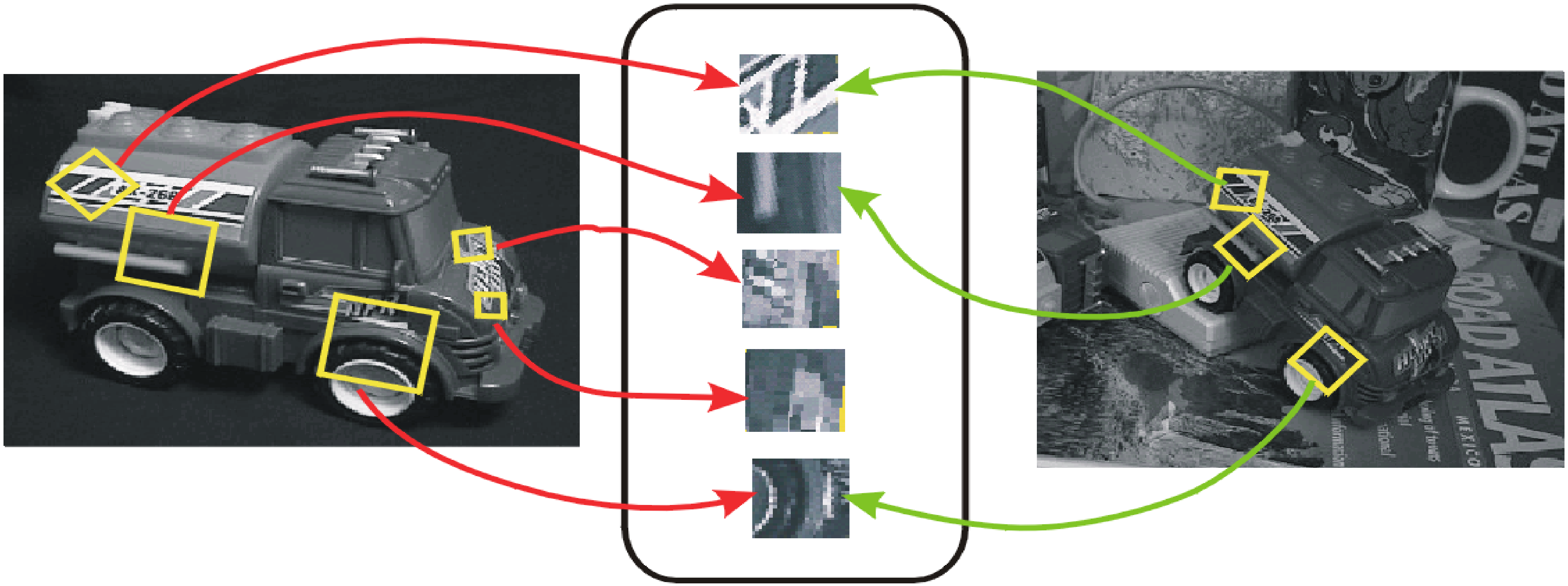
# Invariant Local Features

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



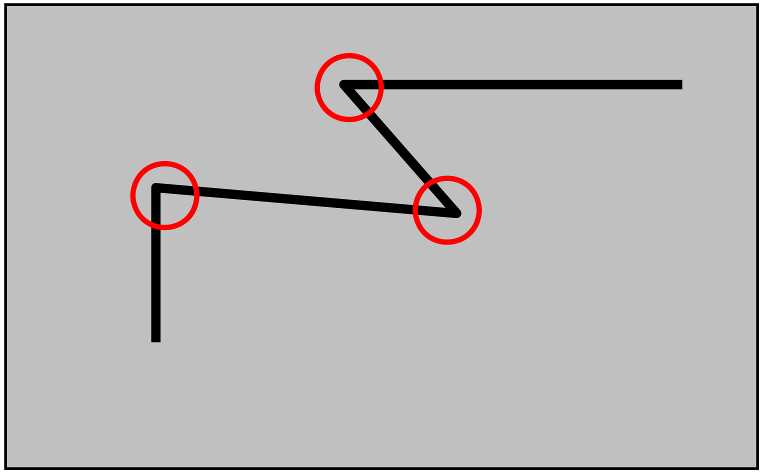
**Features Descriptors**

# More motivation…

- Feature points are used for:
  - Image alignment (homography, fundamental matrix)
  - 3D reconstruction
  - Motion tracking
  - Object recognition
  - Indexing and database retrieval
  - Robot navigation
  - … other

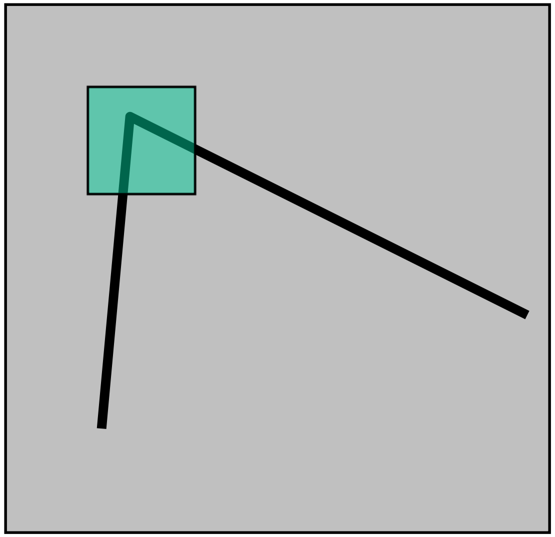
# Corner detector

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

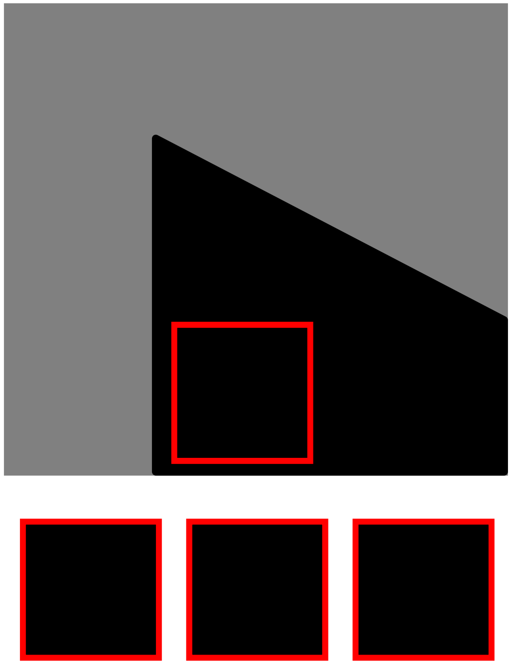


# The Basic Idea

- 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

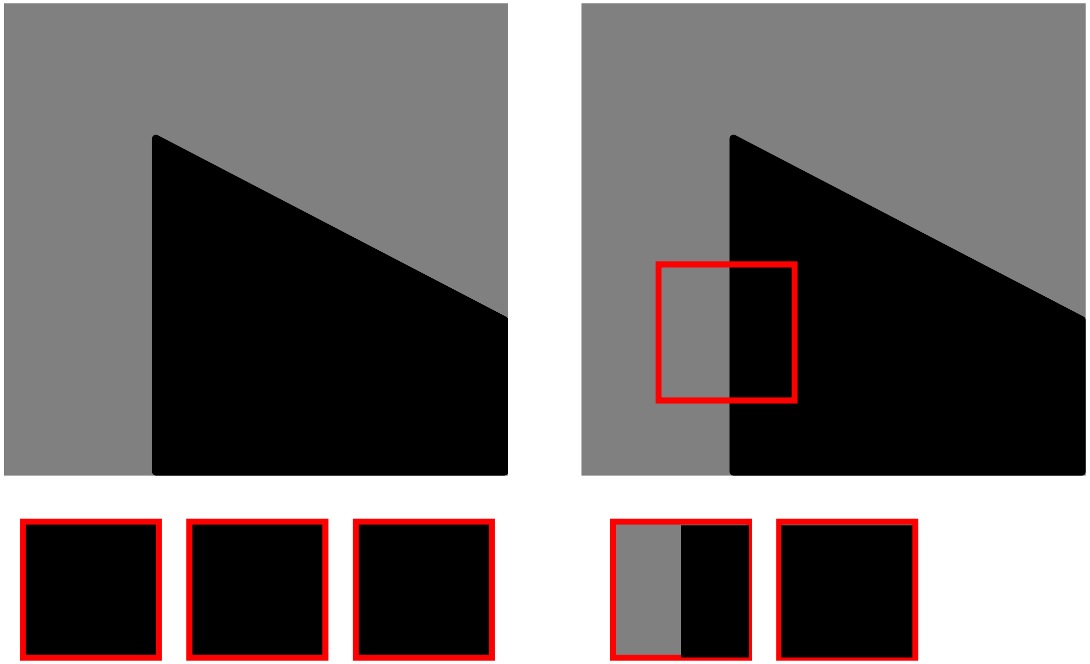


# Moravec corner detector



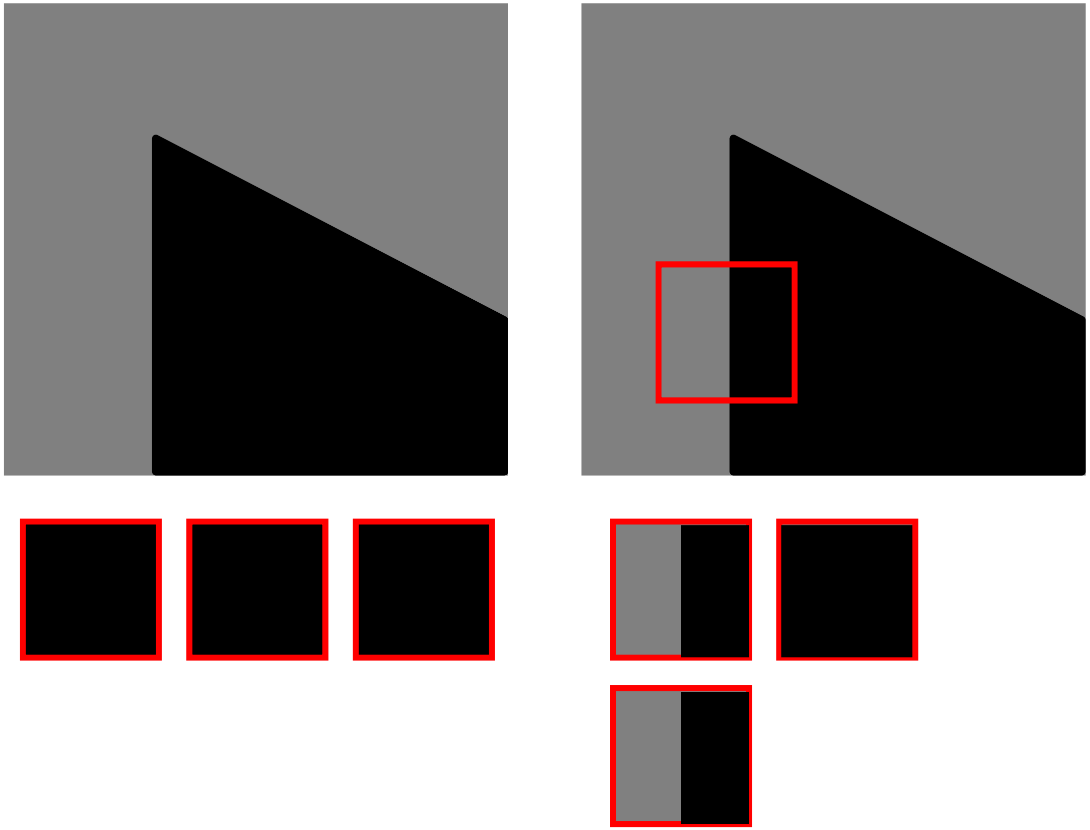
flat

# Moravec corner detector



flat

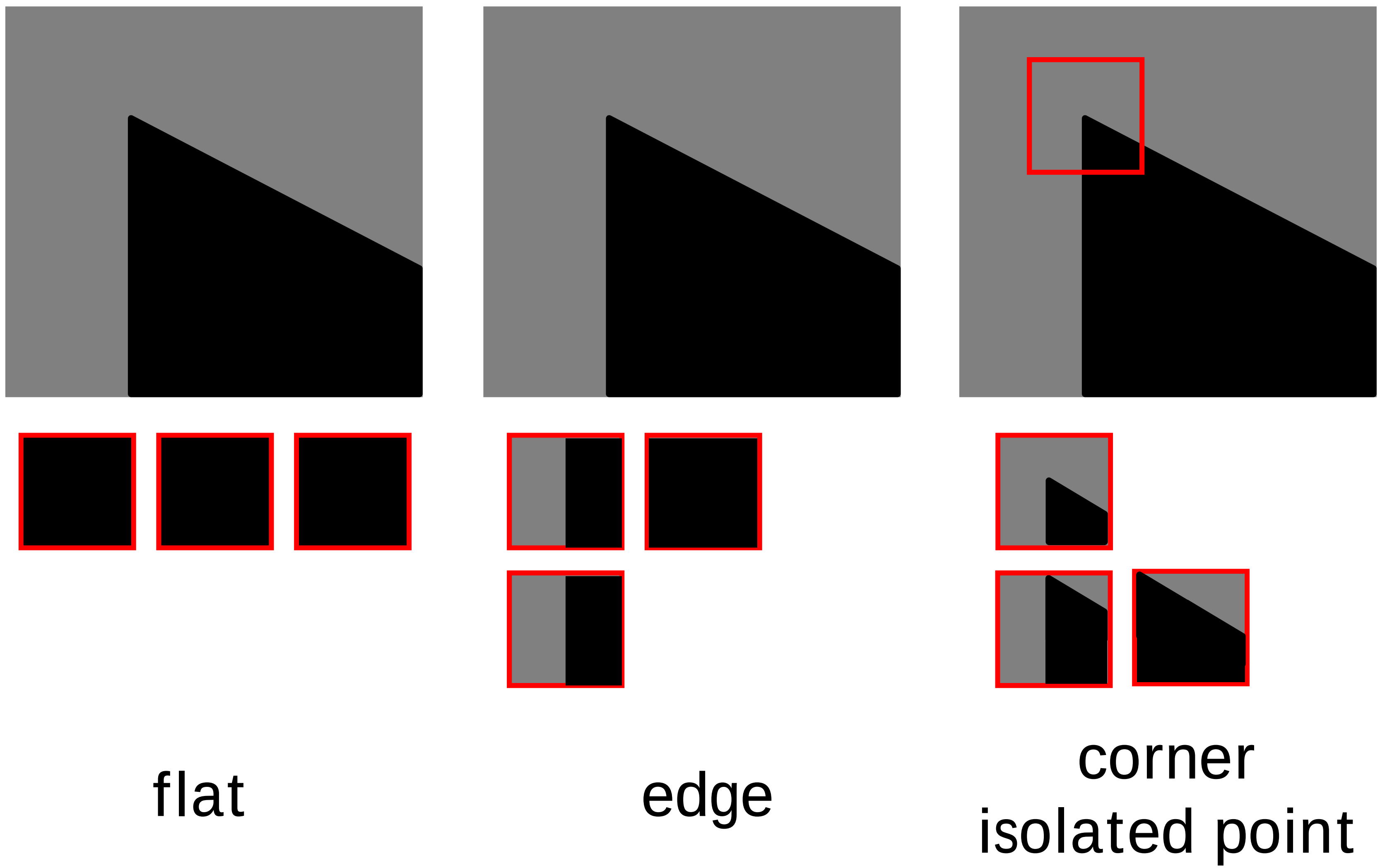
# Moravec corner detector



flat

edge

# Moravec corner detector



# Moravec corner detector

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

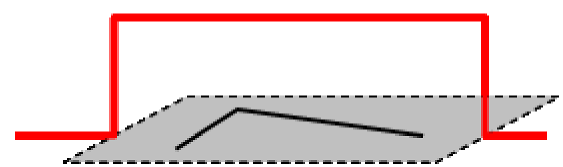
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

Intensity

Window function $w(x,y) =$



1 in window, 0 outside

Four shifts: (u,v) = (1,0), (1,1), (0,1), (-1, 1)

Look for local maxima in *min{E}*

When does this idea fail?

# Problems of Moravec detector

- Only a set of shifts at every 45 degree is considered
- Noisy response due to a binary window function
- Only minimum of E is taken into account

⇨ Harris corner detector (1988) solves these problems.

