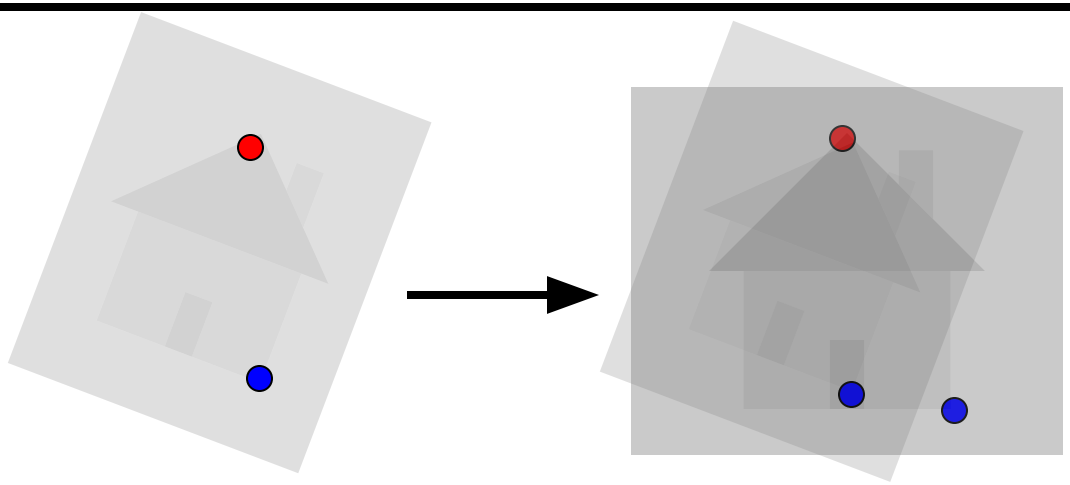
CSCE 448/748 - Computational Photography

Automatic Image Alignment and RANSAC

Nima Kalantari

Slides from Alexei A. Efros, James Hays, Richard Szeliski, and Steve Seitz

Image Alignment



How do we align two images automatically?

Two broad approaches:

- Feature-based alignment
  - Find a few matching features in both images
  - compute alignment
- Direct (pixel-based) alignment
  - Search for alignment where most pixels agree

Feature-based alignment

1. **Feature Detection**: find a few important features (aka Interest Points) in each image separately
2. **Feature Matching**: match them across two images
3. **Compute image transformation**: RANSAC

Feature-based alignment

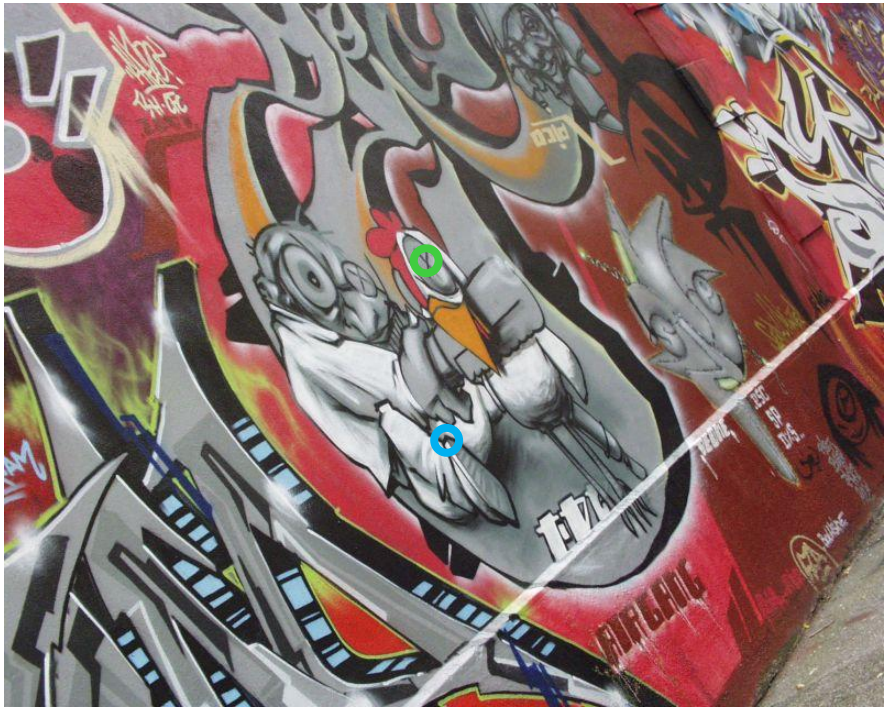
1. **Feature Detection**: find a few important features (aka Interest Points) in each image separately
2. **Feature Matching**: match them across two images
3. **Compute image transformation**: RANSAC

Feature detection

How do we choose good features automatically?

- They must be prominent in both images
- Easy to localize
- Think how you do that by hand

Example



Feature detection

How do we choose good features automatically?

- They must be prominent in both images
- Easy to localize
- Think how you do that by hand
- Corners!

Feature-based alignment

1. **Feature Detection**: find a few important features (aka Interest Points) in each image separately
2. **Feature Matching**: match them across two images
3. **Compute image transformation**: RANSAC

Feature-based alignment

1. **Feature Detection**: find a few important features (aka Interest Points) in each image separately
2. **Feature Matching**: match them across two images
3. **Compute image transformation**: RANSAC

Feature Matching

How do we match the features between the images?

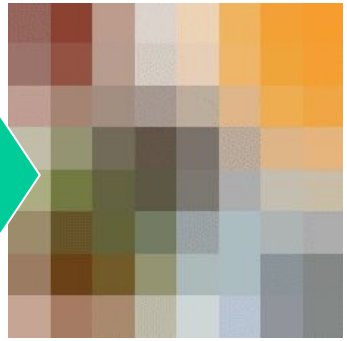
- Need a way to describe a region around each feature
  - e.g. image patch around each feature

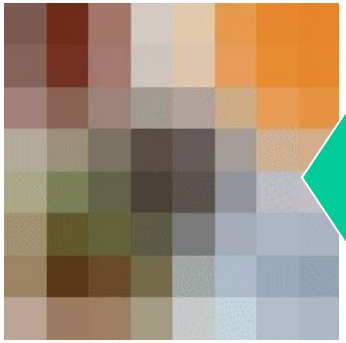
Issues:

- What if the image patches for several interest points look similar?
  - Make patch size bigger
- What if the image patches for the same feature look different due to scale, rotation, exposure etc.
  - Need an invariant descriptor

Invariant Feature Descriptors

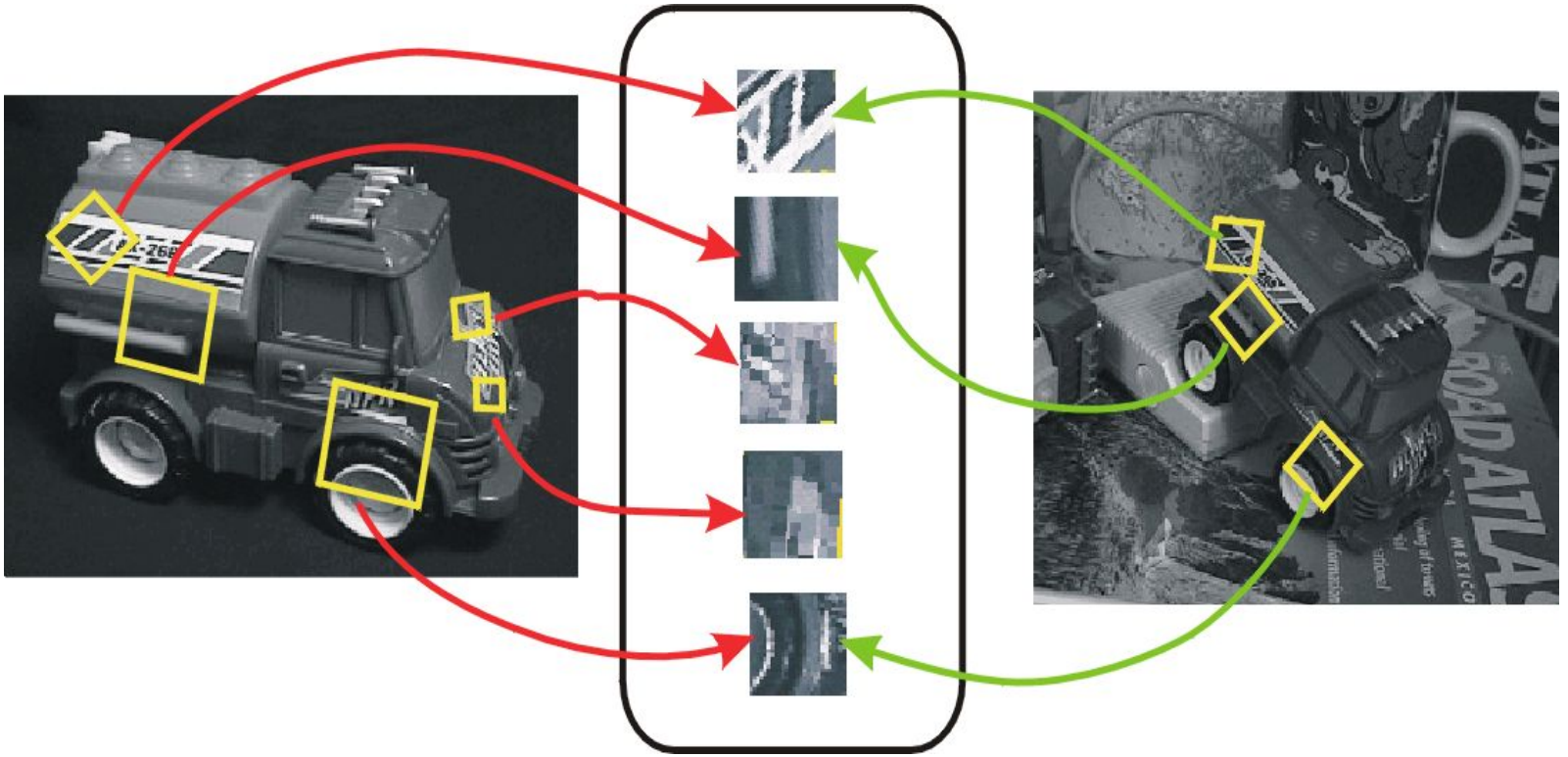
Schmid & Mohr 1997, Lowe 1999, Baumberg 2000, Tuytelaars & Van Gool 2000, Mikolajczyk & Schmid 2001, Brown & Lowe 2002, Matas et. al. 2002, Schaffalitzky & Zisserman 2002





Invariant Local Features

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



Features Descriptors

Applications

Feature points are used for:

- Image alignment (homography, fundamental matrix)
- 3D reconstruction
- Motion tracking
- Object recognition
- Scene categorization
- Indexing and database retrieval
- Robot navigation
- … other

Feature-based alignment

1. **Feature Detection**: find a few important features (aka Interest Points) in each image separately
2. **Feature Matching**: match them across two images
3. **Compute image transformation**: RANSAC

Feature-based alignment

1. **Feature Detection**: find a few important features (aka Interest Points) in each image separately
2. **Feature Matching**: match them across two images
3. **Compute image transformation**: RANSAC

Computing transformation

- Use successful matches to estimate homography
  - Need to do something to get rid of outliers

Outline

- How to automatically align two images
  - Feature detection
  - Feature description
  - Feature matching
  - RANSAC

Reading:

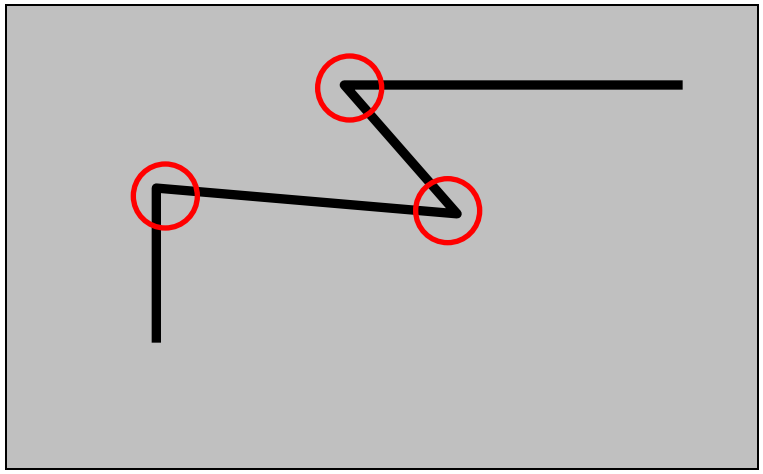
Multi-image Matching using Multi-scale image patches, CVPR 2005

