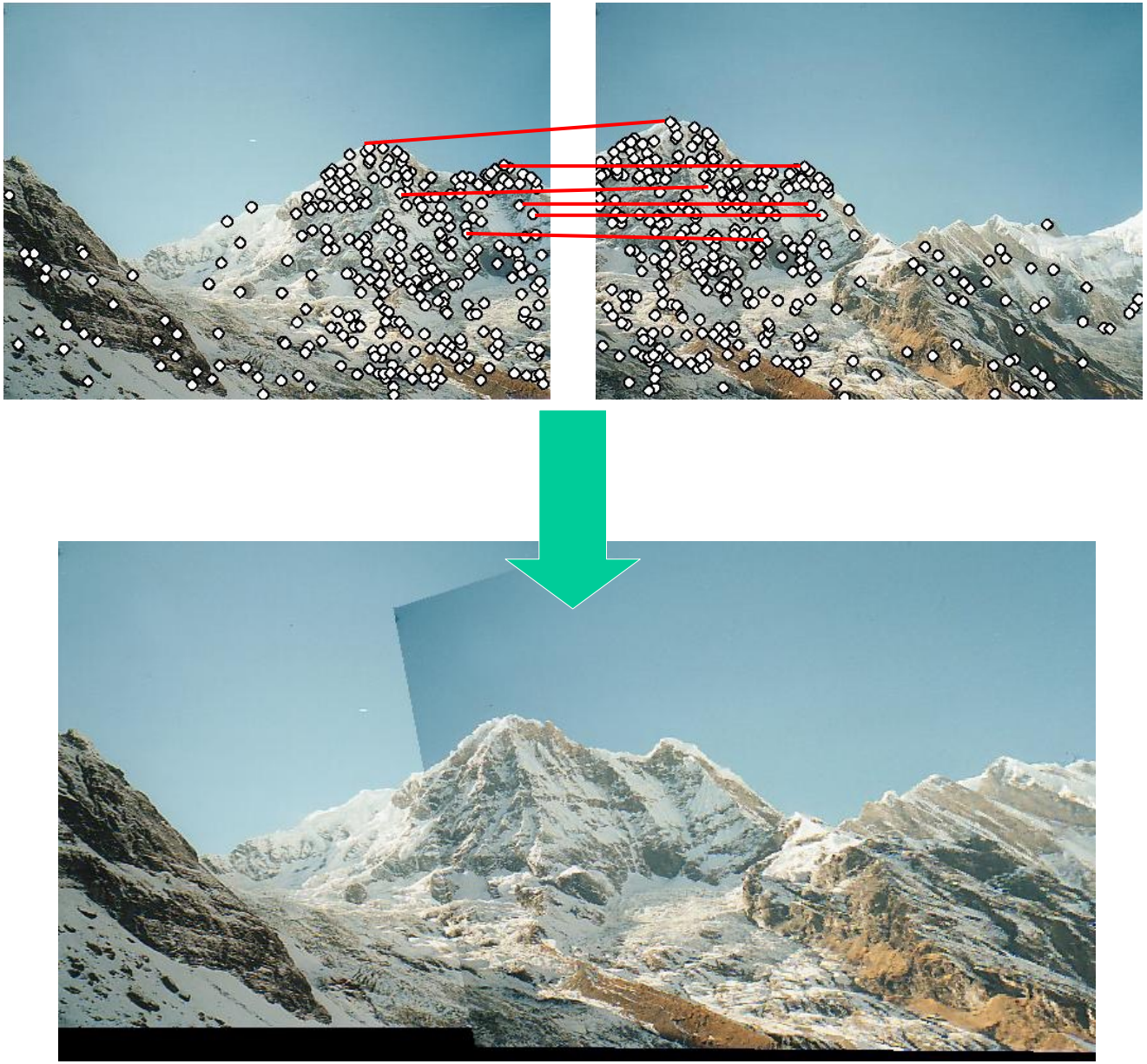
$$k = \frac{\log(1-p)}{\log(1-w^n)}$$



n = 7
p = {.99,.999,.9999}

“With more inliers, fewer trials are needed”

RANSAC for alignment

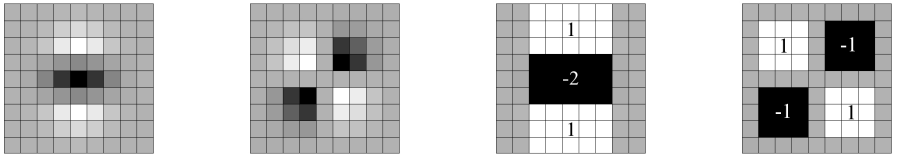


Outline

- Motivation
- Interest point detection
  - Cornerness
  - Optimization (first-order,second-order)
  - Scale-space
- Descriptors
  - SIFT
  - **Other fast descriptors (SURF, BRIEF)**
- Ransac [interlude]

SURF (SIFT++)
(Speeded Up Robust Features)

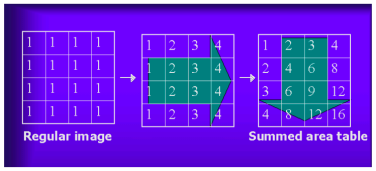
- Replace Gaussians with box-filters



Fast approximation of first and second derivatives

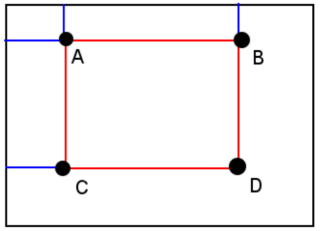
Recall that one can implement a box filter (of all 1's) that is *independant* of filter size