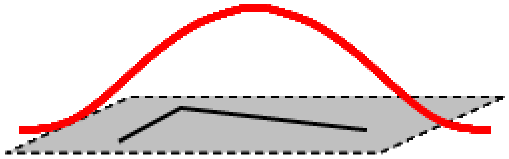
# Harris corner detector

Noisy response due to a binary window function

- Use a Gaussian function

$$w(x,y) = \exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$

Window function $w(x,y) =$



Gaussian

# Harris corner detector

Only a set of shifts at every 45 degree is considered

- Consider all small shifts by Taylor's expansion

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

$$E(u,v)=Au^2+2Cuv+Bv^2$$

$$A=\sum_{x,y} w(x,y)I_x^2(x,y)$$

$$B=\sum_{x,y} w(x,y)I_y^2(x,y)$$

$$C=\sum_{x,y} w(x,y)I_x(x,y)I_y(x,y)$$

# Harris corner detector

Equivalently, for small shifts [$u,v$] we have a *bilinear* approximation:

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

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

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

# Harris corner detector

Only minimum of E is taken into account

- A new corner measurement

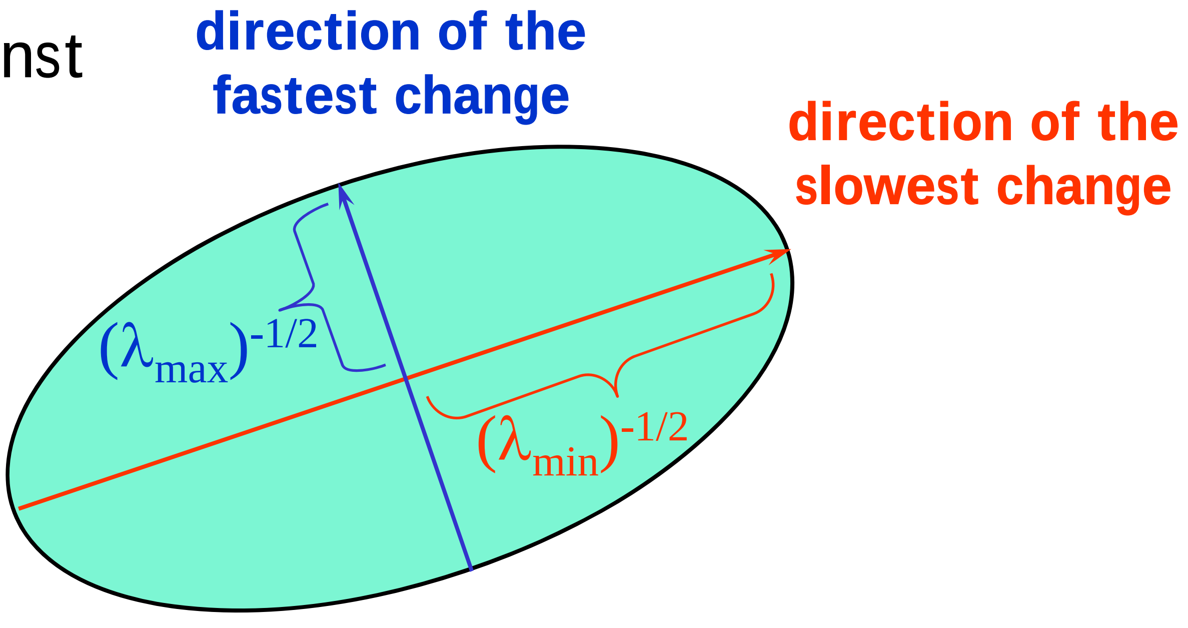
# Harris corner detector

Intensity change in shifting window: eigenvalue analysis

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

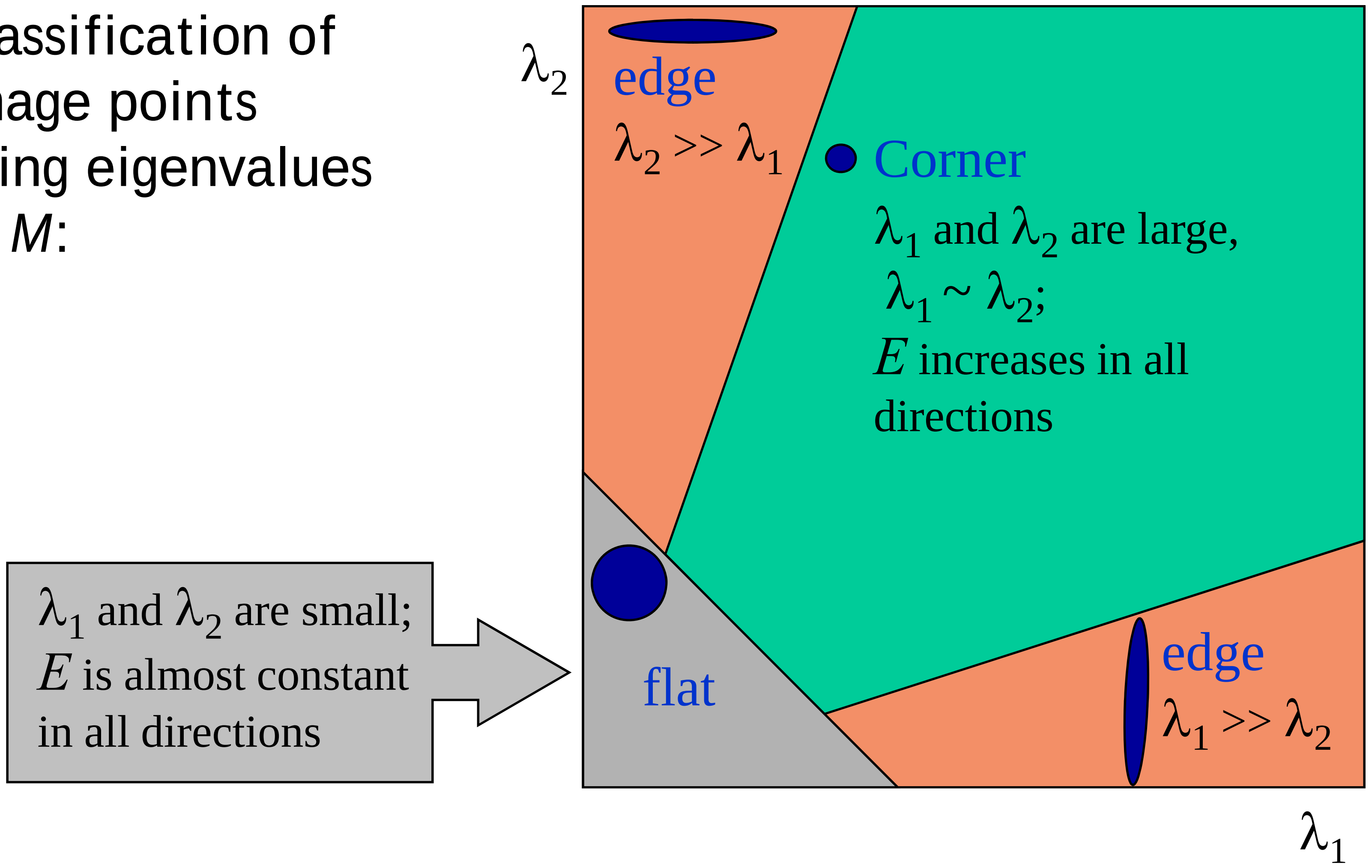
$\lambda_1, \lambda_2$ – eigenvalues of $M$

Ellipse *E(u,v)* = const



# Harris corner detector

Classification of image points using eigenvalues of *M*:



# Harris corner detector

Measure of corner response:

$$R = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{\det M}{\operatorname{Trace} M}$$

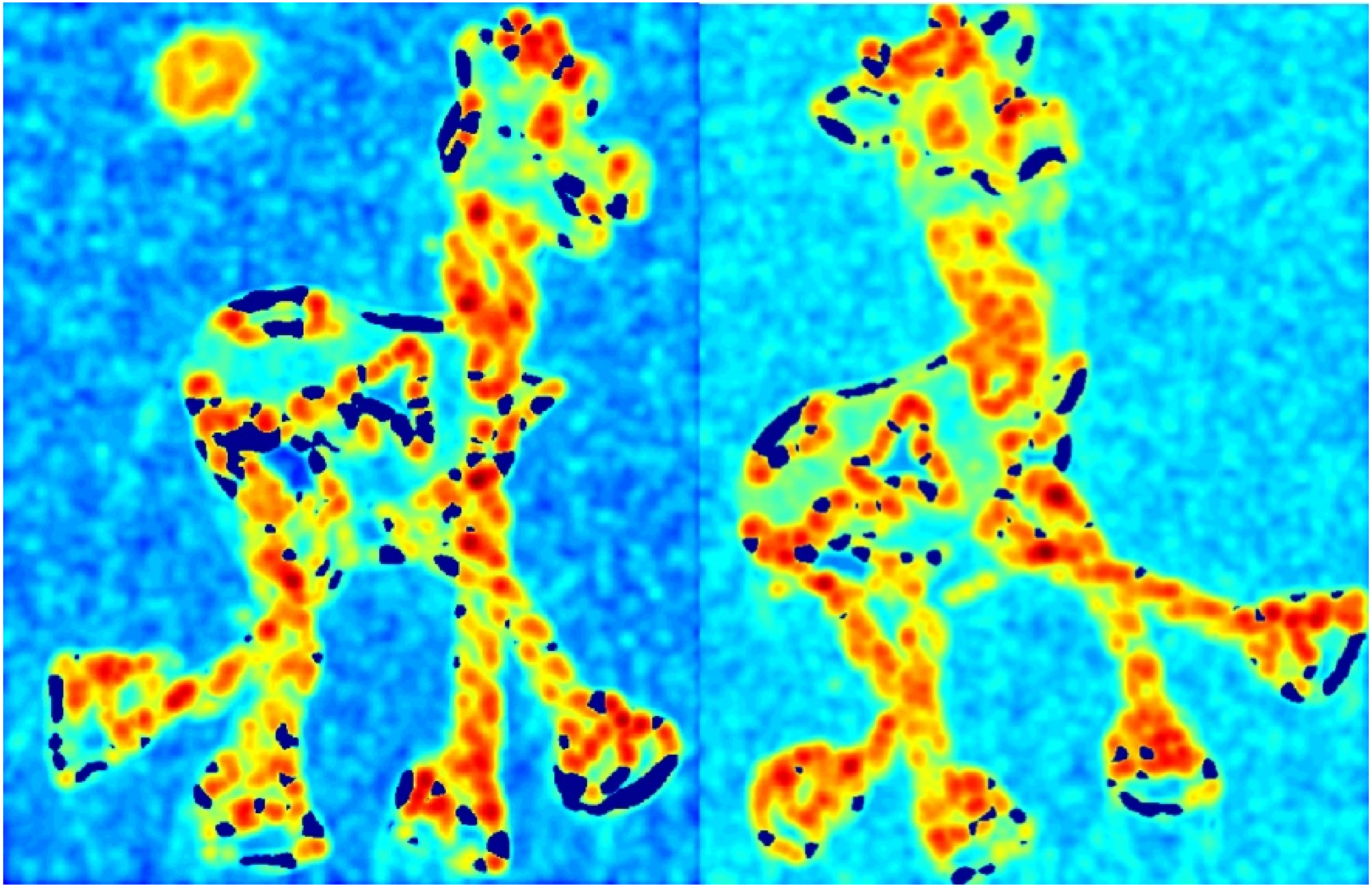
# Harris Detector

- The Algorithm:
  - Find points with large corner response function $R$ ($R$ > threshold)
  - Take the points of local maxima of $R$

# Harris Detector: Workflow

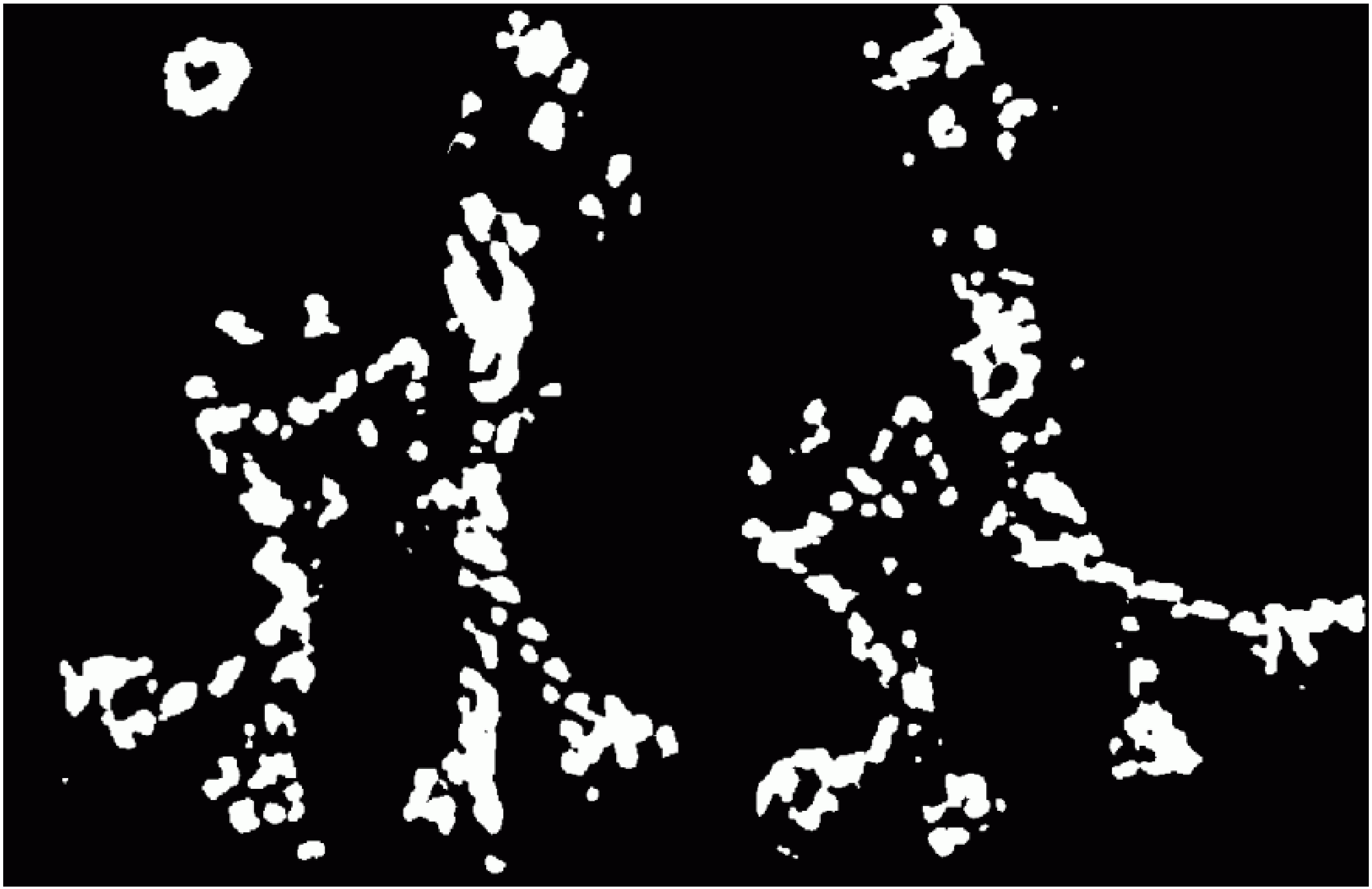
# Harris Detector: Workflow

Compute corner response $R$



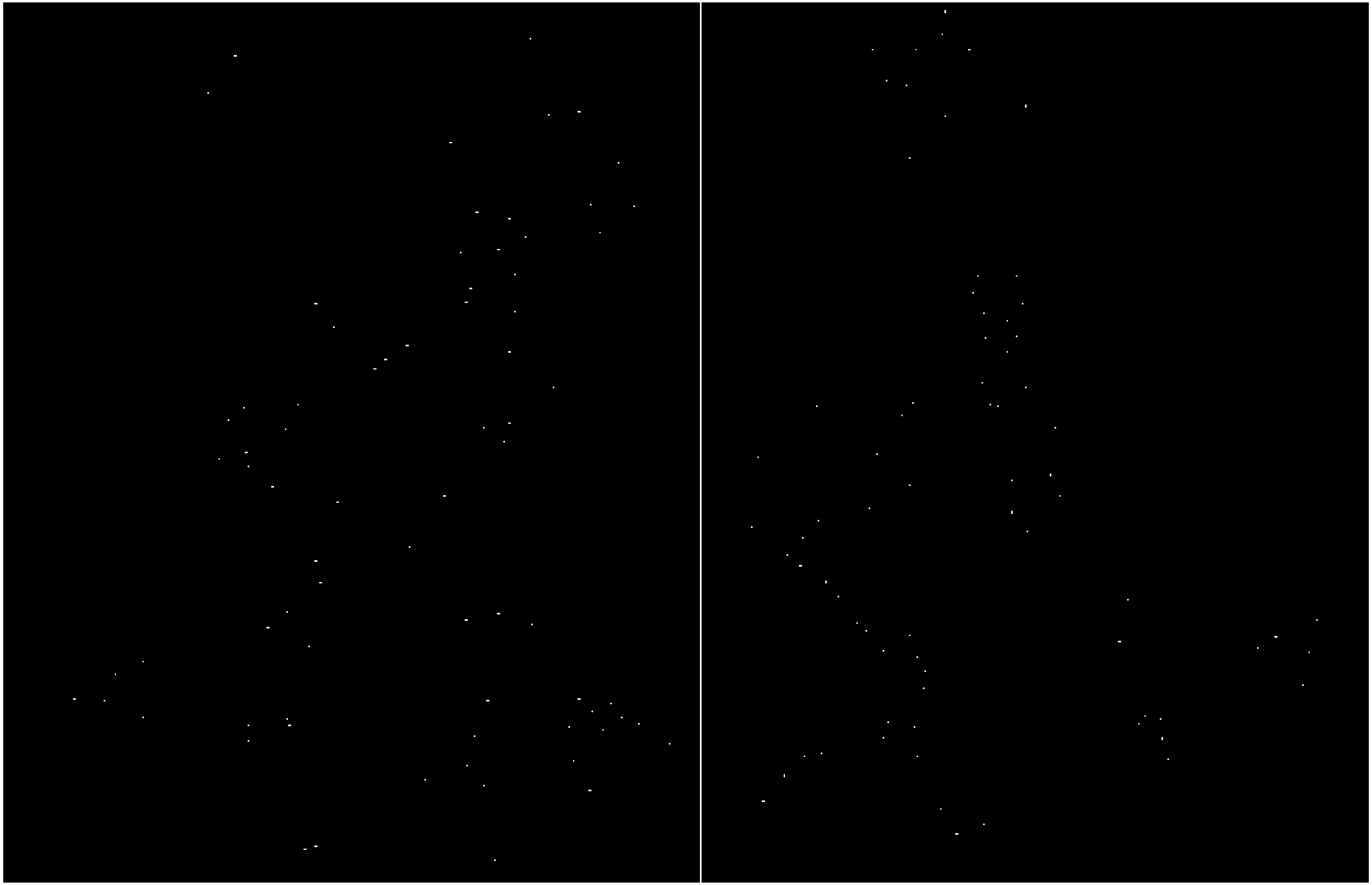
# Harris Detector: Workflow

Find points with large corner response: $R >$ threshold



# Harris Detector: Workflow

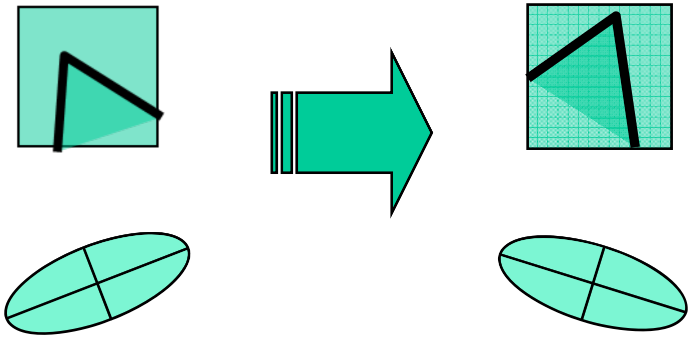
Take only the points of local maxima of $R$



# Harris Detector: Workflow

# Harris Detector: Some Properties

- Rotation invariance



Ellipse rotates but its shape (i.e. eigenvalues) remains the same

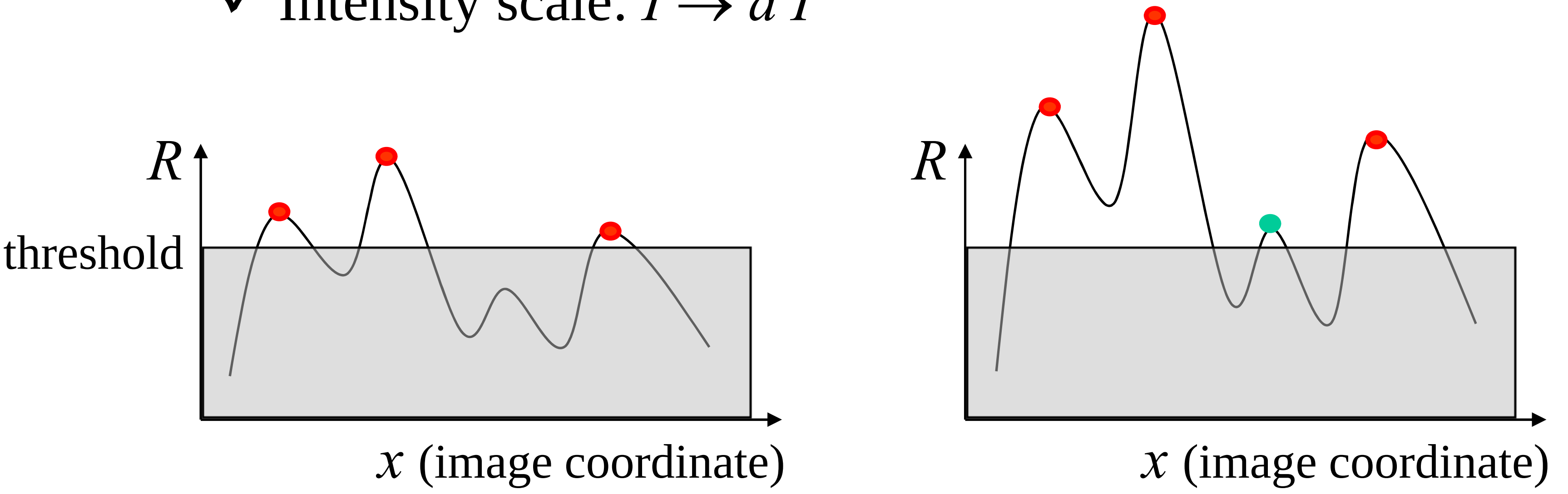
*Corner response R* is invariant to image rotation