Outline

- How to automatically align two images
  - Feature detection
  - Feature description
  - Feature matching
  - RANSAC

Harris corner detector

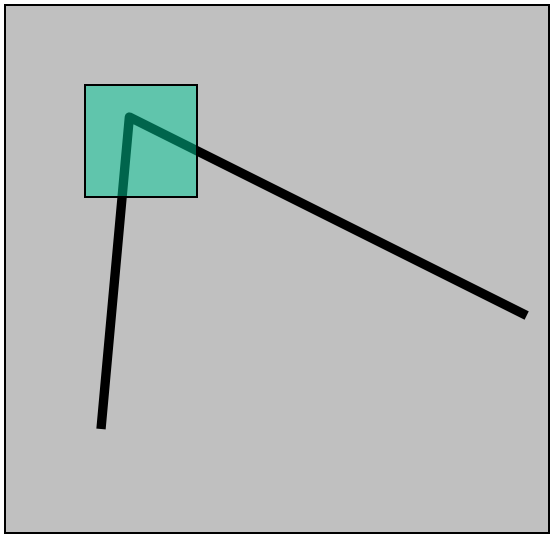
C.Harris, M.Stephens. “A Combined Corner and Edge Detector”. 1988



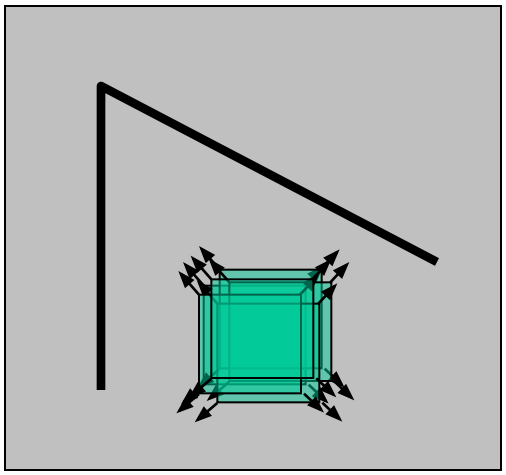
The Basic Idea

We should easily recognize the point by looking through a small window

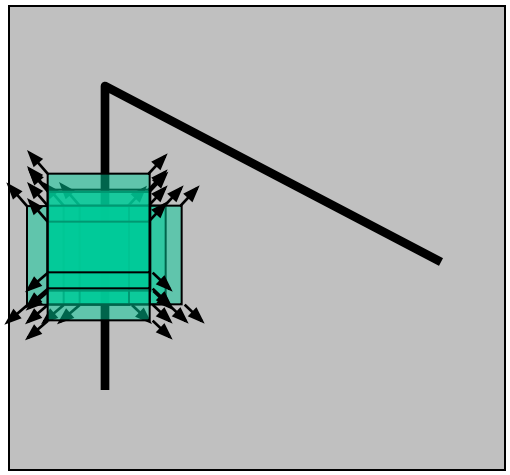
Shifting a window in *any direction* should give *a large change* in intensity



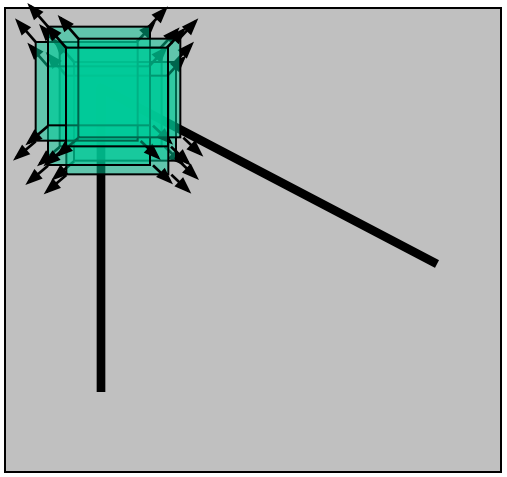
Harris Detector: Basic Idea



"flat" region:
no change in
all directions



"edge":
no change along
the edge direction



"corner":
significant change
in all directions

Harris Detector: Mathematics

Change of intensity for the shift $[u,v]$:

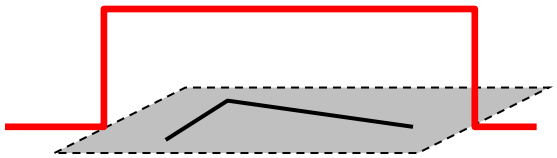
$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2$$

Window function

Shifted intensity

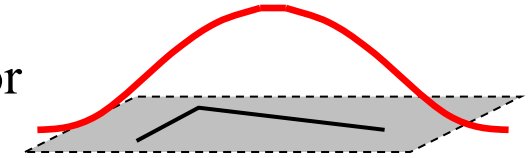
Intensity

Window function $w(x,y)$ =



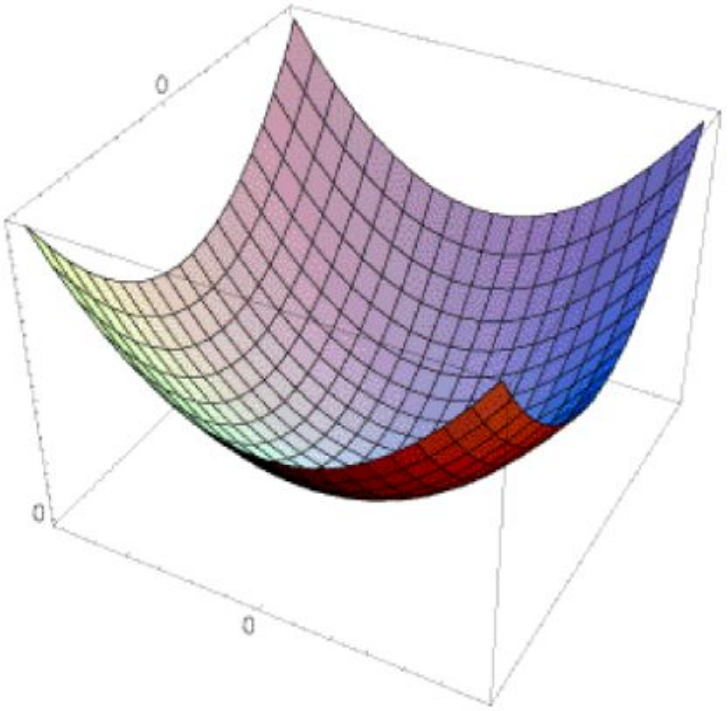
1 in window, 0 outside

or



Gaussian

Error surface



Harris Detector: Mathematics

For small shifts $[u,v]$ we can use Taylor Series expansion to get:

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

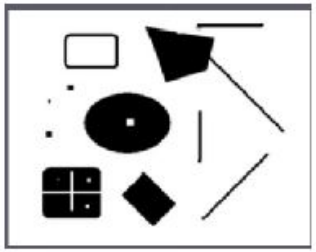
where M is a 2×2 matrix computed from image derivatives:

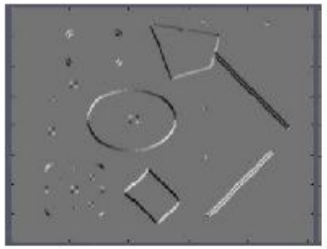
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Corners as distinctive interest points

$$M = \sum \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

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





Notation:

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

Slide from Kavita Bala

Harris Detector

The Algorithm:

- Construct matrix M
- Compute the response function R
- Find points with large corner response function R (R > threshold)
- Take the points of local maxima of R

Harris Detector: Mathematics

Measure of corner response:

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

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{a} & \boldsymbol{b} \\ \boldsymbol{c} & \boldsymbol{d} \end{bmatrix} \qquad \boldsymbol{\det M = ab - cd}$$

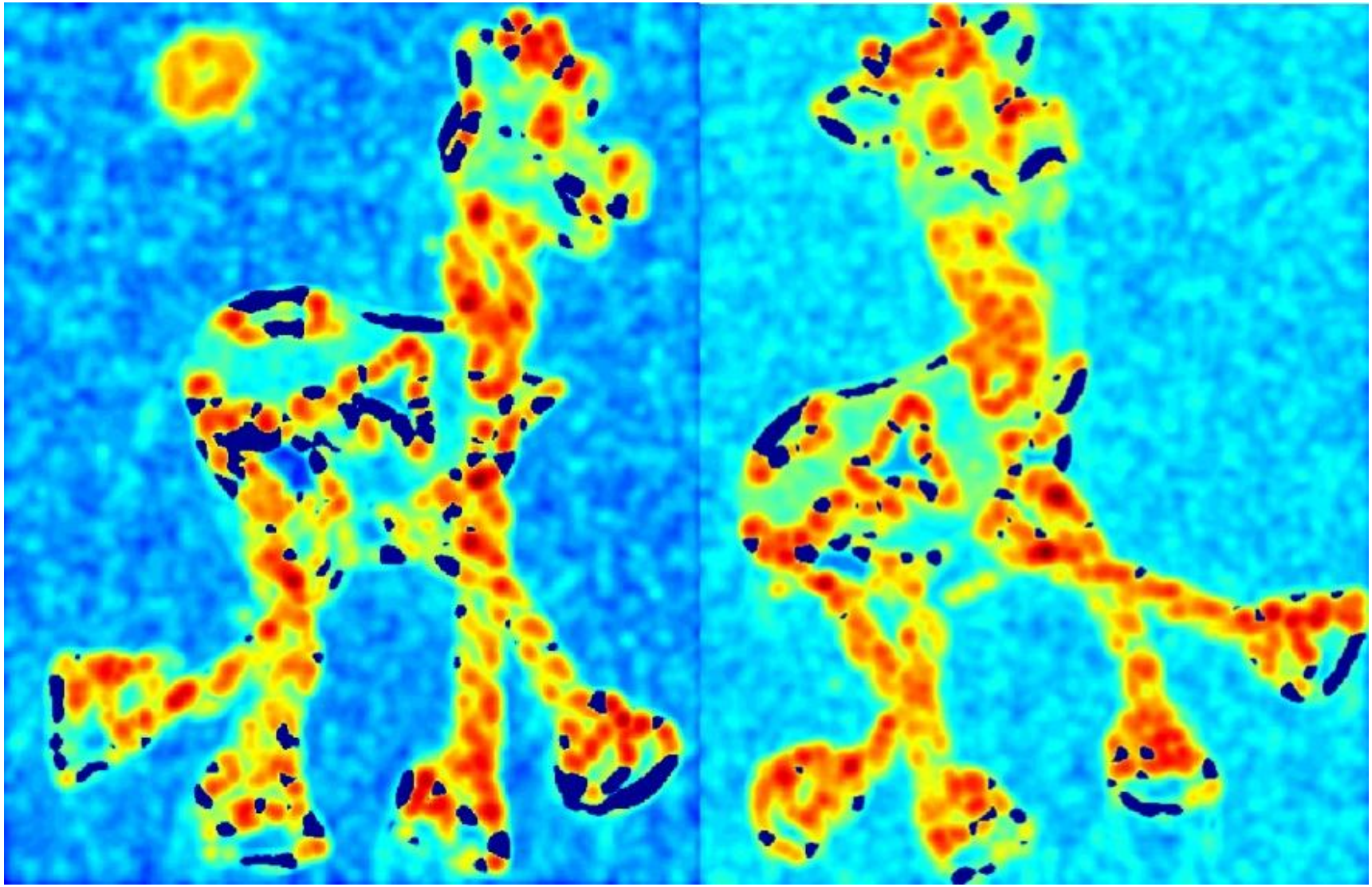
$$\textbf{trace M} = \textbf{a} + \textbf{d}$$