http://en.wikipedia.org/wiki/Summed_area_table



$$I(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} i(x',y')$$

$$I(x,y) = i(x,y) + I(x-1,y) + I(x,y-1) - I(x-1,y-1)$$



Text

Sum = D - B - C + A

SURF

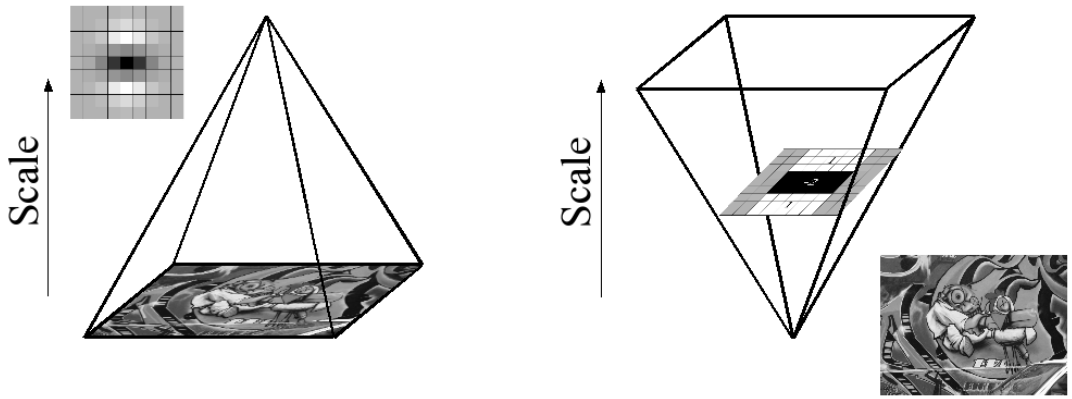


Fig. 4. Instead of iteratively reducing the image size (left), the use of integral images allows the up-scaling of the filter at constant cost (right).

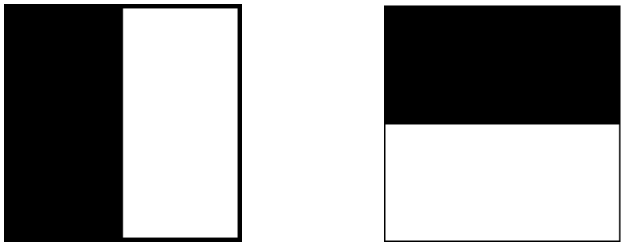


Fig. 9. Haar wavelet filters to compute the responses in x (left) and y direction (right). The dark parts have the weight -1 and the light parts $+1$.

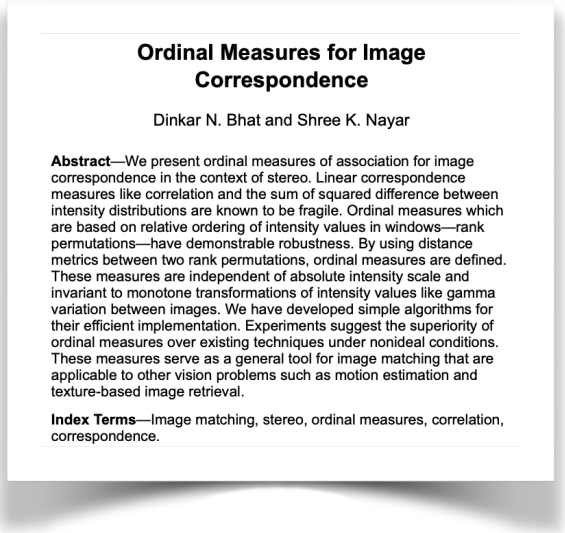
Alternate family of approaches: rank-based representations

28	50	70
5	10	80
3	1	30

Order:
6,3,2,9,1,5,4,7,8

position of brightest pixel

position of darkest pixel



Ordinal Measures for Image Correspondence

Dinkar N. Bhat and Shree K. Nayar

Abstract—We present ordinal measures of association for image correspondence in the context of stereo. Linear correspondence measures like correlation and the sum of squared difference between intensity distributions are known to be fragile. Ordinal measures which are based on relative ordering of intensity values in windows—rank permutations—have demonstrable robustness. By using distance metrics between two rank permutations, ordinal measures are defined. These measures are independent of absolute intensity scale and invariant to monotone transformations of intensity values like gamma variation between images. We have developed simple algorithms for their efficient implementation. Experiments suggest the superiority of ordinal measures over existing techniques under nonideal conditions. These measures serve as a general tool for image matching that are applicable to other vision problems such as motion estimation and texture-based image retrieval.

Index Terms—Image matching, stereo, ordinal measures, correlation, correspondence.

125	154	176
87	98	189
92	85	140

Order:
6,3,2,9,1,5,7,4,8

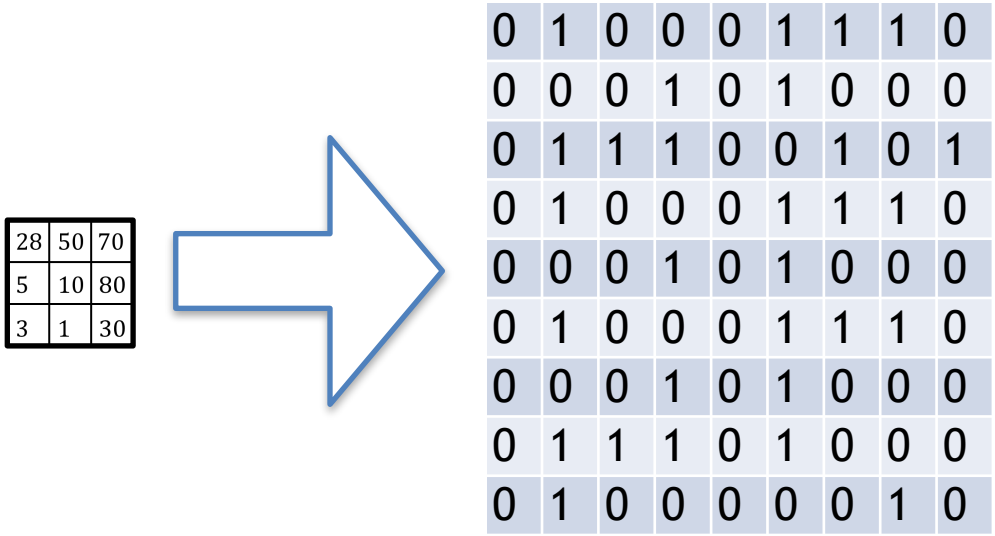
Positive: rank ordering is invariant to monotonic transformations of intensity (not just linear transformations!)

