

Features, Harris Corners and Descriptors

Frédo Durand
MIT EECS 6.8370/6.8371



Julian Beever

e.g. <http://users.skynet.be/J.Beever/>
http://www.crystalinks.com/julian_beever.html

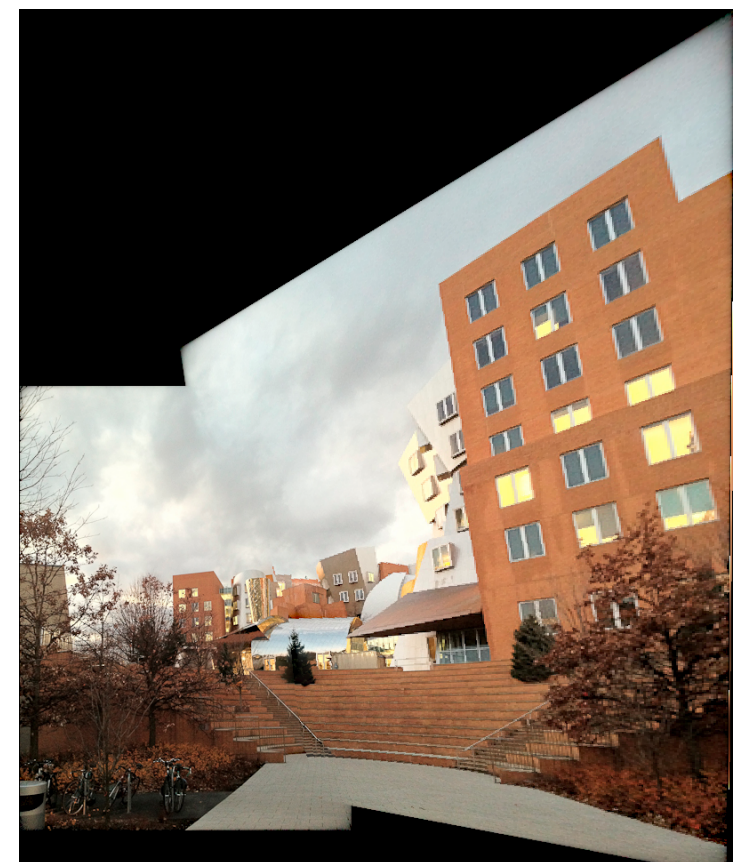
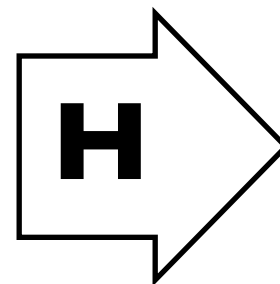


Recap



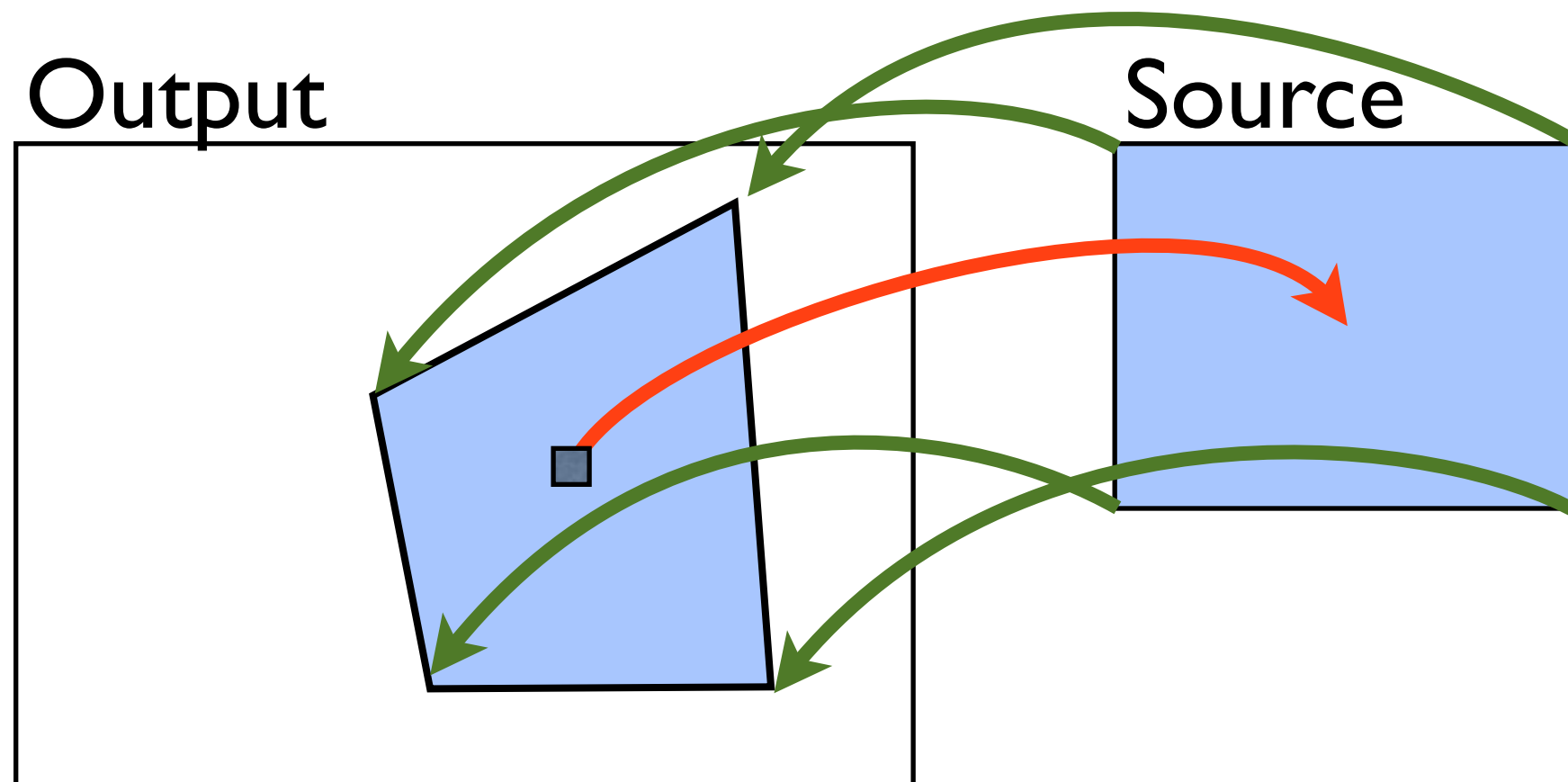
Manual linear panoramas

- Create a virtual wide angle view from 2 images
- Choose one image as reference
- User gives 4 correspondences
- Deduce homography matrix
- Reproject (warp) second image into first one



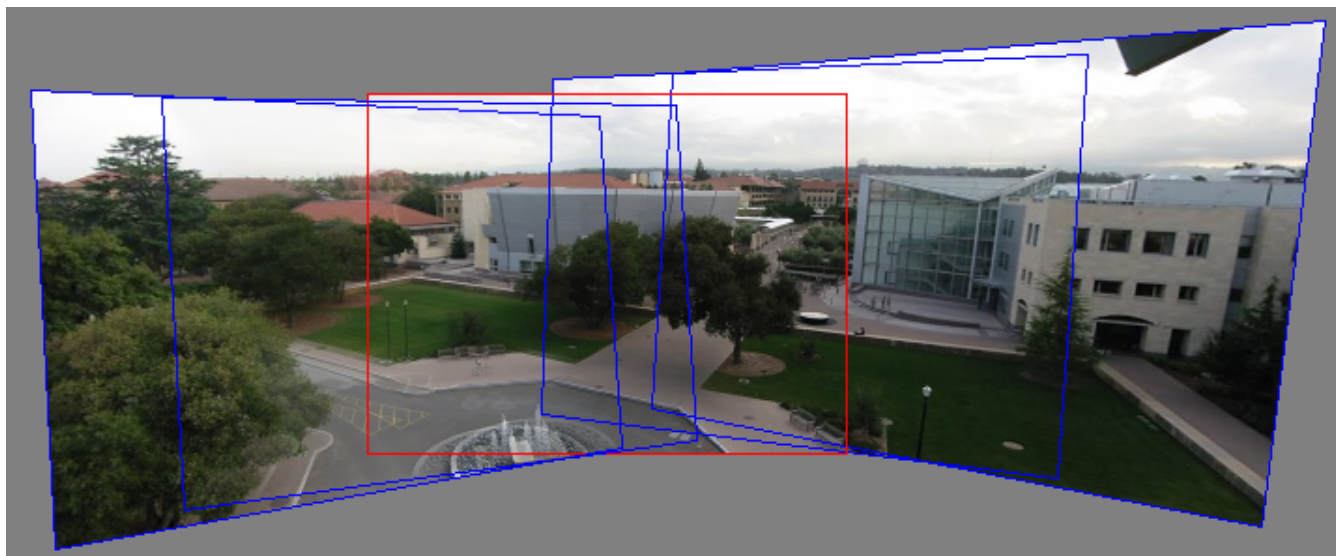
Forward backward Homography

- To know bbox:
 - Go from corner in source space to output domain
- To make sure we fill all output pixels
 - loop over output pixels=> need inverse homography



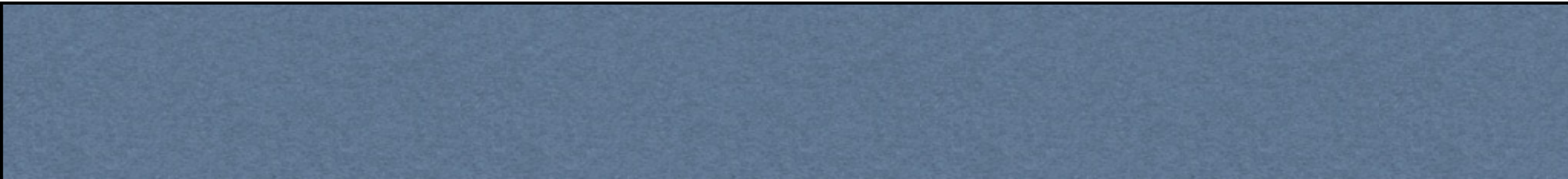
Recap

- **Panorama = reprojection**
- **3D rotation $\rightarrow$ homography**
 - Homogeneous coordinates are kewl
- **Use feature correspondence**
- **Solve linear problem**
 - Set of linear equations
- **Warp all images to a reference one**
- **Use your favorite blending**





Old-style panorama



Recall brute-force alignment?

- Try all possible translation
- compute sum of square differences
- Keep best



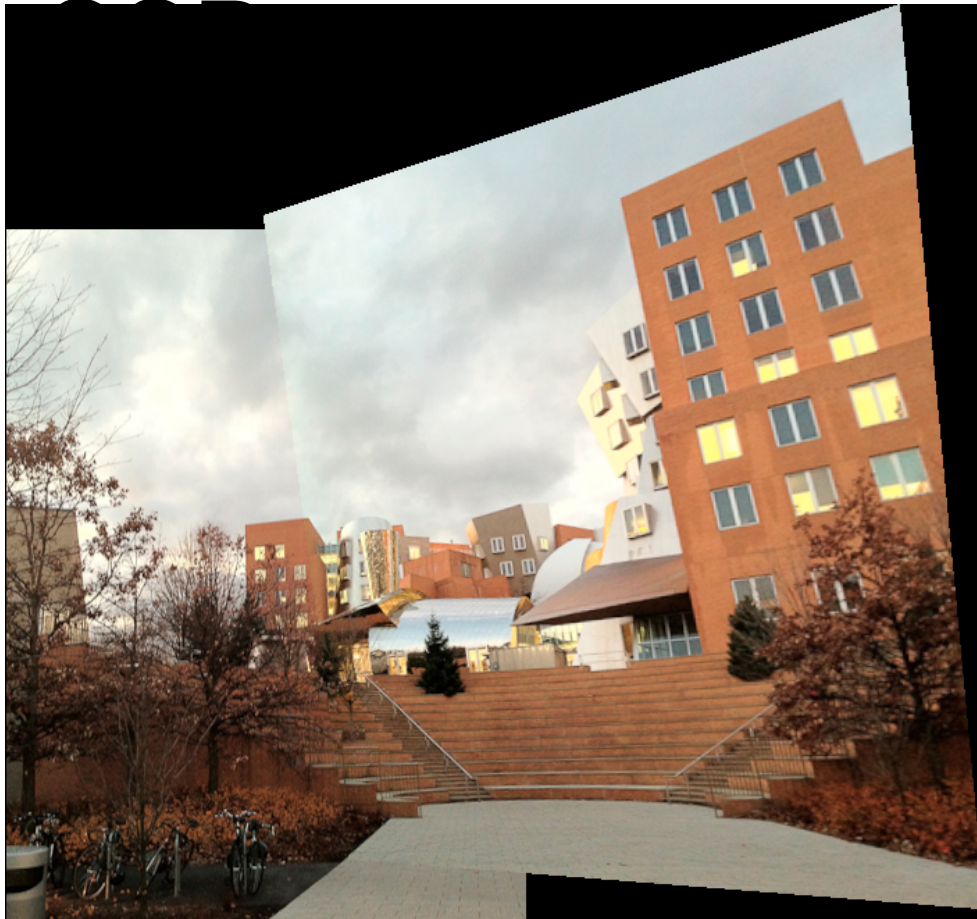
Recall brute-force alignment?

- We sought a **dense** match:
we evaluated error for all the pixels



Brute force homography?

- Discretize the space of 8 coefficients?
 - Crazy combinatorics!
 - Even if you know the focal length and reduce it to 3 rotational degrees of freedom
 - Costly evaluation: need to warp and compute



(Old) dense approach

- Reduce to 1 or 2 degrees of freedom
 - Assume no rotation around optical axis (camera stays horizontal)
 - Use smart optimization, gradients
 - Work in cylindrical coordinates: rotation becomes translation and warping is easier
 - <http://research.microsoft.com/apps/pubs/default.aspx?>



(a)

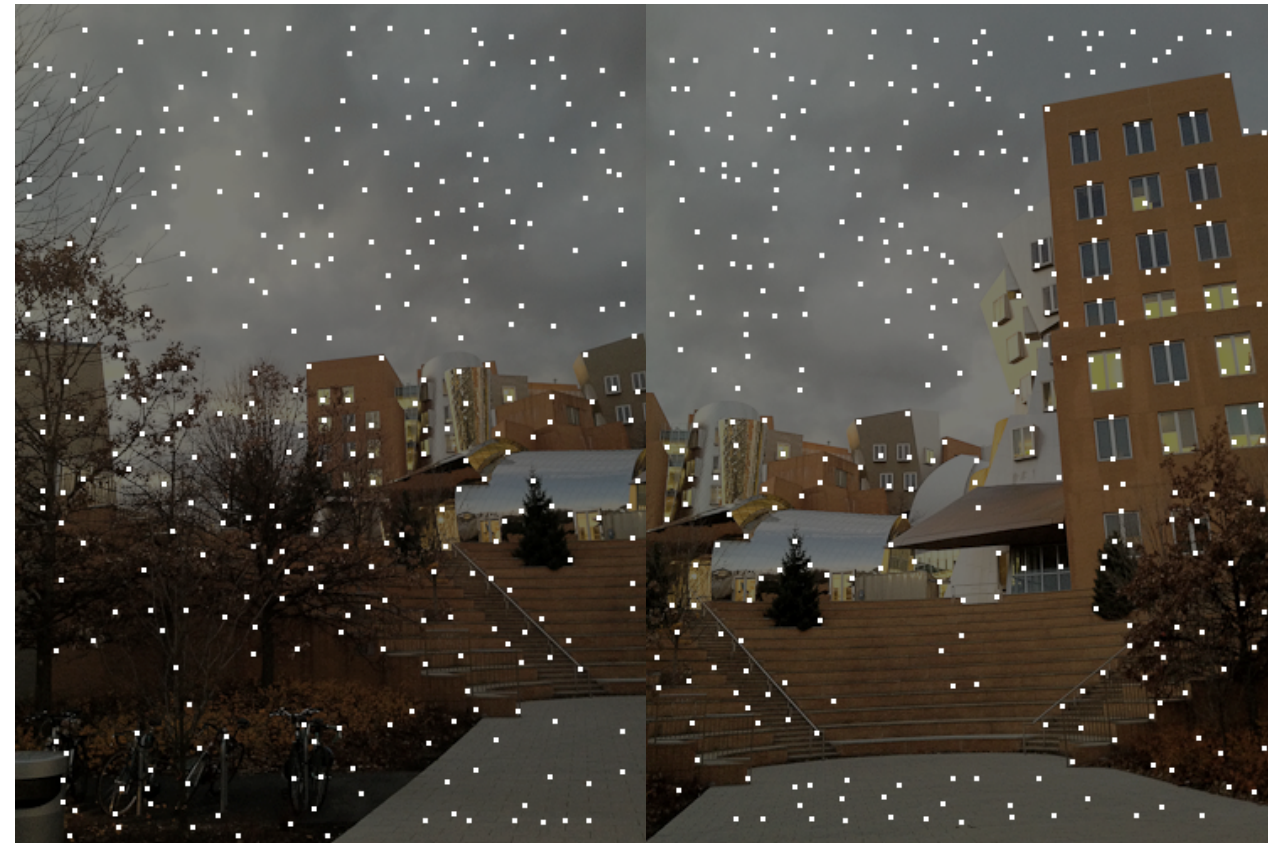


(b)

Fig. 2.7 An example of a cylindrical panorama: (a) two cylindrically warped images related by a horizontal translation; (b) part of a cylindrical panorama composited from a sequence of images.