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

Harris Detector: Workflow

Harris Detector: Workflow

Compute corner response R



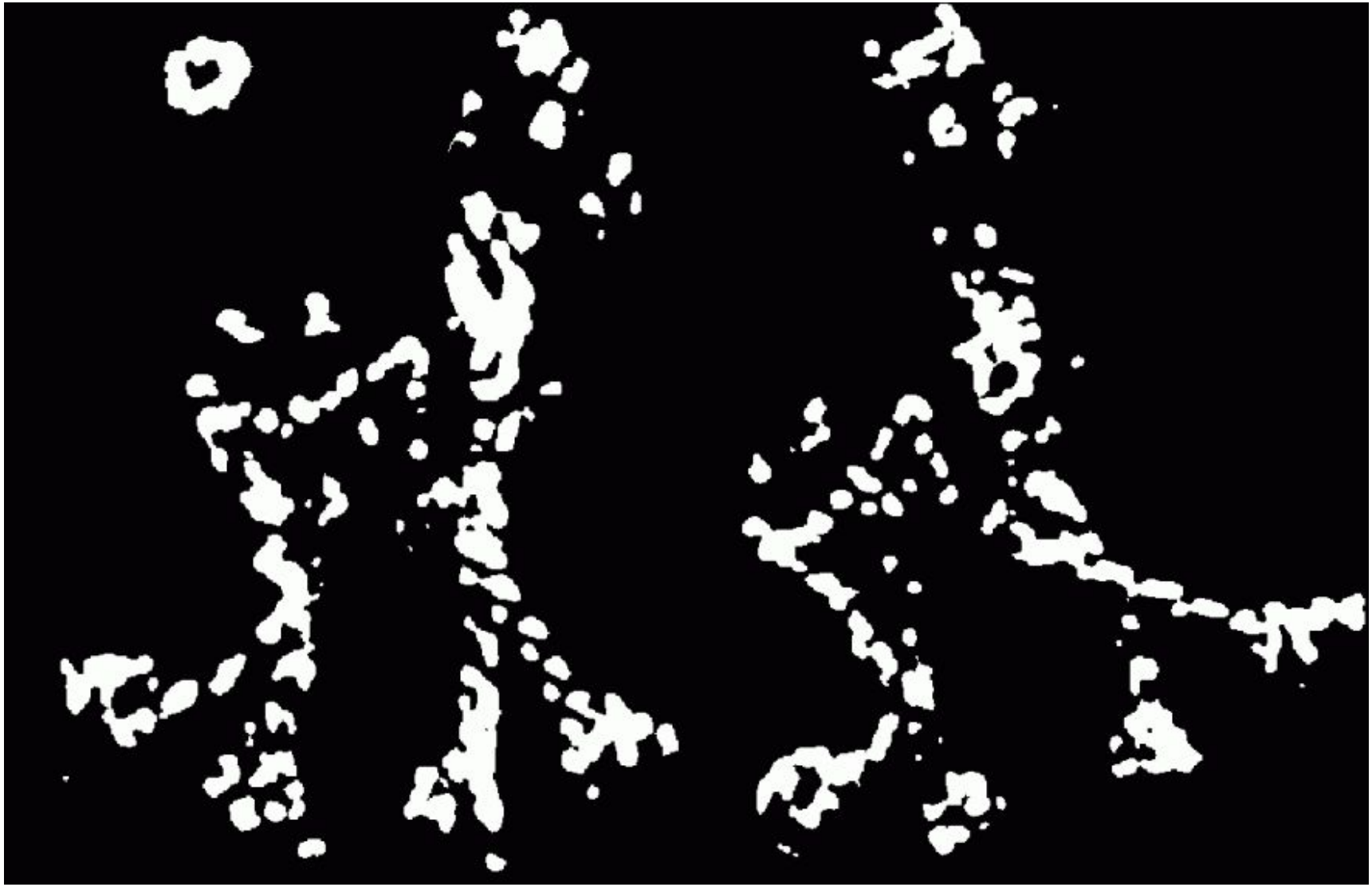
Harris Detector

The Algorithm:

- Construct matrix M
- Compute the response function R
- Find points with large corner response function R (R > threshold)
- Take the points of local maxima of R

Harris Detector: Workflow

Find points with large corner response: R>threshold



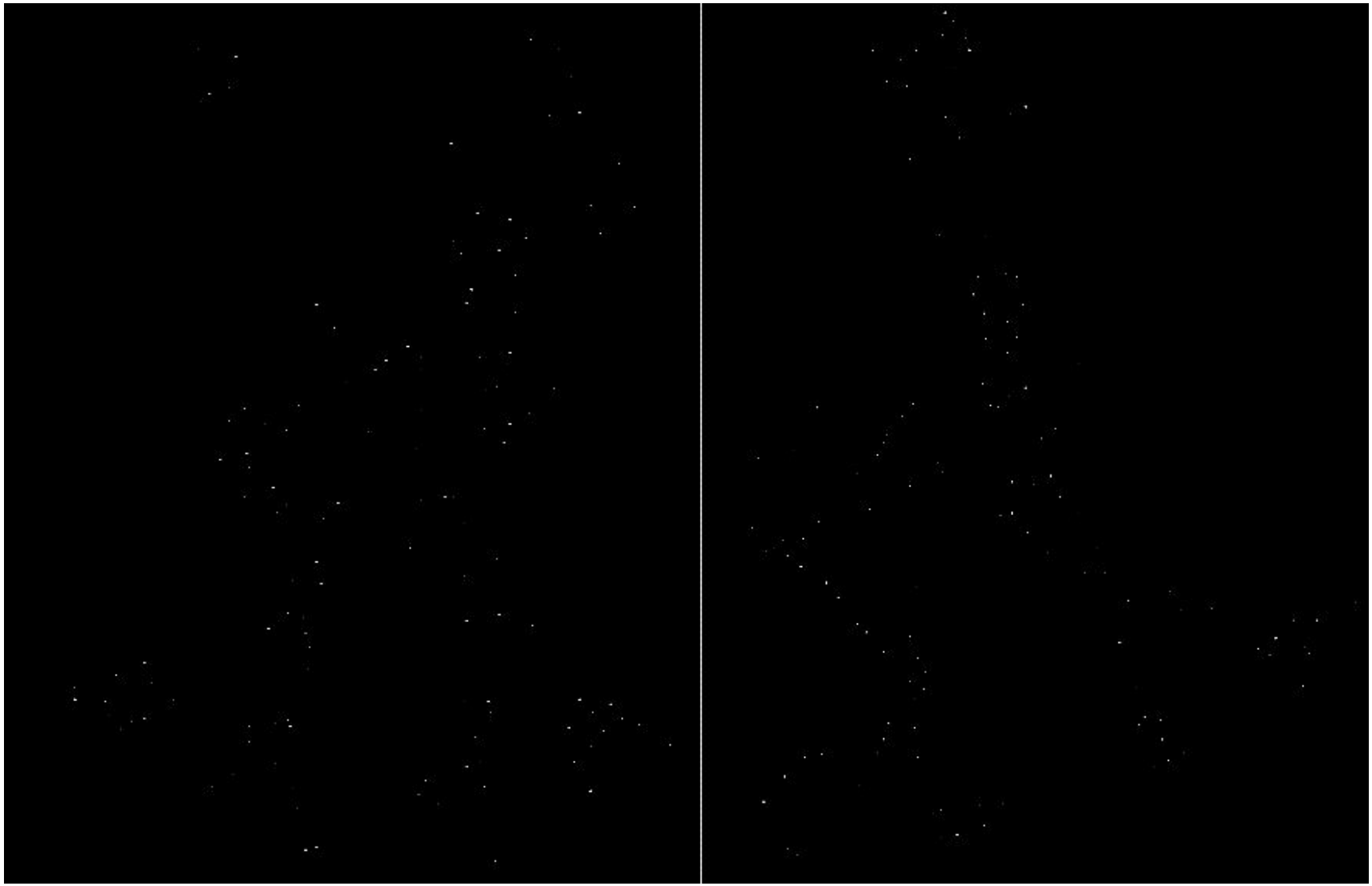
Harris Detector

The Algorithm:

- Construct matrix M
- Compute the response function R
- Find points with large corner response function R (R > threshold)
- Take the points of local maxima of R

Harris Detector: Workflow

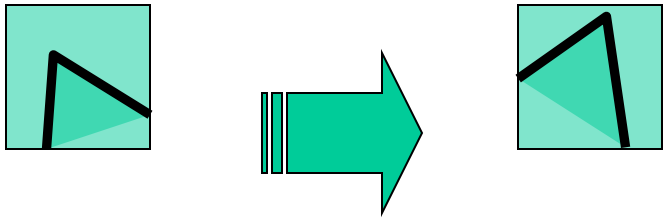
Take only the points of local maxima of R



Harris Detector: Workflow

Harris Detector: Some Properties

Rotation invariance

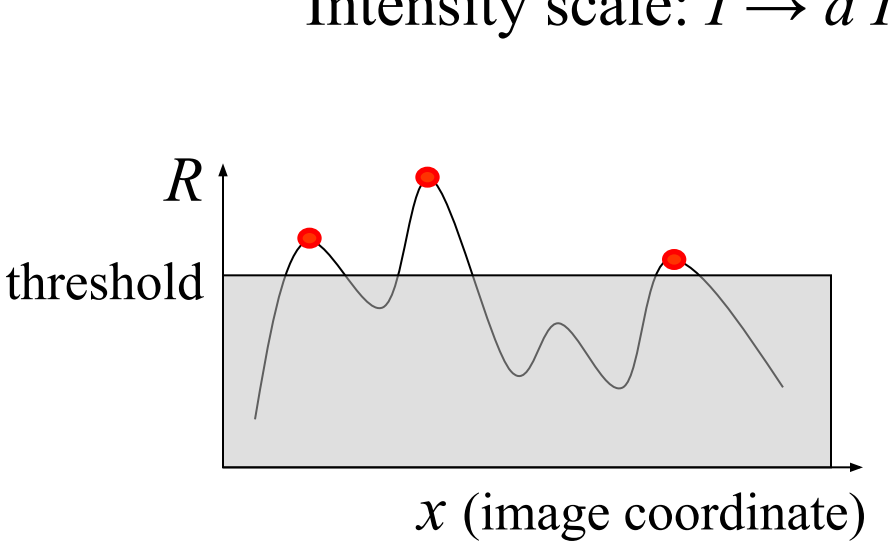


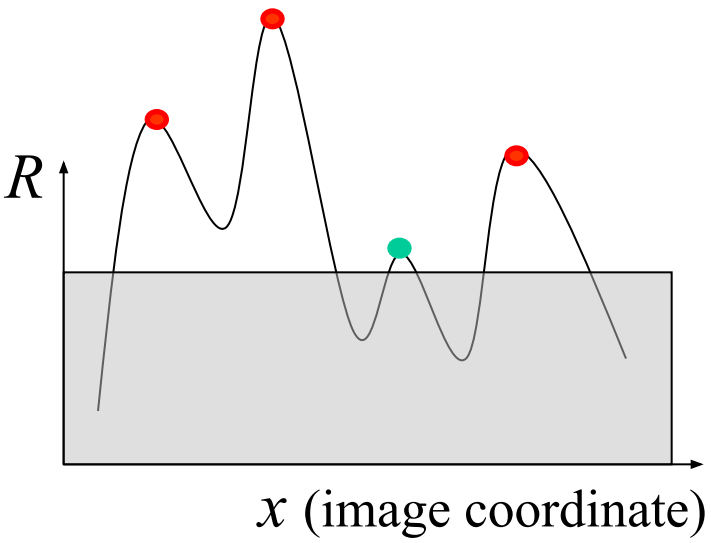
Corner response R is invariant to image rotation

Harris Detector: Some Properties

✔ Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$

Intensity scale: $I \rightarrow a\, I$

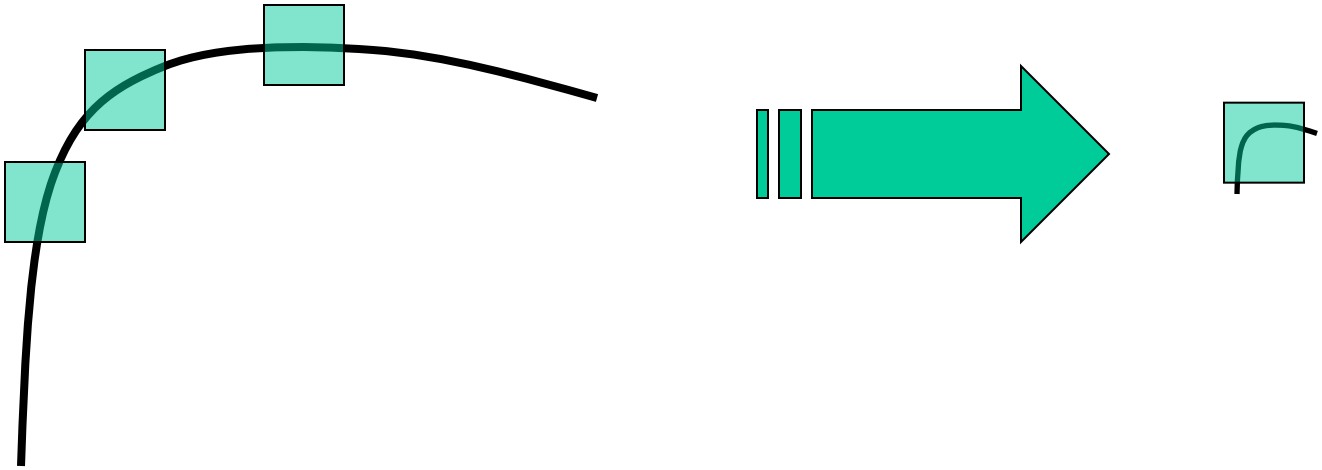




Corner response R is partially invariant to intensity scale

Harris Detector: Some Properties

But: non-invariant to *image scale*!