# Harris Detector: Some Properties

- Partial invariance to *affine intensity* change

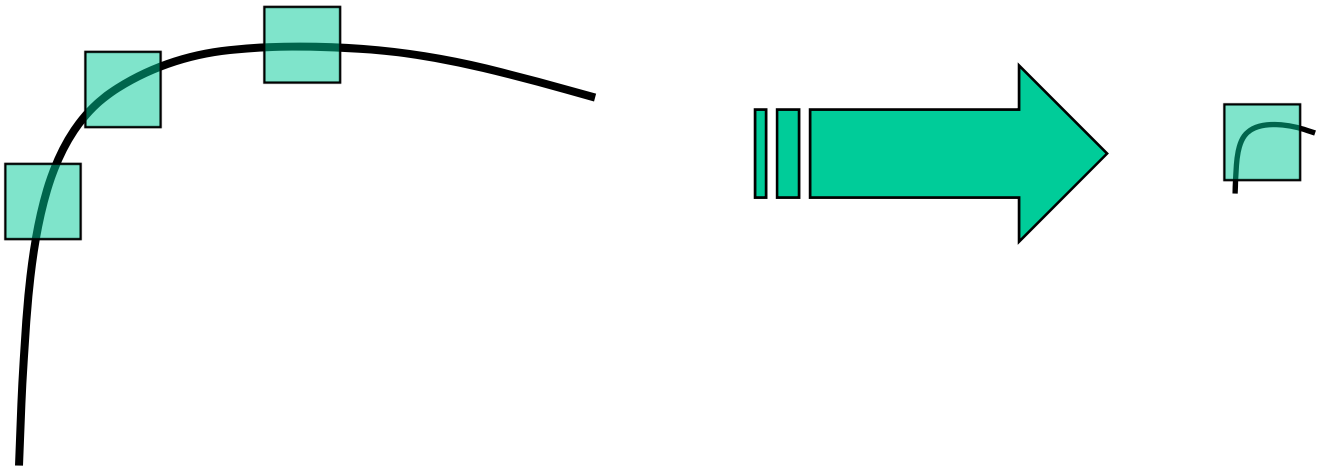
✓ Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$

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



# Harris Detector: Some Properties

- But: non-invariant to *image scale*!

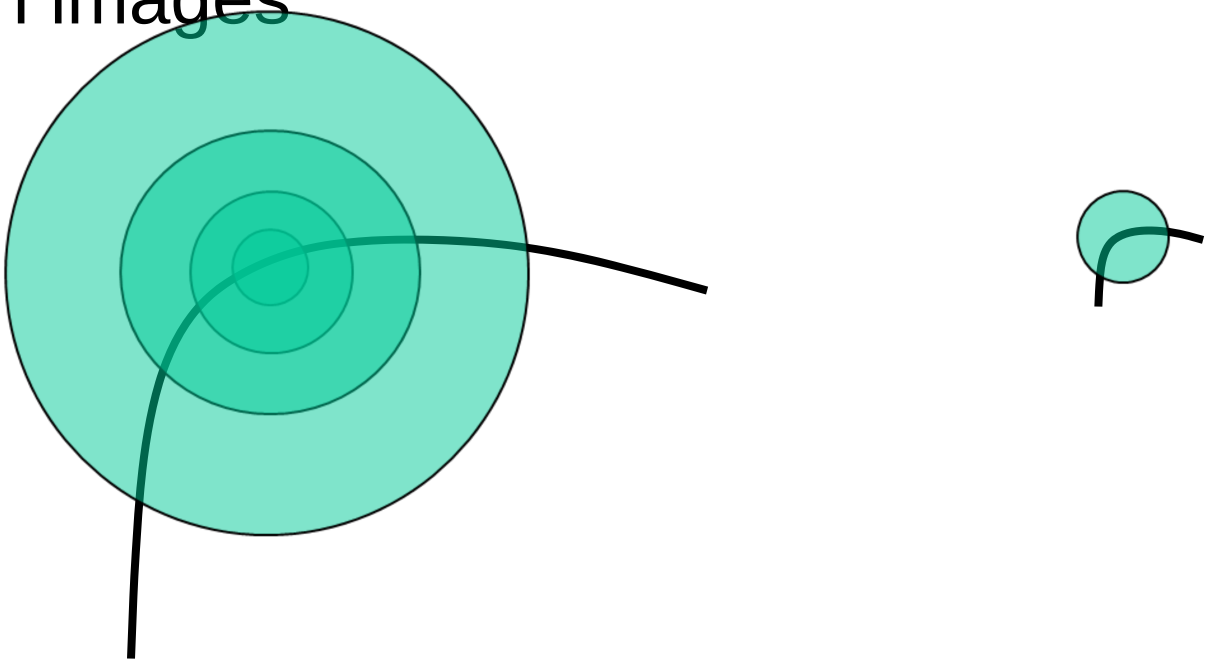


All points will be classified as edges

Corner !

# Scale Invariant Detection

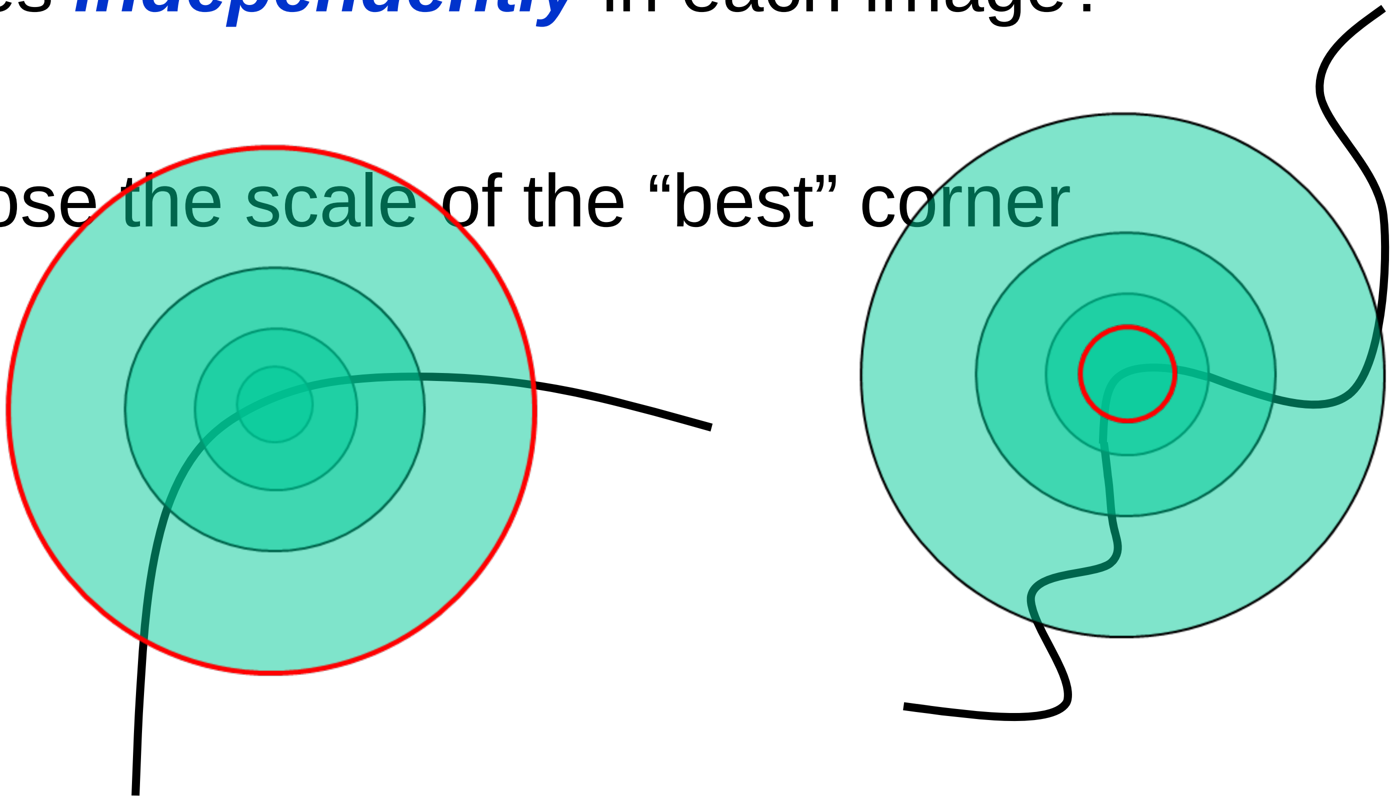
- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images



# Scale Invariant Detection

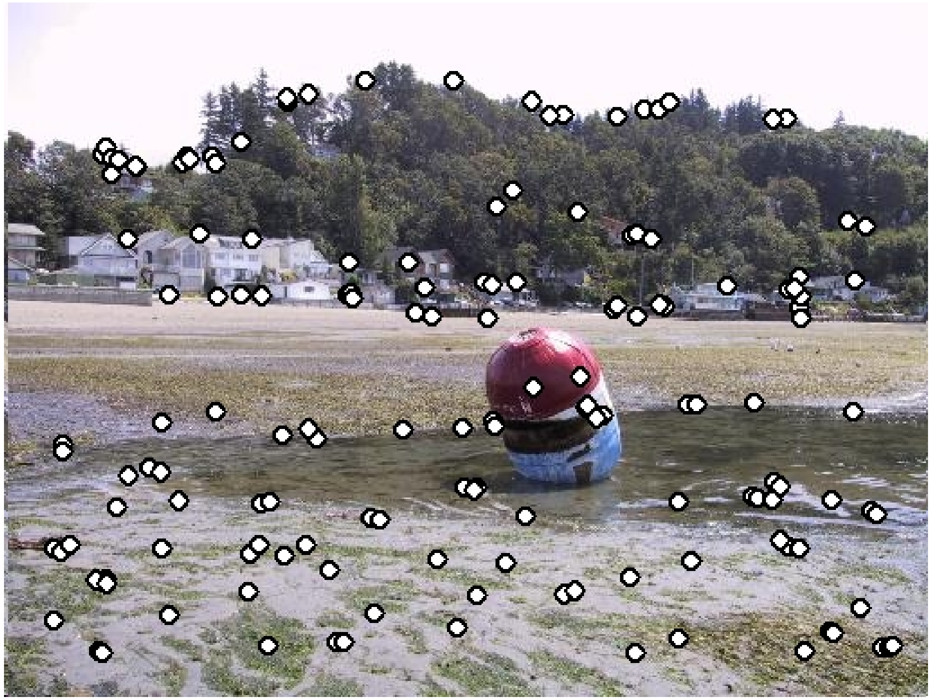
- The problem: how do we choose corresponding circles ***independently*** in each image?

- Choose the scale of the "best" corner



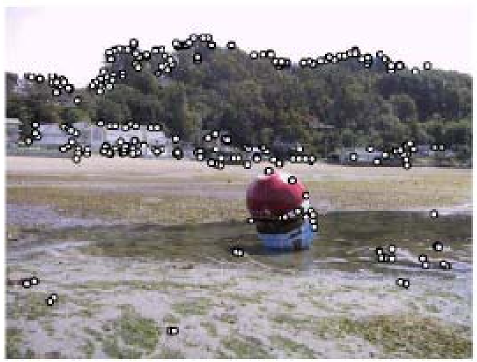
# Feature selection

- Distribute points evenly over the image

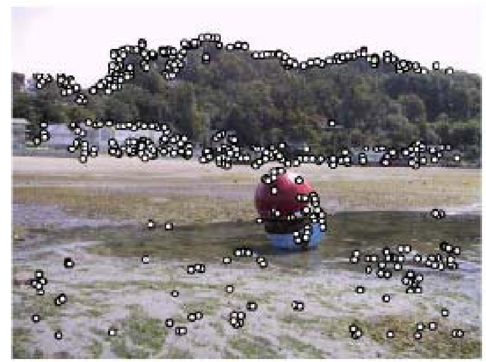


# Adaptive Non-maximal Suppression

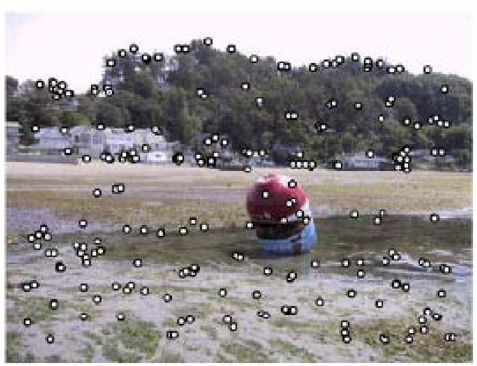
- Desired: Fixed # of features per image
  - Want evenly distributed spatially…
  - Sort ponts by non-maximal suppression radius [Brown, Szeliski, Winder, CVPR'05]



(a) Strongest 250



(b) Strongest 500

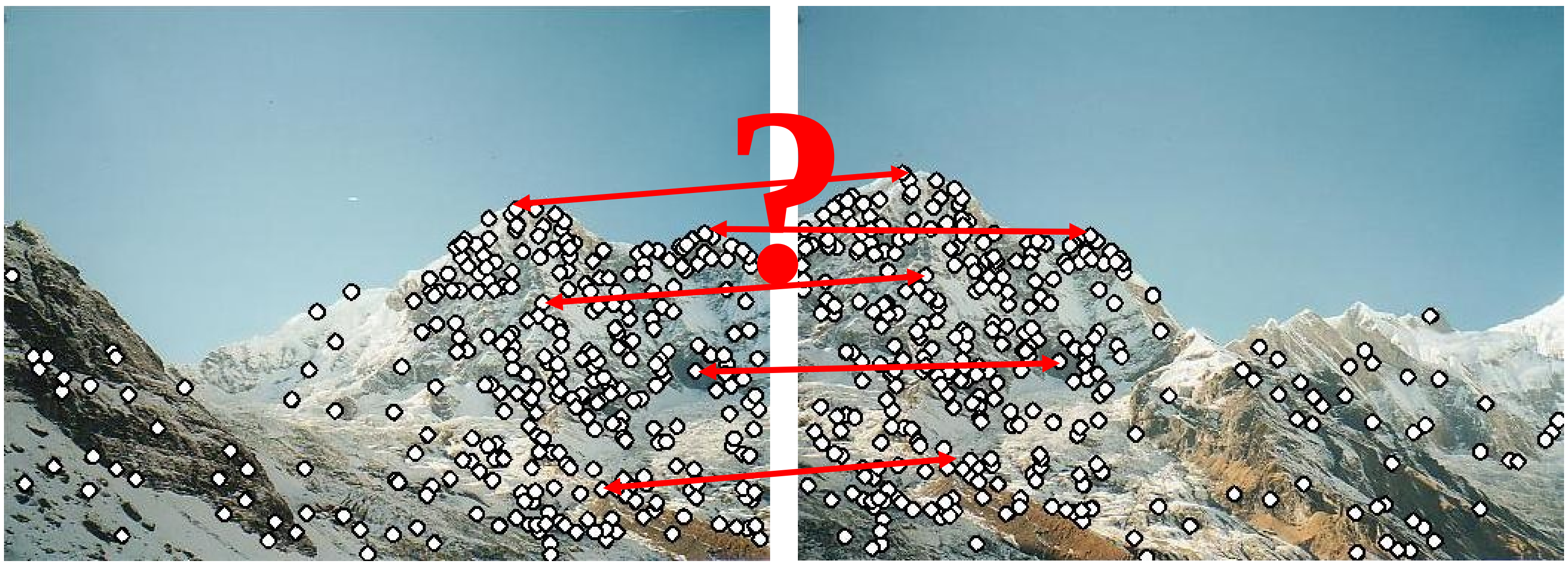


(c) ANMS 250, $r = 24$

(d) ANMS 500, $r = 16$

# Feature descriptors

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



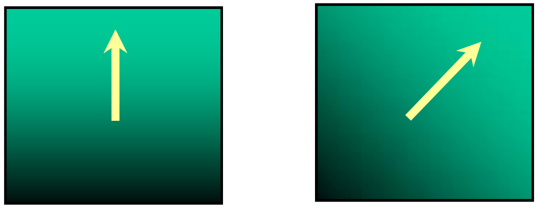

Point descriptor should be:

1. Invariant
2. Distinctive

# Descriptors Invariant to Rotation

- Find local orientation

Dominant direction of gradient



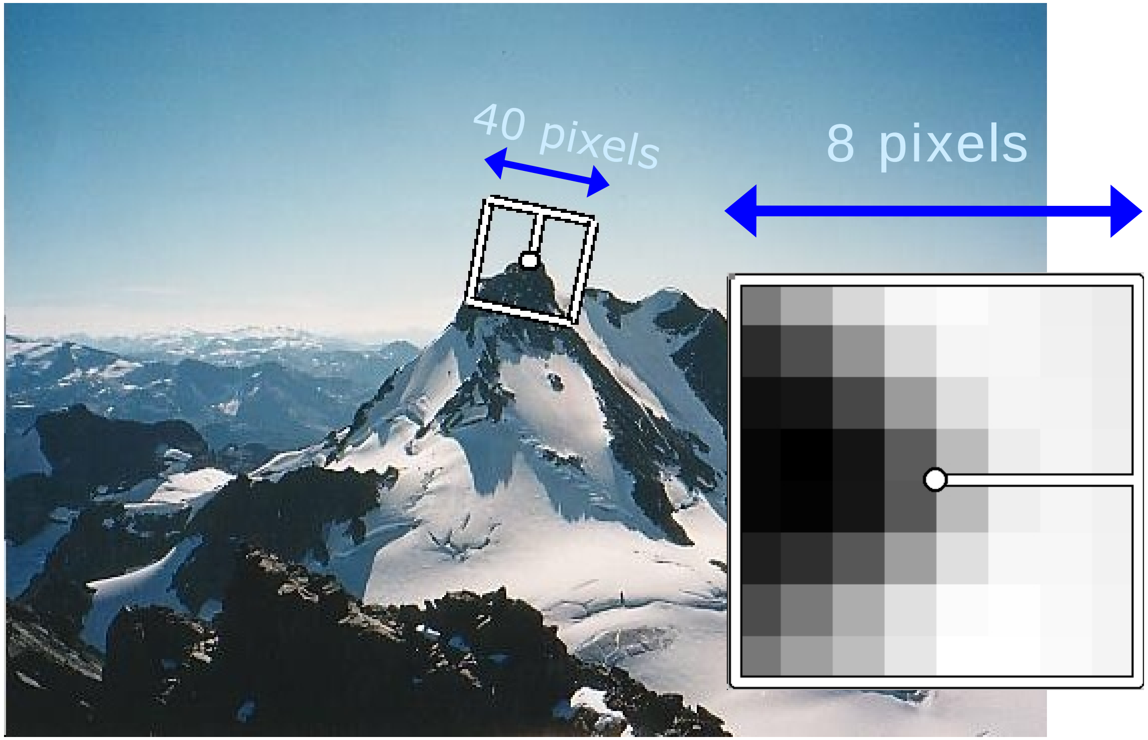
- Extract image patches relative to this orientation