Negative: comparing two different ranks is expensive

Alternate approach: binary patterns

Convert rank-order vector into a N^2 binary matrix of relative comparisons

A[i,j] = is pixel i > pixel j ?



1. Create a descriptor that is a *fixed but random projection* of the vectorized matrix
2. Compare two descriptors by # of matching bits (Hamming distance with bitwise operations)

Next few slides will describe various approaches for selecting effectie random masks

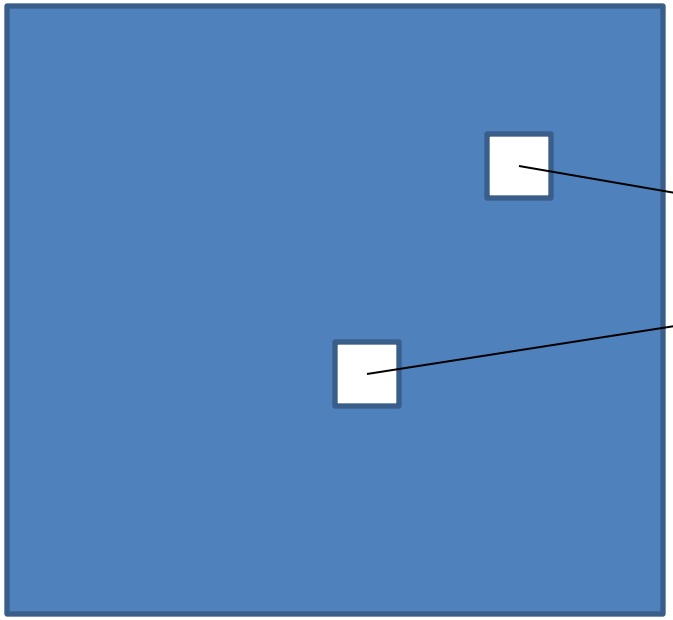
Ho, Tin (1995) "Random Decision Forests" Int'l Conf. Document Analysis and Recognition
Breiman, Leo (2001) "Random Forests" Machine Learning
Amit, Y. & Geman, D. (1997). Shape quantization and recognition with randomized trees. Neural Computation, 1997

Example of random binary descriptor: (BRIEF)

Binary Robust Independent Elementary Features

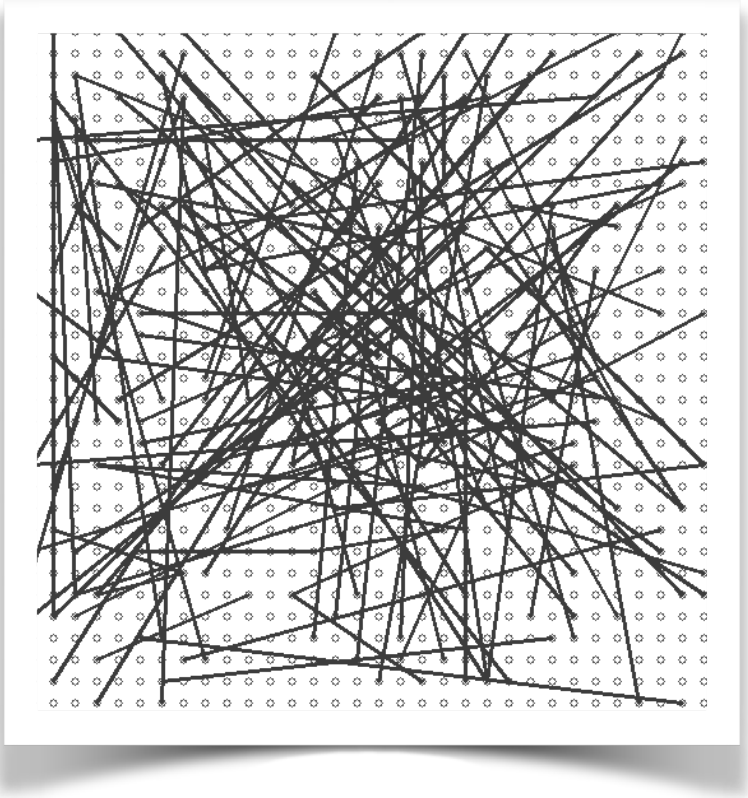
(still used in contemporary robotics packages like ORB-SLAM)

- *Randomly* select pairs p and p' for comparison
- Design choice: How to sample p,p'?

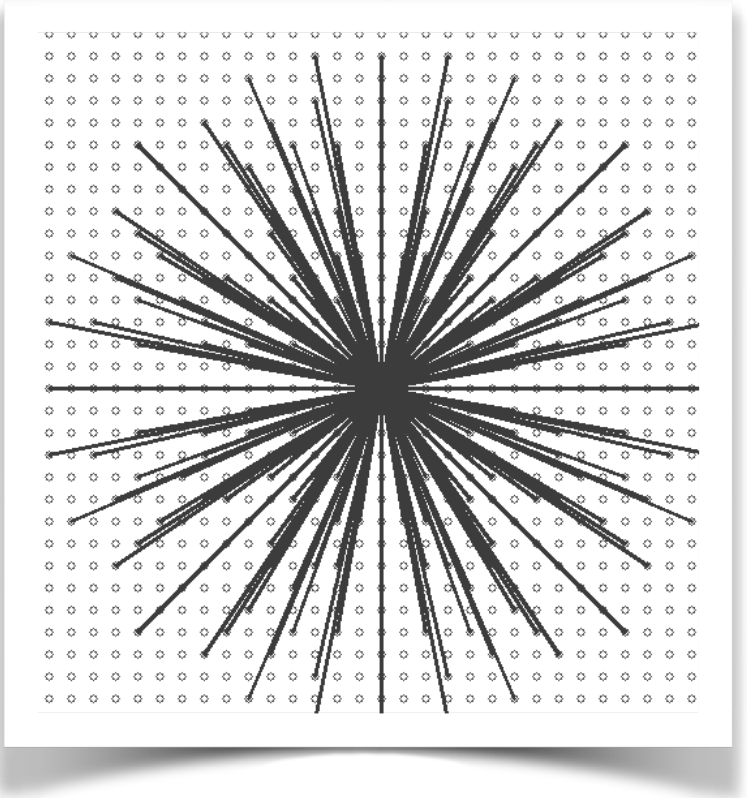


$$b = \begin{cases} 1 \; if \; I(p) > I(p') \\ 0 \; otherwise \end{cases}$$

Example sampling strategies

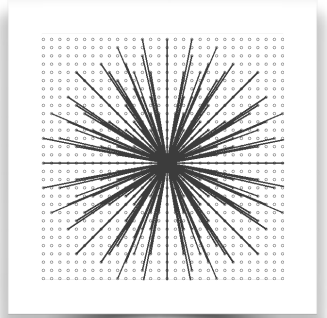


p1,p2 ~ uniform(x,y)