Modern *sparse* matching

- Focus on areas that are very distinctive:
Interest points
 - Forget uniform areas
- Do the matching locally:
which point matches
which point
- Add mechanisms to be robust to local mistakes



**Auto stitch
with sparse
features**

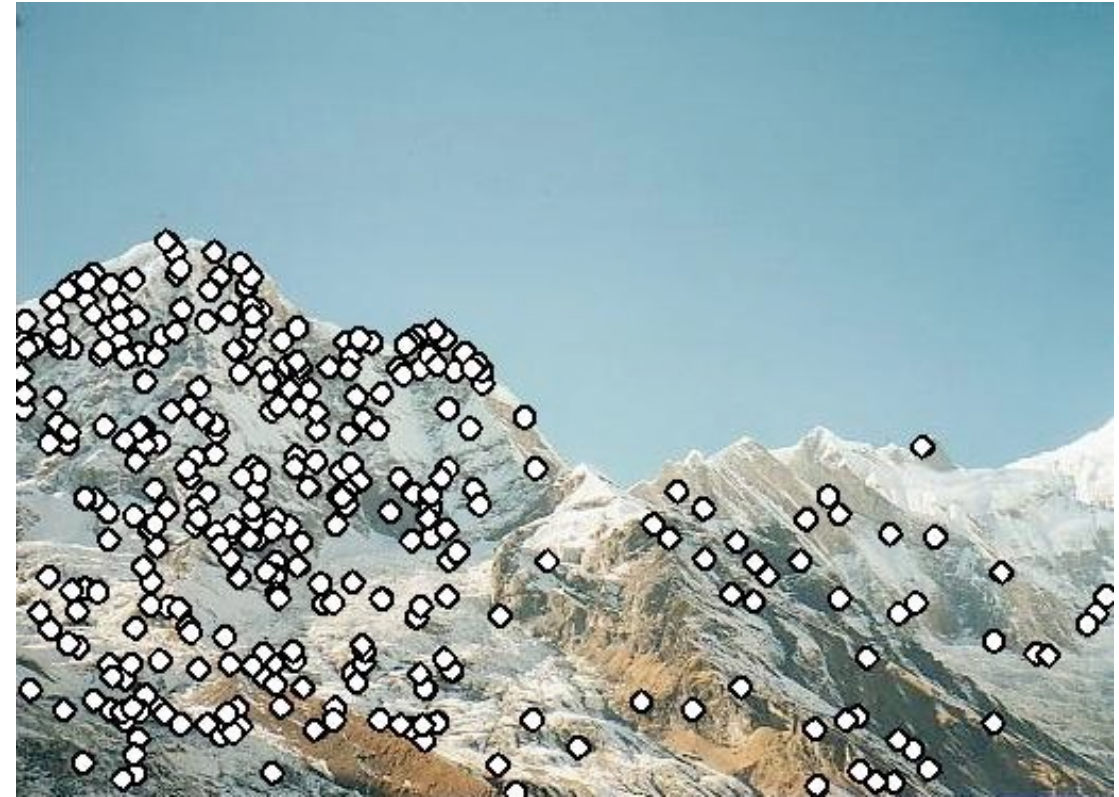
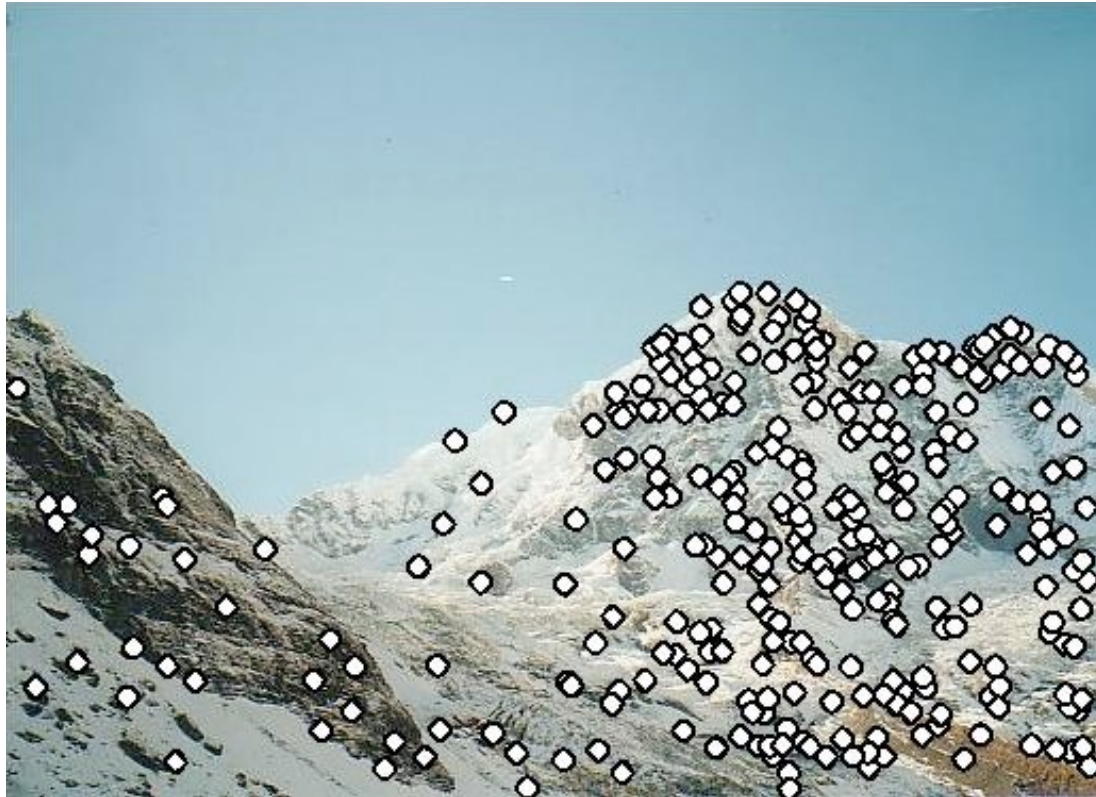
Automatic panorama

- We need to match (align) images



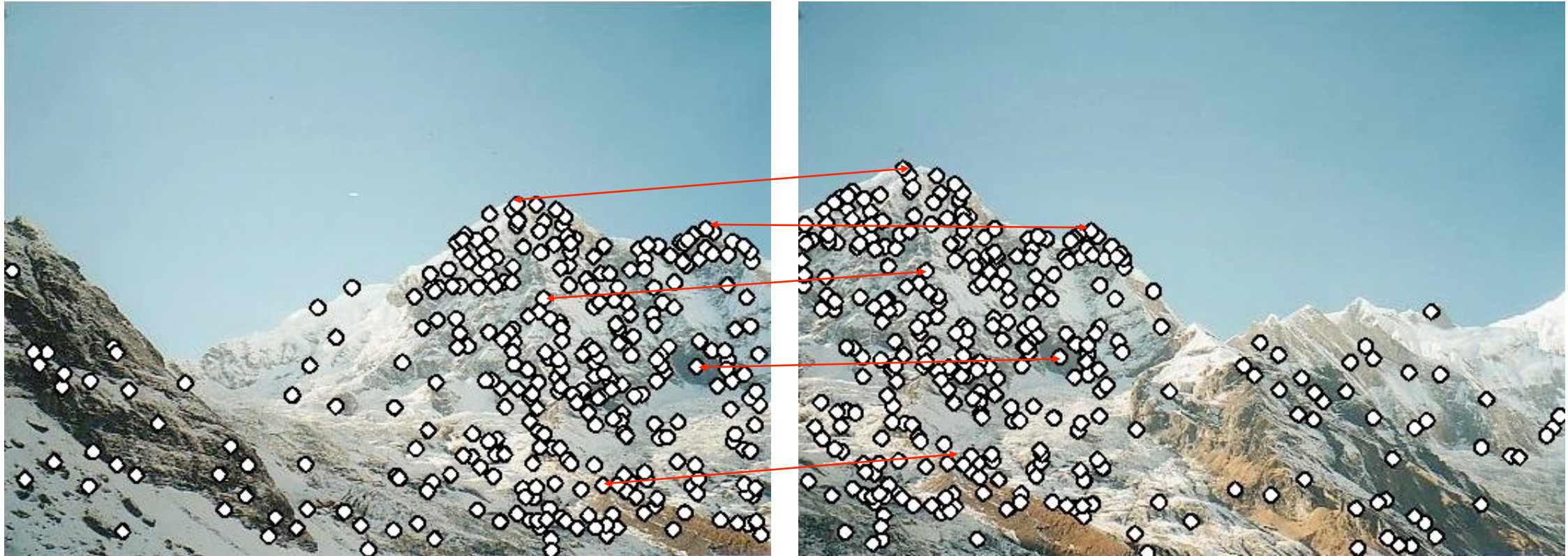
Matching with Features

- Detect feature points in both images



Matching with Features

- Detect feature points in both images
- Find corresponding pairs



Matching with Features

- Detect feature points in both images
- Find corresponding pairs
- Use these pairs to align images - we know this



Matching with Features

- Problem 1:
 - Detect the *same* point *independently* in both images

counter-example:

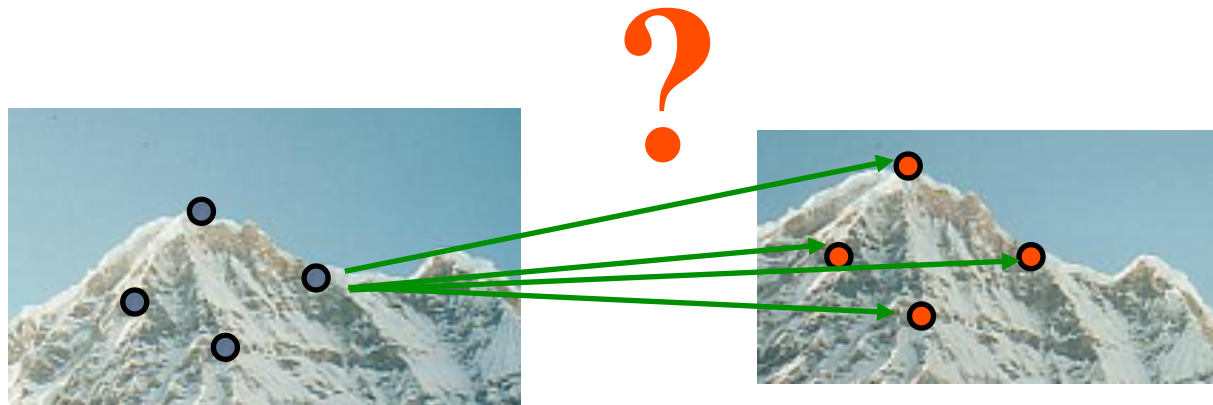


no chance to match!

We need a repeatable
detector

Matching with Features

- Problem 2:
 - For each point correctly recognize the corresponding one



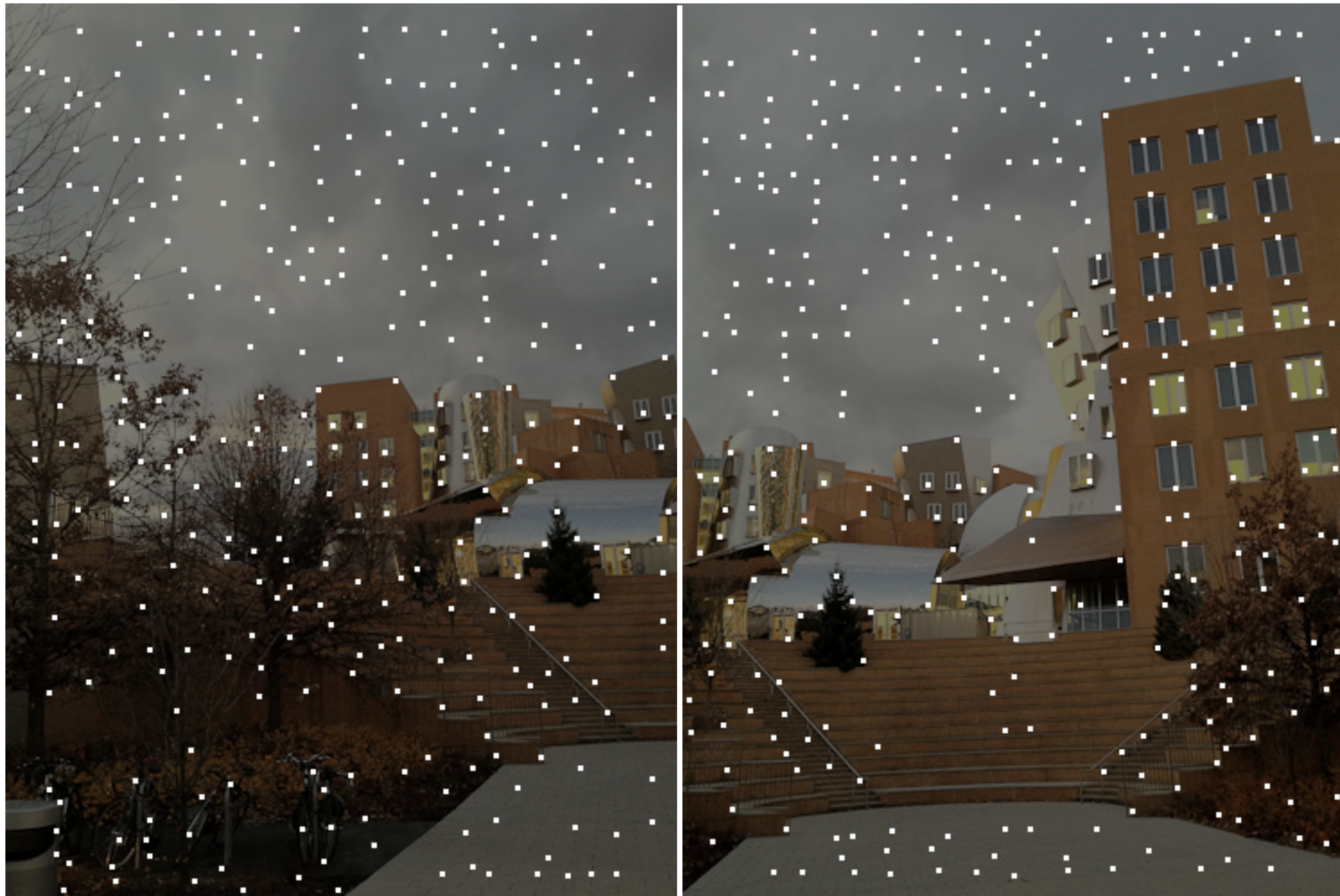
We need a reliable and distinctive **descriptor**

Discussion

- In this lecture, we'll seek to derive detectors and descriptors that work perfectly
- But in reality, they will never be 100% perfect
- Next time, we'll see how we can use the large number of feature points we can extract to reject outliers and get good homographies despite error.
- Plus we will only implement a simplistic detector/descriptor