# Descriptor Vector

- Rotation Invariant Frame
- Orientation = blurred gradient

# MOPS descriptor vector

- 8x8 oriented patch
  - Sampled at 5 x scale
- Bias/gain normalisation: $I' = (I - \mu)/\sigma$



# Detections at multiple scales

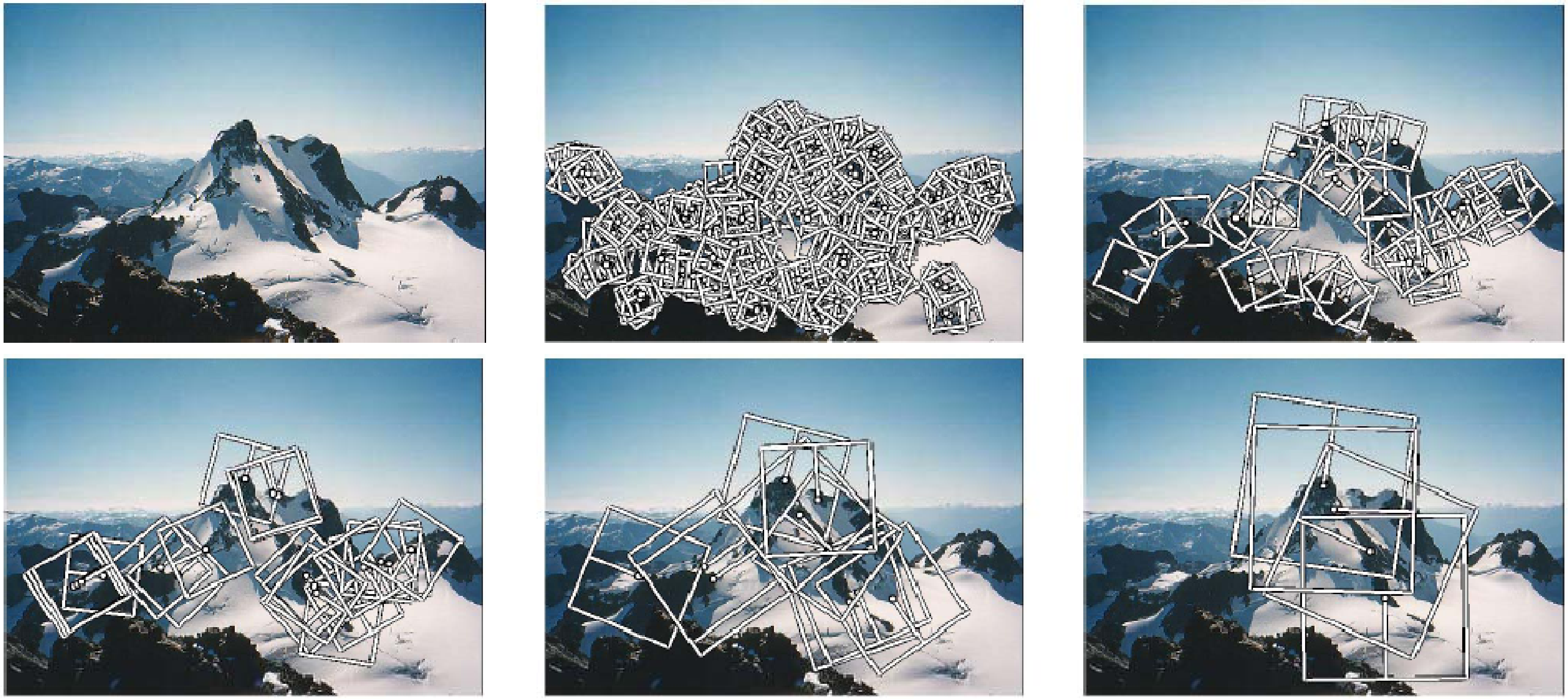
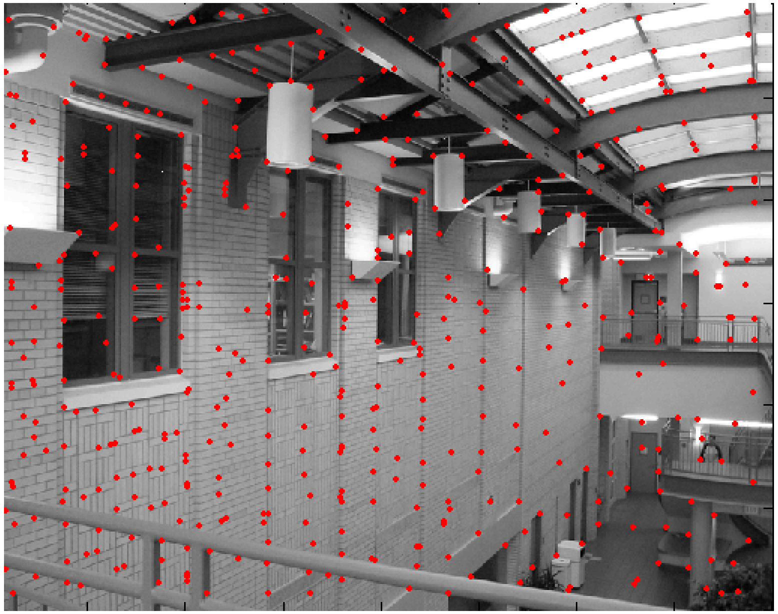


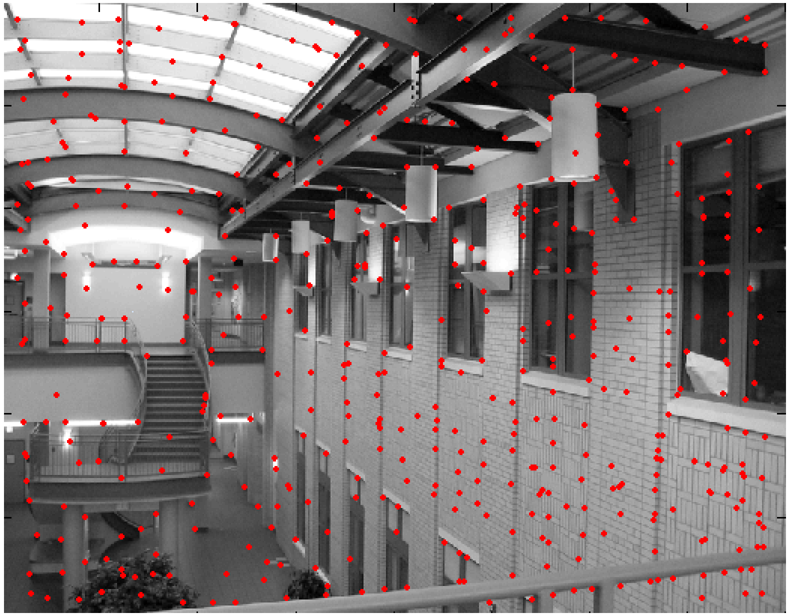
*Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.*

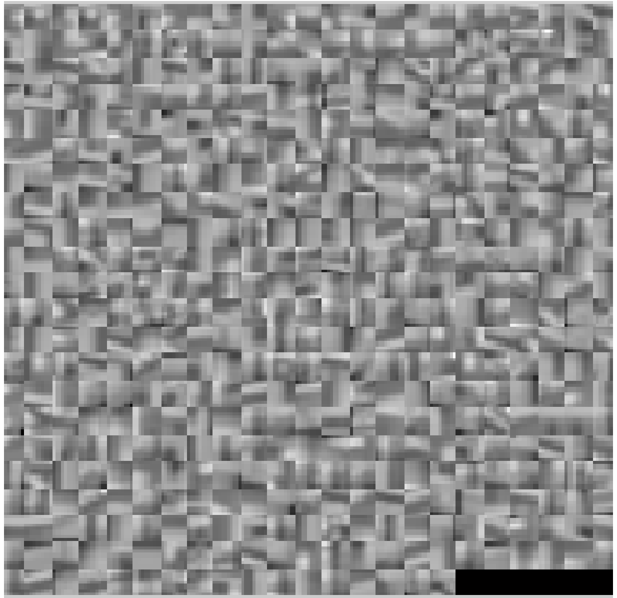
# Multi-Scale Oriented Patches (Summary)

- Interest points
  - Multi-scale Harris corners
  - Orientation from blurred gradient
  - Geometrically invariant to rotation
- Descriptor vector
  - Bias/gain normalized sampling of local patch (8x8)
  - Photometrically invariant to affine changes in intensity
- [Brown, Szeliski, Winder, CVPR’2005]

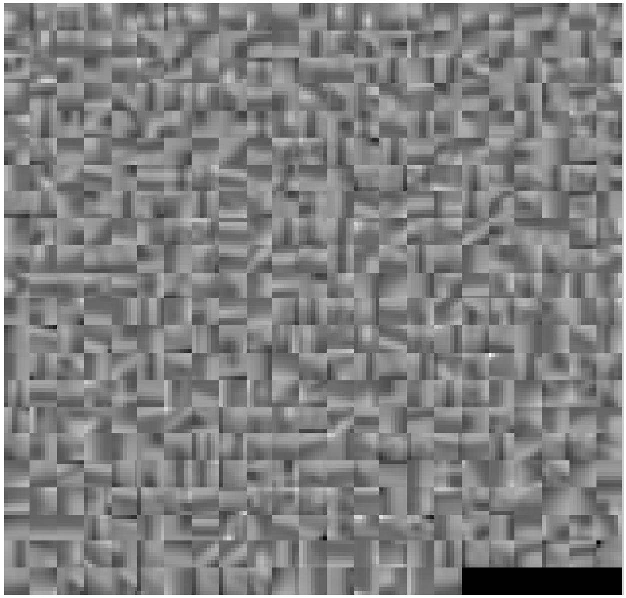
# Feature matching







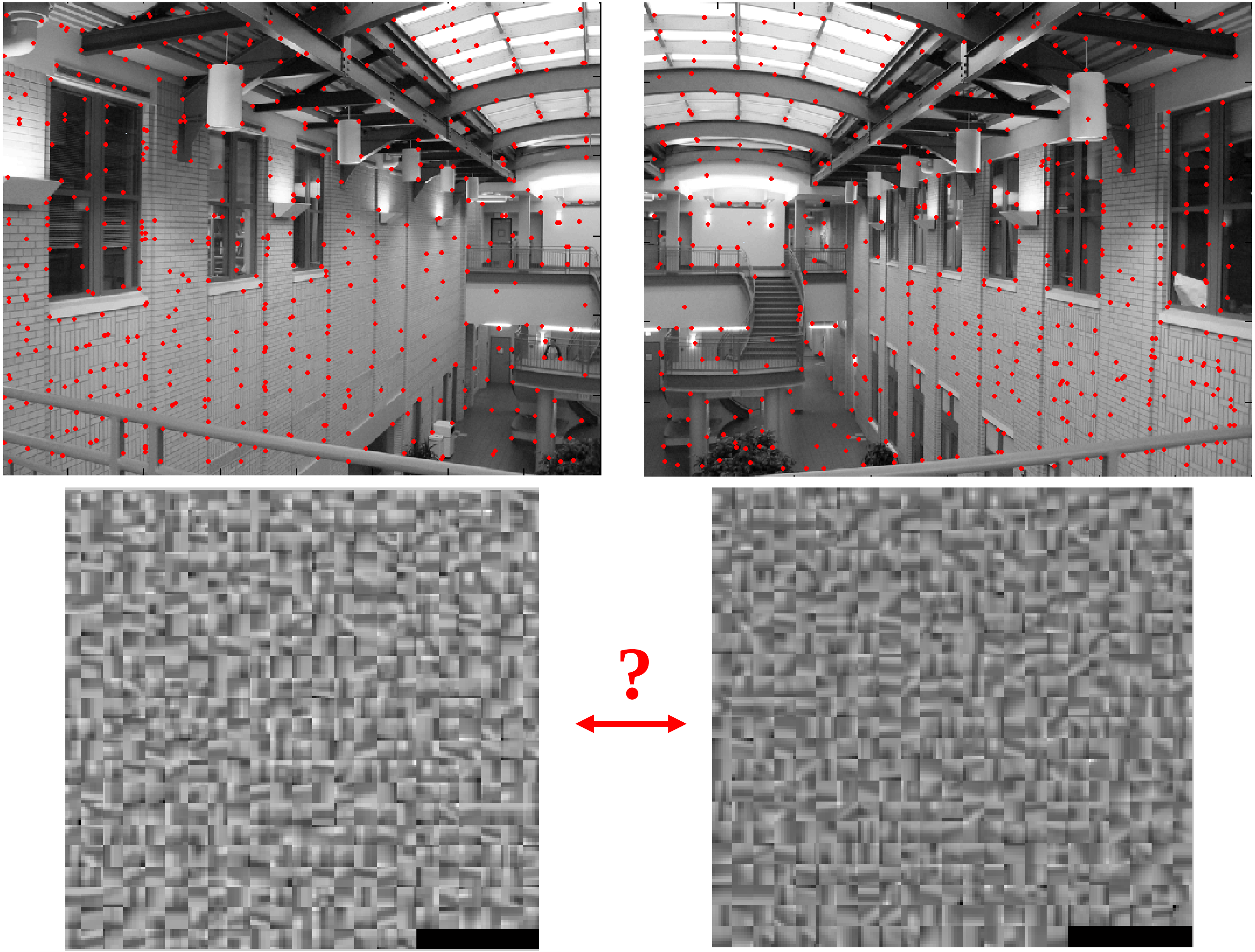




# Feature matching

- Exhaustive search
  - for each feature in one image, look at *all* the other features in the other image(s)
- Hashing
  - compute a short descriptor from each feature vector, or hash longer descriptors (randomly)
- Nearest neighbor techniques
  - *kd*-trees and their variants

# What about outliers?



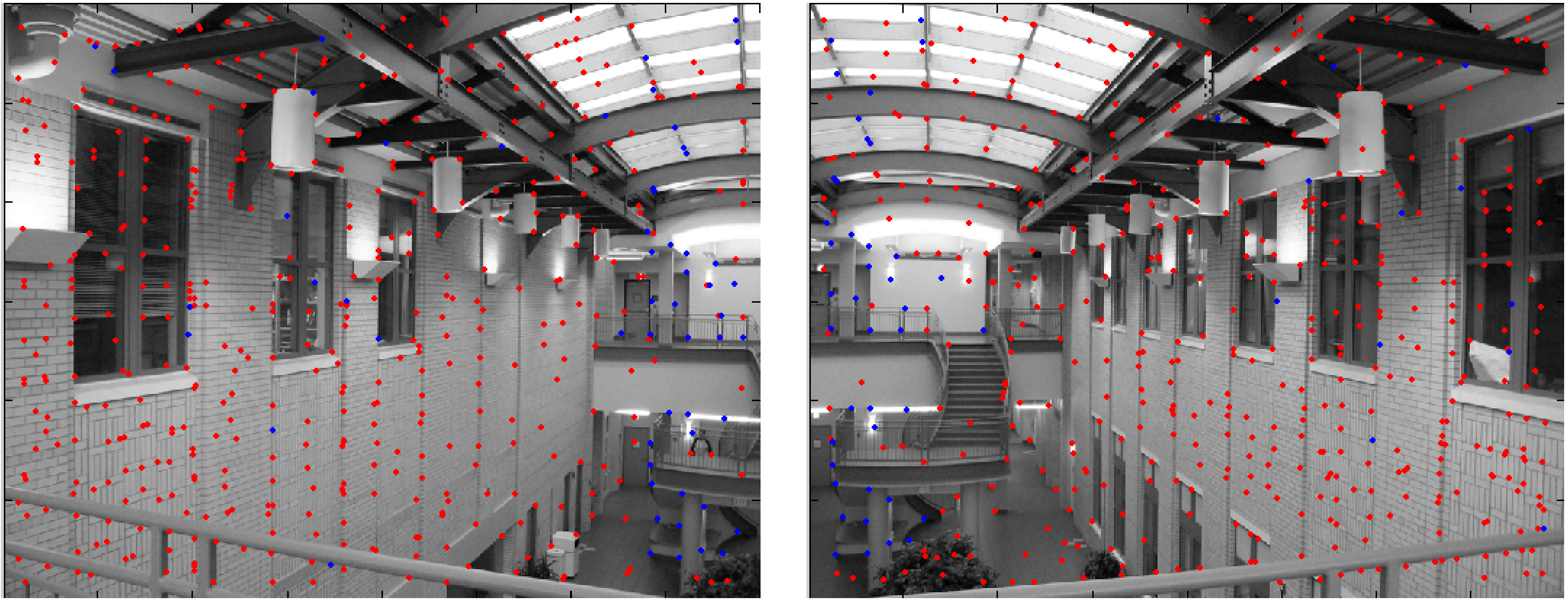
# Feature-space outlier rejection

- Let's not match all features, but only these that have "similar enough" matches?
- How can we do it?
  - SSD(patch1,patch2) < threshold
  - How to set threshold?