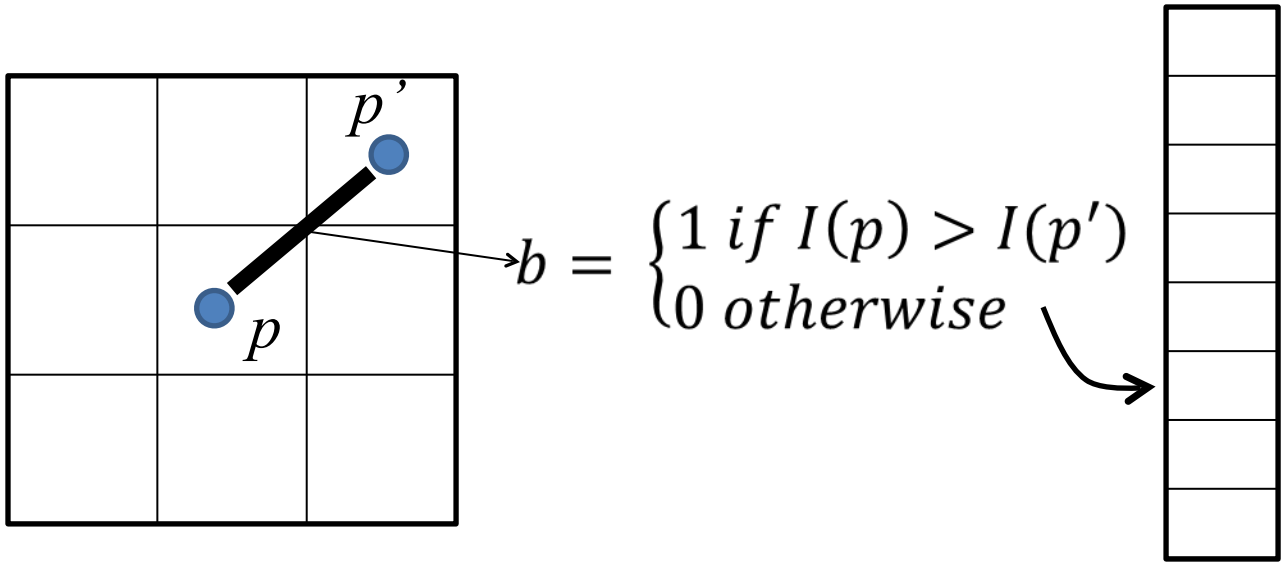


p1 = 0
p2 = grid(r,theta)

LBP: Local Binary Patterns

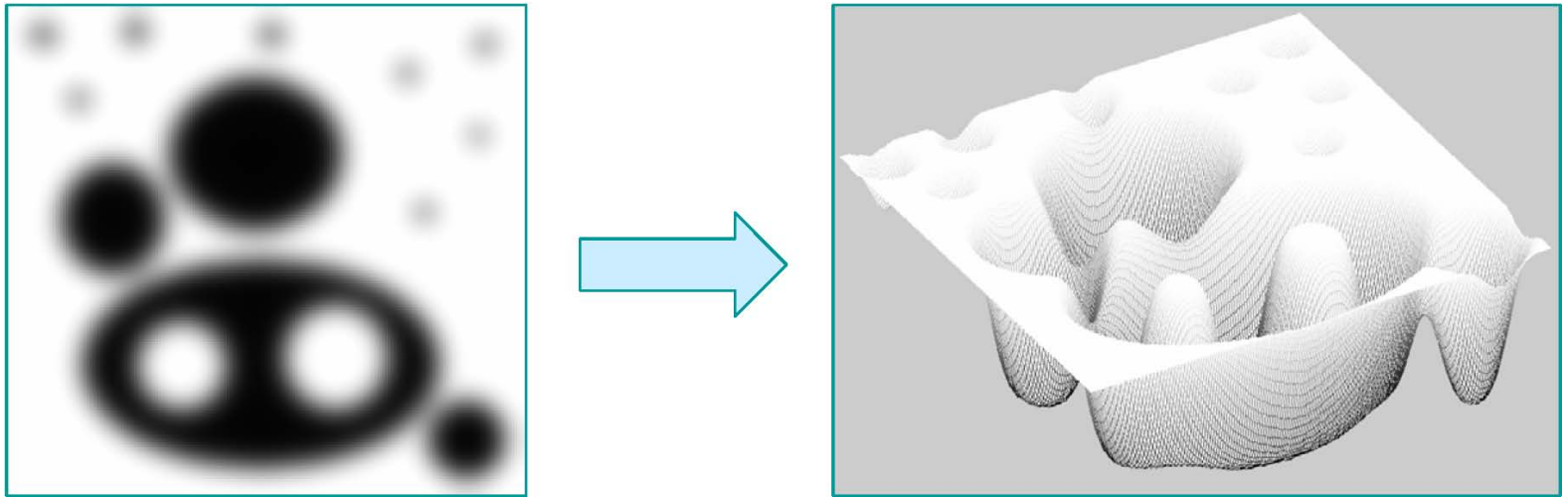


Special case for representing a 3x3 patch with a 8-bit vector:
Compare center pixel (p1=0) with with its (8) neighbor