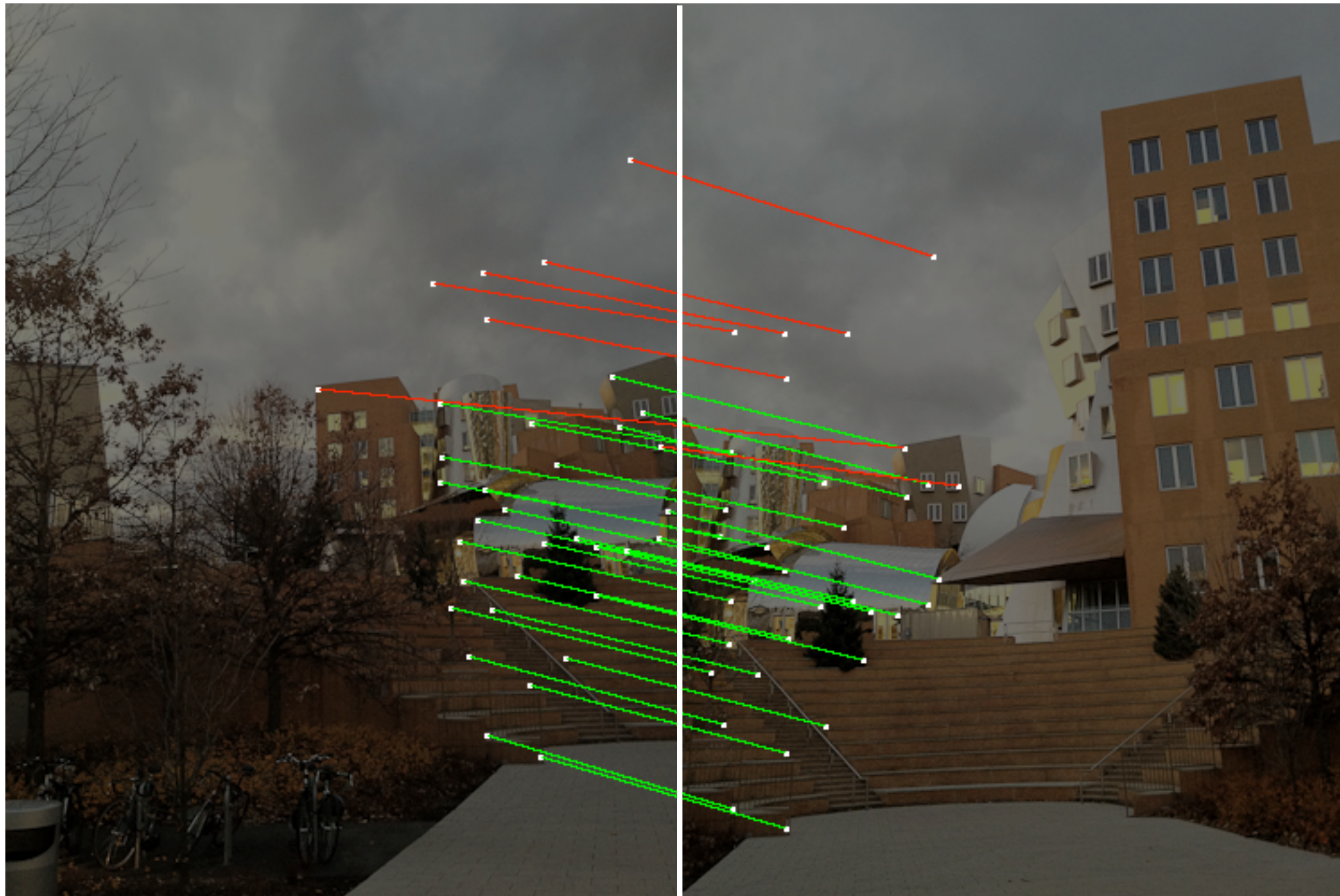
Preview of pset 7

- Automatically-extracted feature points



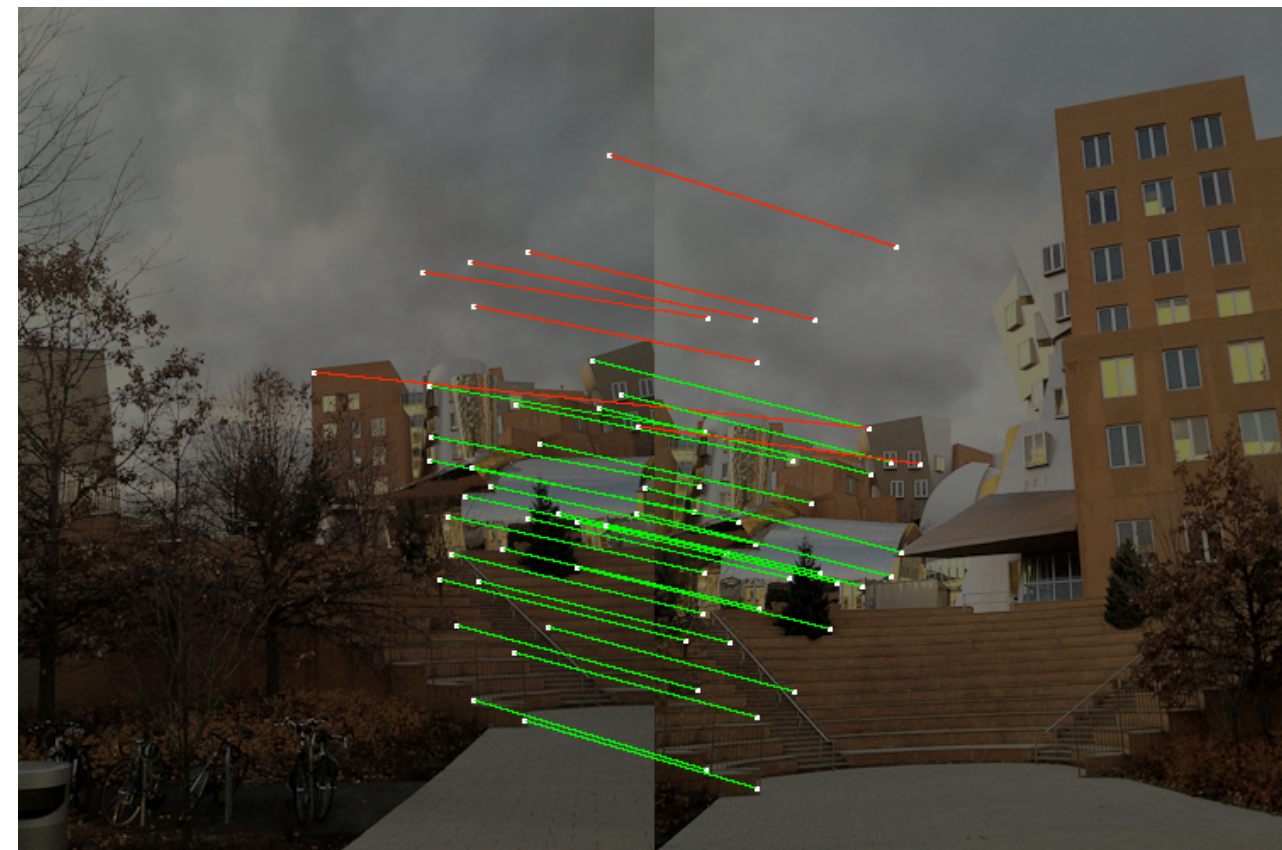
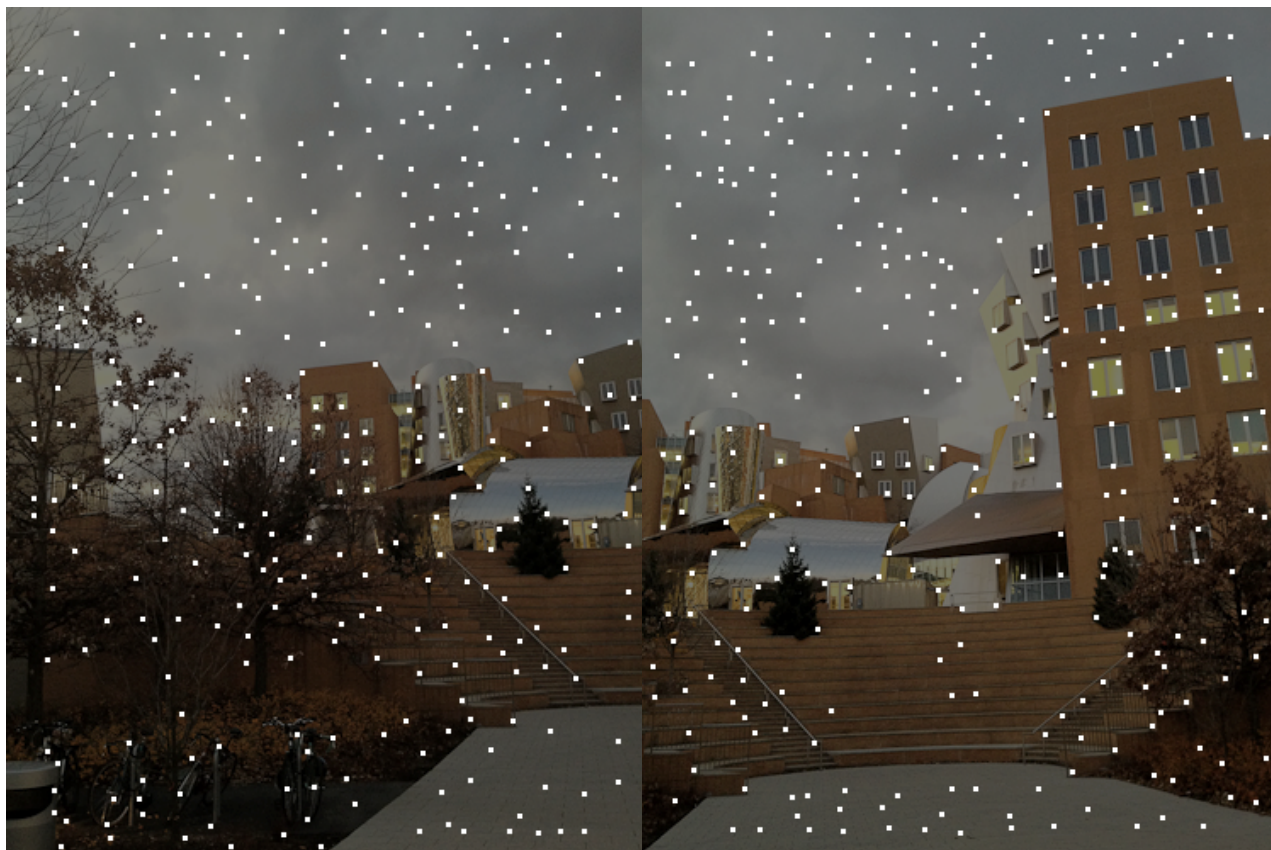
Preview of pset 7

- Only a subset of features are matched
- In green are matches determined correct



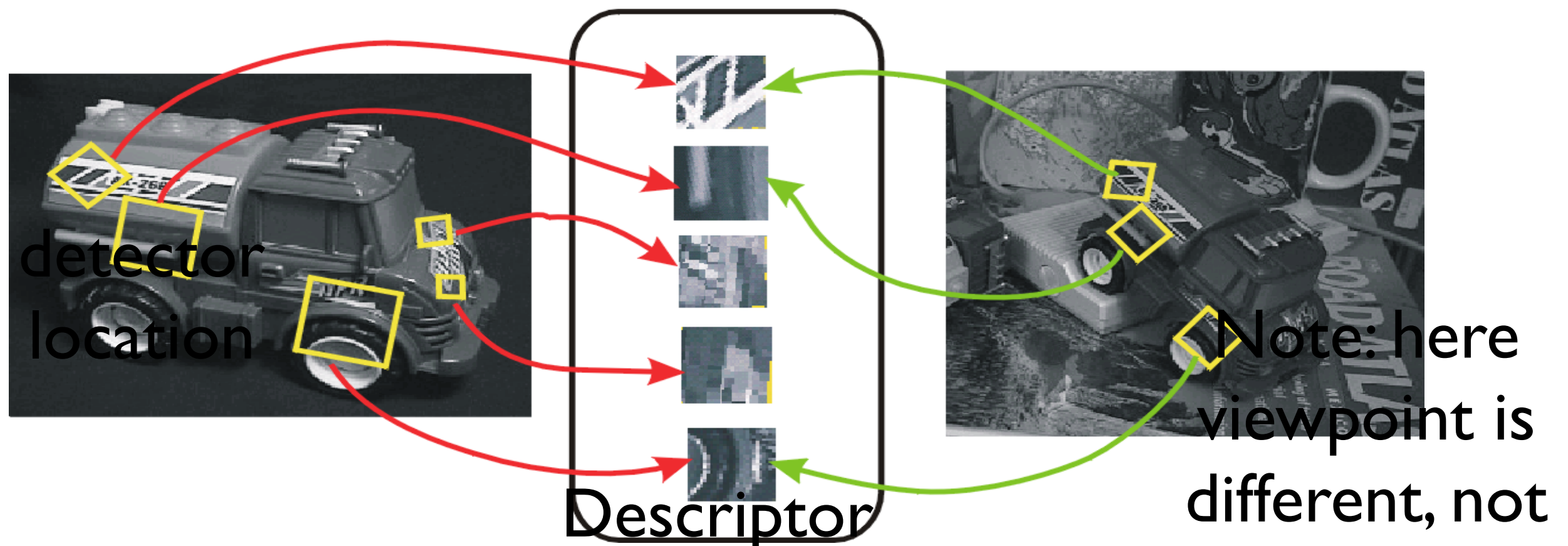
In a nutshell

- We will replace your manual clicks
- ***Detectors*** tell us where to click
- ***Descriptors*** tells us which click in an image corresponds to which one in the other one
- 2NN & RANSAC correct for mistakes



Preview

- Detector: detect same scene points independently in both images
- Descriptor: encode local neighboring window
 - Note how scale & rotation of window are the same in both image (but computed independently)
- Correspondence: find most similar descriptor in other image

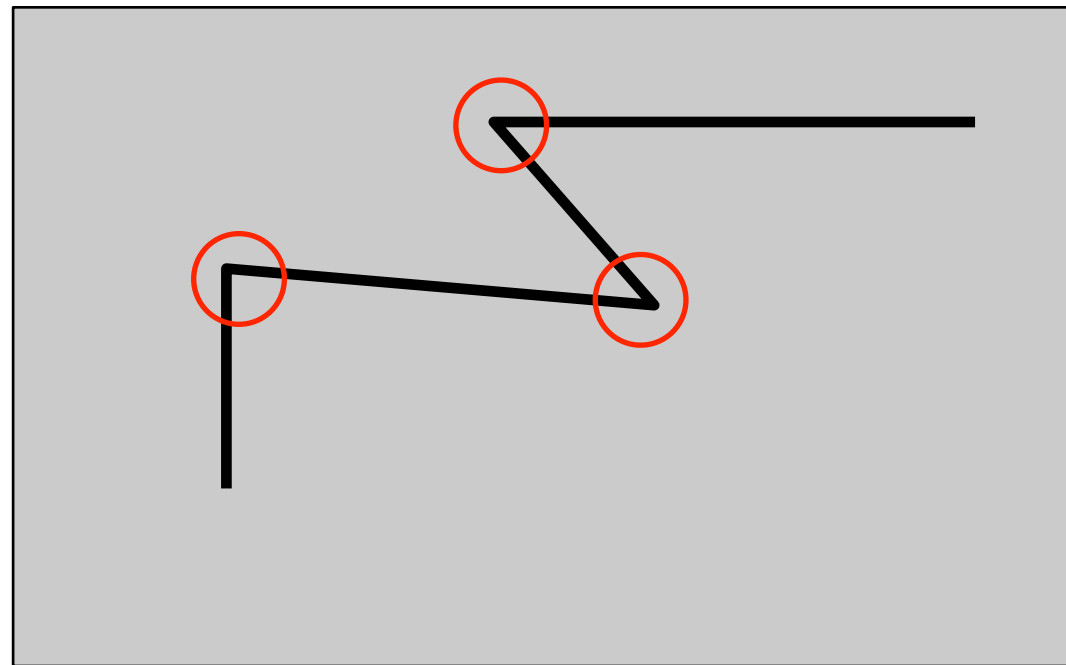


Important

- We run the detector and compute descriptors **independently** in the various images.
- It is only at the end, given a bag of feature points and their descriptors, that we compute correspondences
- But for the derivation, we will look how the detector and descriptor behave for corresponding scene points, to check for consistency

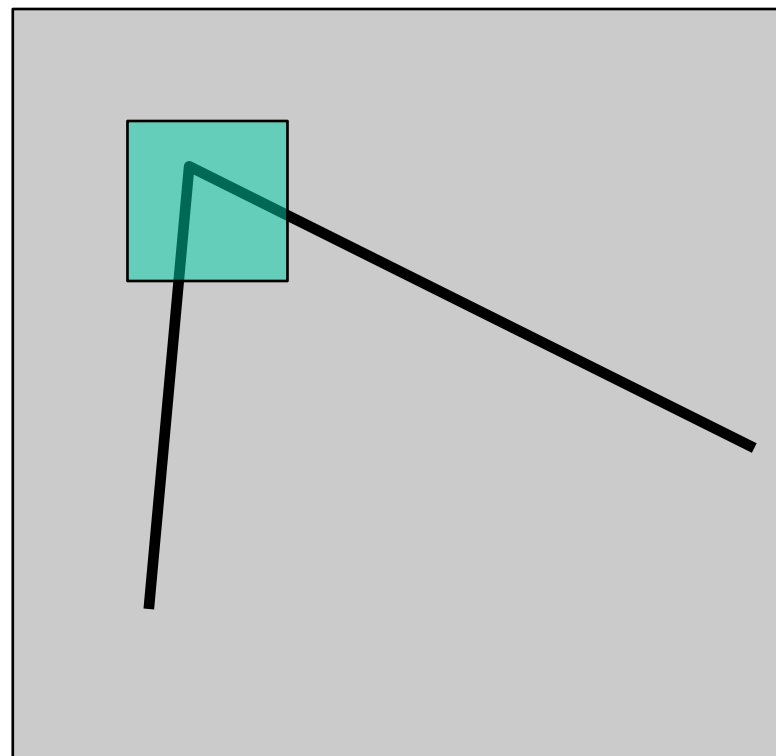
Harris corner detector

Harris corner detector

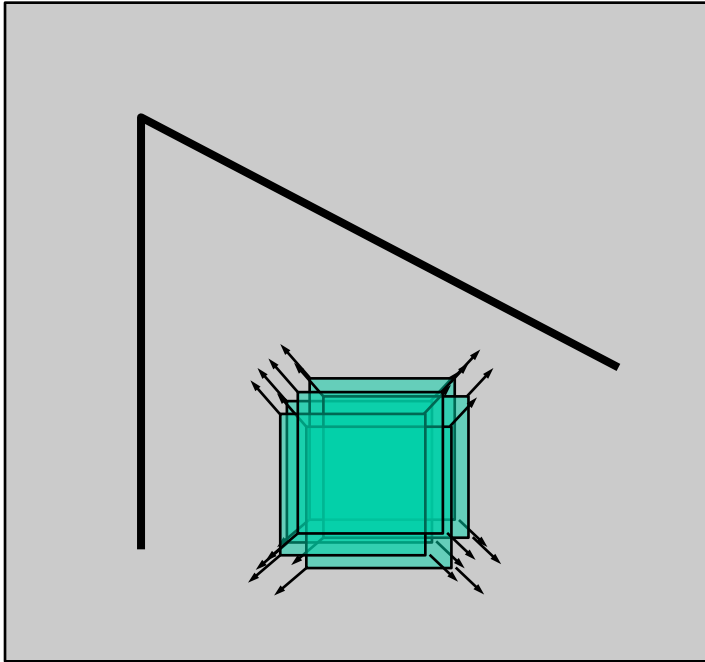


The Basic Idea

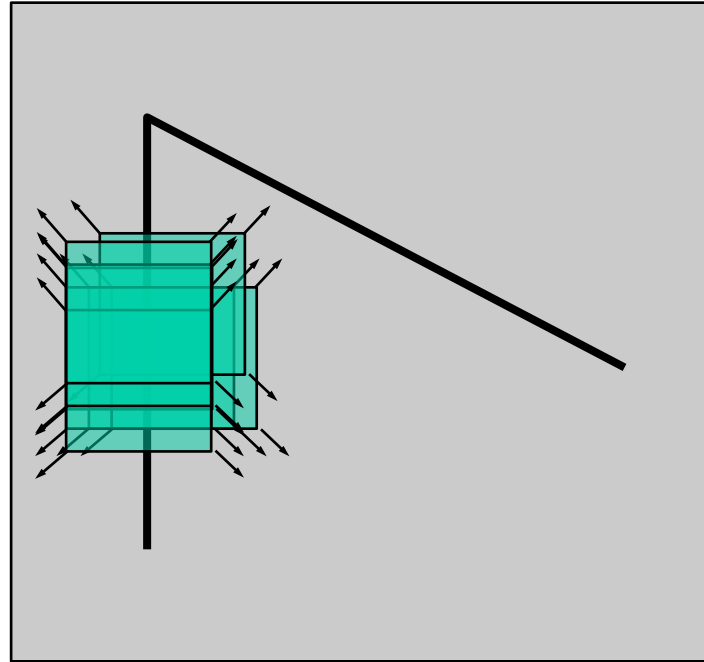
- We should easily localize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in pixels intensities in window
 - makes location precisely define