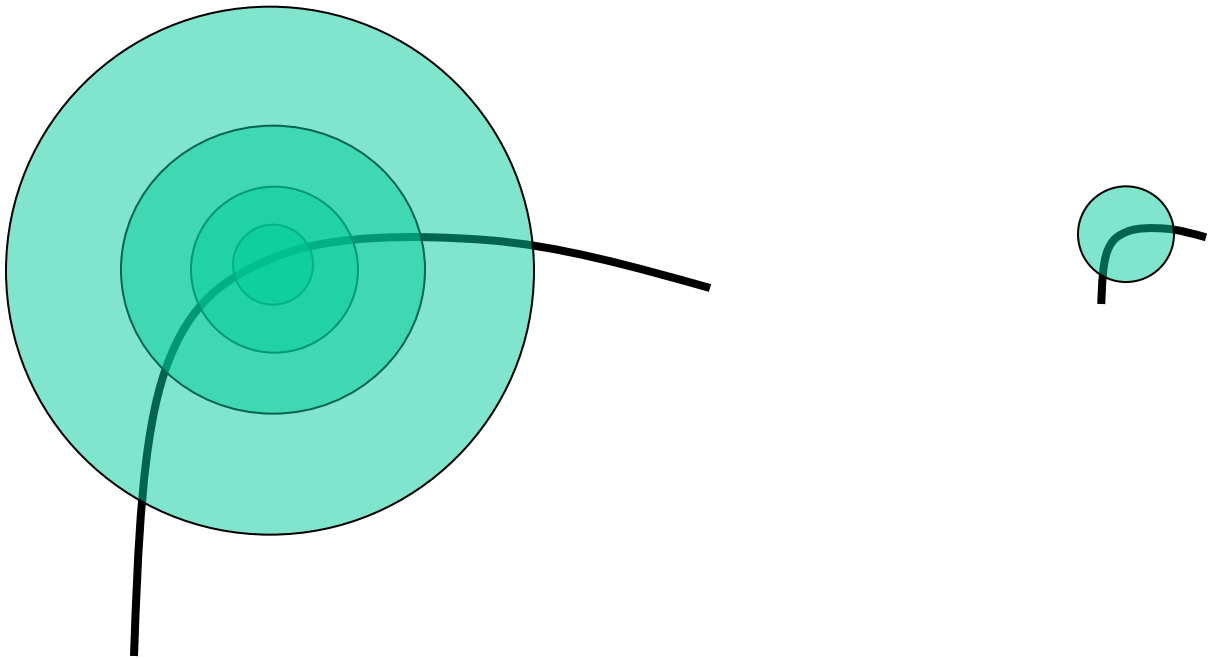
All points will be classified as edges

Corner !

Scale Invariant Detection

Consider regions (e.g. circles) of different sizes around a point

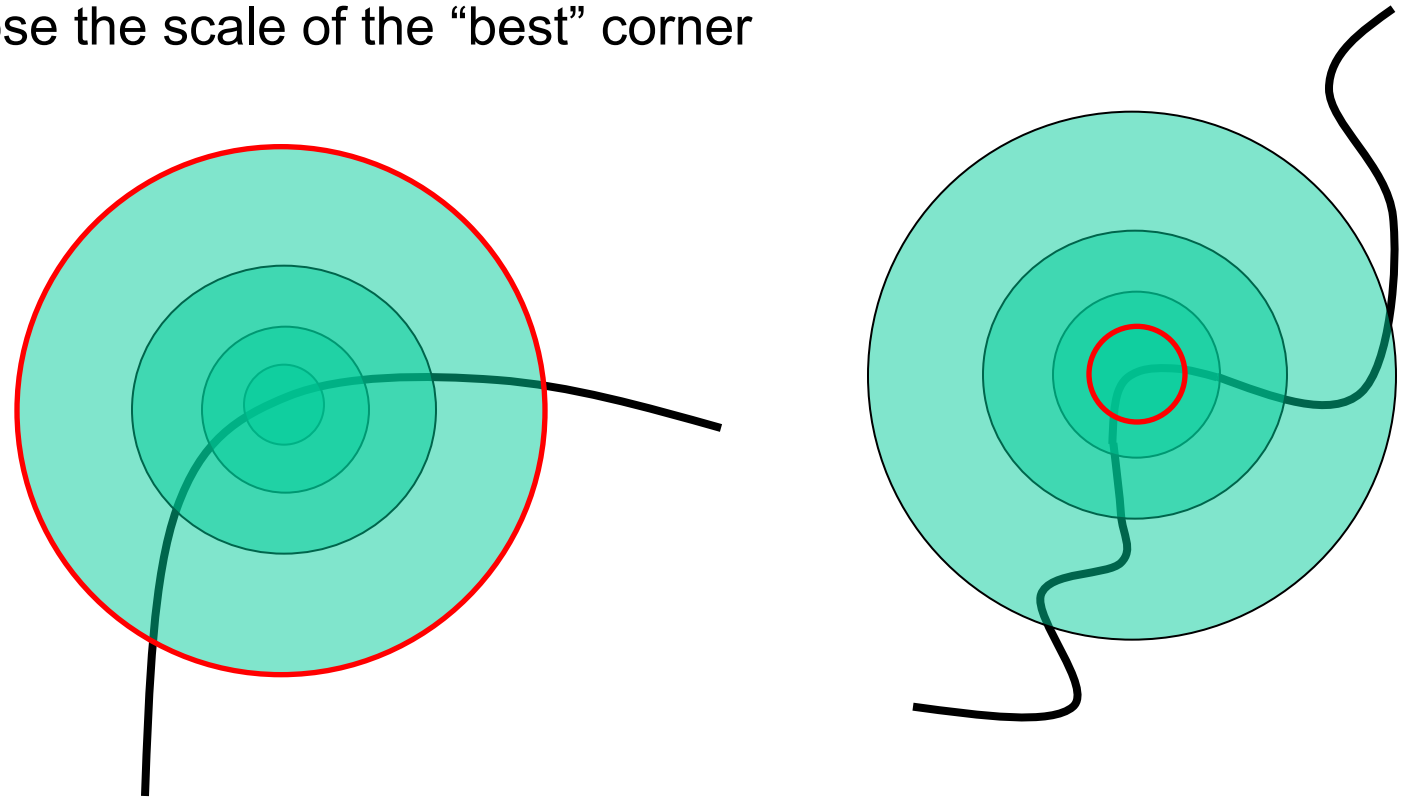
Regions of corresponding sizes will look the same in both images