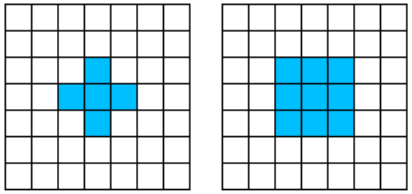


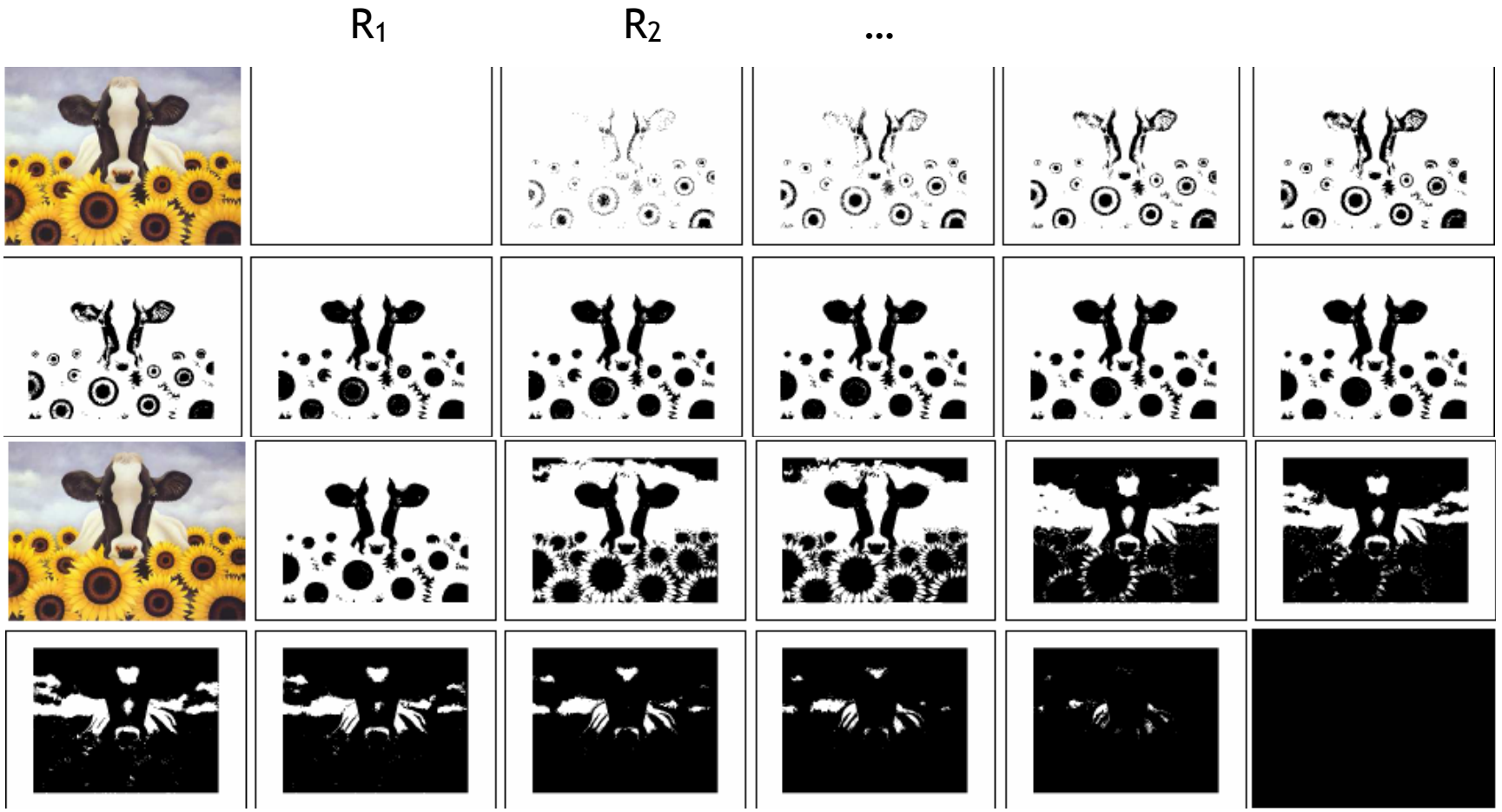
Regions instead of points

Maximally Stable Extremal Regions (MSER)



Extremal region R_i : Set of connected pixels such that the intensity values inside R_i are greater (or lower) than those at the boundary of R_i

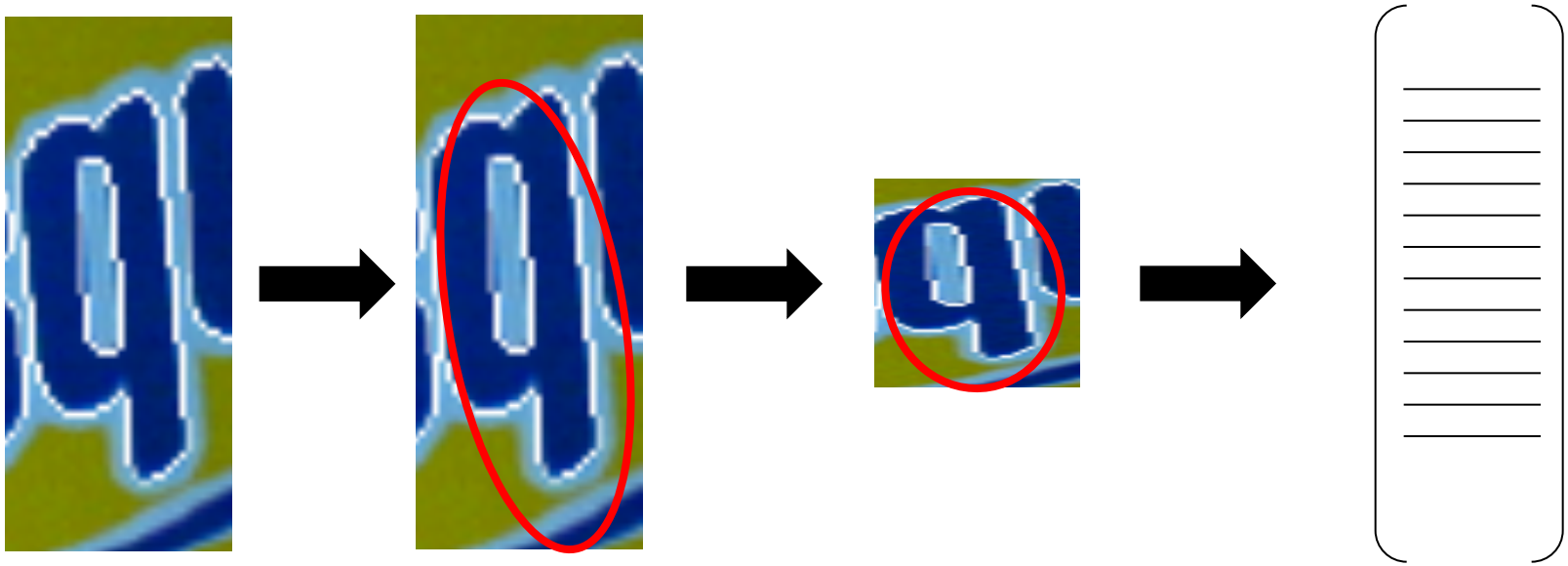




Stable region: connected region of pixels that doesn't change across different thresholds