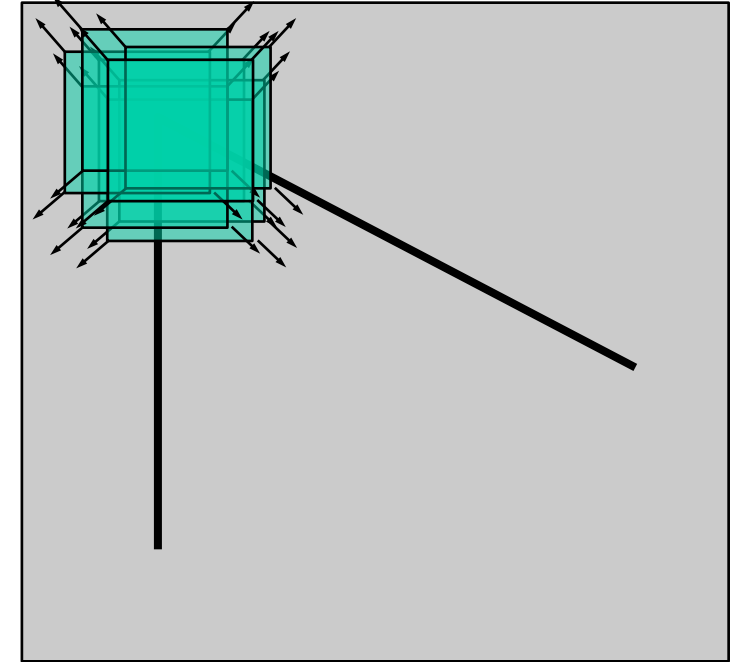
Corner Detector: Basic Idea



“flat” region:
no change in
all directions



“edge”:
no change along
the edge direction



“corner”:
significant change
in all directions

Harris Detector: Mathematics

Window-averaged change of intensity induced
by shifting the image data by $[u, v]$:

Harris Detector: Mathematics

Window-averaged change of intensity induced
by shifting the image data by $[-u, -v]$:

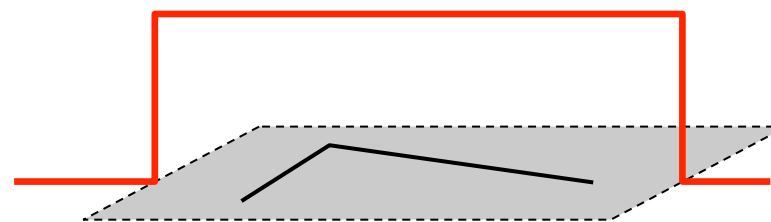
$$E(u, v) = \sum_{x, y} w(x, y) [I(x + u, y + v) - I(x, y)]^2$$

Window
function

Shifted
intensity

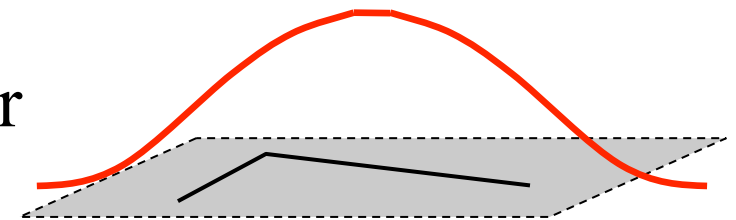
Intensity

Window function $w(x, y) =$



1 in window, 0 outside

or



Gaussian

Practical corner detection

- Computing the error for all neighbors is costly
- Let's derive a more direct criterion
- Taylor expansions to the rescue!
 - derive changes in error in neighborhood of pixel

Taylor expansion

Taylor expansion

- 1D:

$$I(x + u) = I(x) + uI'(x) + O(u^2)$$

- 2D:

$$I(x + u, y + v) = I(x, y) + \begin{pmatrix} u \\ v \end{pmatrix} \cdot \nabla I(x) + O(u^2 + v^2)$$

Taylor series approximation

Recall full equation:

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

Taylor series approximation to shifted image gives quadratic form for error as function of image shifts.

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

$$I(x + u, y + v) \approx I(x, y) + \begin{pmatrix} u \\ v \end{pmatrix} \cdot \begin{pmatrix} I_x \\ I_v \end{pmatrix}$$

where subscripts denote derivatives

Note: derivatives computed at each x, y

Taylor series approximation to shifted image gives quadratic form for error as function of image shifts.

$$\begin{aligned} E(u, v) &\approx \sum_{x,y} w(x, y) [I(x, y) + uI_x + vI_y - I(x, y)]^2 \\ &= \sum_{x,y} w(x, y) [uI_x + vI_y]^2 \\ &= (u \quad v) \sum_{x,y} w(x, y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} \end{aligned}$$

where subscripts denote derivatives

Note: derivatives computed at each x,y

Harris Detector: Mathematics

Expanding $I(x,y)$ in a Taylor series expansion, we have, for small shifts $[u,v]$, a *quadratic* approximation to the error surface between a patch and itself, shifted by $[u,v]$:

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

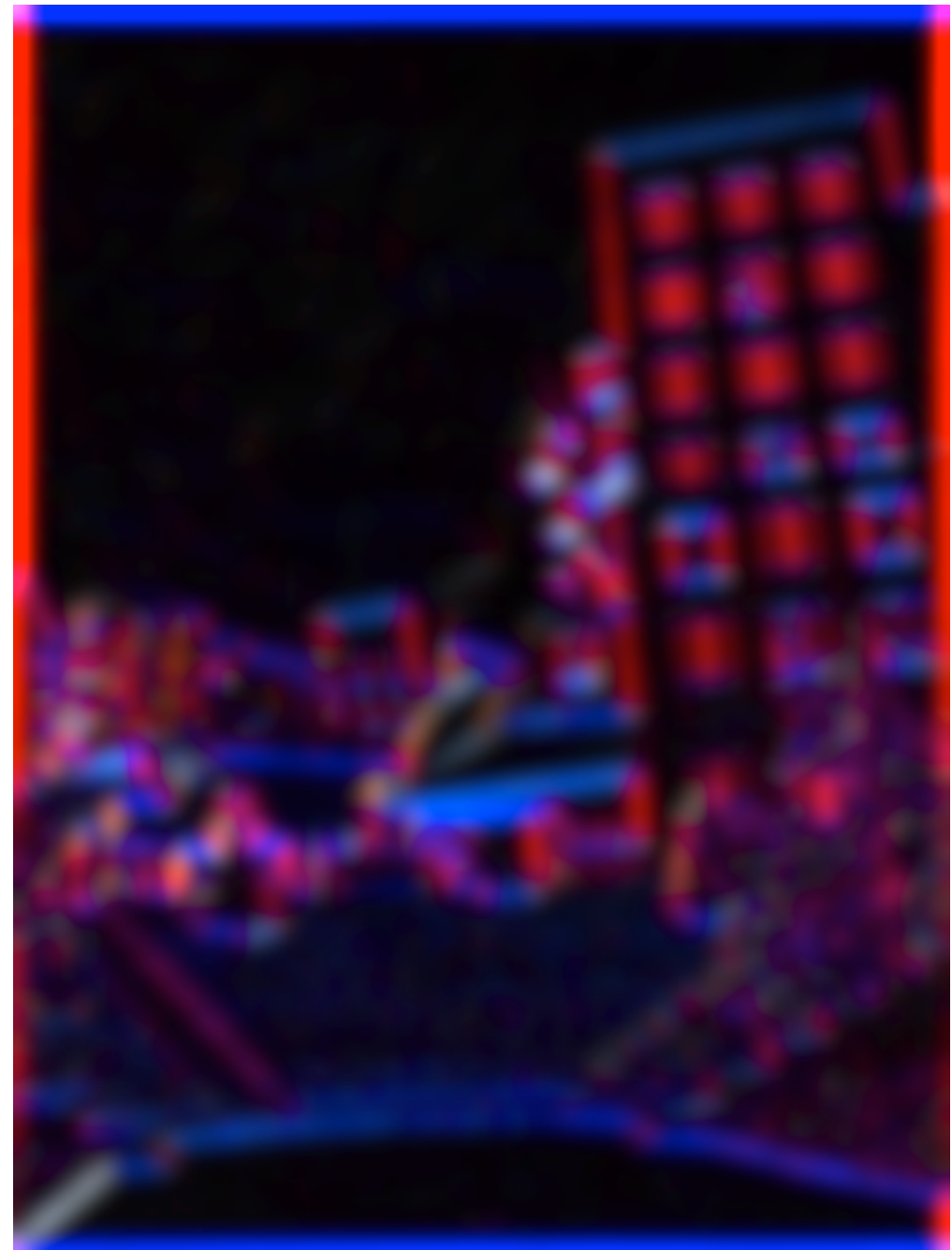
where M is a 2×2 matrix computed from image derivatives:

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

M is often called structure tensor

Structure tensor: Stata

- Blue: xx , red: yy , green: xy



Harris Detector: Mathematics

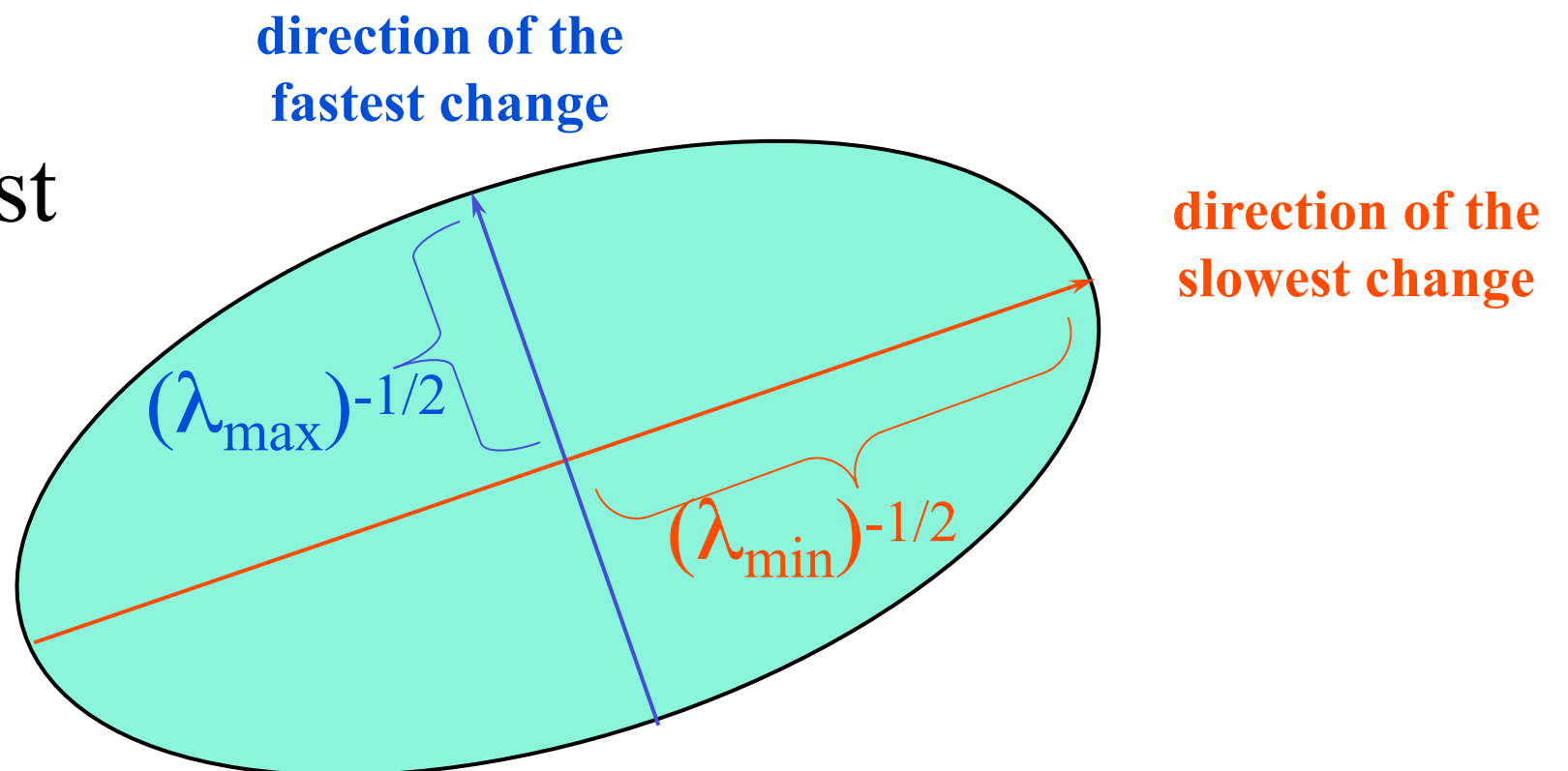
Intensity change in shifting window: eigenvalue analysis