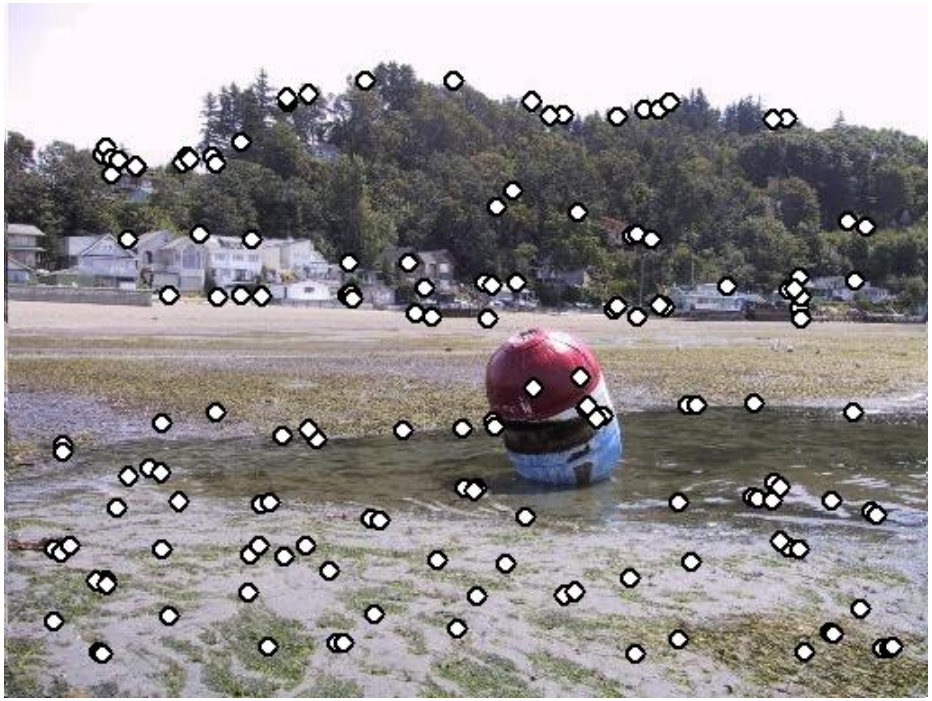
Scale Invariant Detection

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

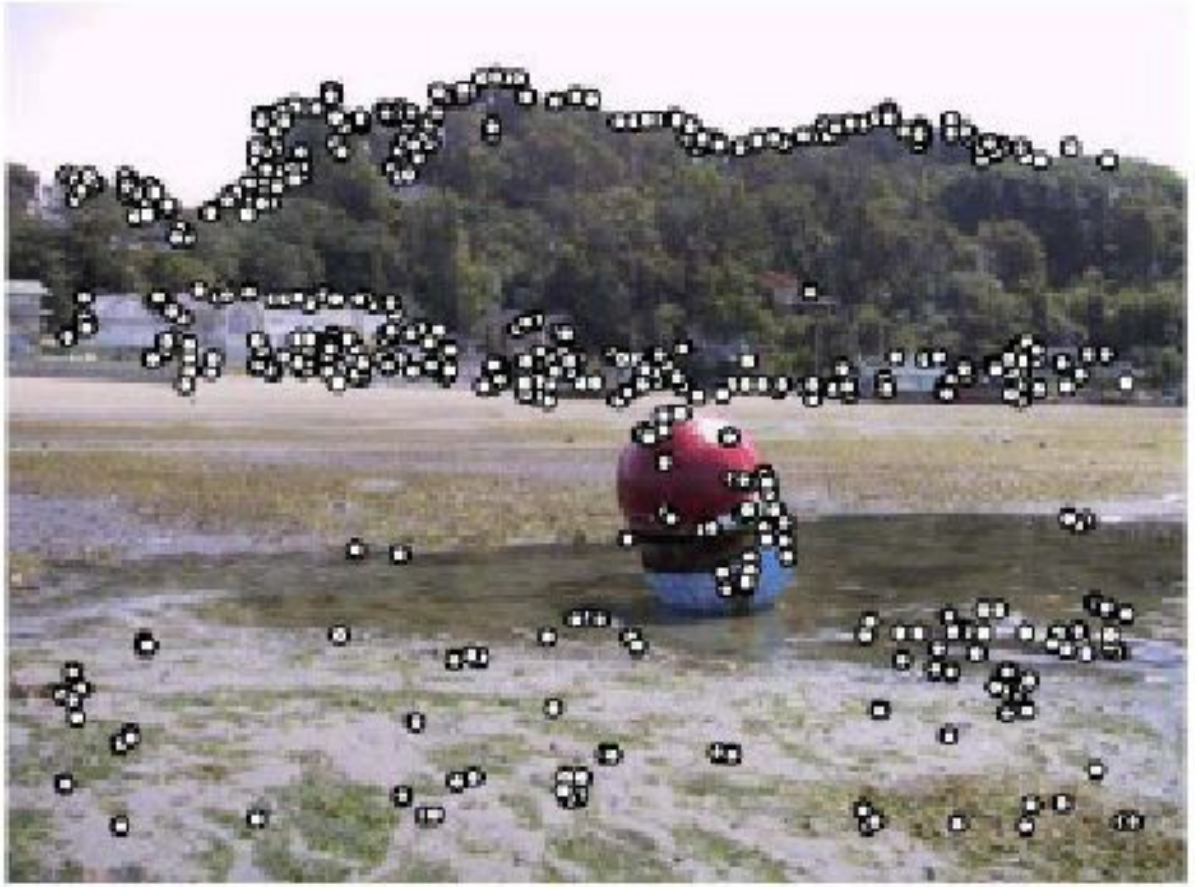
Choose the scale of the “best” corner



Feature selection



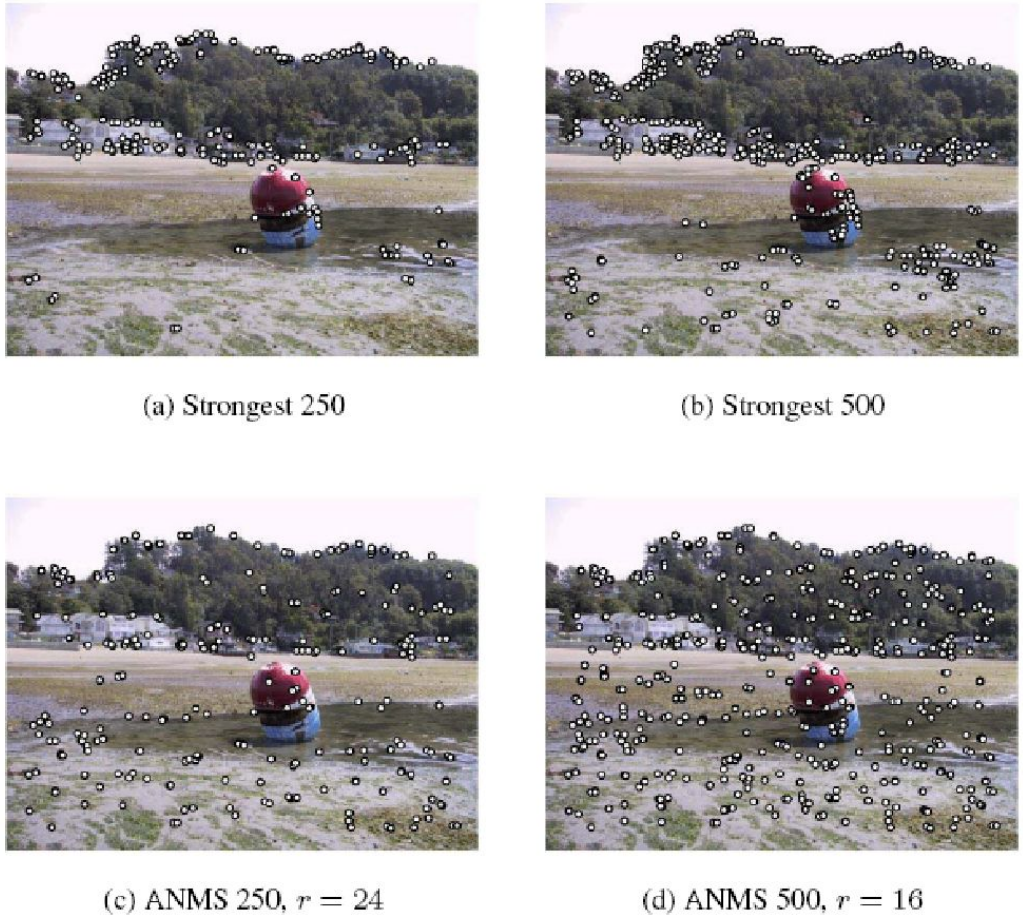
Feature selection



Adaptive Non-maximal Suppression

Desired: Fixed # of features per image

- Want evenly distributed spatially…
- Sort points by non-maximal suppression radius [Brown, Szeliski, Winder, CVPR'05]



(a) Strongest 250

(b) Strongest 500

(c) ANMS 250, $r = 24$

(d) ANMS 500, $r = 16$

Outline

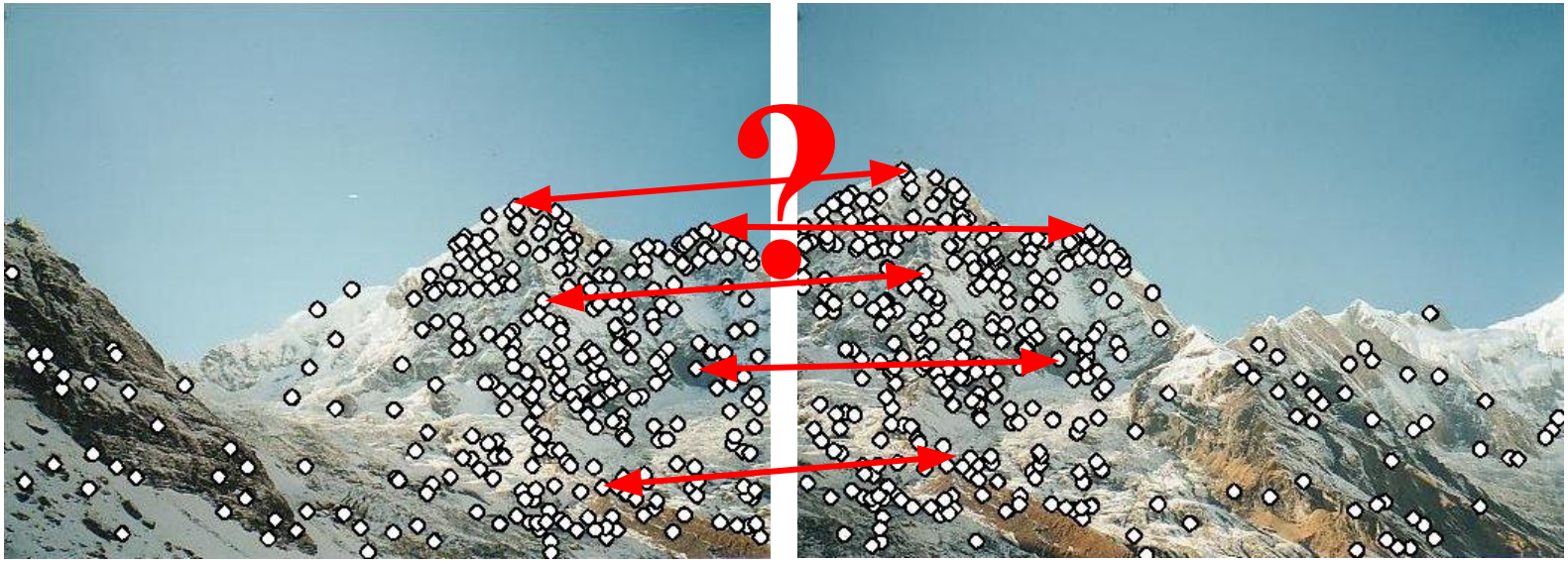
- How to automatically align two images
  - Feature detection
  - Feature description
  - Feature matching
  - RANSAC

Outline

- How to automatically align two images
  - Feature detection
  - Feature description
  - Feature matching
  - RANSAC

Feature descriptors

We know how to detect points
Next question: **How to match them?**



Point descriptor should be:
1. Invariant
2. Distinctive

Multi-Scale Oriented Patches (MOPS)

Find local orientation

Dominant direction of gradient

- Extract image patches relative to this orientation

Detect Features, setup Frame

Orientation = blurred gradient

Rotation Invariant Frame

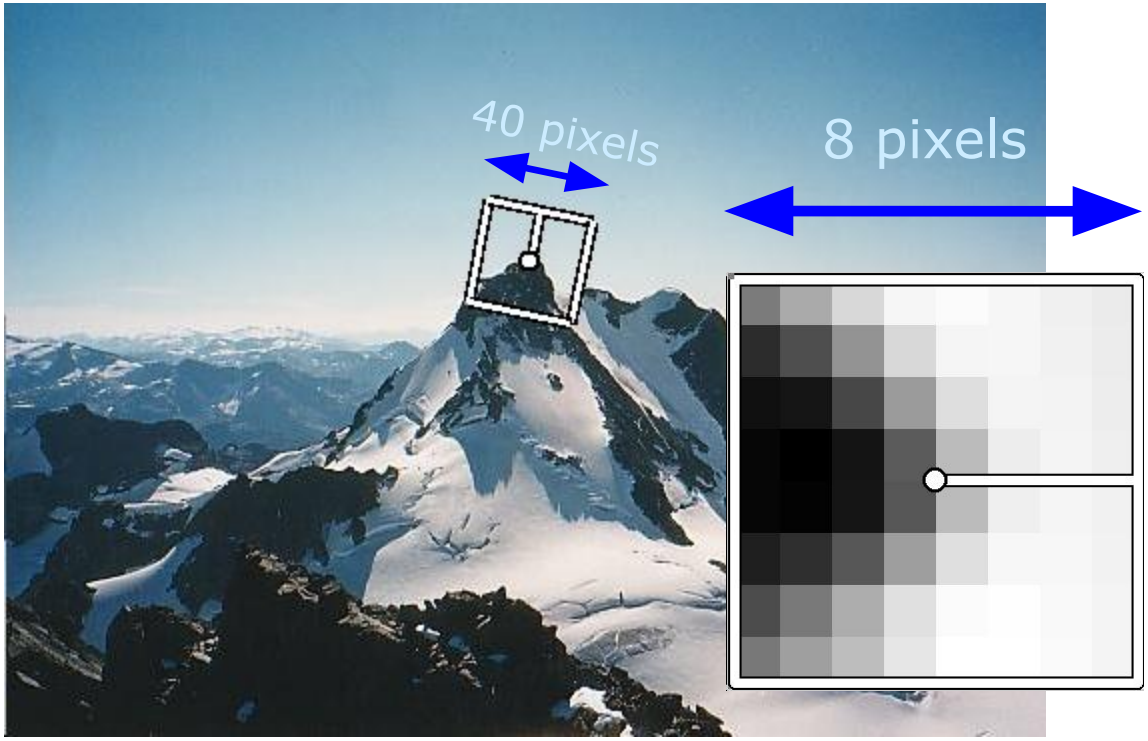
- Scale-space position (x, y, s) + orientation (θ)