$$(|R_{i+1}| - |R_i|)/ |R_i|$$

From MSER to SIFT



Find extremal region

Fit ellipse

Rotate + scale to canonical patch

Compute 128-dim SIFT vector

Examples from image matching

Example from Matas et al. BMCV 2002

Outline

- Motivation
- Interest point detection
  - Cornerness
  - Optimization (Guass-Newton, coarse-to-fine)
  - Scale-space
- Ransac
- Descriptors
  - SIFT
  - Other fast descriptors (SURF, BRIEF,MSER)
- Recent (neural extensions; superpoint and superglue)

SuperPoint: Self-Supervised Interest Point Detection and Description

Daniel DeTone
Magic Leap
Sunnyvale, CA
ddetone@magicleap.com

Tomasz Malisiewicz
Magic Leap
Sunnyvale, CA
tmalisiewicz@magicleap.com

Andrew Rabinovich
Magic Leap
Sunnyvale, CA
arabinovich@magicleap.com

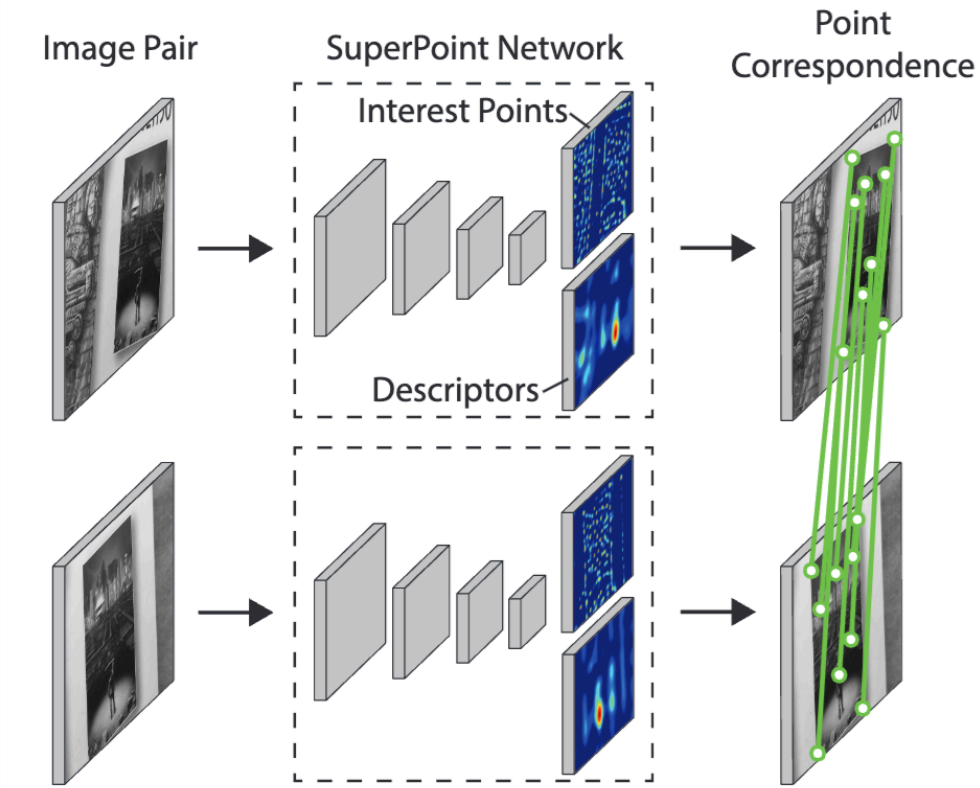


Figure 1. **SuperPoint for Geometric Correspondences.** We present a fully-convolutional neural network that computes SIFT-like 2D interest point locations and descriptors in a single forward pass and runs at 70 FPS on 480×640 images with a Titan X GPU.

SuperPoint: Self-Supervised Interest Point Detection and Description

Daniel DeTone
Magic Leap
Sunnyvale, CA
ddetone@magicleap.com

Tomasz Malisiewicz
Magic Leap
Sunnyvale, CA
tmalisiewicz@magicleap.com

Andrew Rabinovich
Magic Leap
Sunnyvale, CA
arabinovich@magicleap.com

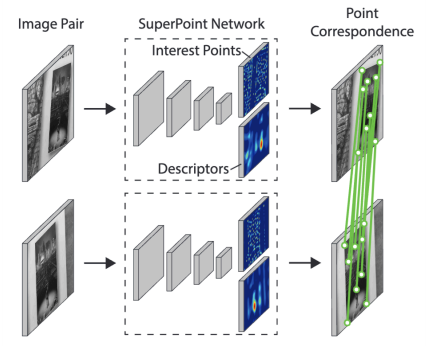


Figure 1. **SuperPoint for Geometric Correspondences.** We present a fully-convolutional neural network that computes SIFT-like 2D interest point locations and descriptors in a single forward pass and runs at 70 FPS on 480 × 640 images with a Titan X GPU.