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

λ_1, λ_2 – eigenvalues of M

Ellipse $E(u, v) = \text{const}$

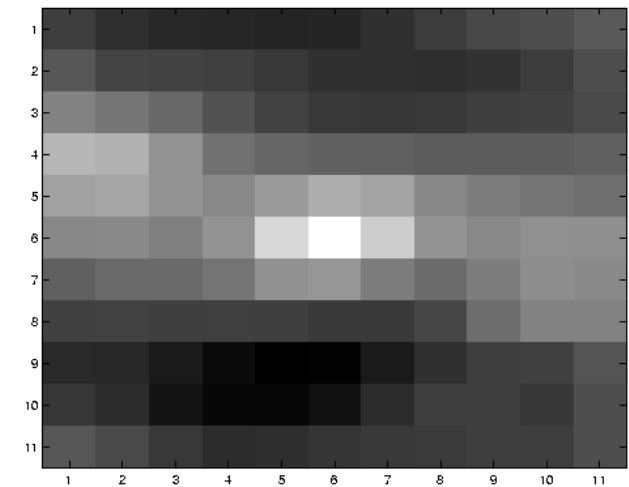
Iso-intensity contour of $E(u, v)$



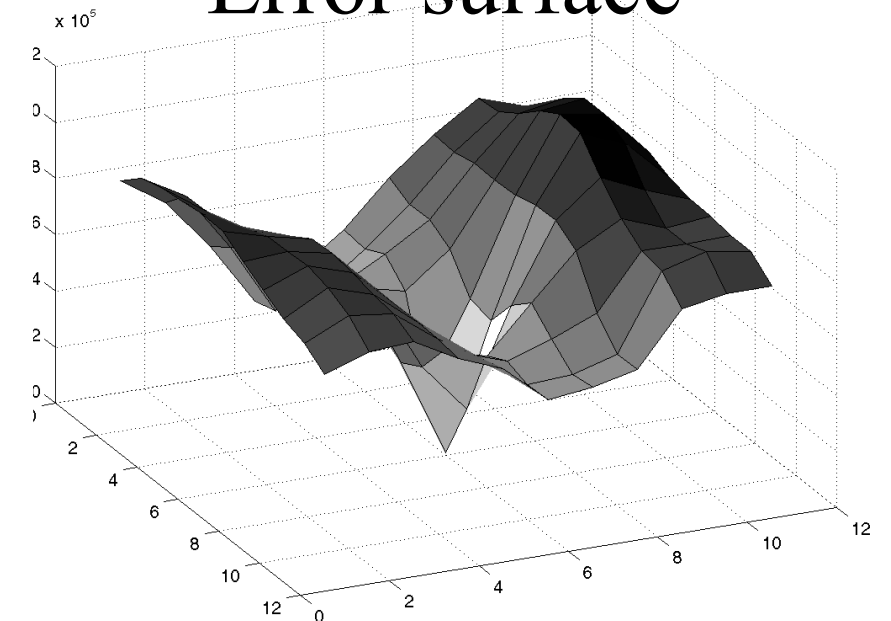
Selecting Good Features



Image patch



Error surface

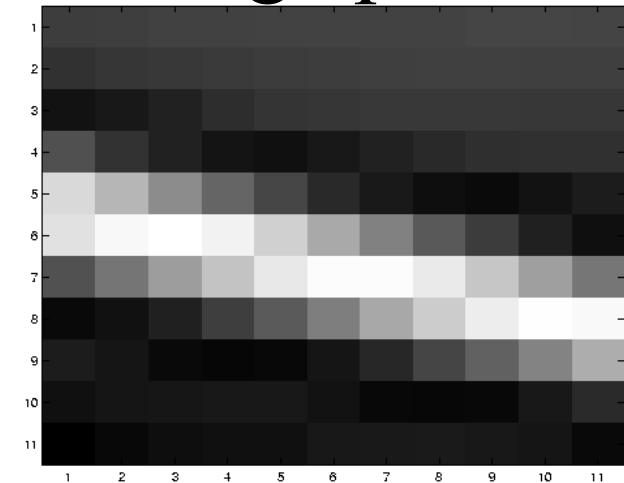


λ_1 and λ_2 are large

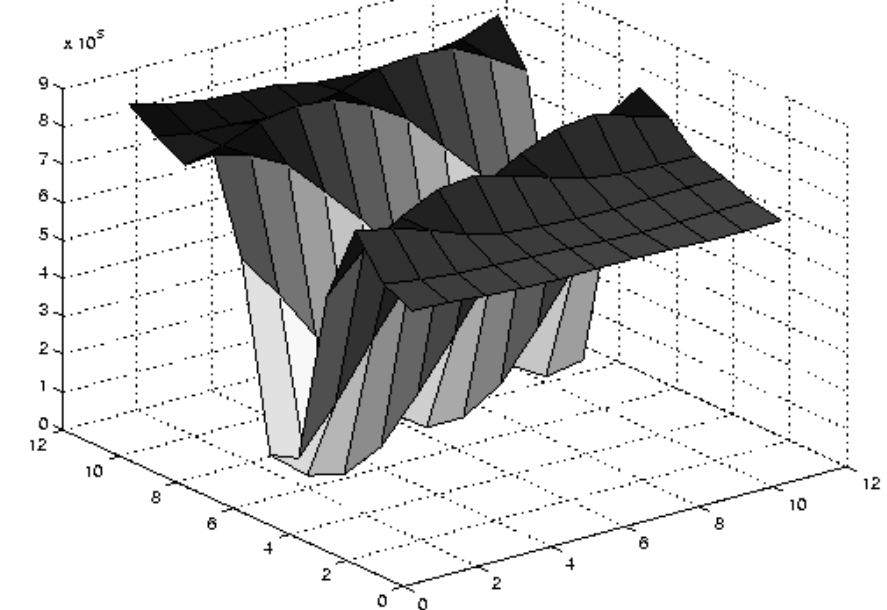
Selecting Good Features



Image patch



Error surface

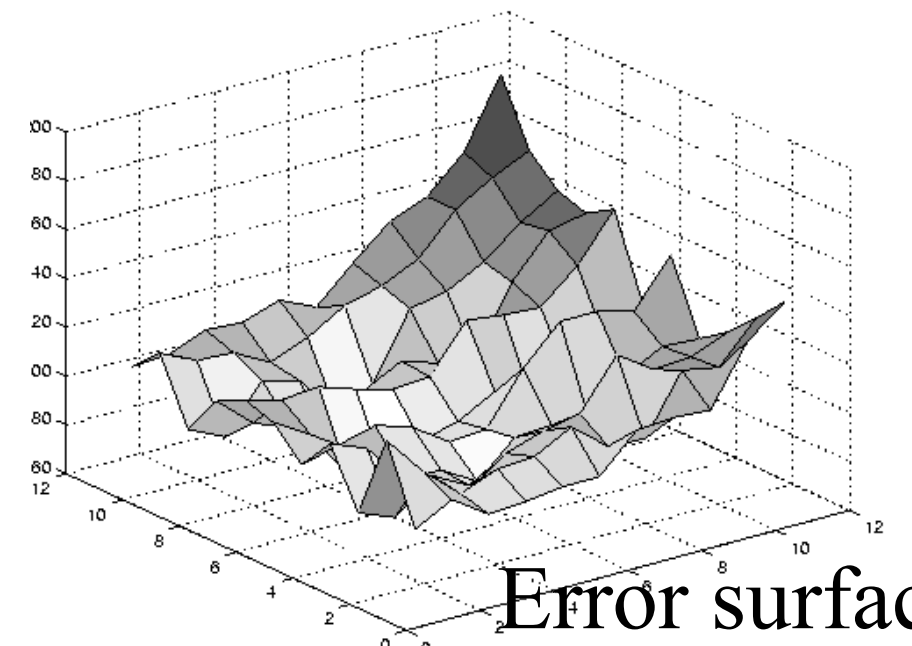
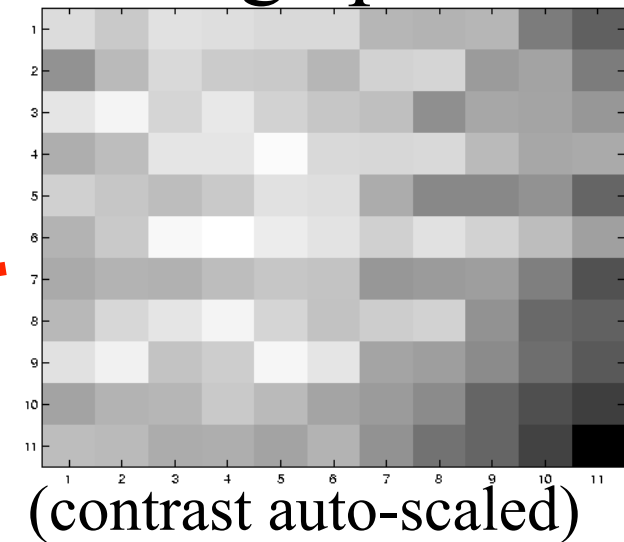


large λ_1 , small λ_2

Selecting Good Features



Image patch



Error surface

(vertical scale exaggerated relative to previous plots)

small λ_1 , small λ_2

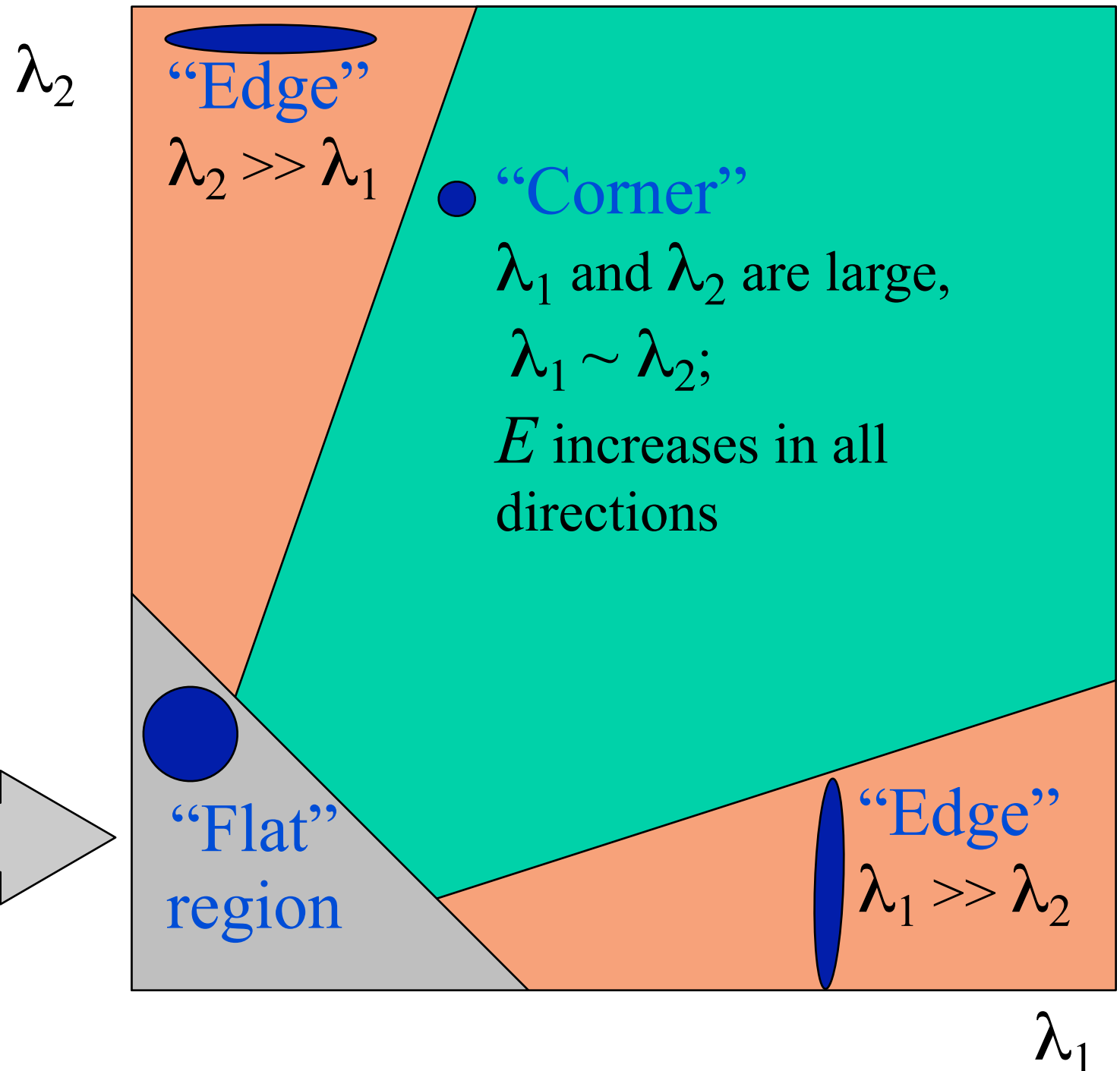
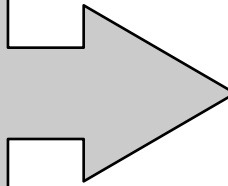
Harris Detector: Mathematics

Classification of
image points using
eigenvalues of M :

Harris Detector: Mathematics

Classification of image points using eigenvalues of M :

λ_1 and λ_2 are small;
 E is almost constant
in all directions



Harris Detector: Mathematics

Measure of corner response:

Harris Detector: Mathematics

Measure of corner response:

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

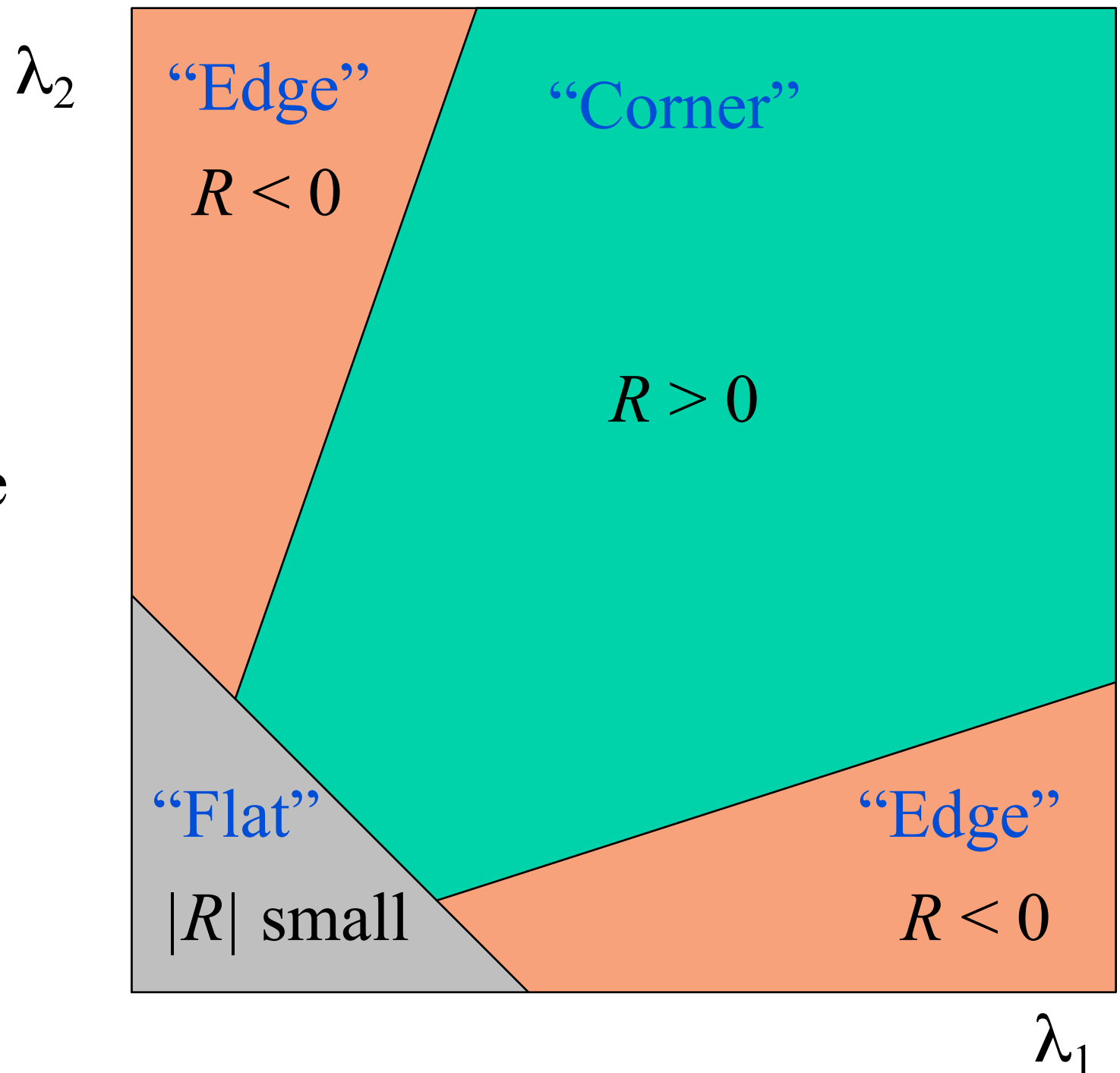
$$\begin{aligned}\det M &= \lambda_1 \lambda_2 \\ \text{trace } M &= \lambda_1 + \lambda_2\end{aligned}$$

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

(Shi-Tomasi variation: use $\min(\lambda_1, \lambda_2)$ instead of R)

Harris Detector: Mathematics

- R depends only on eigenvalues of M
- R is large for a **corner**
- R is negative with large magnitude for an **edge**
- $|R|$ is small for a **flat** region



Harris corner detector algorithm

- Compute image gradients I_x I_y for all pixels
- For each pixel

– Compute

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

by looping over neighbors x,y

– compute

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

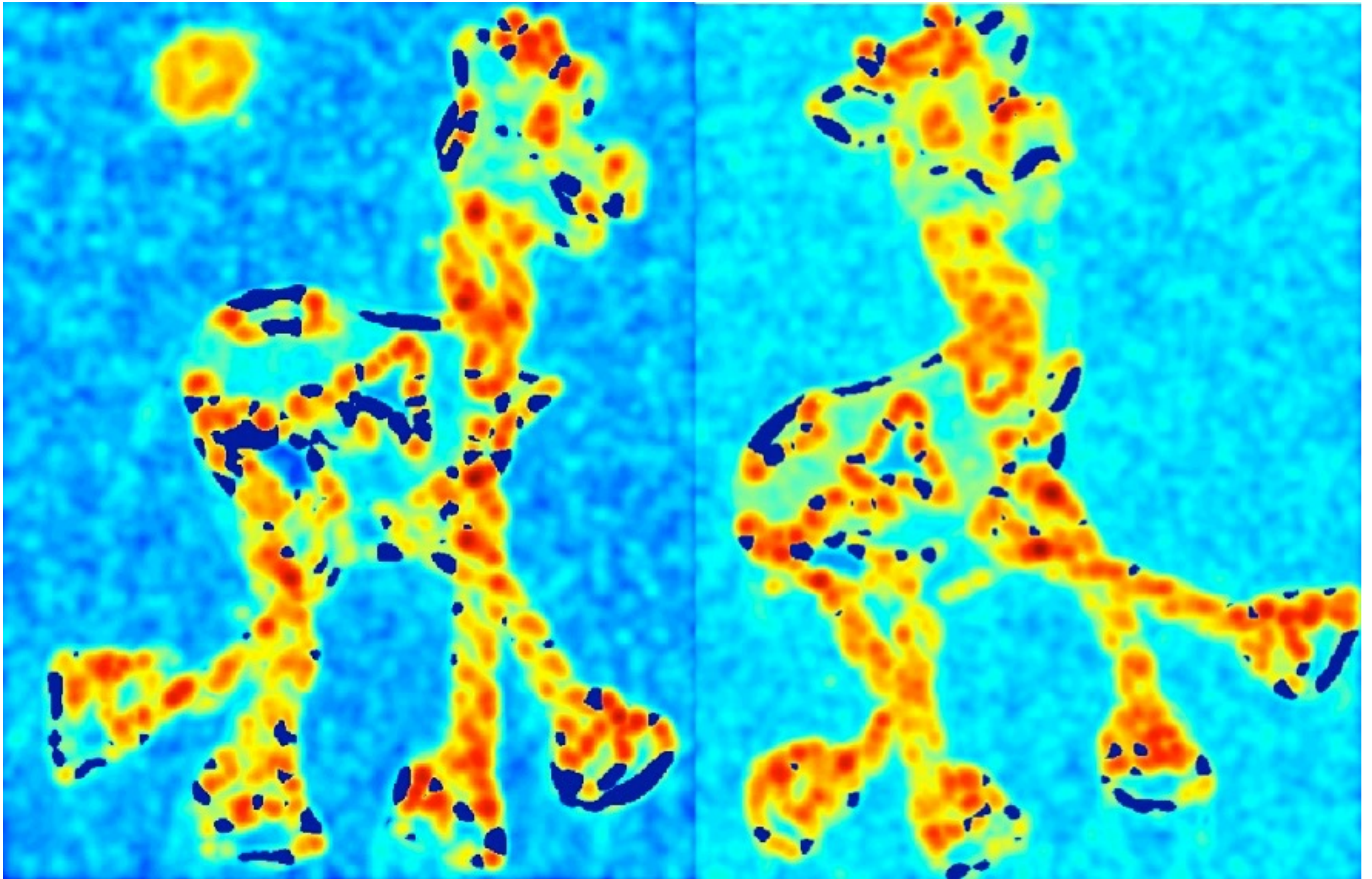
- Find points with large corner response function R ($R > \text{threshold}$)
- Take the points of locally maximum R as the detected feature points
(ie, pixels where R is bigger than for all the $k \times k$ neighbors).