MOPS descriptor vector

8x8 oriented patch

- Sampled at 5 x scale

Bias/gain normalisation: $I' = (I - \mu)/\sigma$



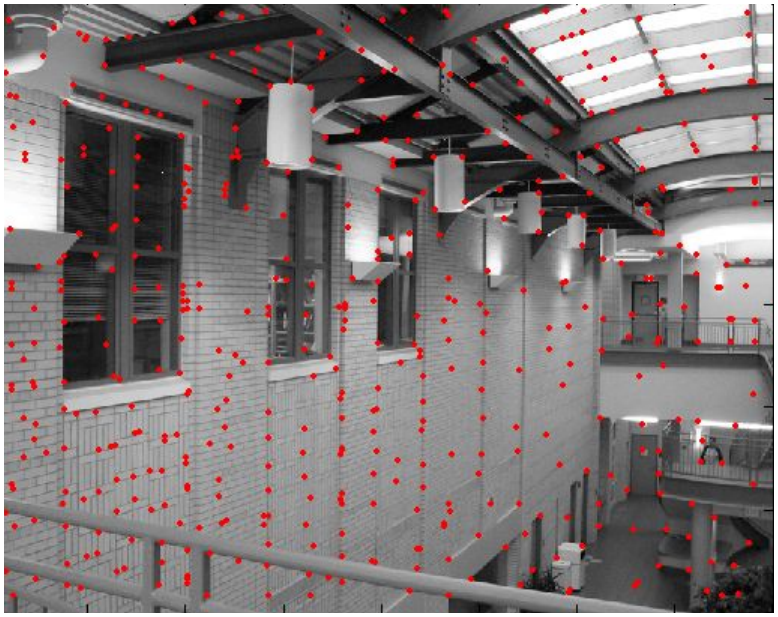
Outline

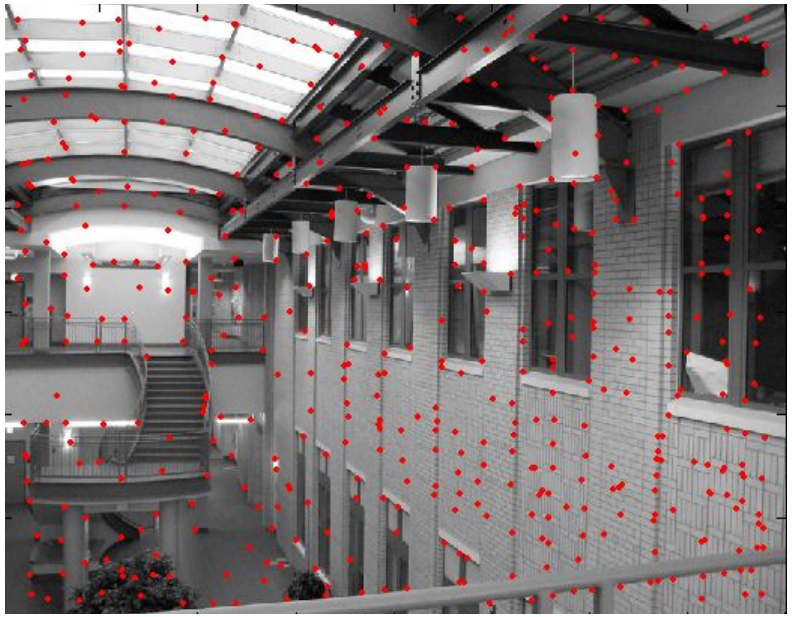
- How to automatically align two images
  - Feature detection
  - Feature description
  - Feature matching
  - RANSAC

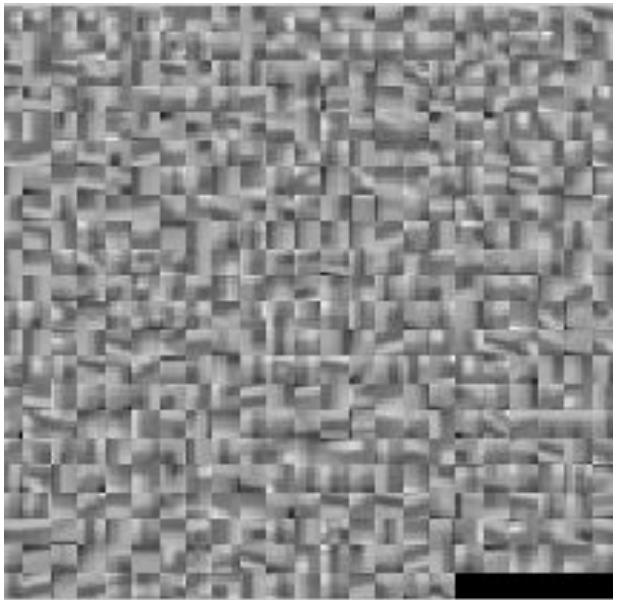
Outline

- How to automatically align two images
  - Feature detection
  - Feature description
  - Feature matching
  - RANSAC

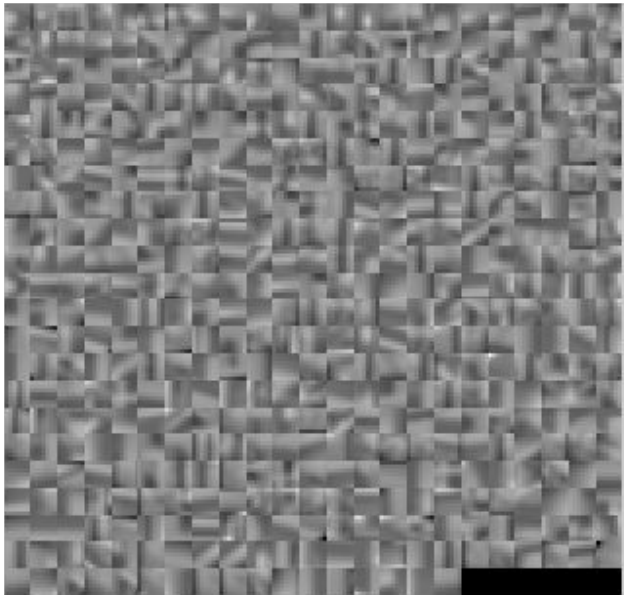
Feature matching







?



Feature matching

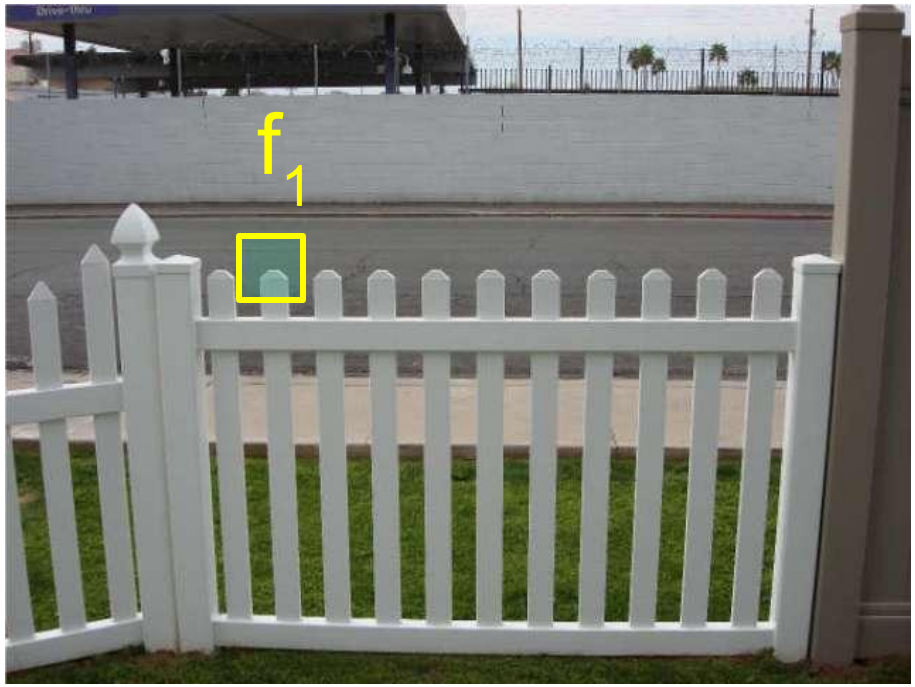
Given a feature in I_1, how to find the best match in I_2?

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

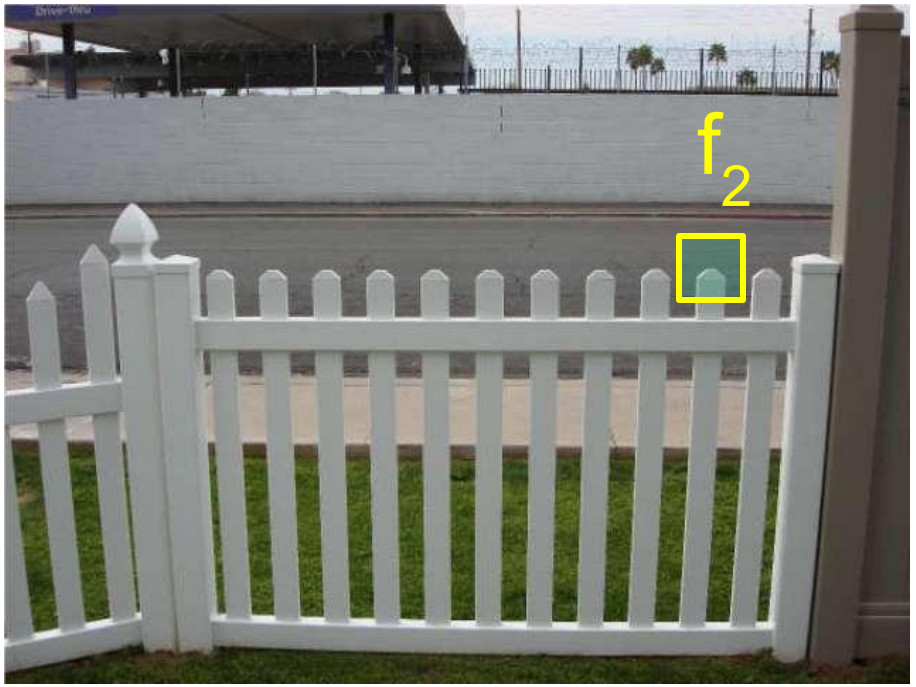
Feature distance

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

- Simple approach is SSD(f_1, f_2)
  - sum of square differences between entries of the two descriptors
  - can give good scores to very ambiguous (bad) matches



I_1

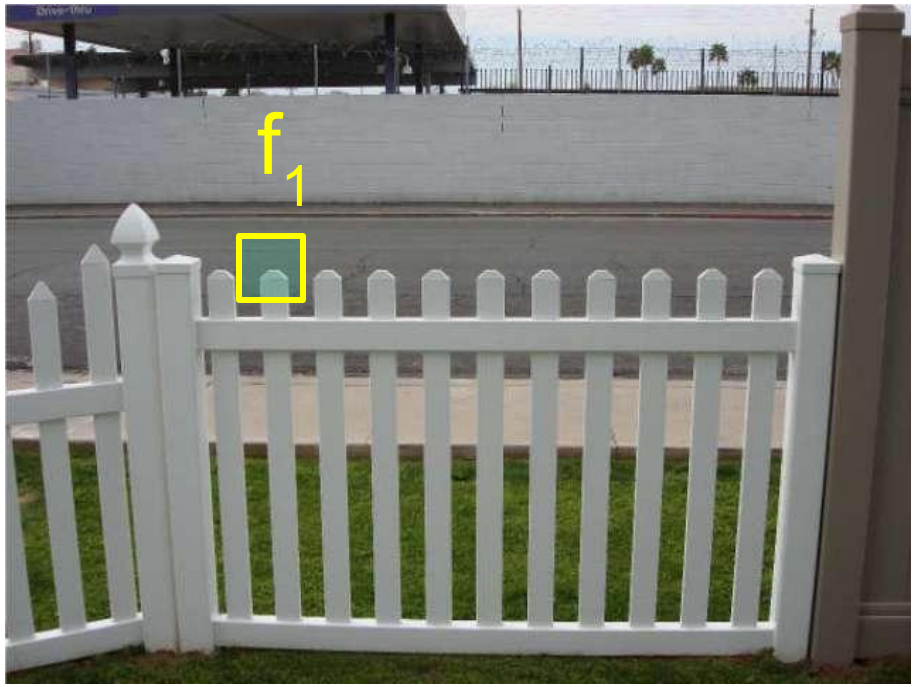


I_2

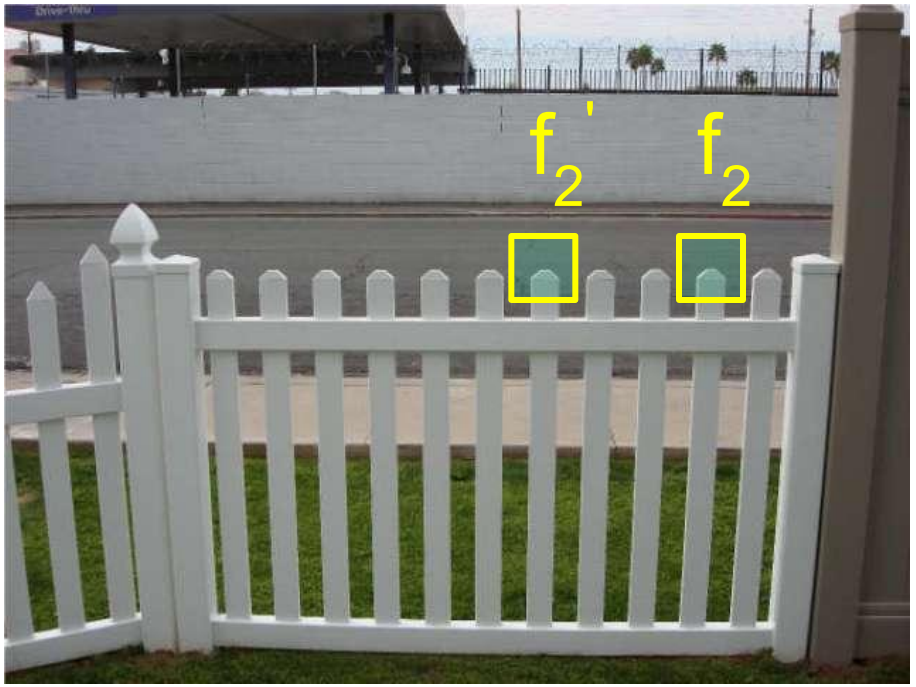
Feature distance

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

- Better approach: ratio distance = SSD(f_1, f_2) / SSD(f_1, f_2')
  - f_2 is best SSD match to f_1 in I_2
  - f_2' is 2nd best SSD match to f_1 in I_2
  - gives large values for ambiguous matches