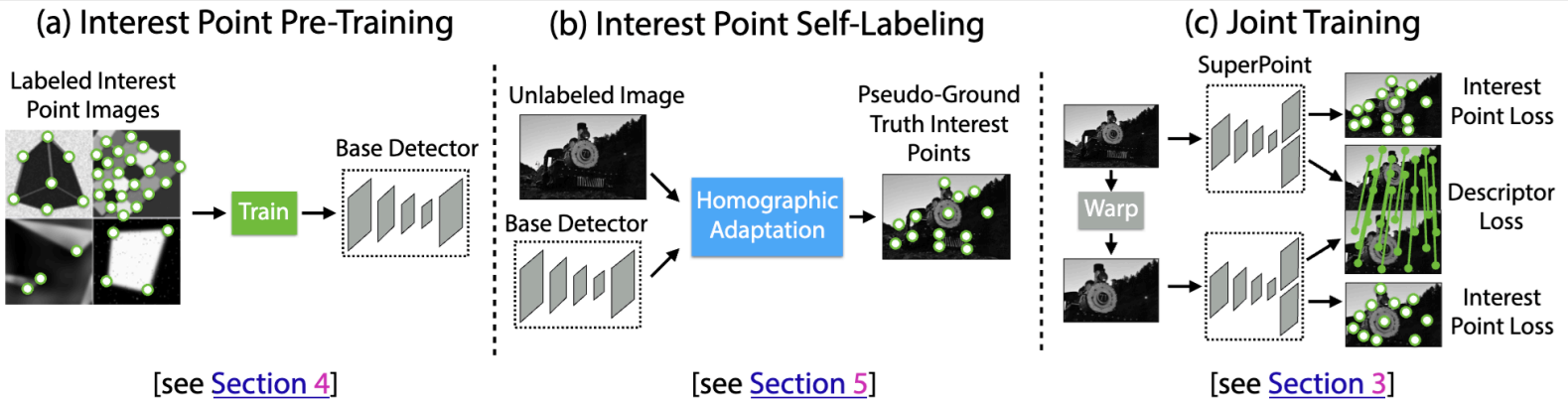


Figure 2. **Self-Supervised Training Overview.** In our self-supervised approach, we (a) pre-train an initial interest point detector on synthetic data and (b) apply a novel Homographic Adaptation procedure to automatically label images from a target, unlabeled domain. The generated labels are used to (c) train a fully-convolutional network that jointly extracts interest points and descriptors from an image.

SuperGlue: Learning Feature Matching with Graph Neural Networks

Paul-Edouard Sarlin[1*] Daniel DeTone[2] Tomasz Malisiewicz[2] Andrew Rabinovich[2]
[1] ETH Zurich [2] Magic Leap, Inc.

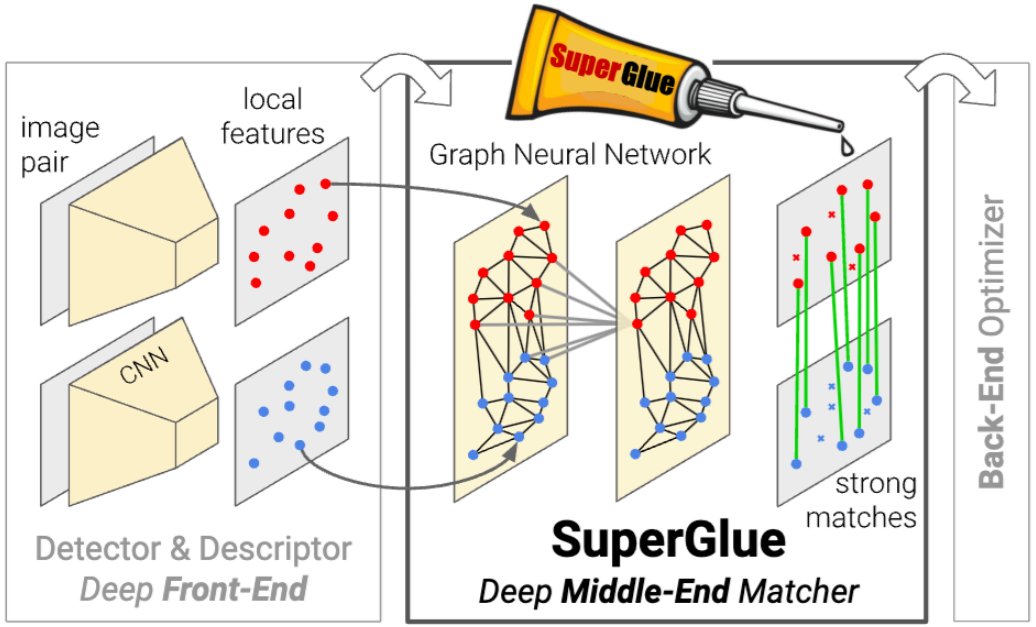


Figure 1: **Feature matching with SuperGlue.** Our approach establishes pointwise correspondences from off-the-shelf local features: it acts as a middle-end between hand-crafted or learned front-end and back-end. SuperGlue uses a graph neural network and attention to solve an assignment optimization problem, and handles partial point visibility and occlusion elegantly, producing a partial assignment.

SuperGlue: Learning Feature Matching with Graph Neural Networks

Paul-Edouard Sarlin[1]* Daniel DeTone[2] Tomasz Malisiewicz[2] Andrew Rabinovich[2]
[1] ETH Zurich [2] Magic Leap, Inc.

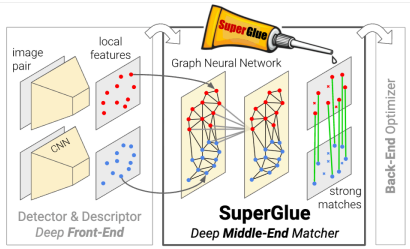


Figure 1: **Feature matching with SuperGlue.** Our approach establishes pointwise correspondences from off-the-shelf local features: it acts as a middle-end between hand-crafted or learned front-end and back-end. SuperGlue uses a graph neural network and attention to solve an assignment optimization problem, and handles partial point visibility and occlusion elegantly, producing a partial assignment.