Harris Detector: Workflow



Harris Detector: Workflow

Compute corner response R



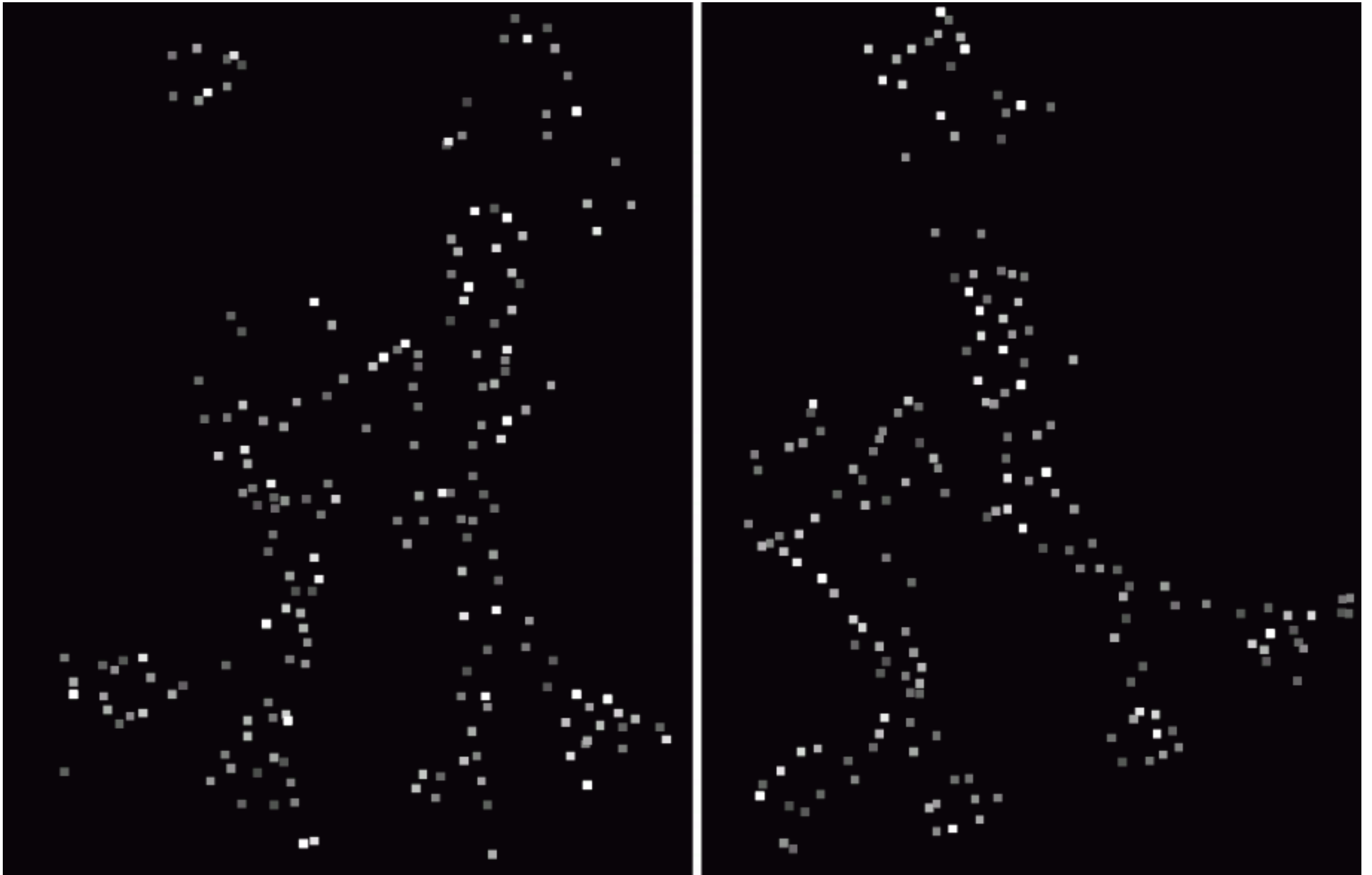
Harris Detector: Workflow

Find points with large corner response: $R > \text{threshold}$

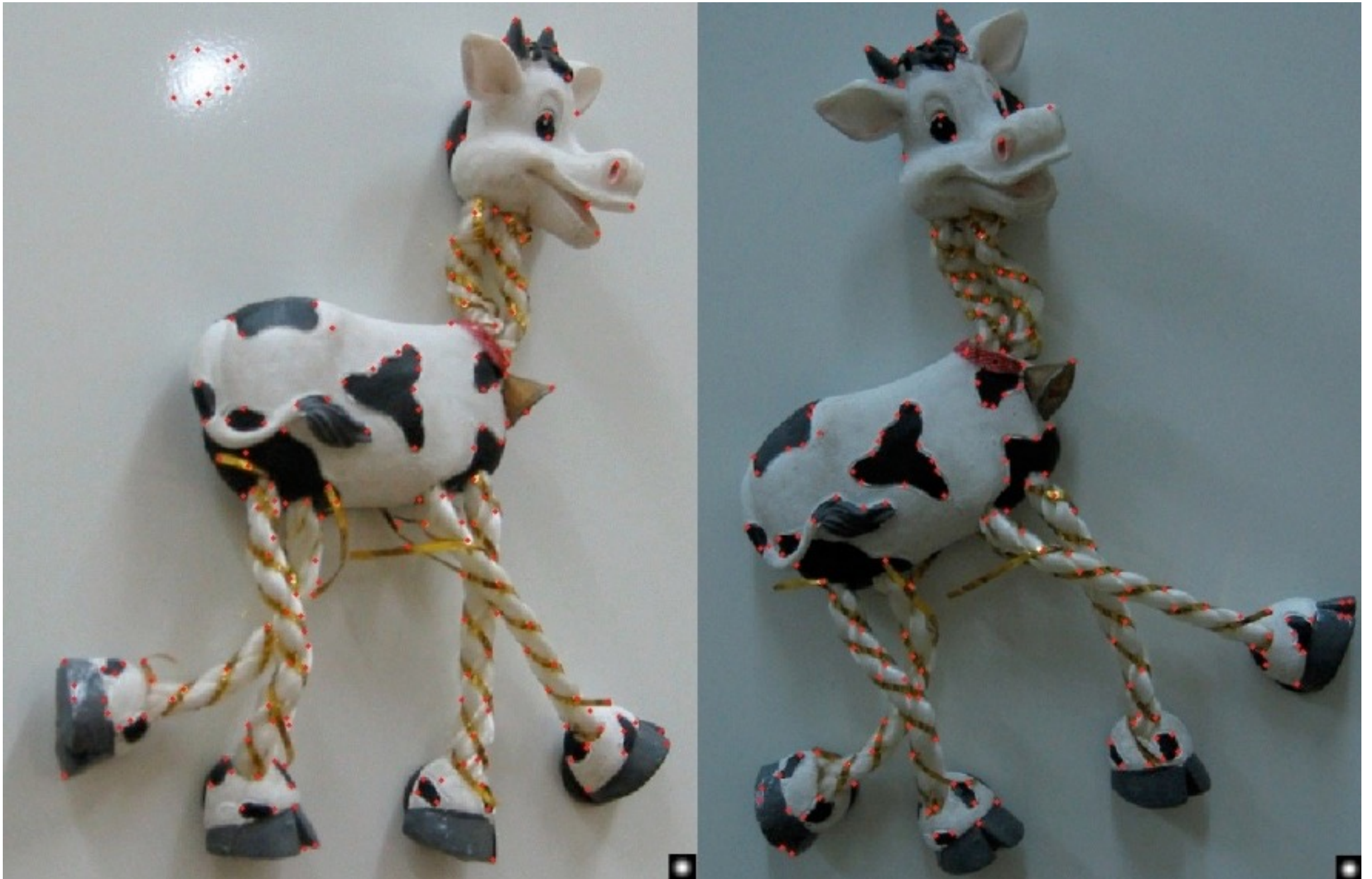


Harris Detector: Workflow

Take only the points of local maxima of R

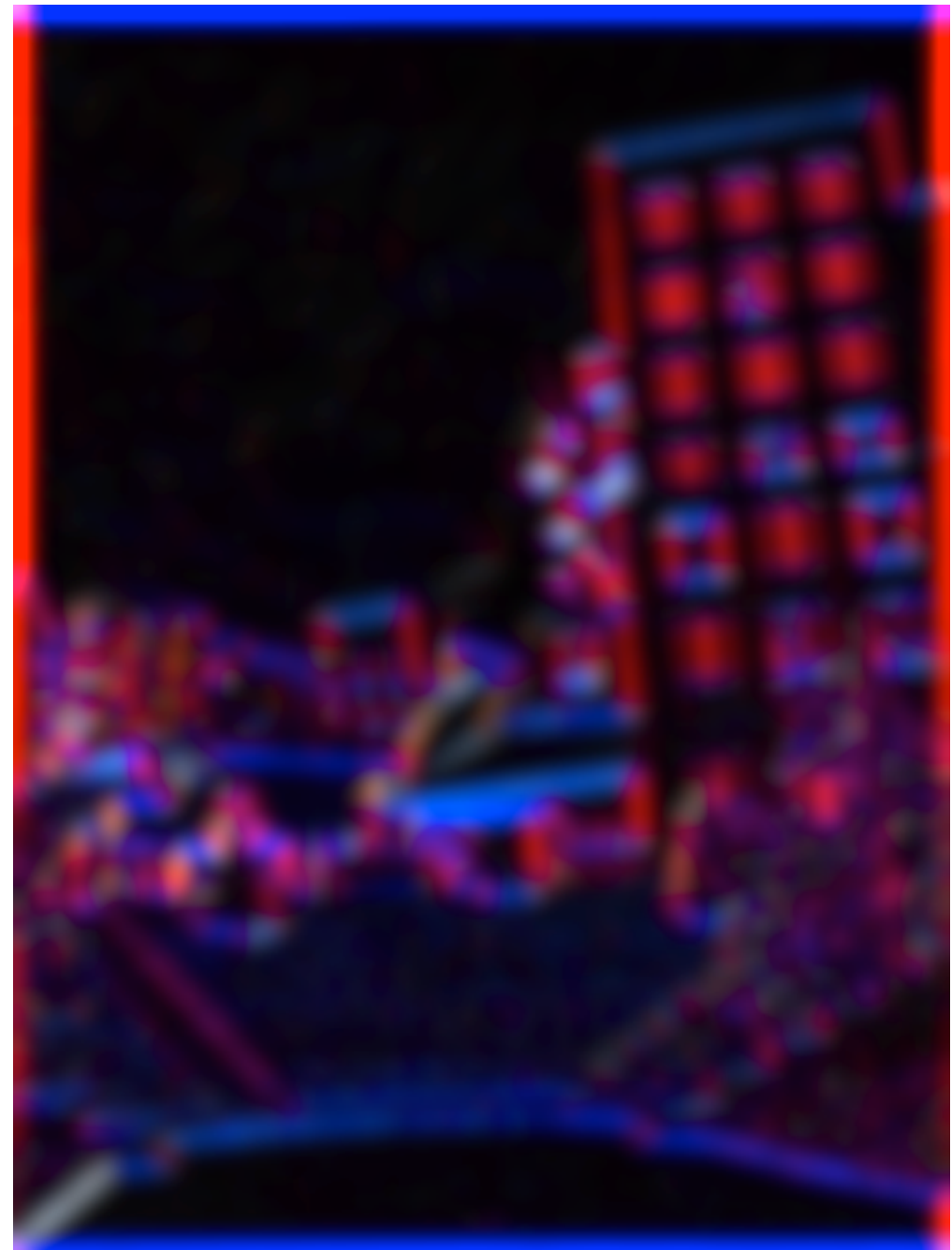


Harris Detector: Workflow

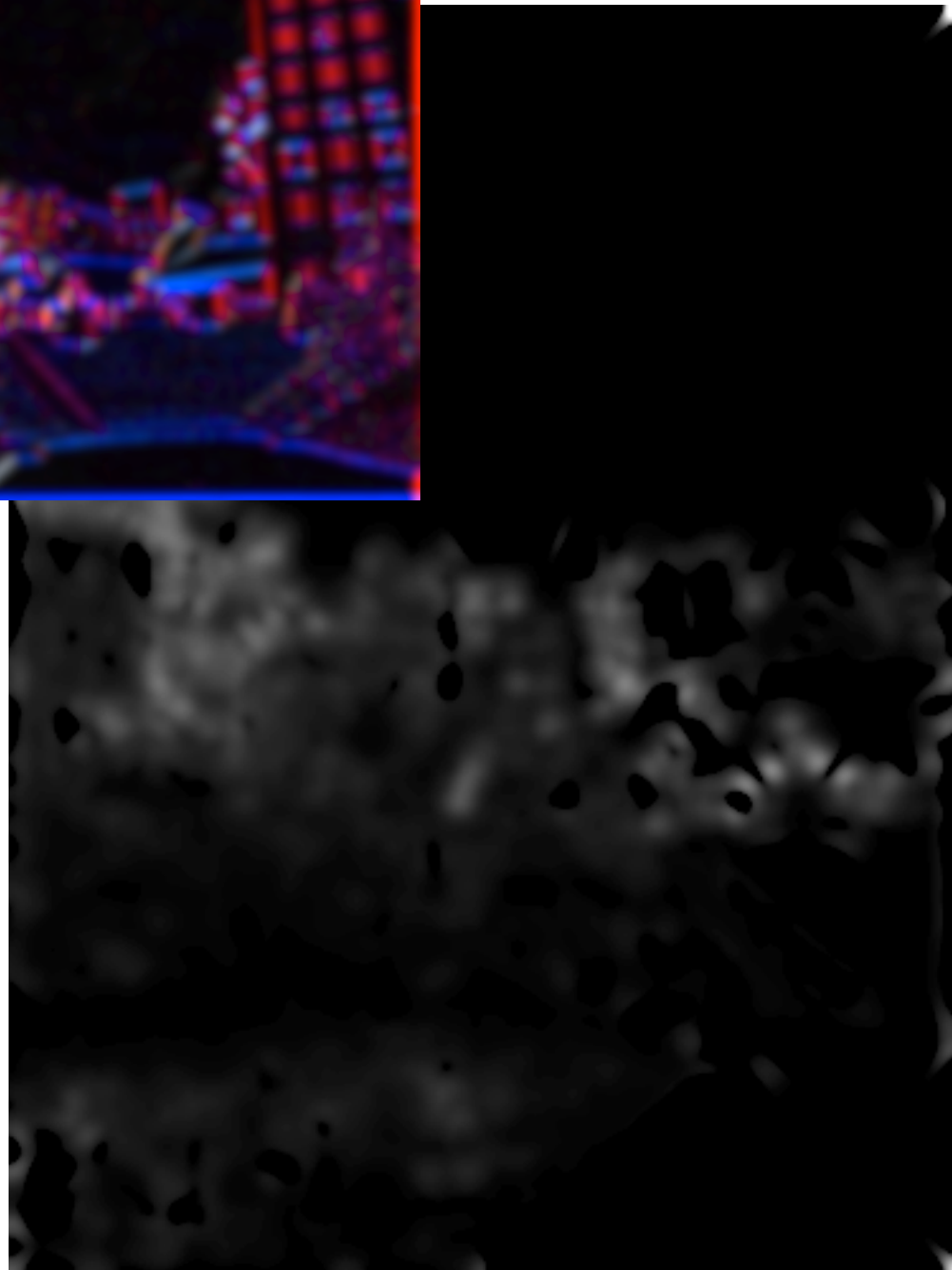
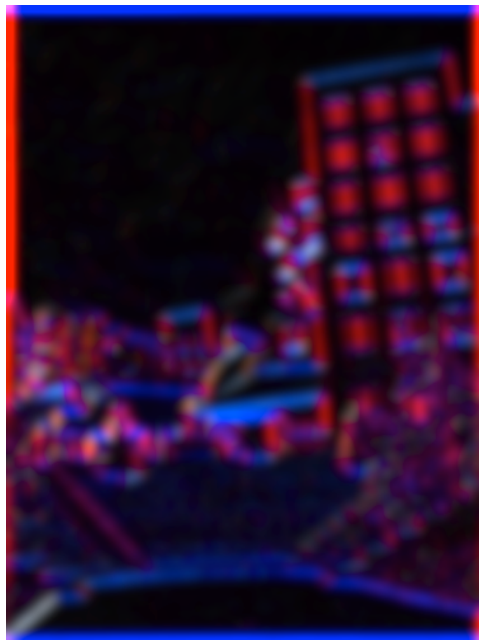


Structure tensor: Stata

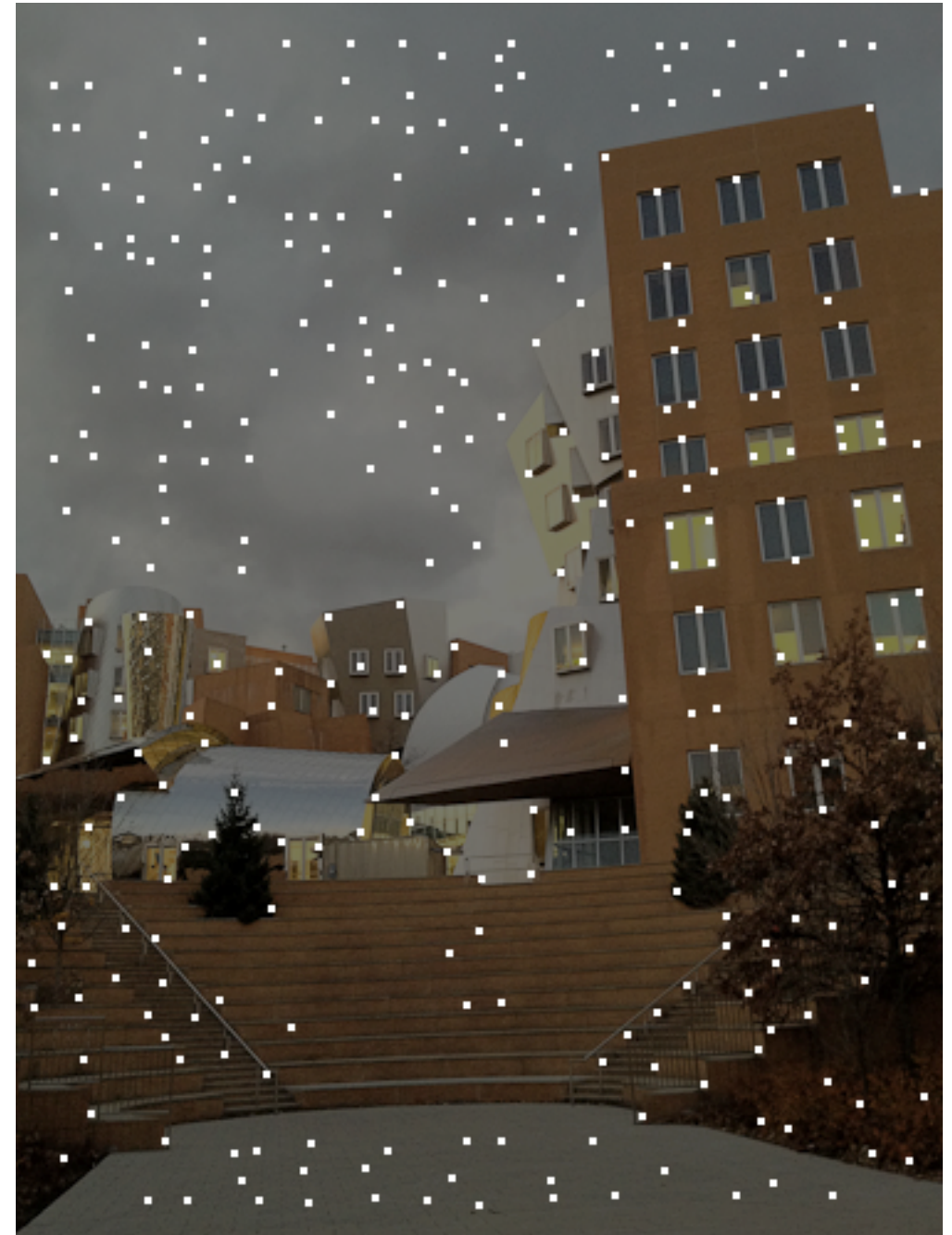
- Blue: xx , red: yy , green: xy



Corner response



Corners



Harris Detector: Summary

- Average intensity change in direction $[u, v]$ can be expressed as a bilinear form:

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

- Describe a point in terms of eigenvalues of M :
measure of corner response

$$R = \lambda_1 \lambda_2 - k (\lambda_1 + \lambda_2)^2$$

- A good (corner) point should have a *large intensity change in all directions*, i.e. R should be large and positive

Correspondences

Recall: Matching with Features

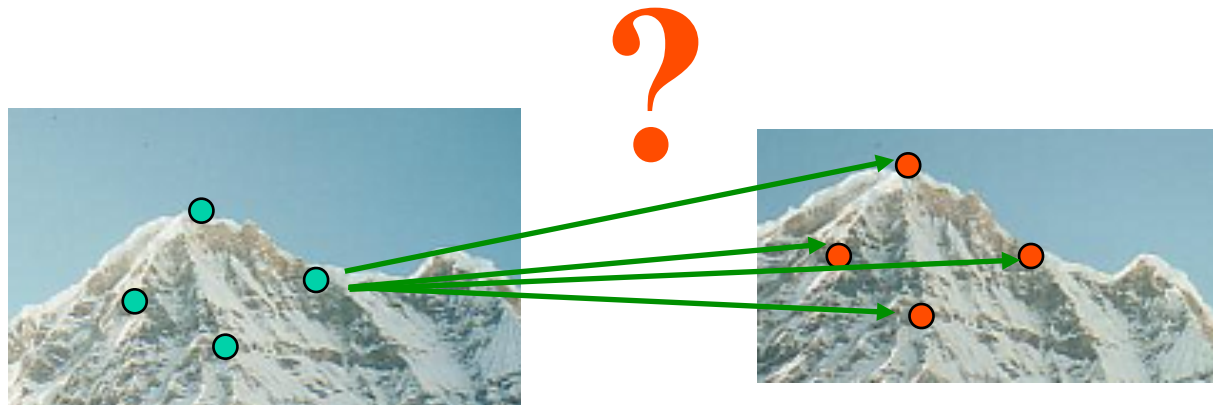
- Problem 1:
 - Detect the *same* point *independently* in both images



We need a repeatable detector:
DONE FOR NOW

Recall: Matching with Features

- Problem 2:
 - For each point correctly recognize the corresponding one



We need a reliable and distinctive descriptor

Using descriptors

- For each interest point (corner) in first image
- Find most similar in second image
- Similarity defined by descriptor difference
- Goal: descriptor should stay the same around a scene feature when we rotate the camera and take the second image

Questions?