# Feature-space outlier rejection

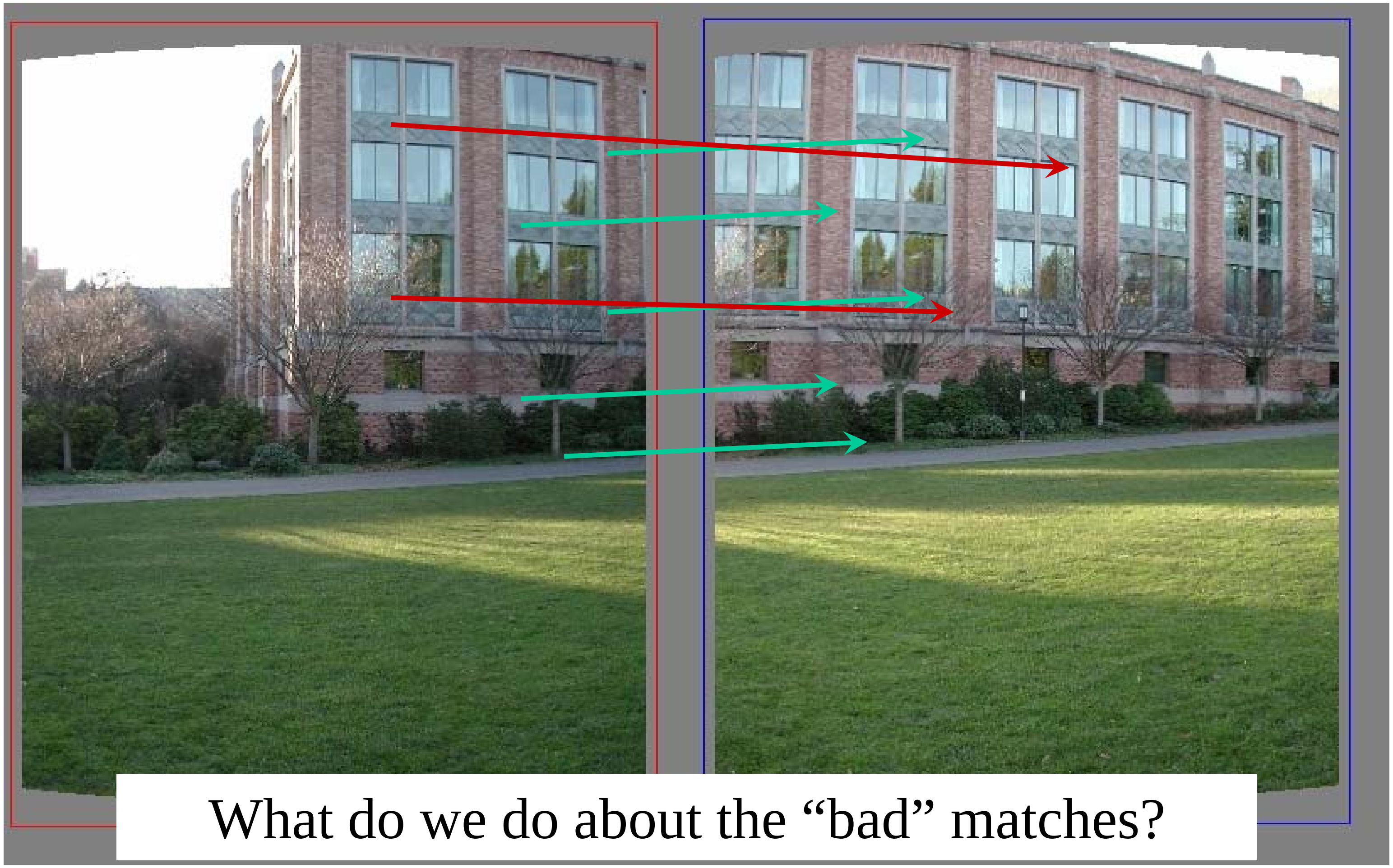
- A better way [Lowe, 1999]:
  - 1-NN: SSD of the closest match
  - 2-NN: SSD of the second-closest match
  - Look at how much better 1-NN is than 2-NN, e.g. 1-NN/2-NN
  - That is, is our best match so much better than the rest?

# Feature-space outliner rejection



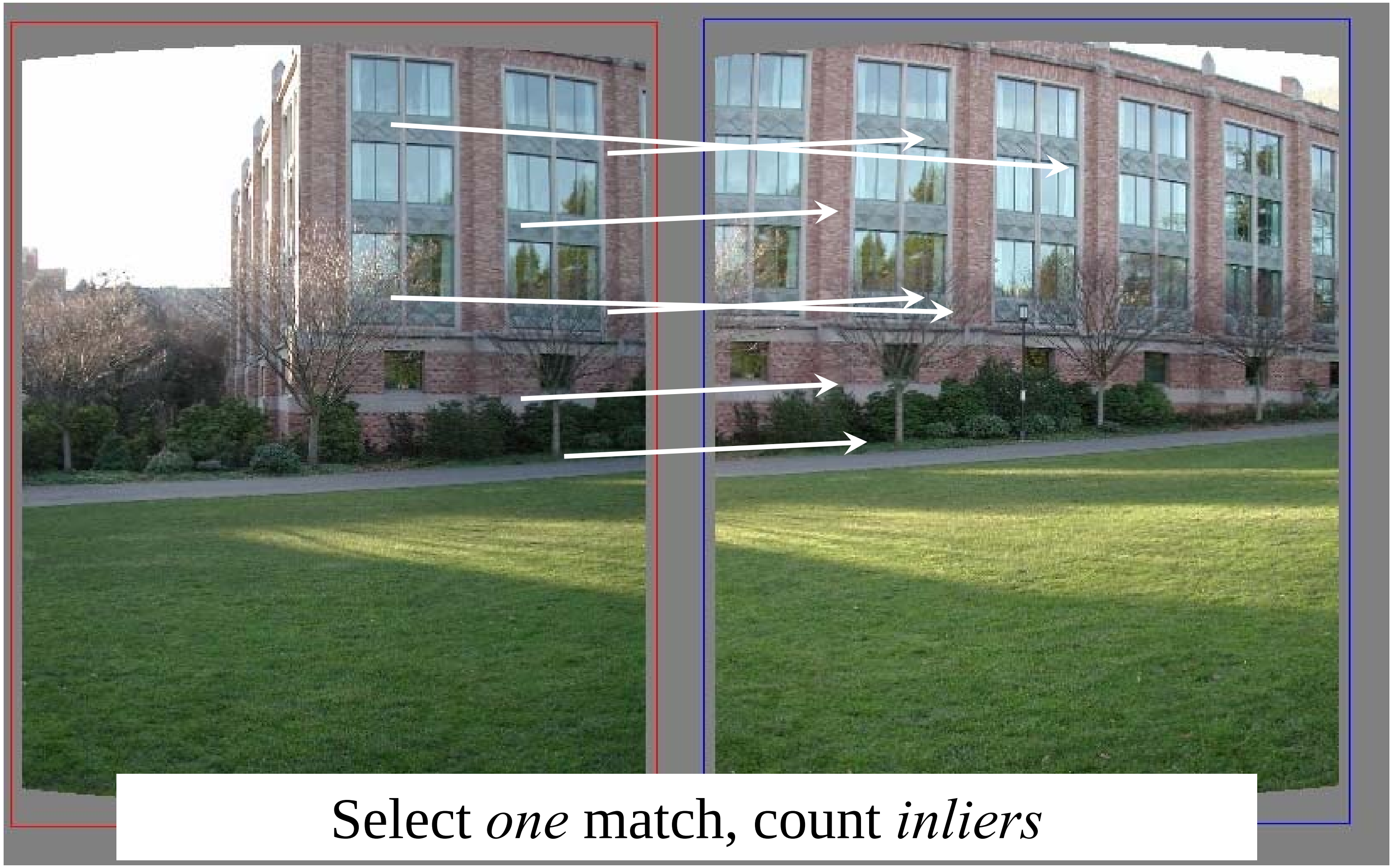
- Can we now compute H from the blue points?
  - No!  Still too many outliers…
  - What can we do?

# Matching features



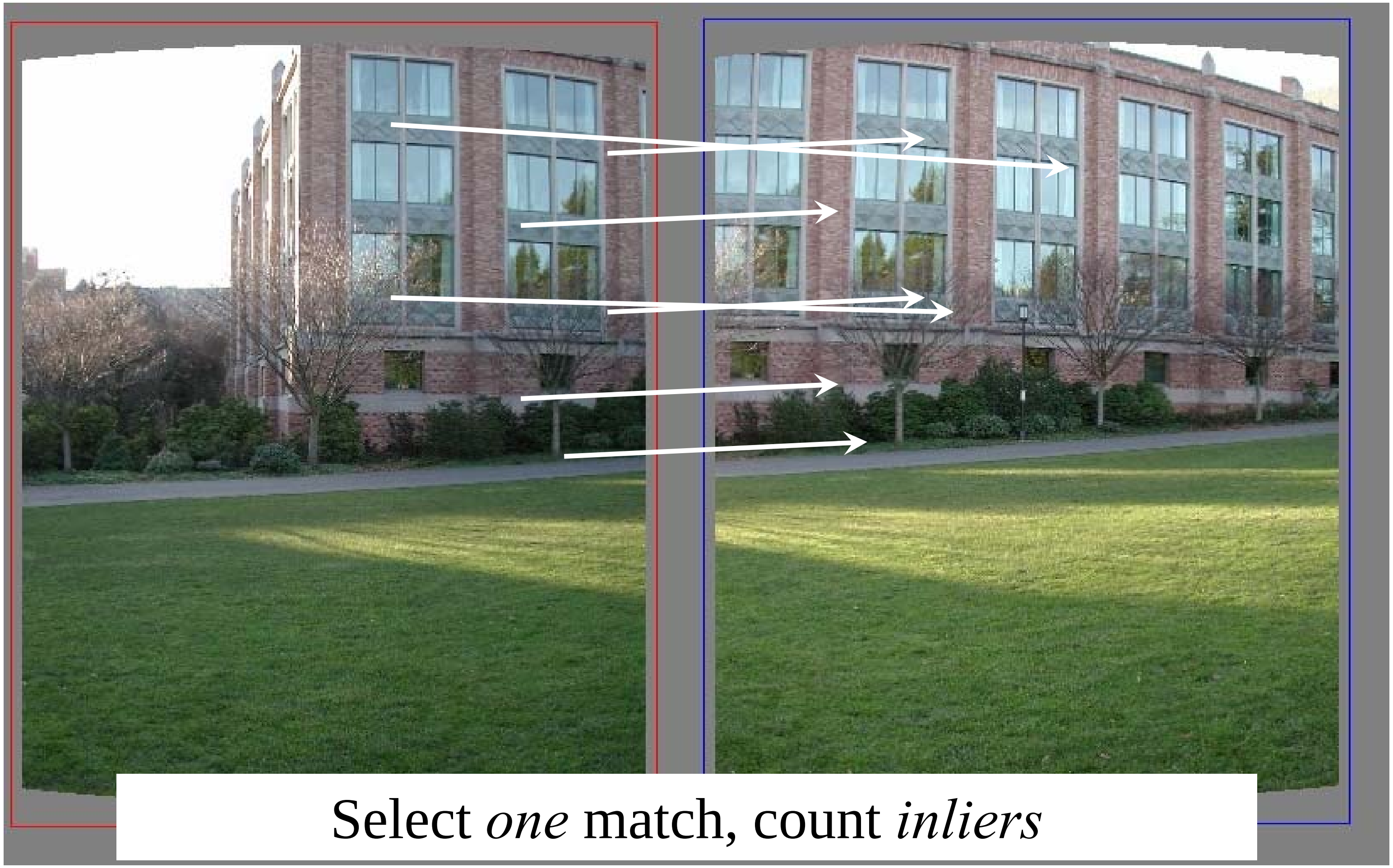
What do we do about the “bad” matches?

# RAndom SAmple Consensus



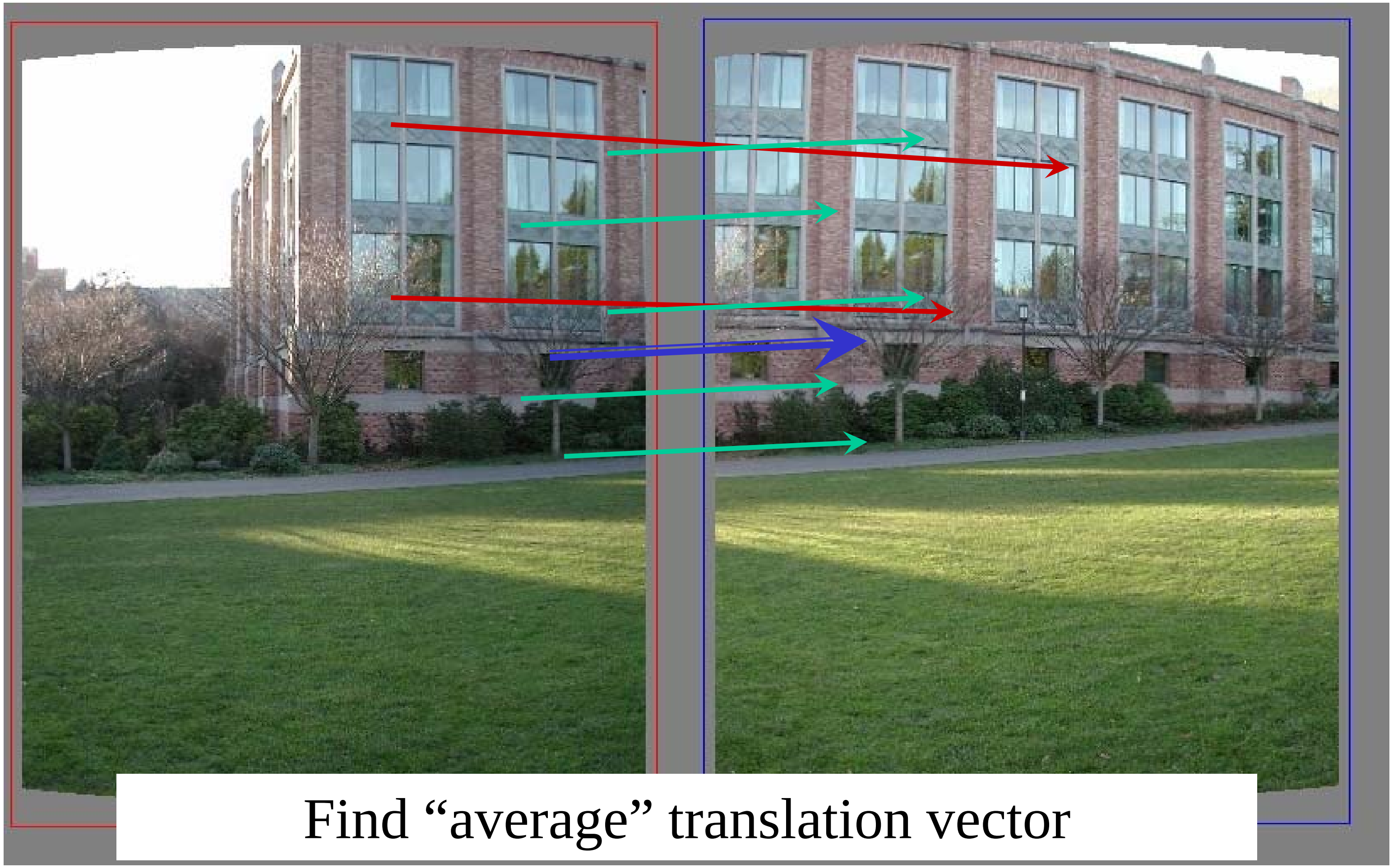
Select *one* match, count *inliers*

# RAndom SAmple Consensus



Select *one* match, count *inliers*

# Least squares fit



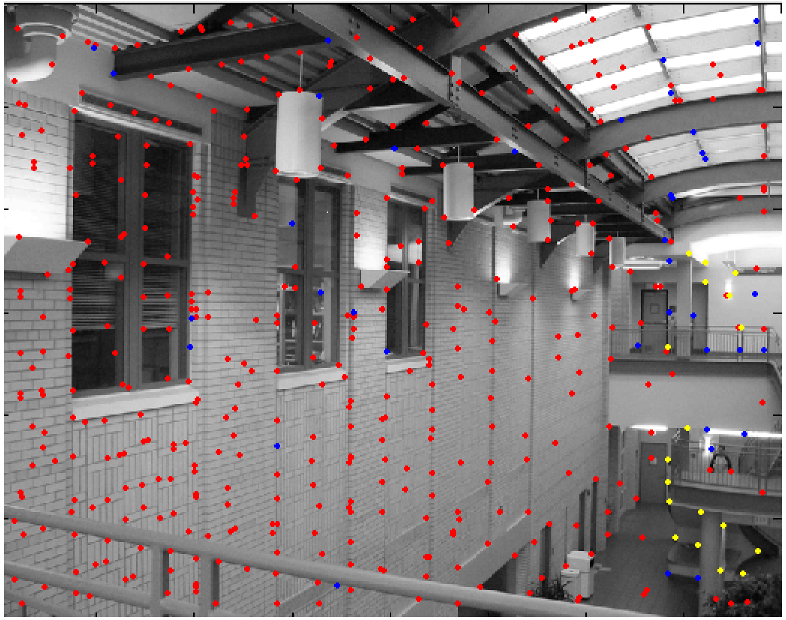
Find “average” translation vector

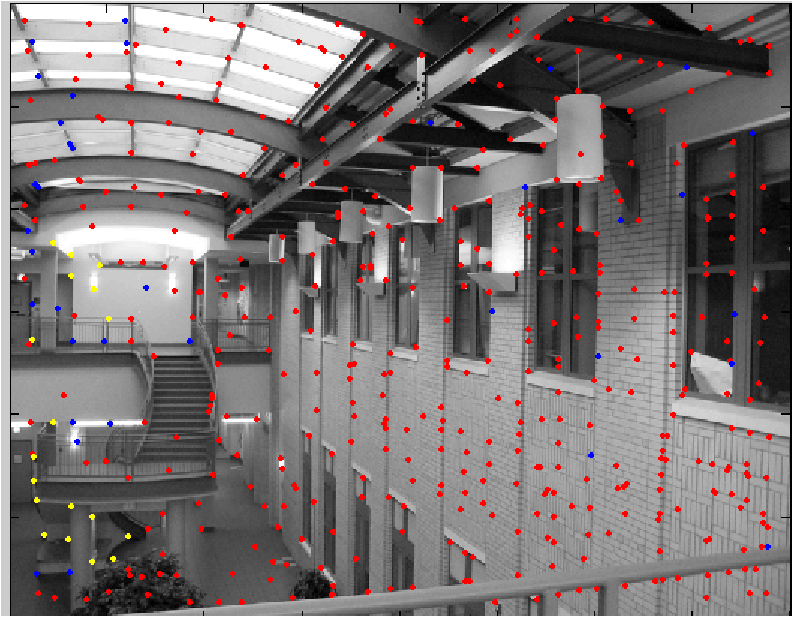
# RANSAC for estimating homography

- RANSAC loop:

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

# RANSAC





# RANSAC in general

- RANSAC = Random Sample Consensus
- an algorithm for robust fitting of models in the presence of many data outliers
- Compare to robust statistics

- Given $N$ data points $x_i$, assume that mjority of them are generated from a model with parameters $\Theta$, try to recover $\Theta$.

# RANSAC algorithm

Run $k$ times: How many times?

(1) draw $n$ samples randomly How big? Smaller is better

(2) fit parameters $\Theta$ with these $n$ samples

(3) for each of other $N$-$n$ points, calculate its distance to the fitted model, count the number of inlier points, $c$

Output $\Theta$ with the largest $c$

How to define? Depends on the problem.