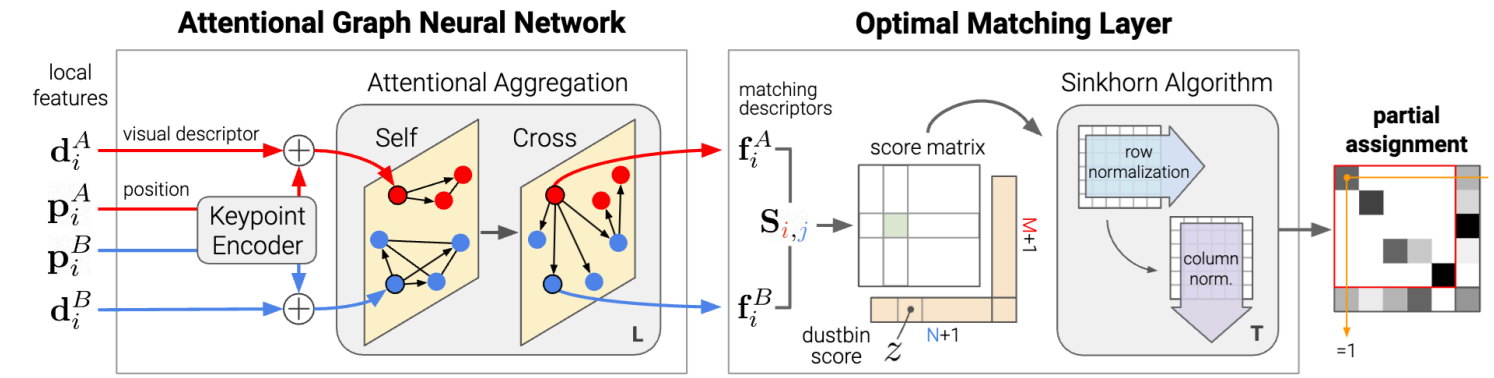
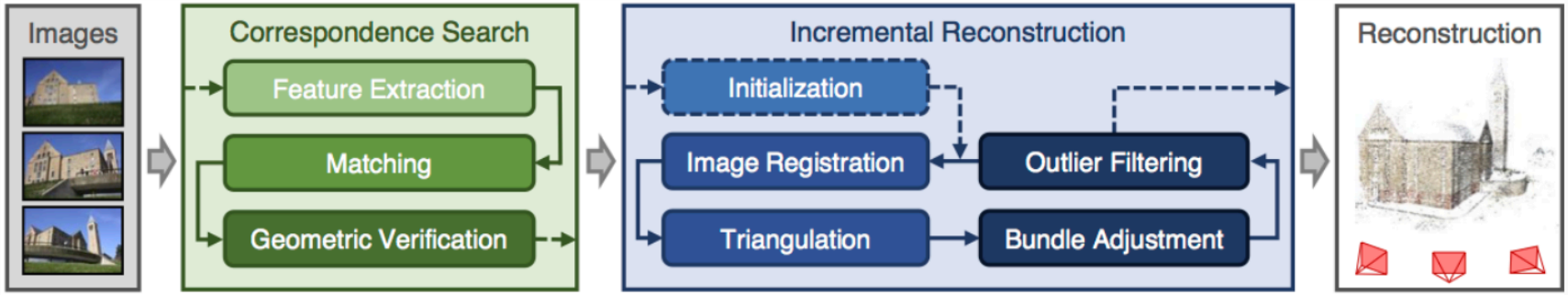
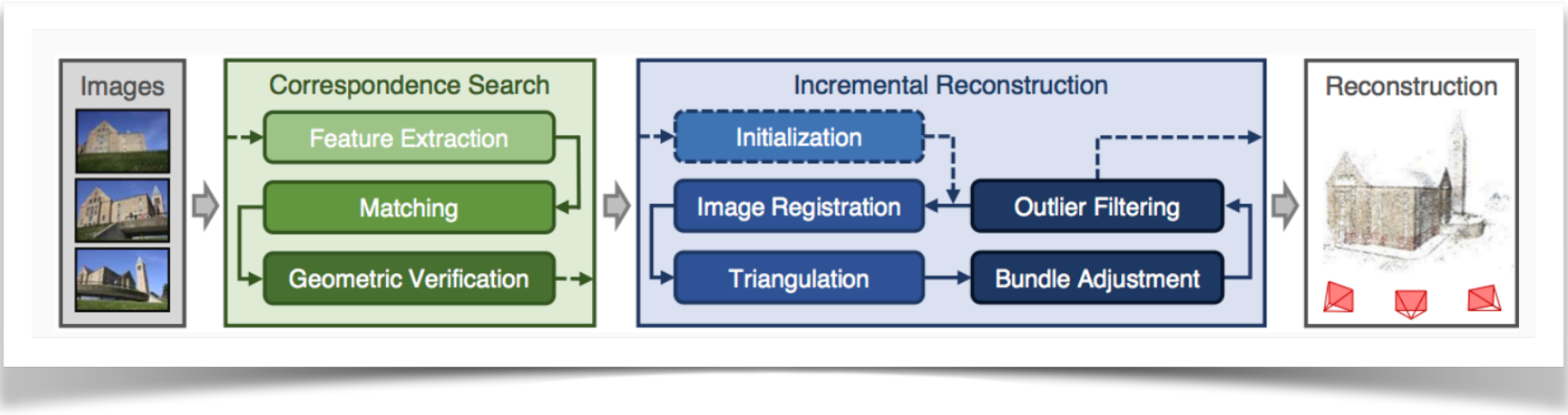


Figure 3: **The SuperGlue architecture.** SuperGlue is made up of two major components: the *attentional graph neural network* (Section 3.1), and the *optimal matching layer* (Section 3.2). The first component uses a *keypoint encoder* to map keypoint positions $\mathbf{p}$ and their visual descriptors $\mathbf{d}$ into a single vector, and then uses alternating self- and cross-attention layers (repeated L times) to create more powerful representations $\mathbf{f}$. The optimal matching layer creates an M by N score matrix, augments it with dustbins, then finds the optimal partial assignment using the Sinkhorn algorithm (for T iterations).

Where we are headed…



Where we are headed…



Deep Visual SLAM Frontends: SuperPoint, SuperGlue, and SuperMaps

Tomasz Malisiewicz
June 14, 2020

Joint Workshop on Long-Term Visual Localization, Visual Odometry and Geometric and Learning-based SLAM @ CVPR 2020

 https://tom.ai/