Patch descriptor



In this lecture

In this lecture

- Simple patch descriptor
 - Describe a corner by the local pixel values
 - Blur a little to reduce effect of sampling
 - Correct for brightness and contrast differences
- Later: more advanced descriptors

Basic idea

- Descriptor = $k \times k$ patch

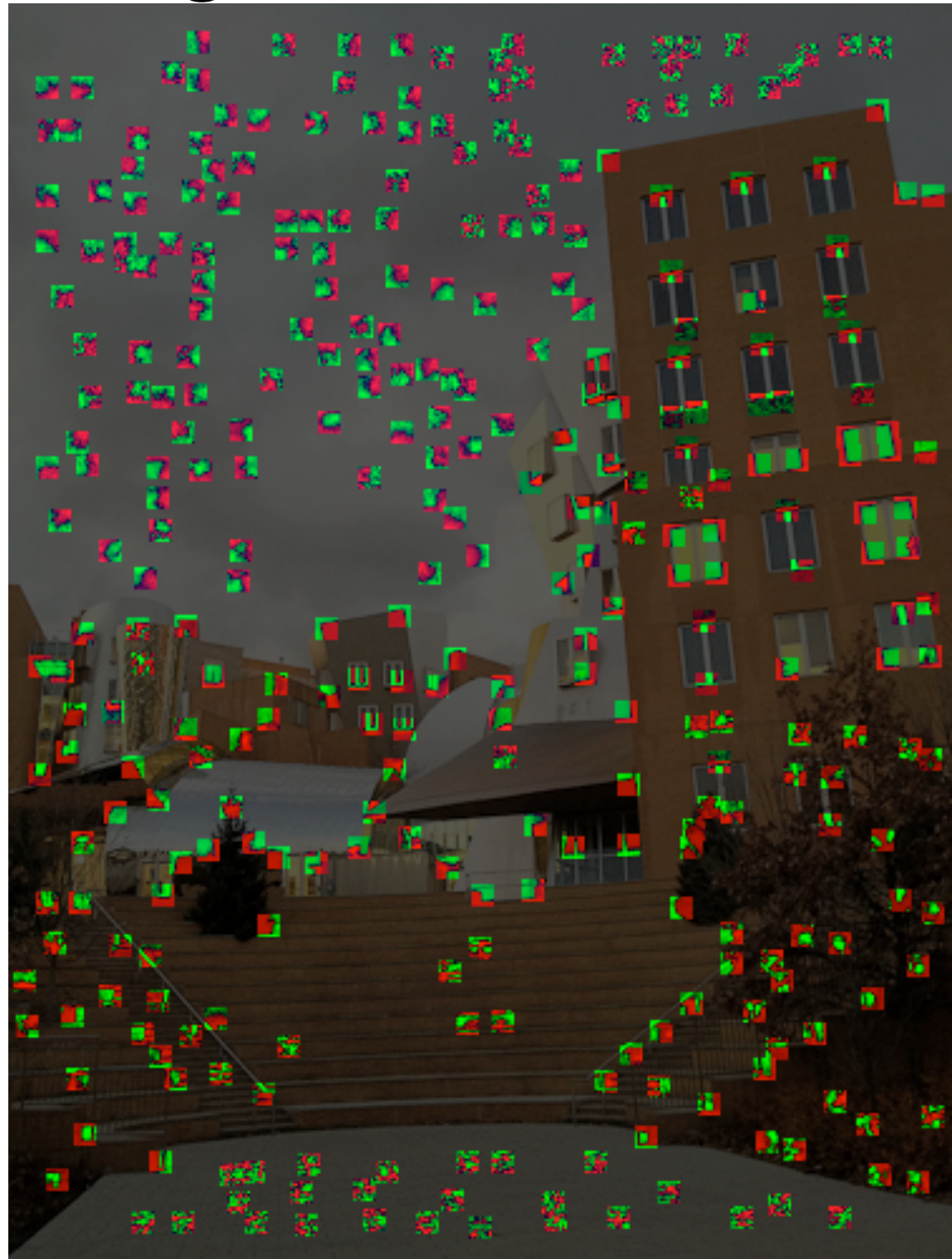
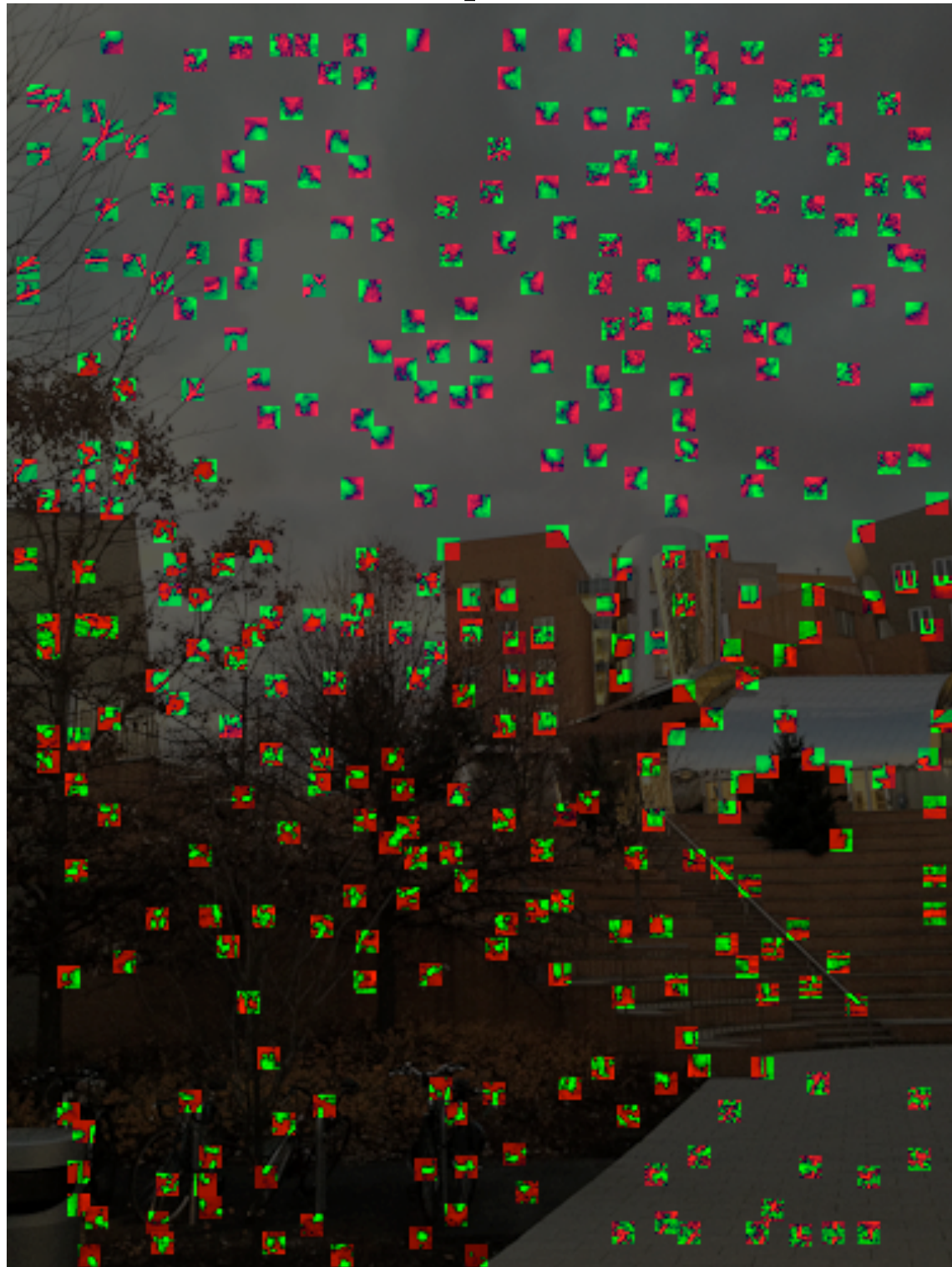


Bells and whistles

- In pset 7, we only use luminance values
- Blur image a little before
 - To reduce sampling/aliasing effects
- Correct for potential brightness/contrast changes
 - e.g. because of exposure, vignetting, clouds
 - Subtract mean
 - normalize by standard deviation
- At the end of the day, a descriptor is $k \times k$ numbers

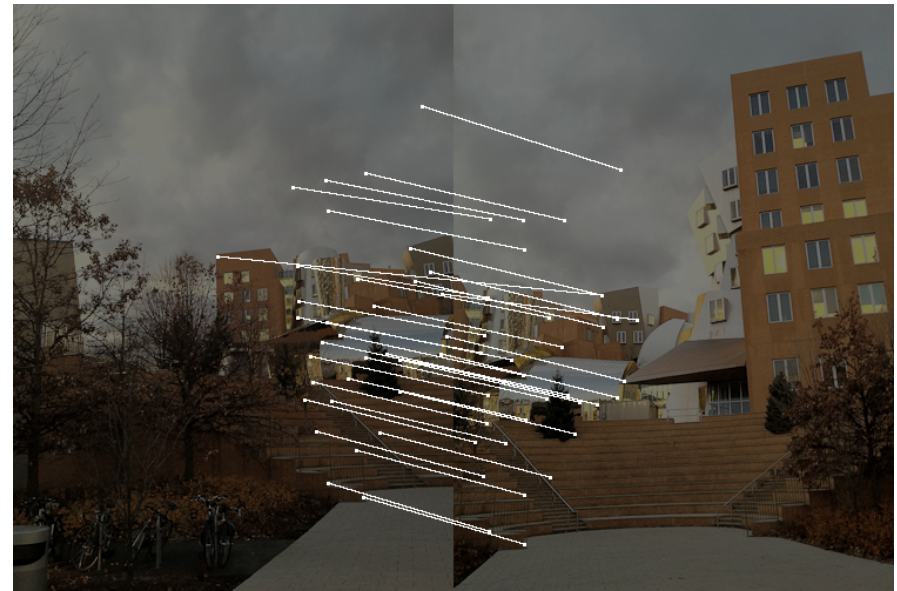
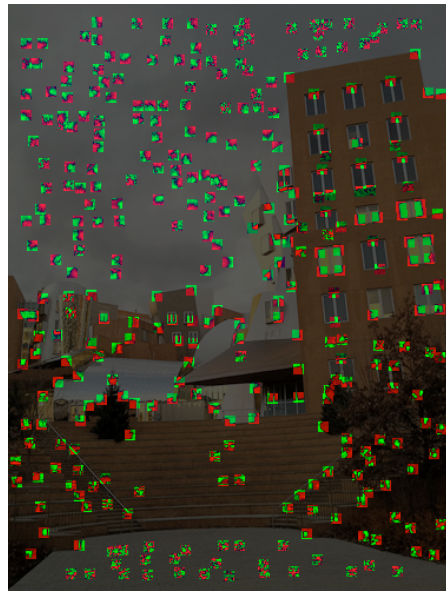
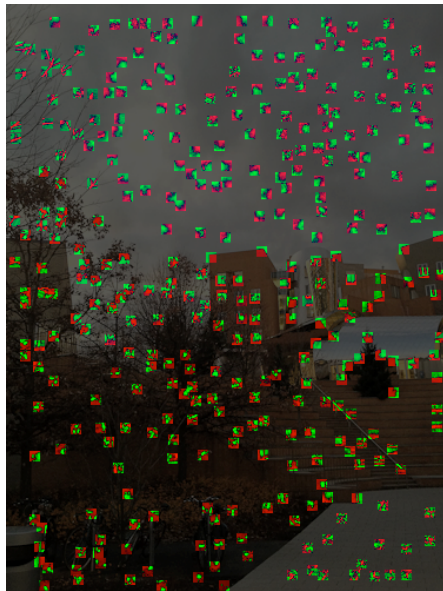
Descriptor Visualization

- Green=positive, red=negative



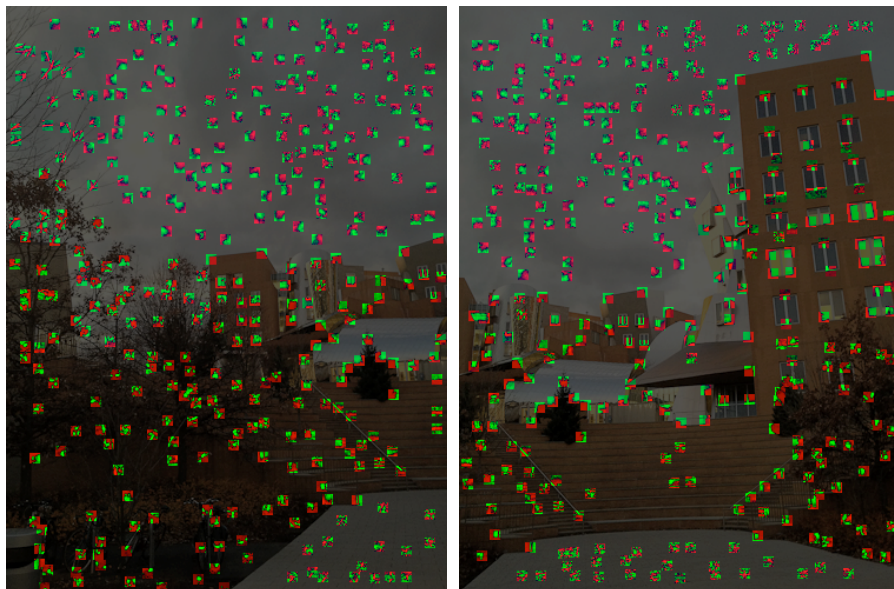
Matching descriptors

Status



Status

- We have extracted N_1 corners from image 1, and N_2 from image 2
- For each corner, we have a $k \times k$ descriptor
- The combination of a corner+descriptor is often called a feature point
- Now we need to match feature points from image 1 to image 2