# How to determine k

$n$: number of samples drawn each iteration

$p$: probability of real inliers

$P$: probability of at least 1 success after k trials

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

$p^n$: n samples are all inliers

$1 - p^n$: a failure

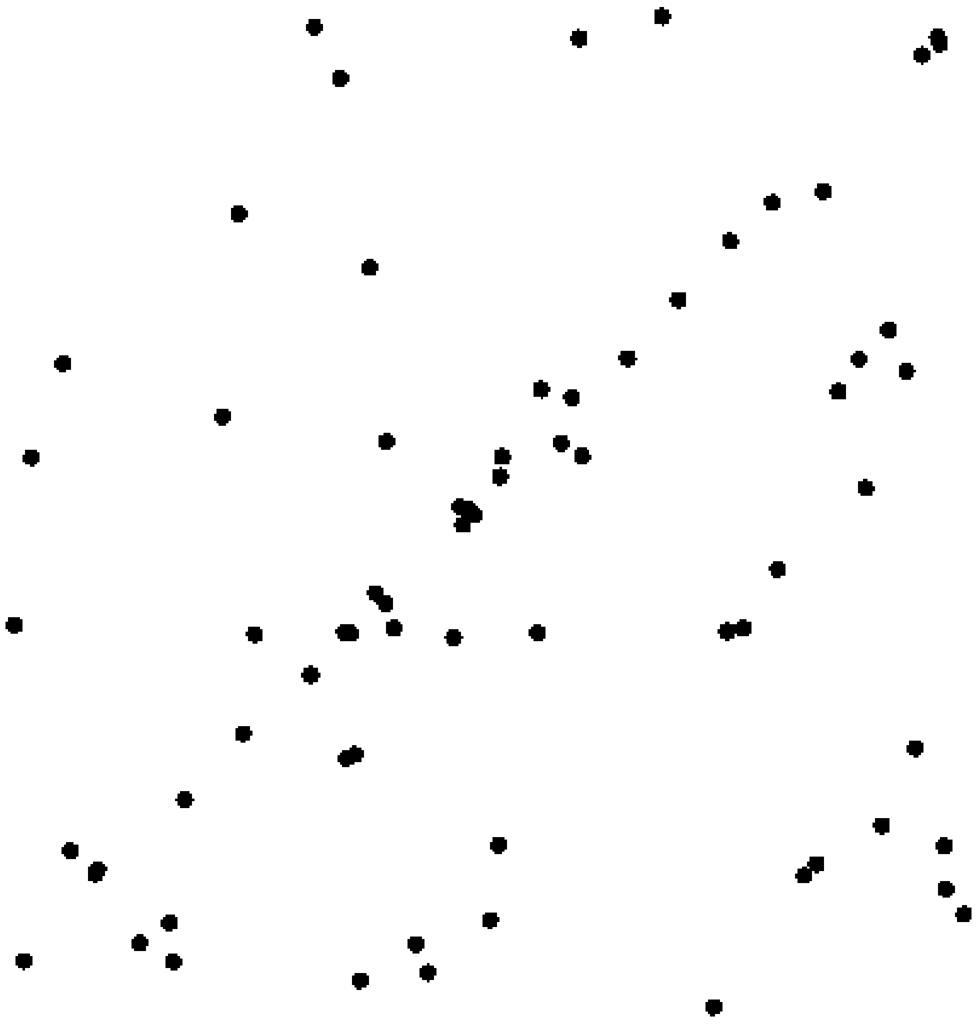
$(1 - p^n)^k$: failure after k trials

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

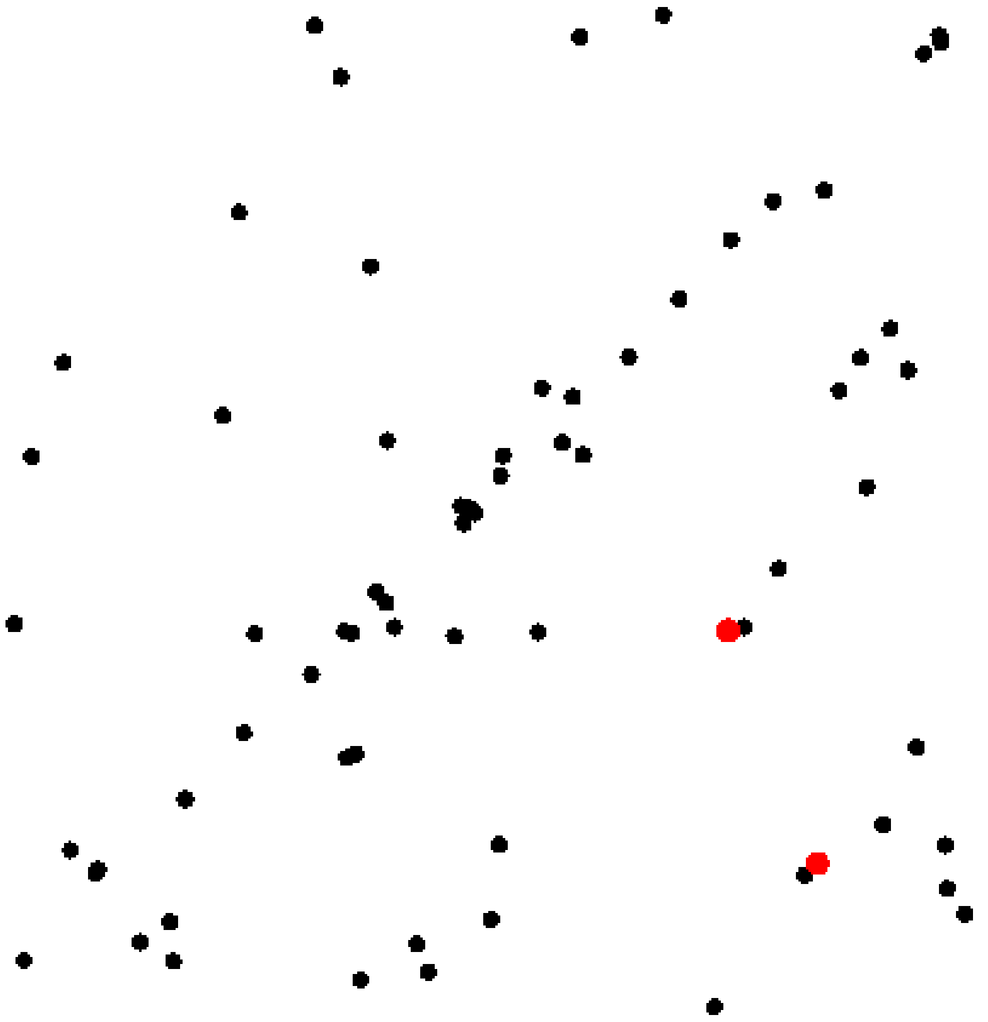
for $P = 0.99$

| $n$ | $p$ | $k$ |
|---|---|---|
| 3 | 0.5 | 35 |
| 6 | 0.6 | 97 |
| 6 | 0.5 | 293 |