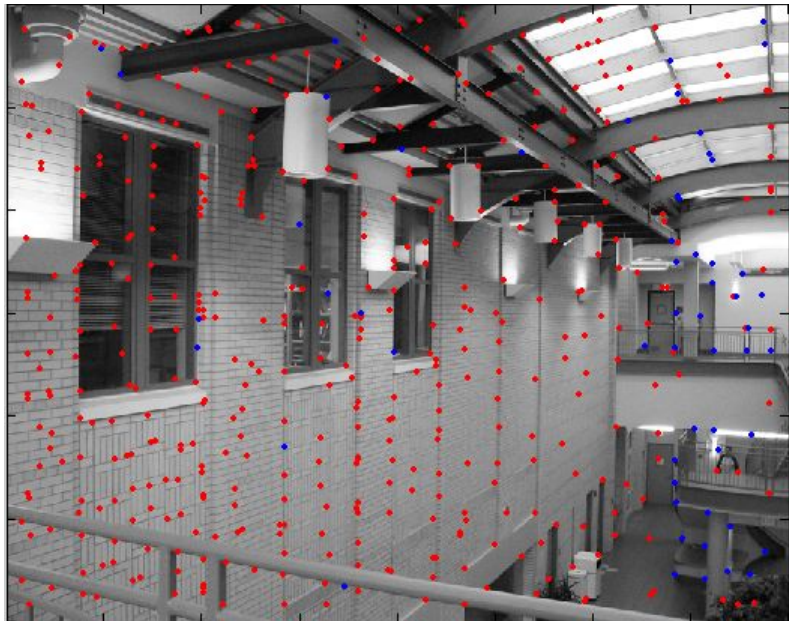


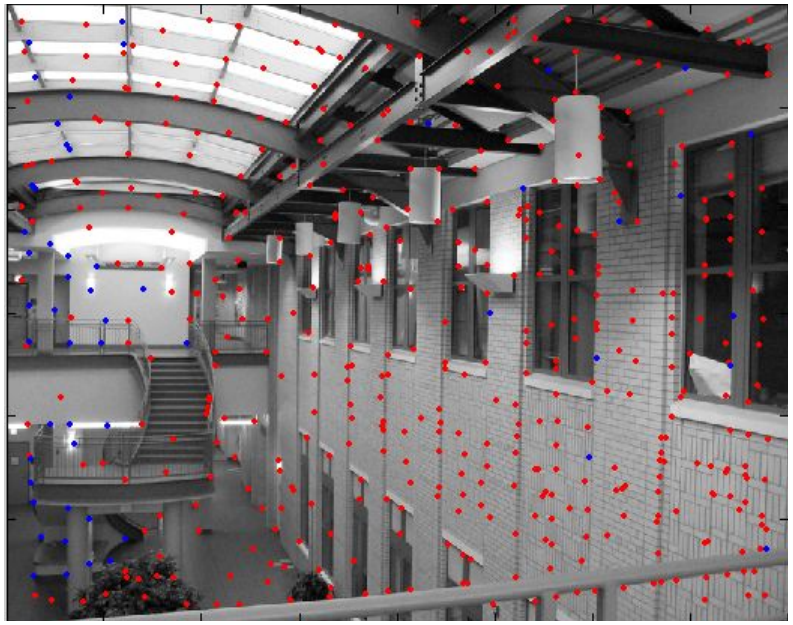
I_1



I_2

Feature-space outliner rejection





Can we now compute H from the blue points?

- No! Still too many outliers…
- What can we do?

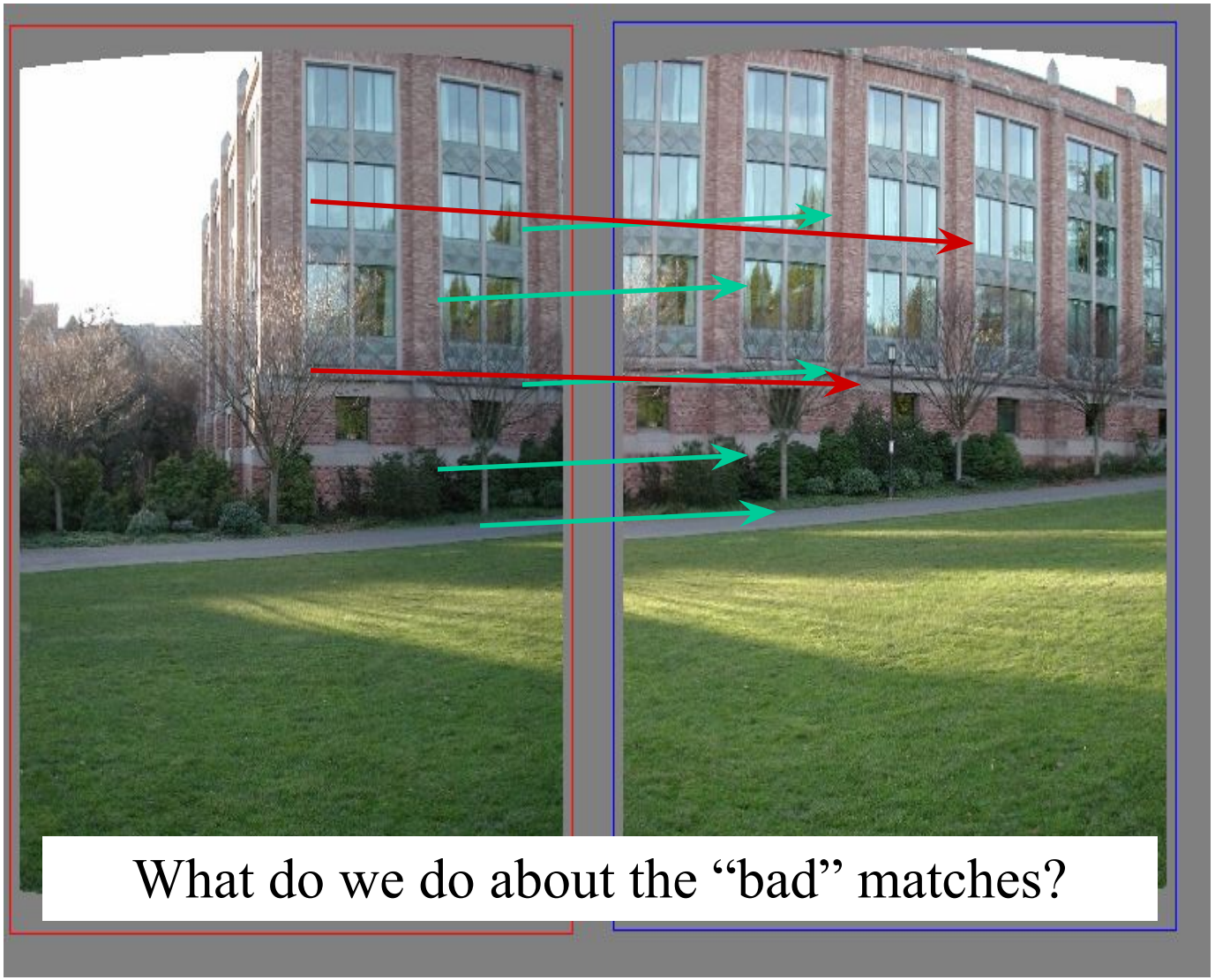
Outline

- How to automatically align two images
  - Feature detection
  - Feature description
  - Feature matching
  - RANSAC

Outline

- How to automatically align two images
  - Feature detection
  - Feature description
  - Feature matching
  - RANSAC

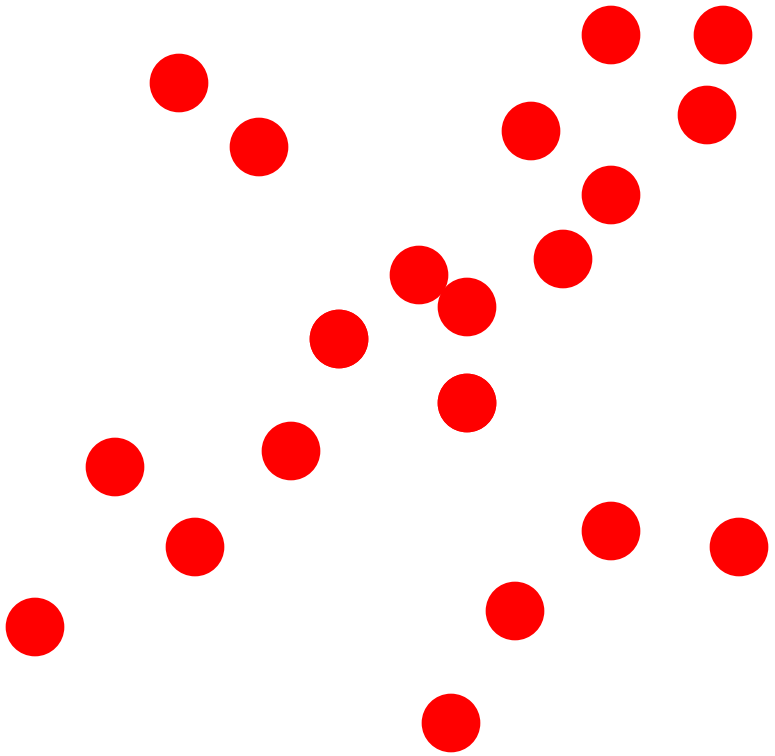
Matching features



RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.