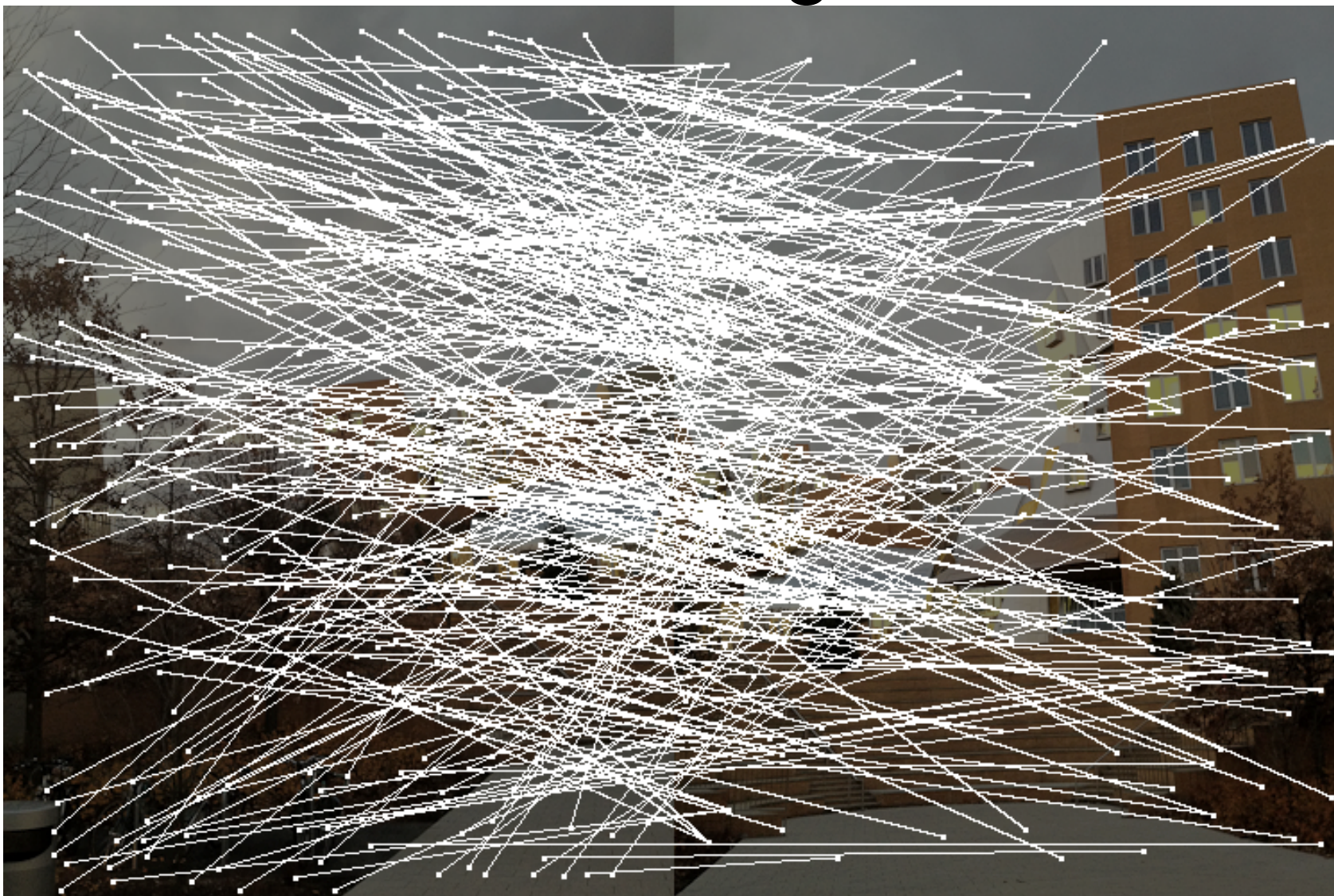
Brute force search

Brute force search

- For each feature point i in image 1
 - scan all feature points j in image 2
 - compute (squared) distance between descriptors i & j
 - i.e. sum of square difference for $k \times k$ numbers
 - Keep descriptor with closest distance

Status

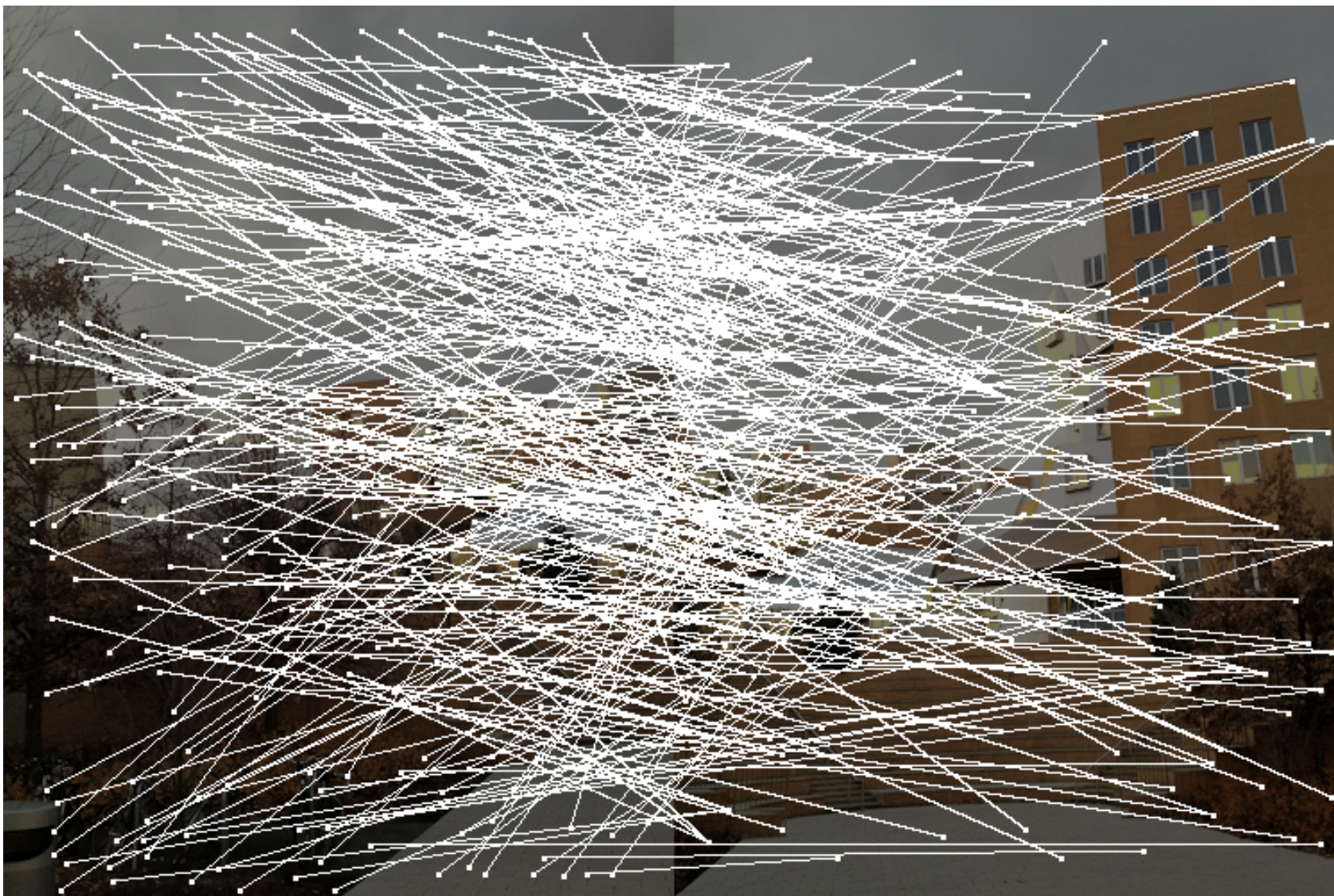
- We have a match for each corner of an image
 - But lots of wrong matches
 - Even scene points that are not on the overlap between the images have a match!



**Second-
Nearest-
Neighbor test**

Problem

- How do we tell good matches from bad ones?

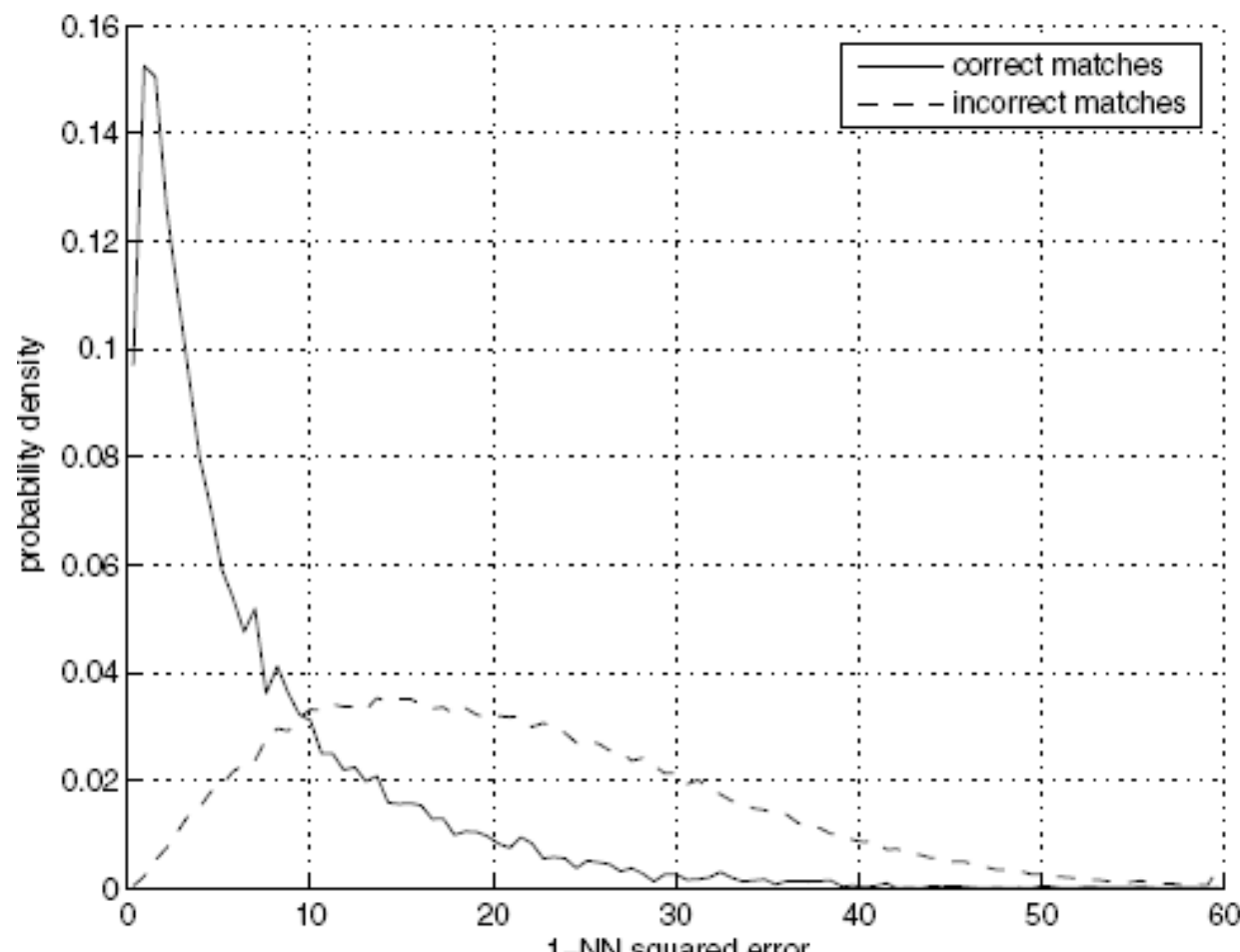


Naive idea

- If distance is too big, ignore match

Evaluating naive idea

- Using ground truth
 - e.g. user clicks to tell us which corner matches which
- distribution of good vs. bad matches as function of descriptor difference
 - slightly different descriptor from us though, smarter)

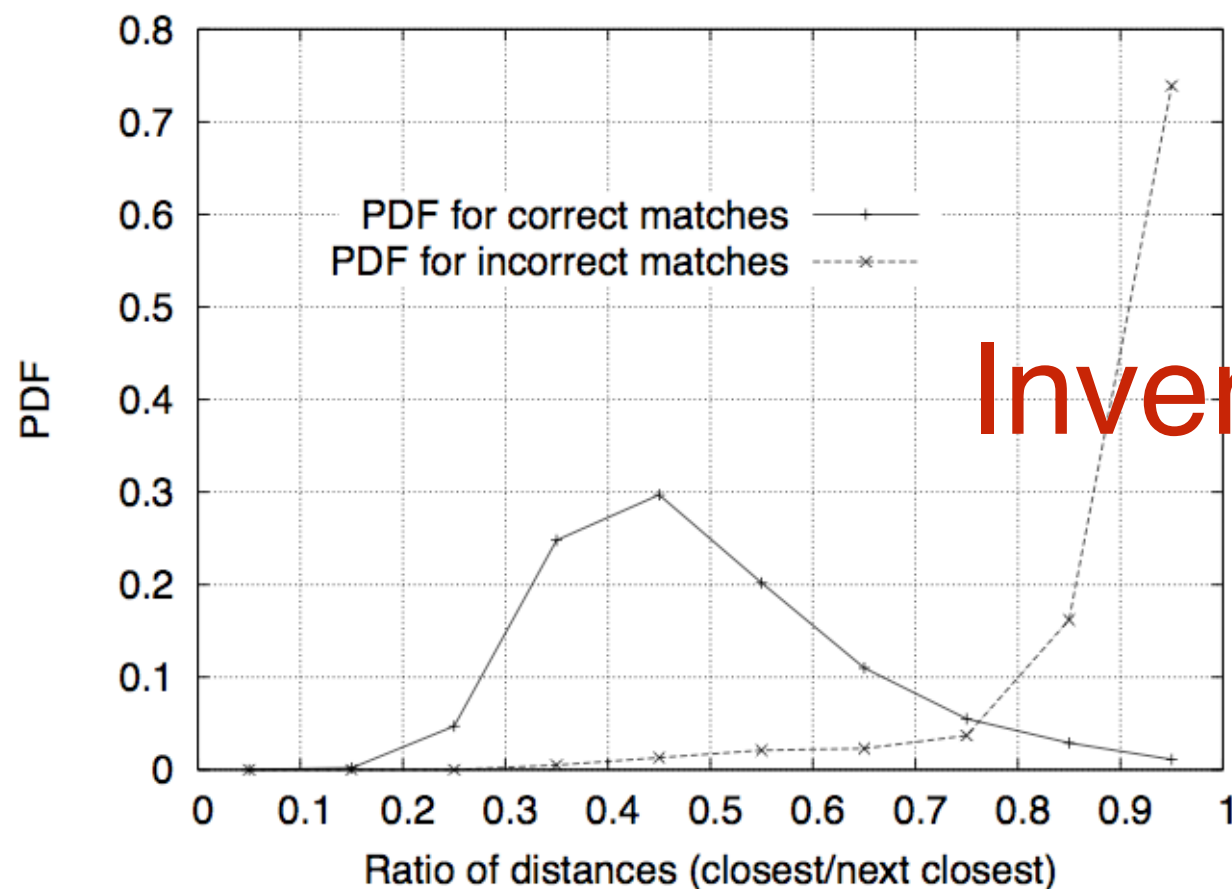


- Not a clear threshold

Better idea: Second NN

Better idea: Second NN

- Evaluate ratio of distance to best descriptor to that of second best
- If they are too similar, the match is ambiguous



Inverse of ratio???

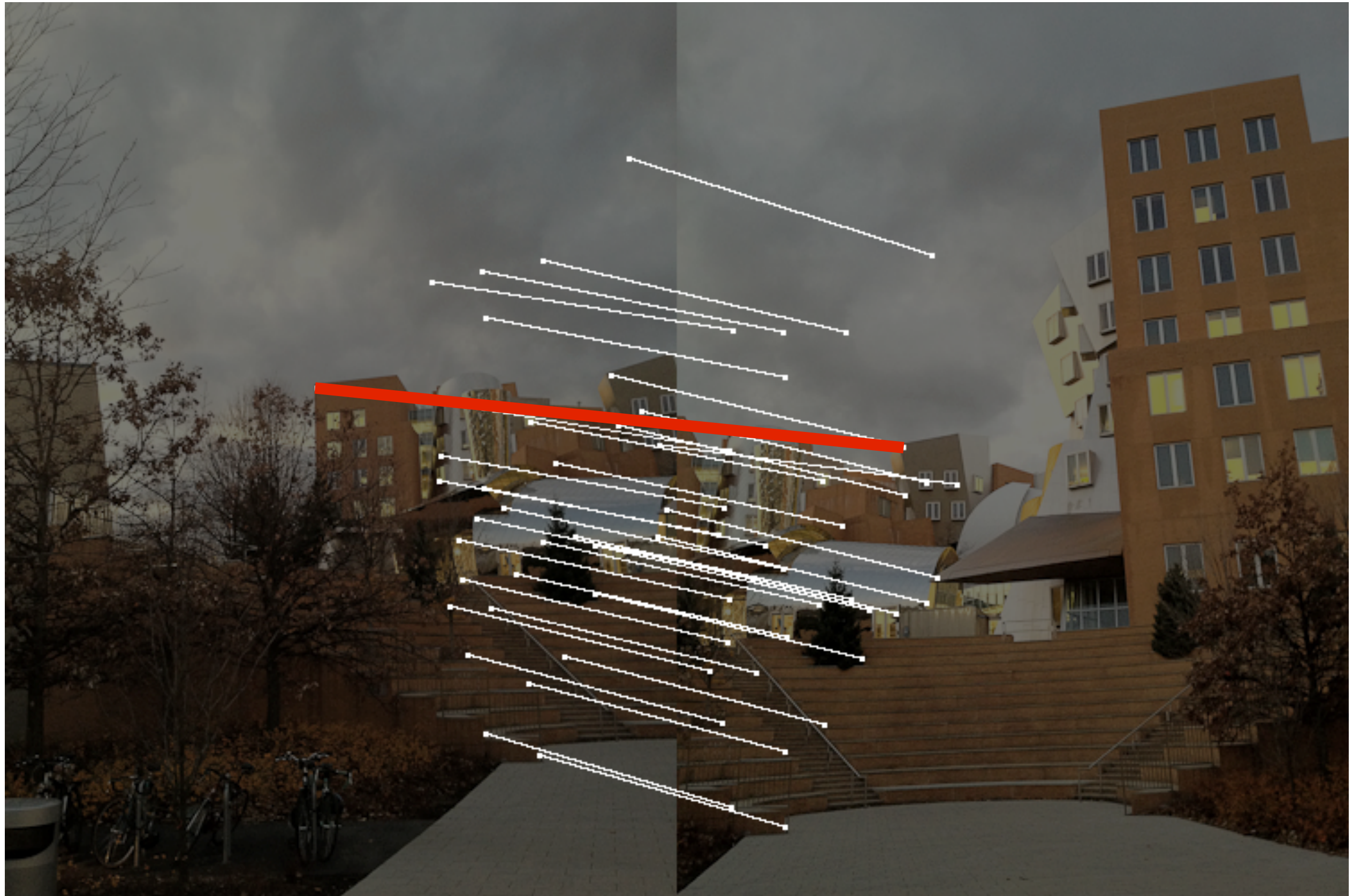
Figure 11: The probability that a match is correct can be determined by taking the ratio of distance from the closest neighbor to the distance of the second closest. Using a database of 40,000 keypoints, the solid line shows the PDF of this ratio for correct matches, while the dotted line is for matches that were incorrect.

Status

- We have feature points
- Matches between two images
 - Most of them are good
 - But not all of them. There can still be error.
 - Next time: RANSAC to remove remaining



Example of outlier



Still to be learned

- Better detector: across scales
- Better descriptor
 - robust to rotation and scale.
 - e.g. SIFT
- Homography fitting despite the presence of outliers
 - RANSAC
- Correct for optical distortions
- Blending to make seams invisible
- Deal with moving objects