# Example: line fitting

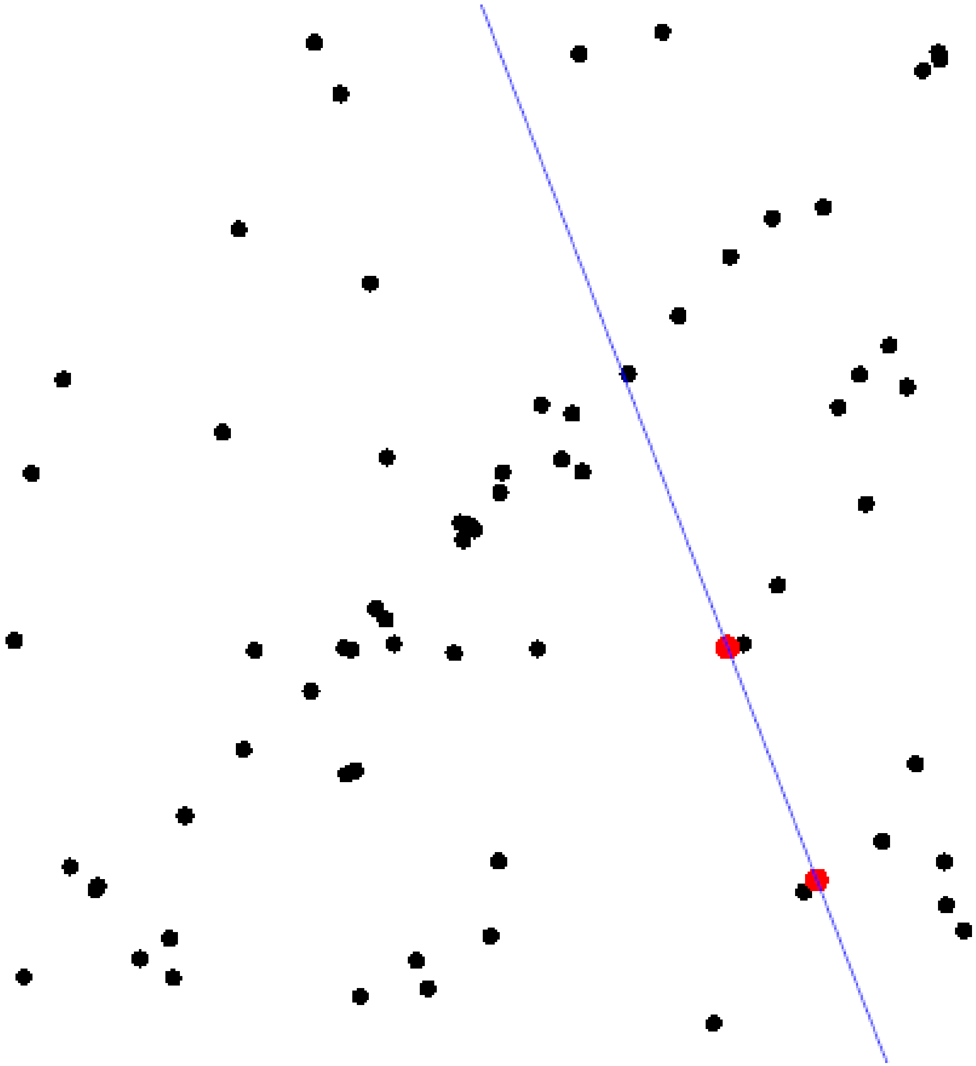


# Example: line fitting

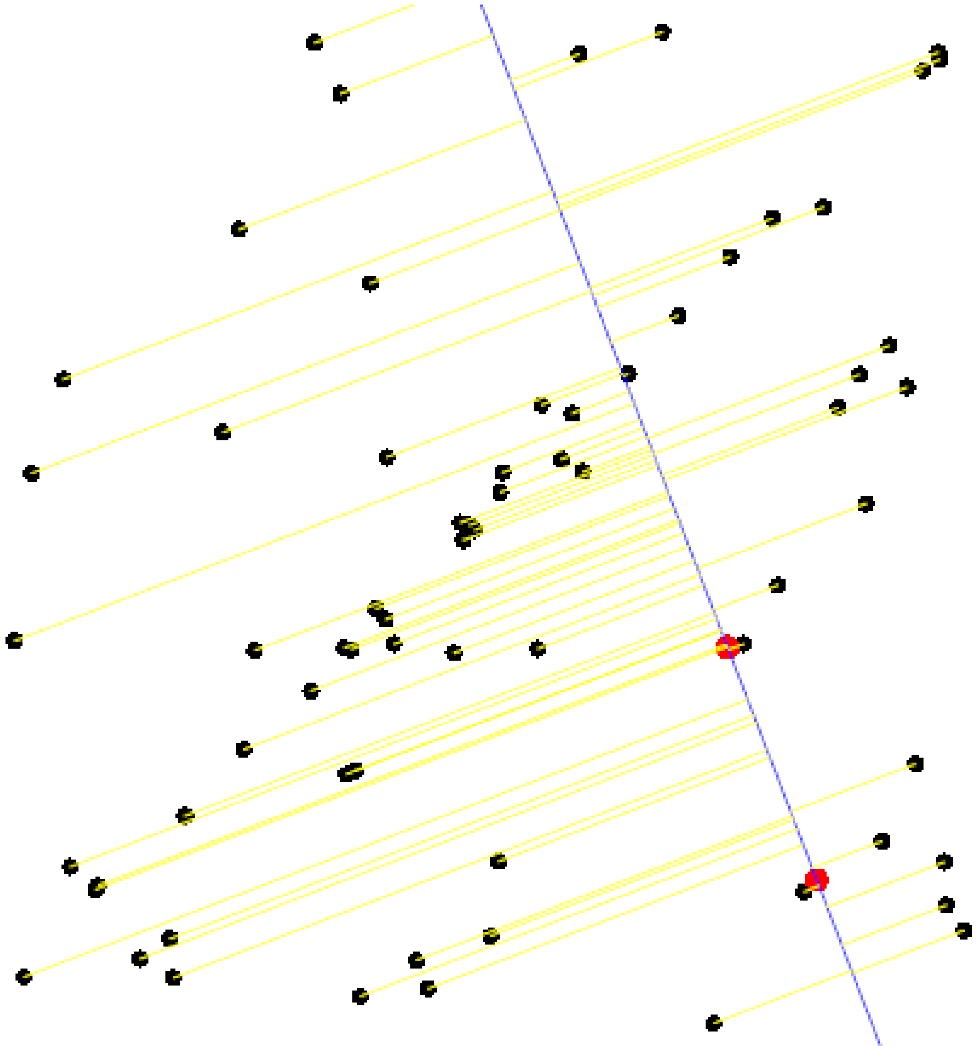


$n=2$

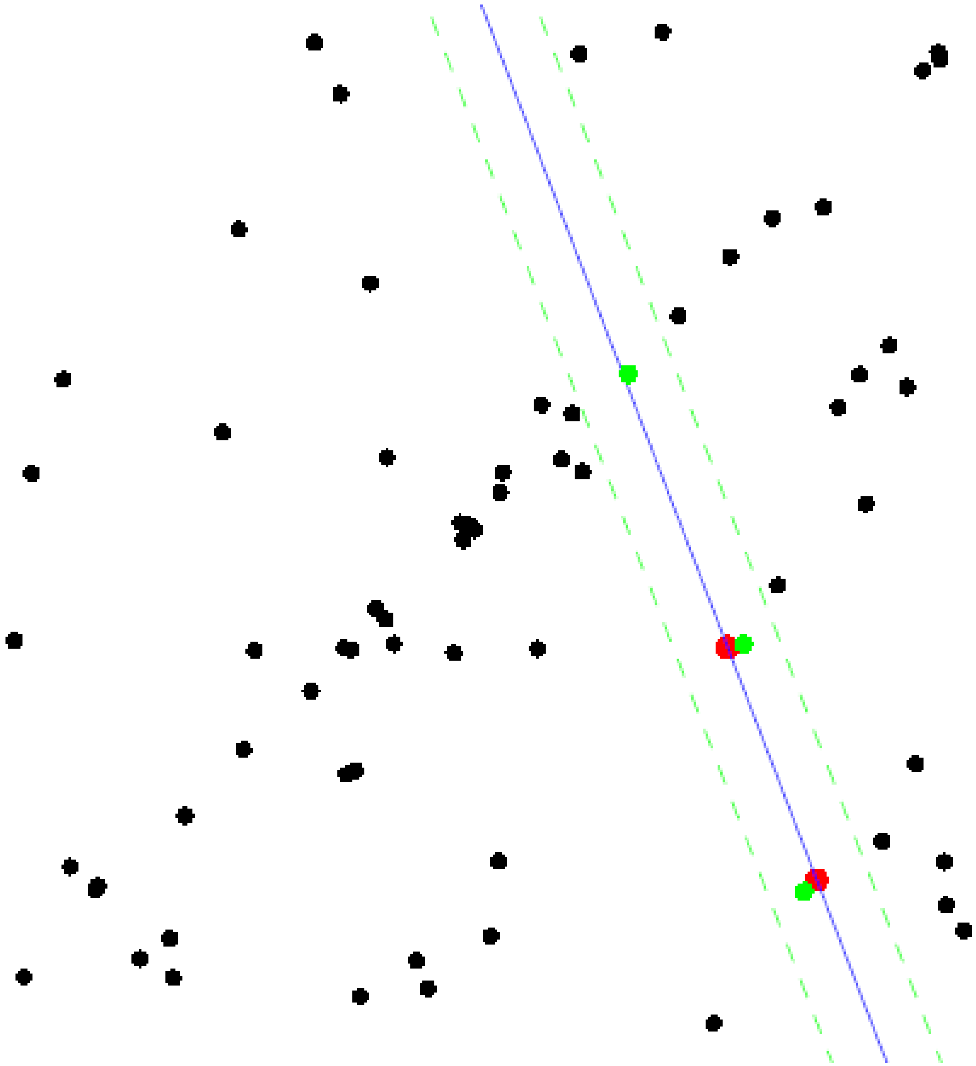
# Model fitting



# Measure distances

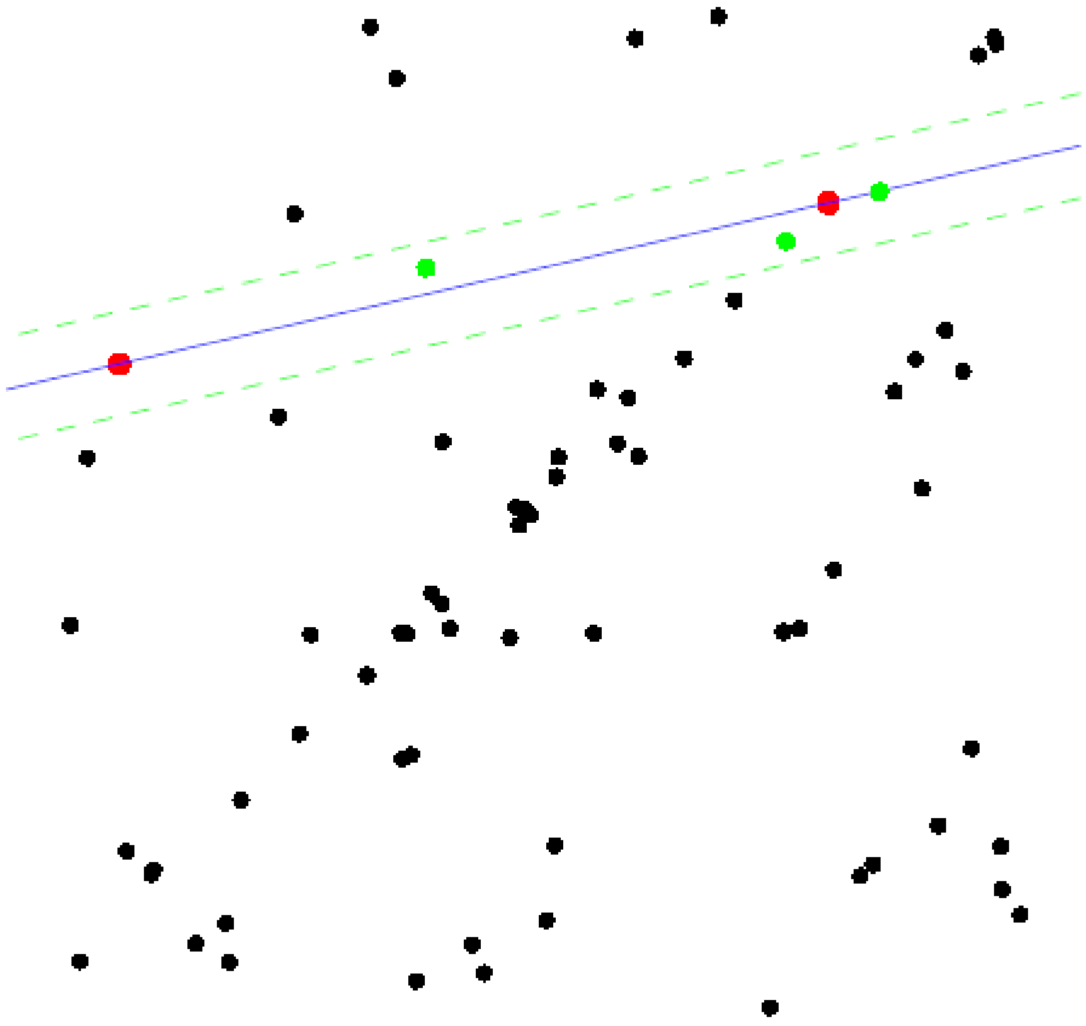


# Count inliers



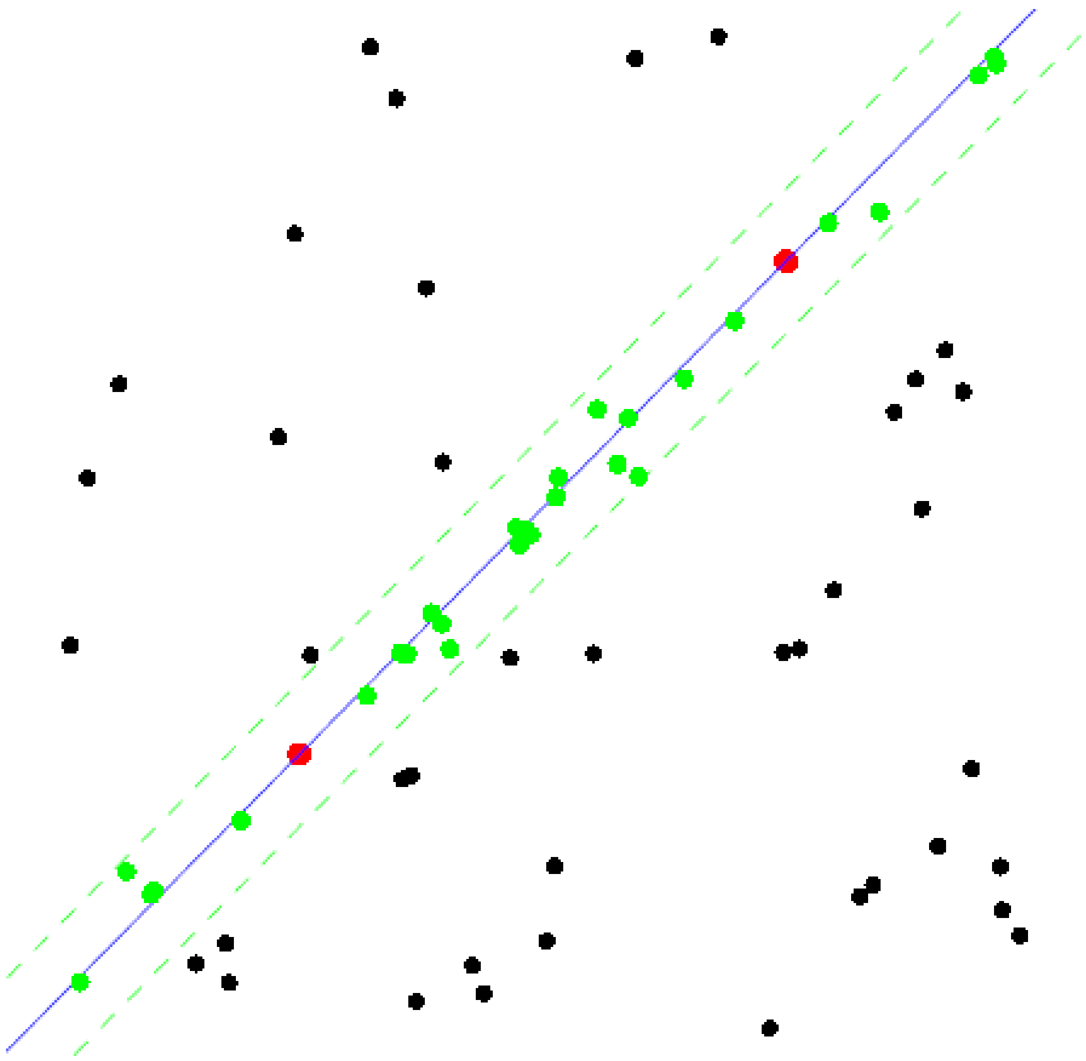
$c=3$

# Another trial



$c=3$

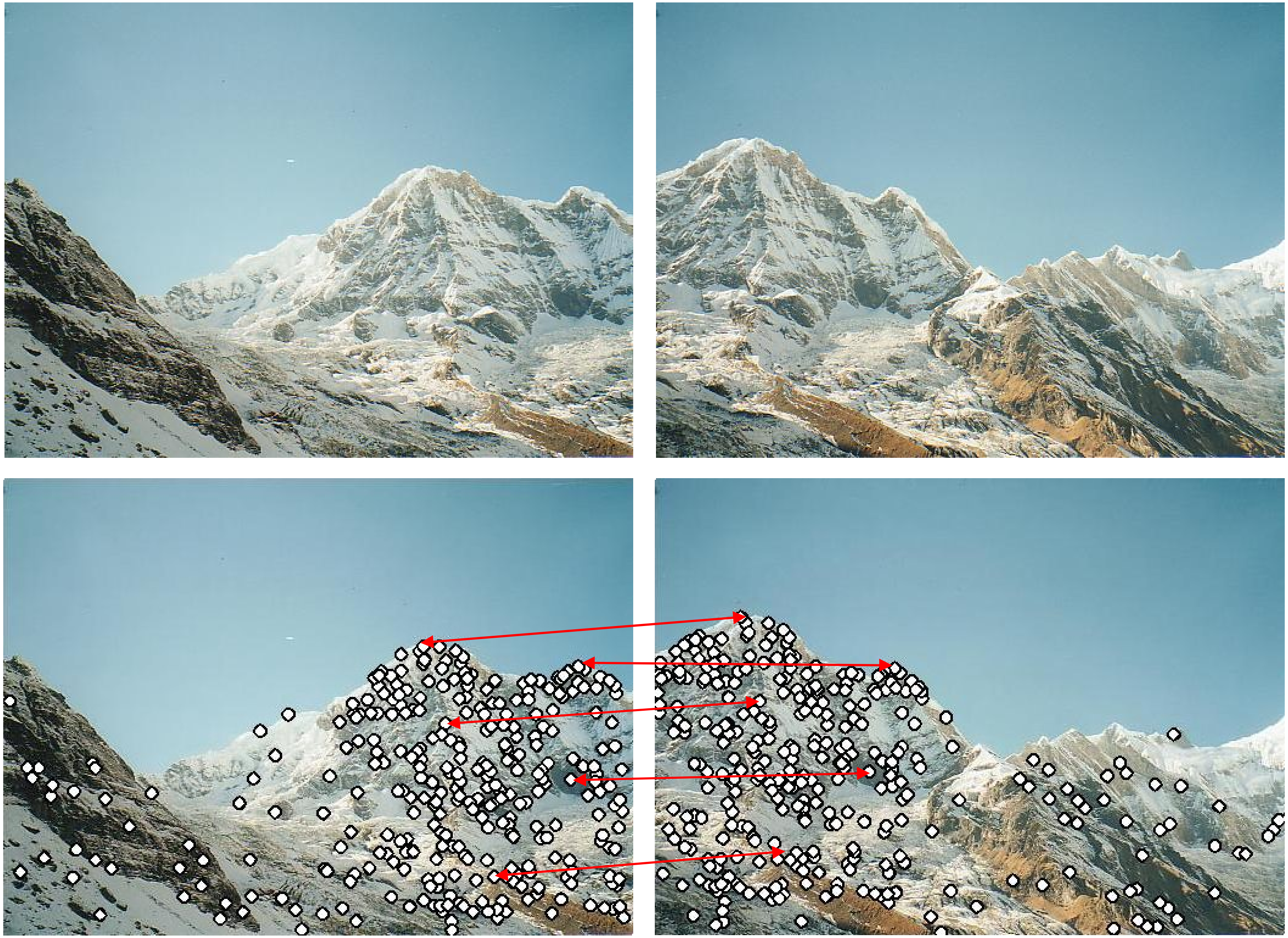
# The best model



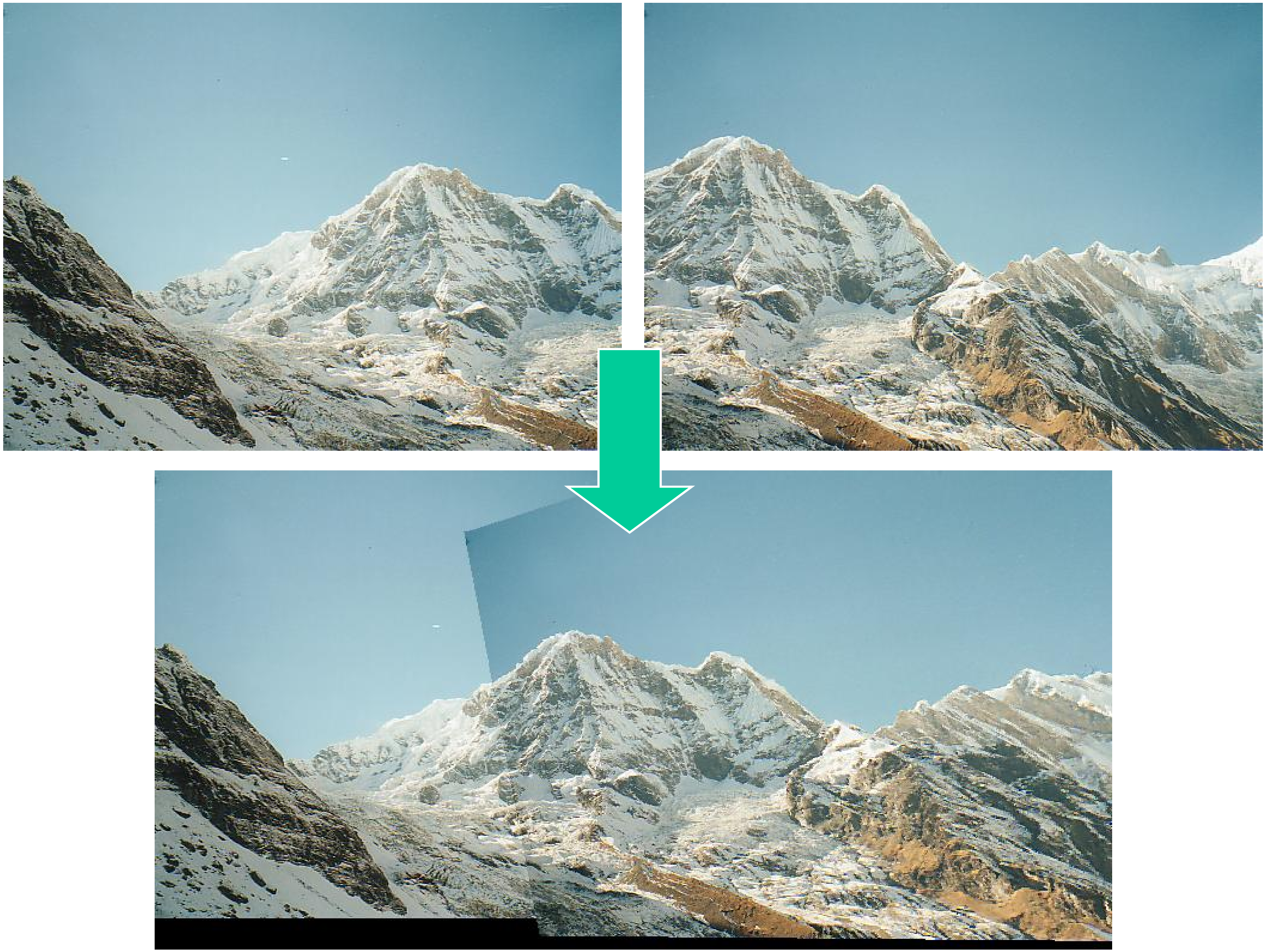
$c=15$

# Applications

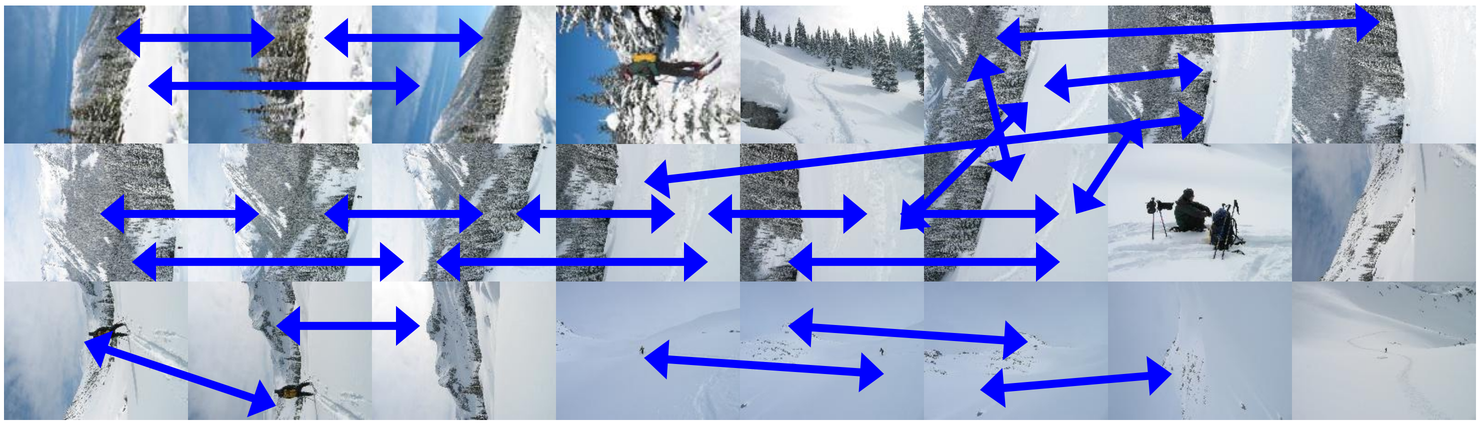
# Automatic image stitching



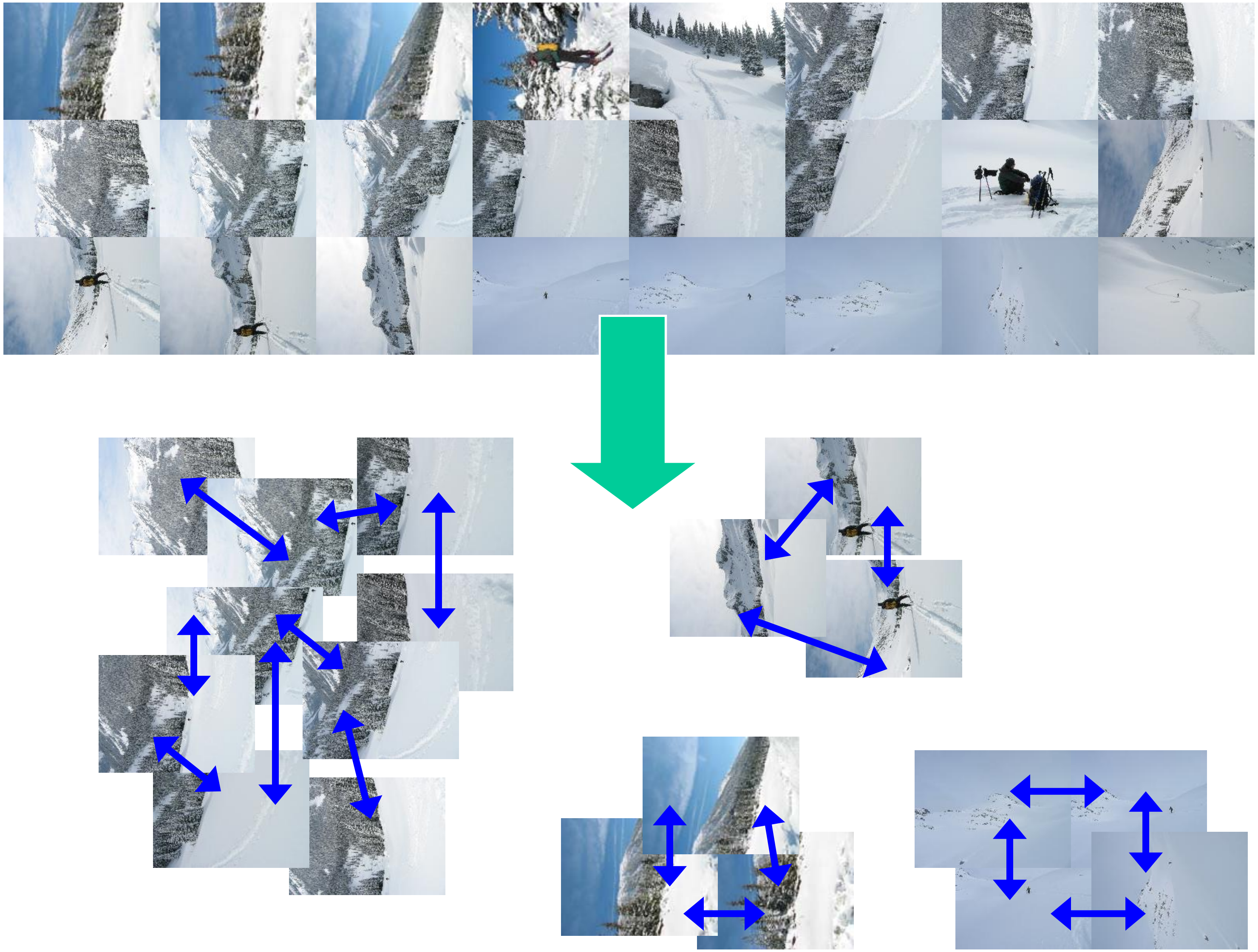
# Automatic image stitching



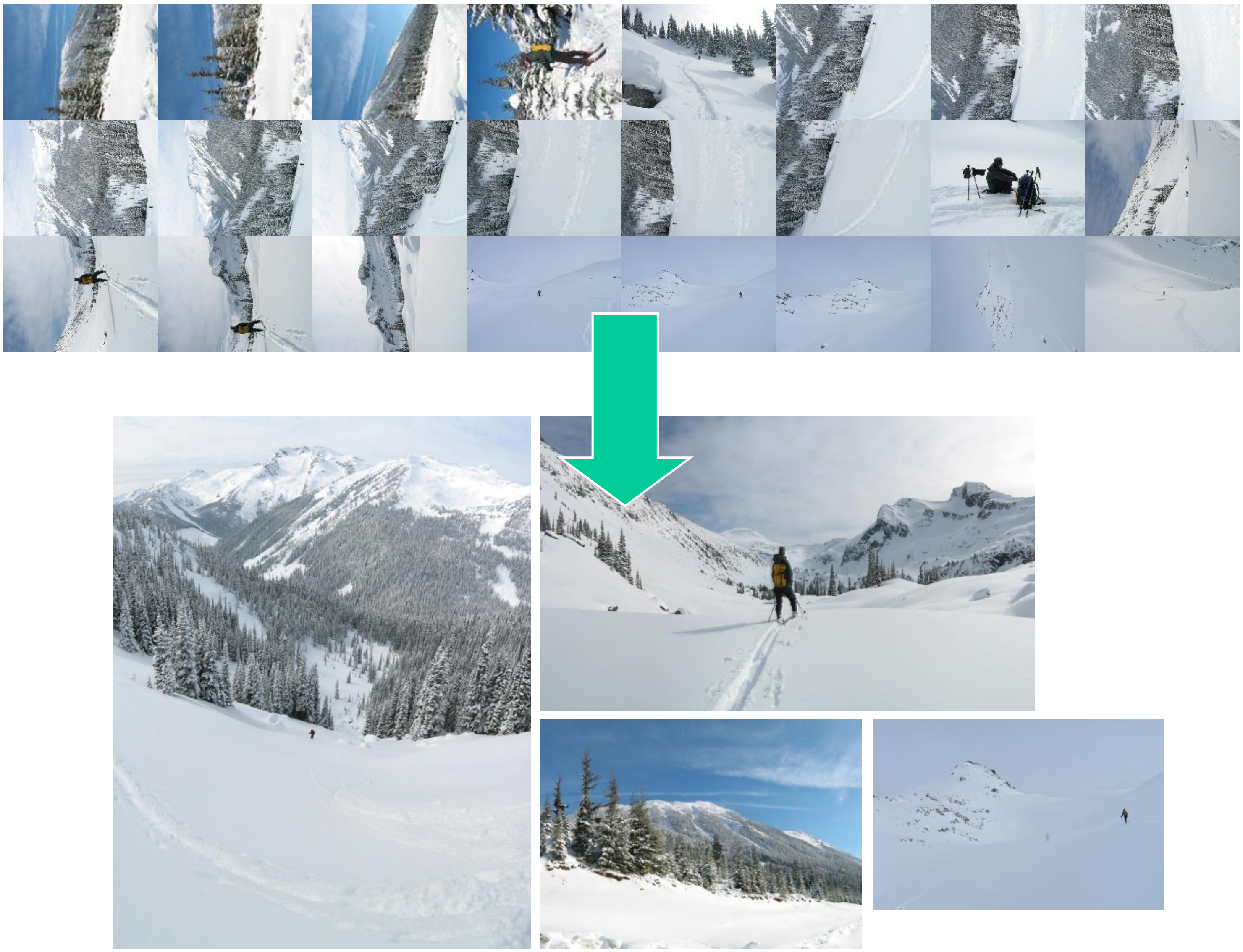
# Automatic image stitching



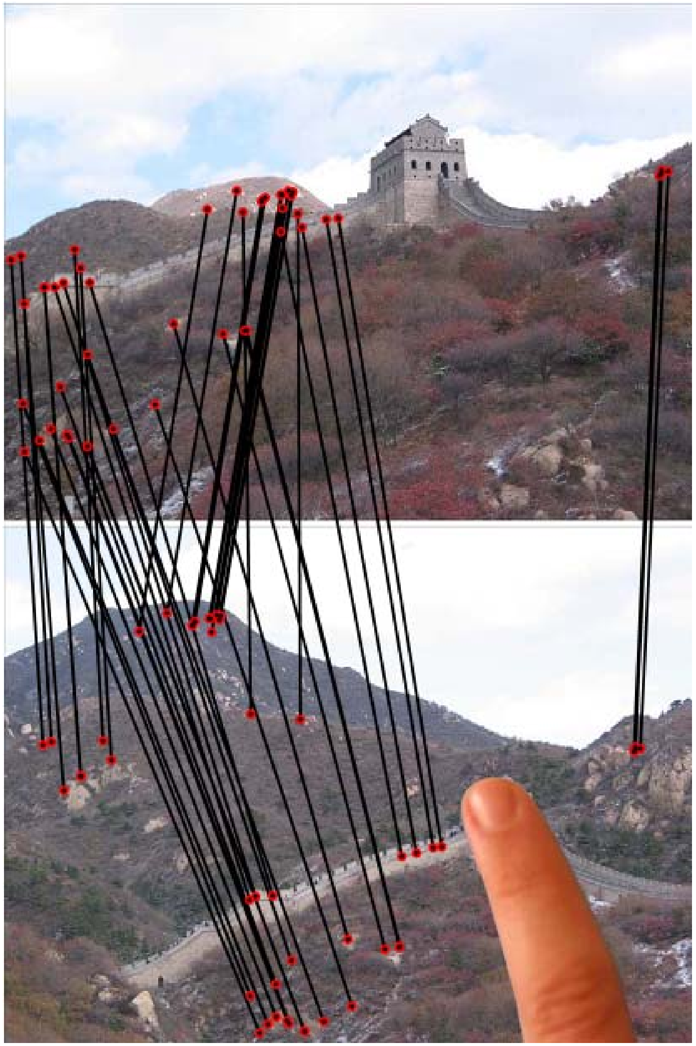