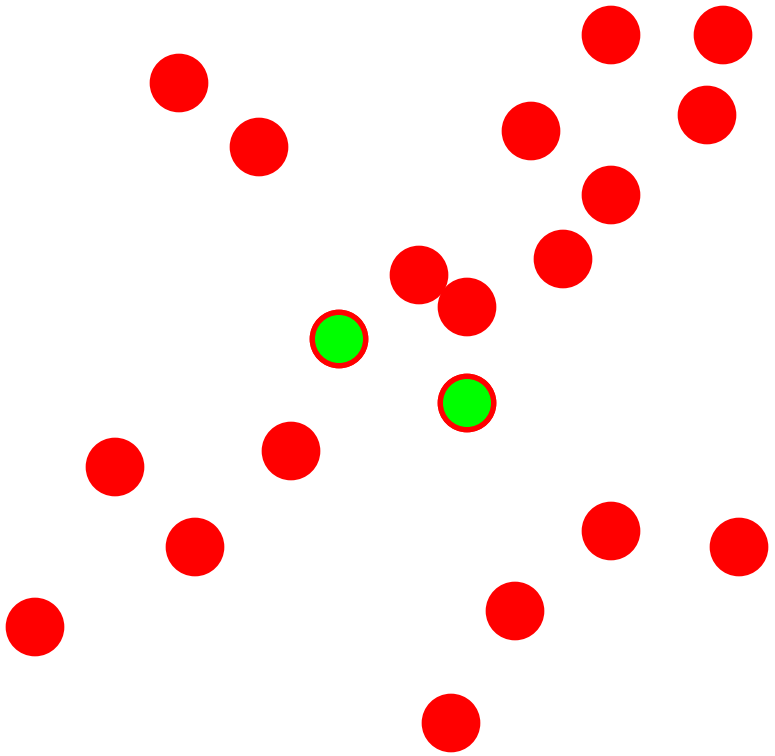
Algorithm:

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

RANSAC

Line fitting example



Algorithm:

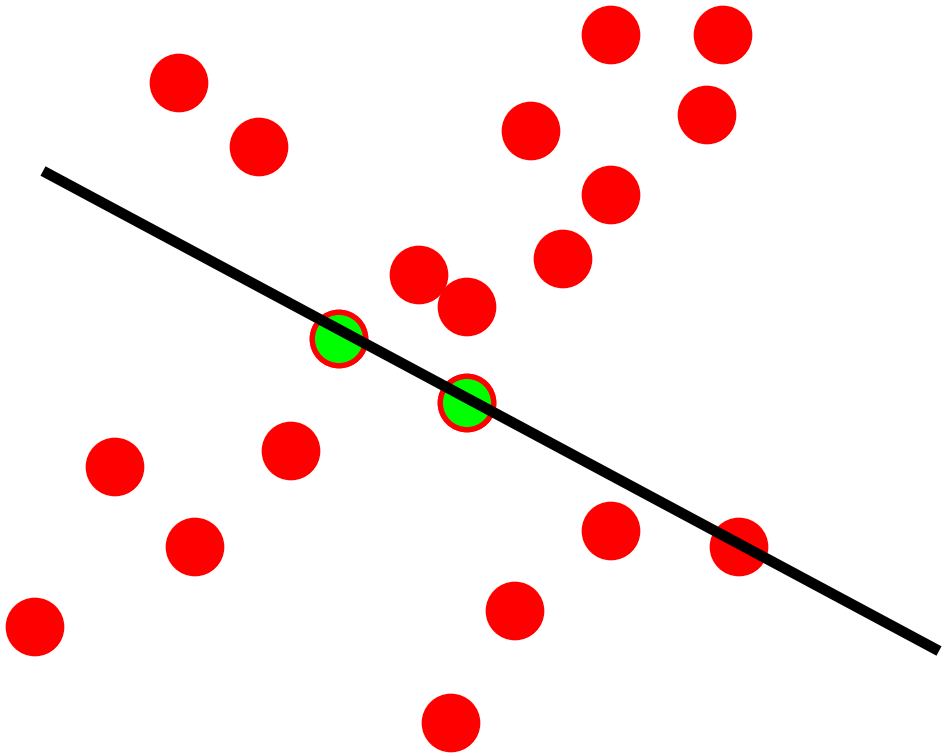
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

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

Illustration by Savarese

RANSAC

Line fitting example



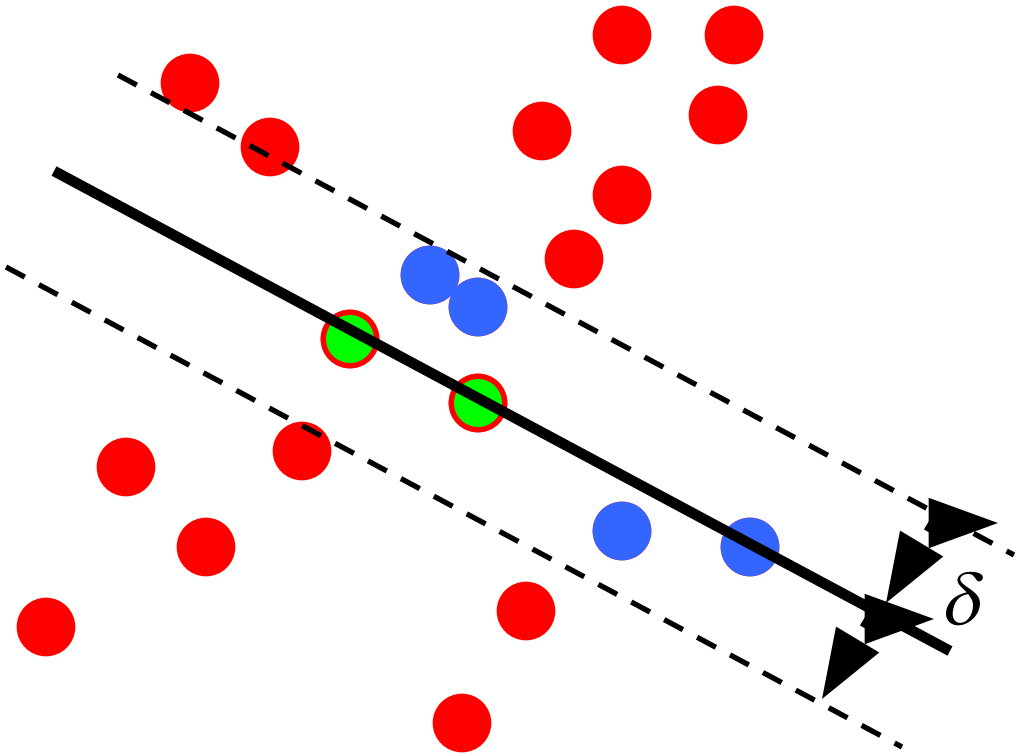
Algorithm:

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



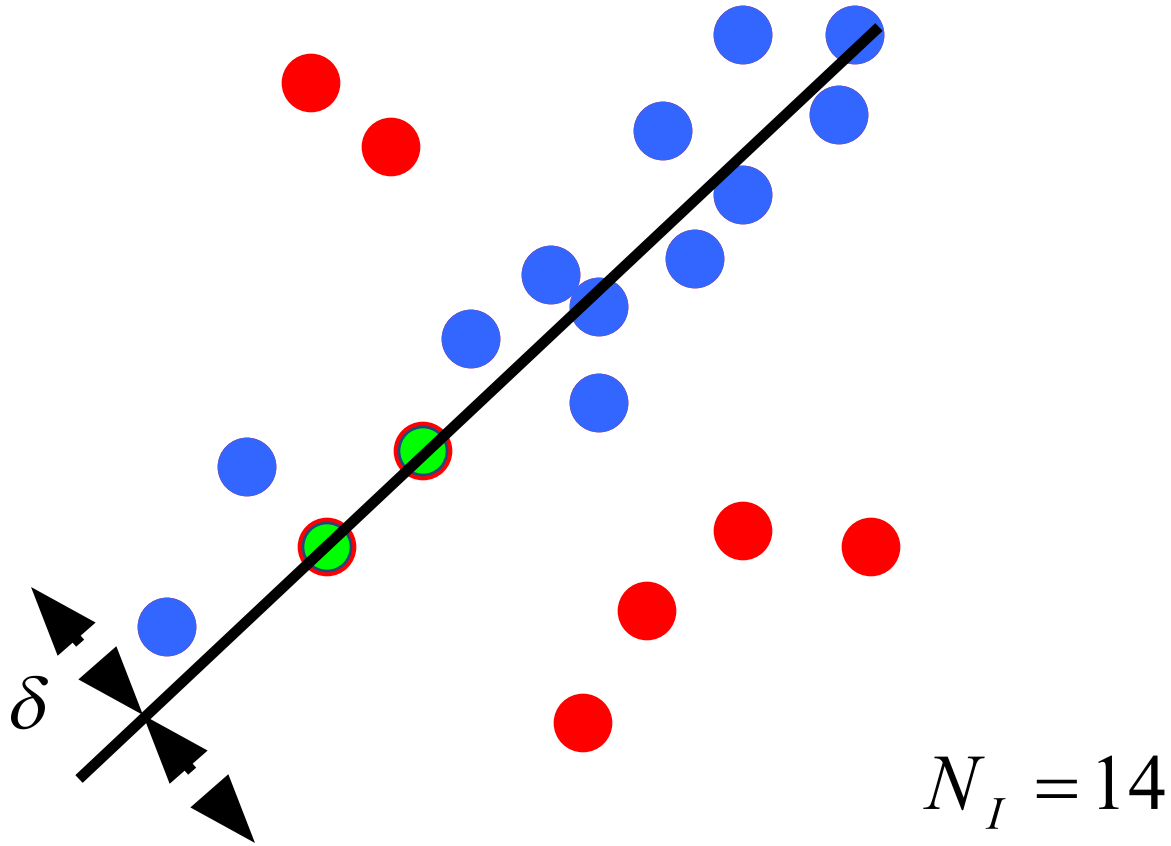
$$N_I = 6$$

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



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

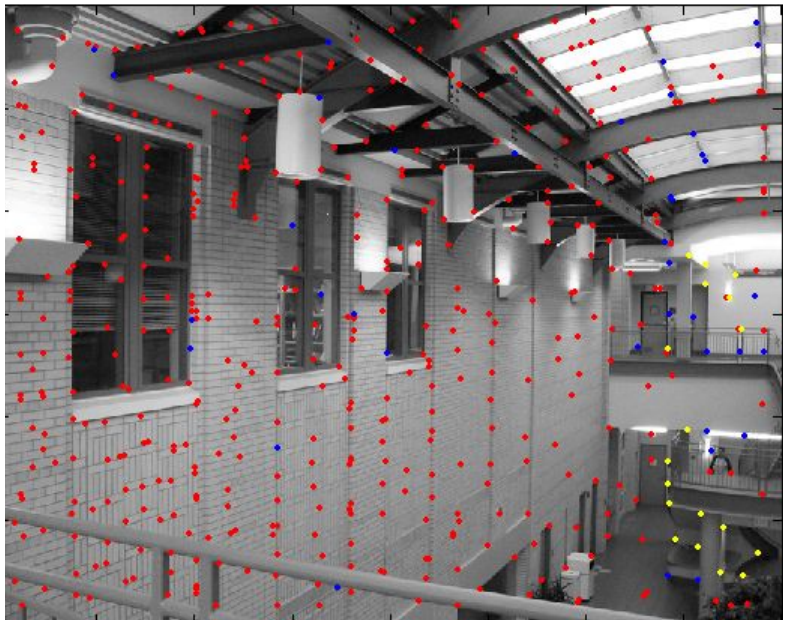
Repeat 1-3 until the best model is found with high confidence

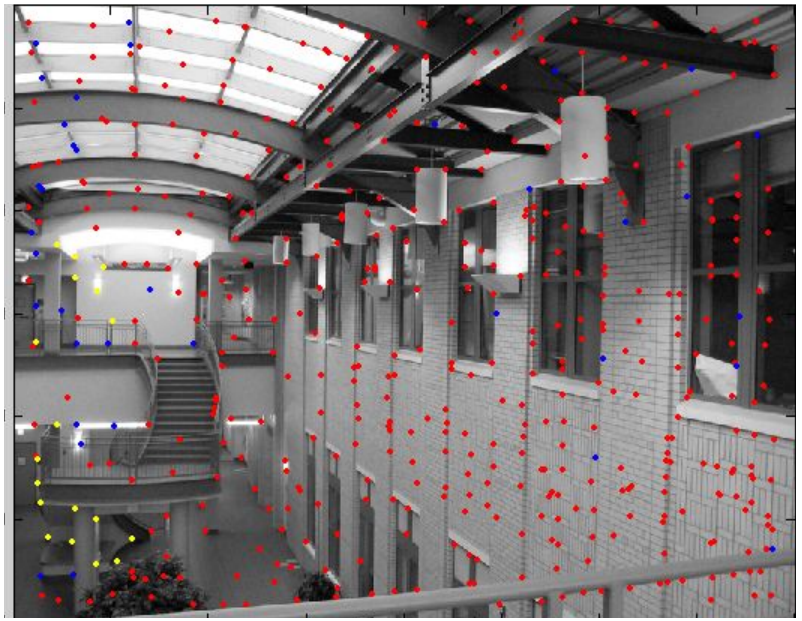
RANSAC for estimating homography

RANSAC loop:

1. Select four feature pairs (at random)
2. Compute homography H (exact)
3. Compute *inliers* where $SSD(p_i', \boldsymbol{H}\, p_i) < \varepsilon$
4. Keep largest set of inliers
5. Re-compute least-squares H estimate on all of the inliers

RANSAC





Alignment

Blending