# Automatic image stitching



# Automatic image stitching

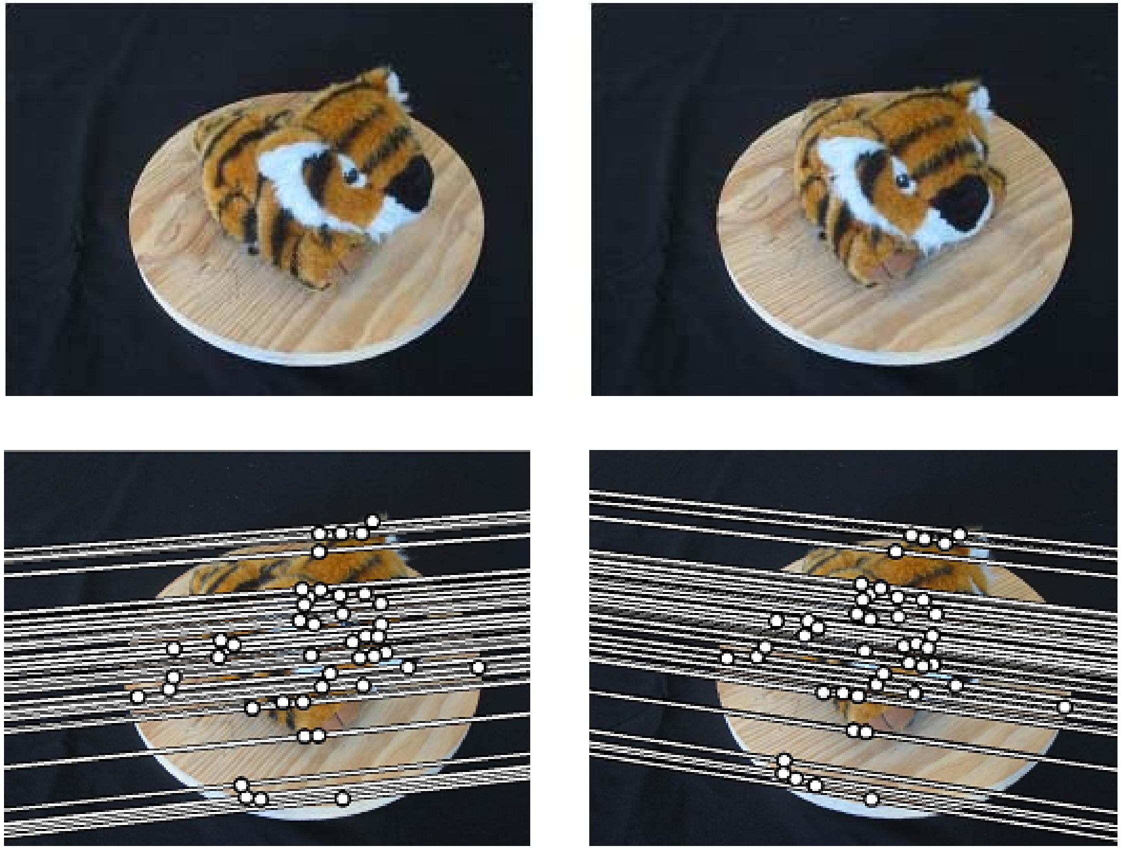


# Correspondence Results



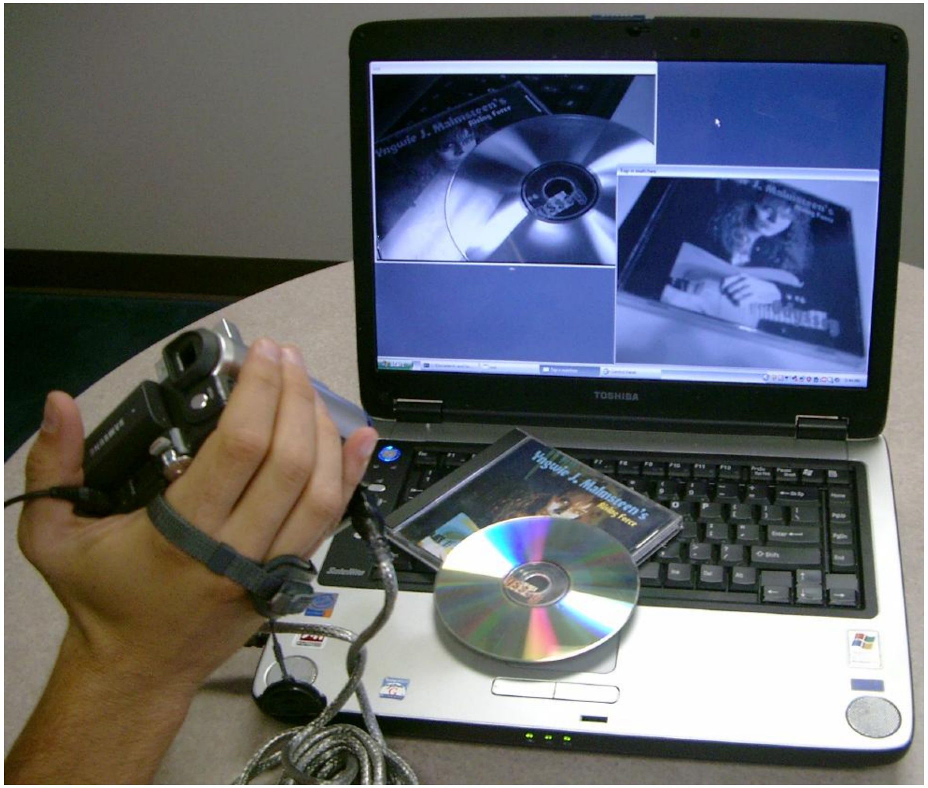
Chum & Matas 2005

# Object Recognition Results



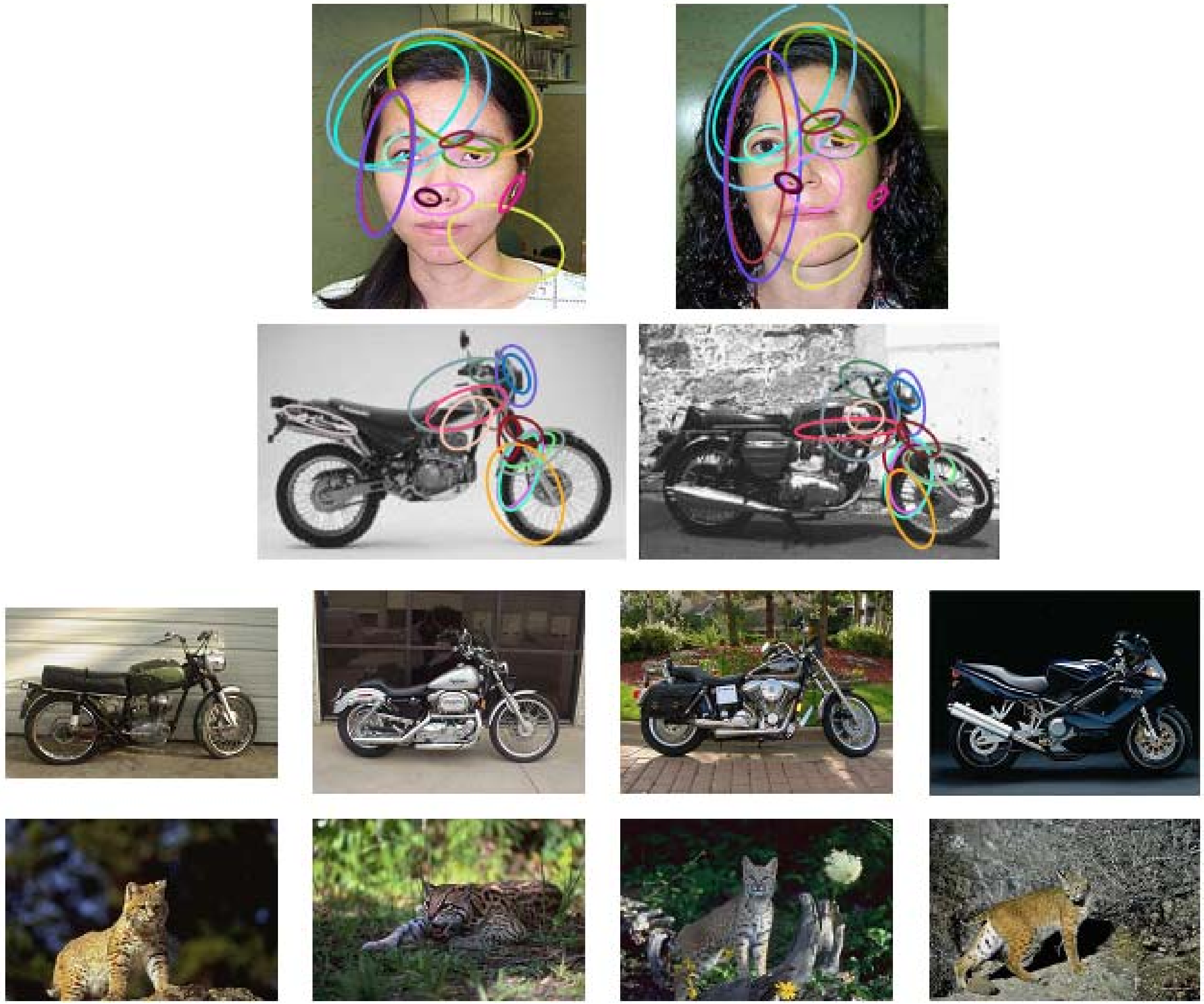
Brown & Lowe 2005

# Object Recognition Results



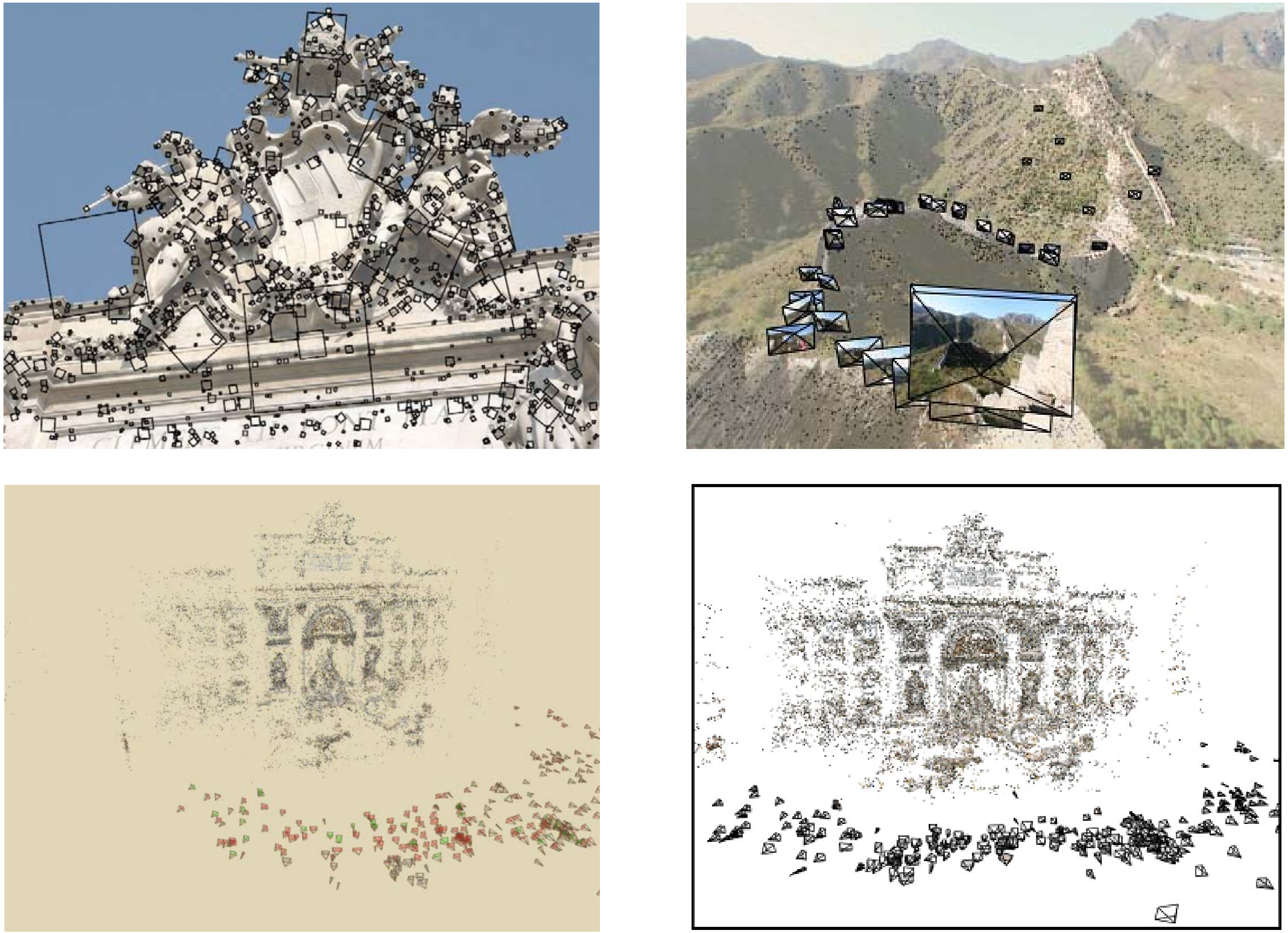

Nister & Stewenius 2006

# Object Classification Results



Grauman & Darrell 2006, Dorko & Schmid 2004

# Geometry Estimation Results



Snavely, Seitz, & Szeliski 2006

# Object Tracking Results

Gordon & Lowe 2005

# Robotics: Sony Aibo

SIFT is used for

- Recognizing charging station
- Communicating with visual cards
- Teaching object recognition
- soccer