Lecture 8

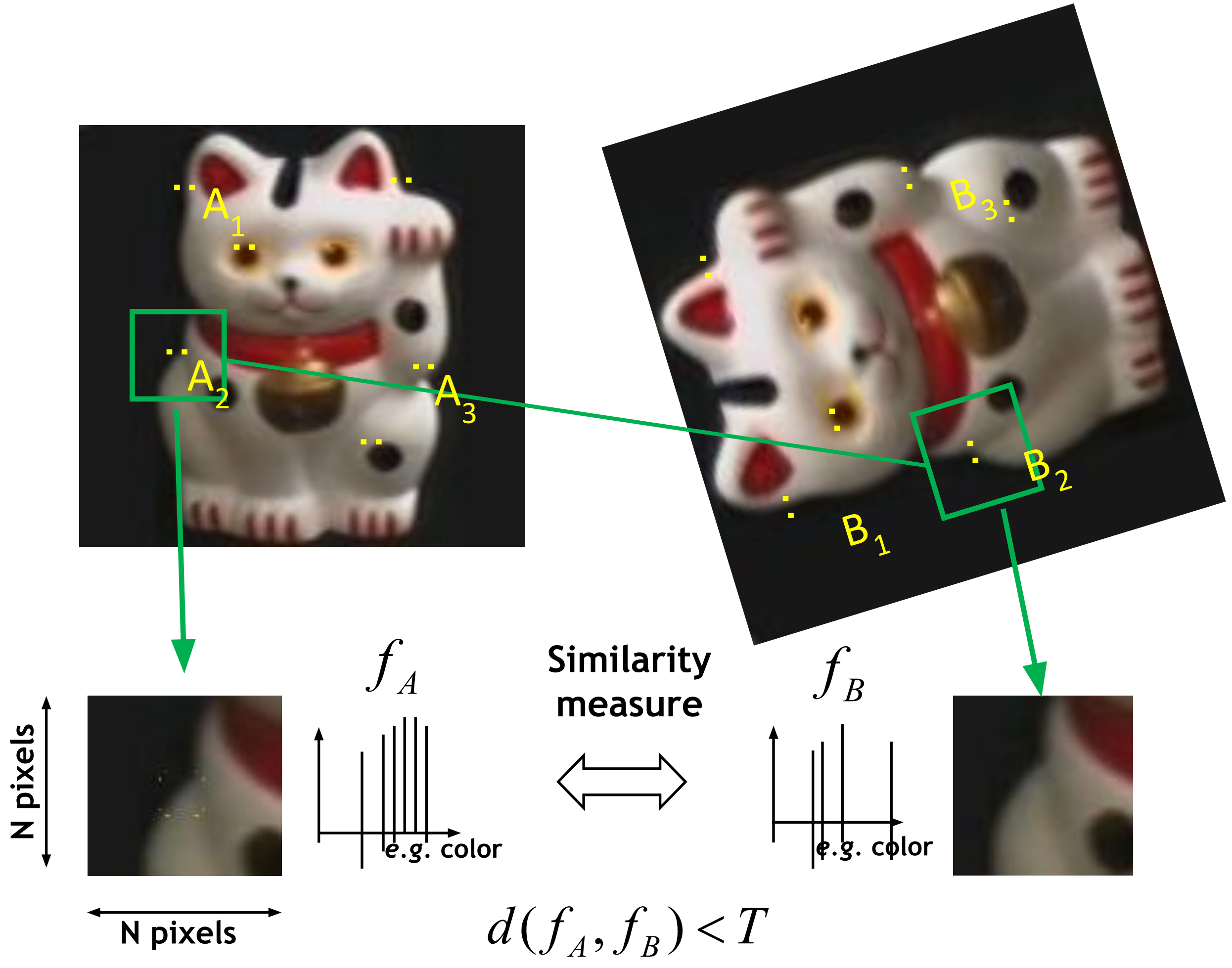
Descriptors & Homographies

Administrative

A2 is out

- Due Feb 7th

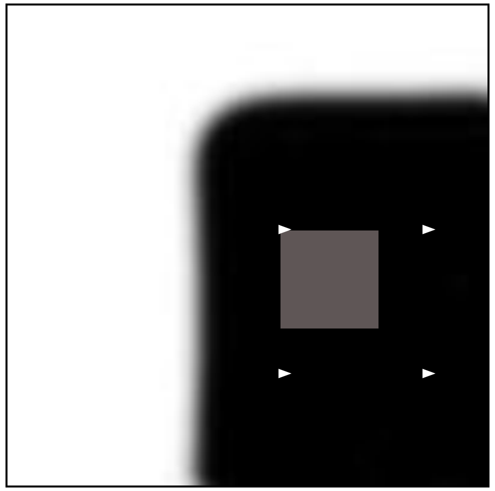
So far: General approach for search

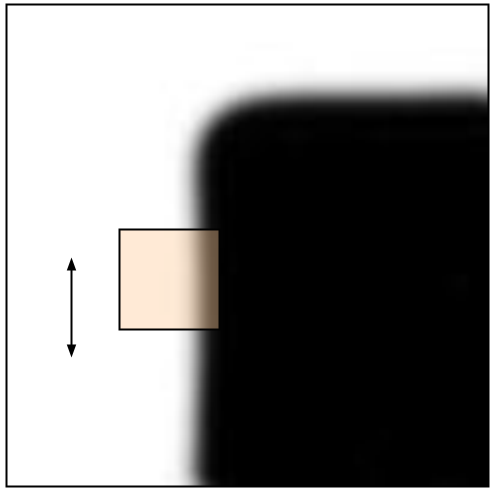


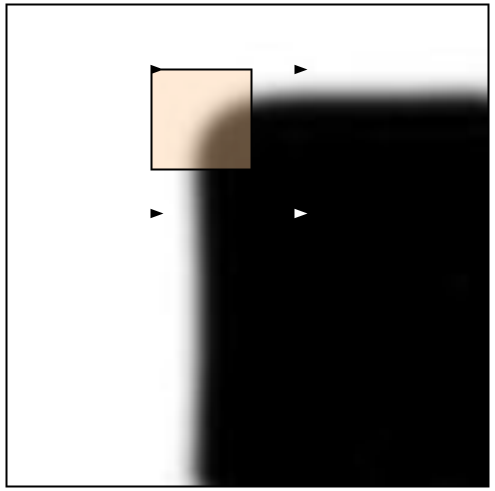
1. Find a set of distinctive key-points
2. Define a region/patch around each keypoint
3. Normalize the region content
4. Compute a local descriptor from the normalized region
5. Match local descriptors

So far: Corners as key-points

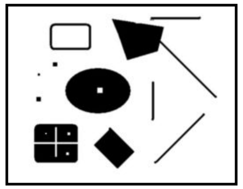
- We should easily recognize the corner point by looking through a small window (*locality*)
- Shifting the window in *any direction* should give *a large change* in intensity (*good localization*)


"flat" region:
no change in
all directions


"edge":
no change along
the edge direction


"corner":
significant change
in all directions

So far: Harris Corner Detector [Harris88]



- Compute second moment matrix (autocorrelation matrix)

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

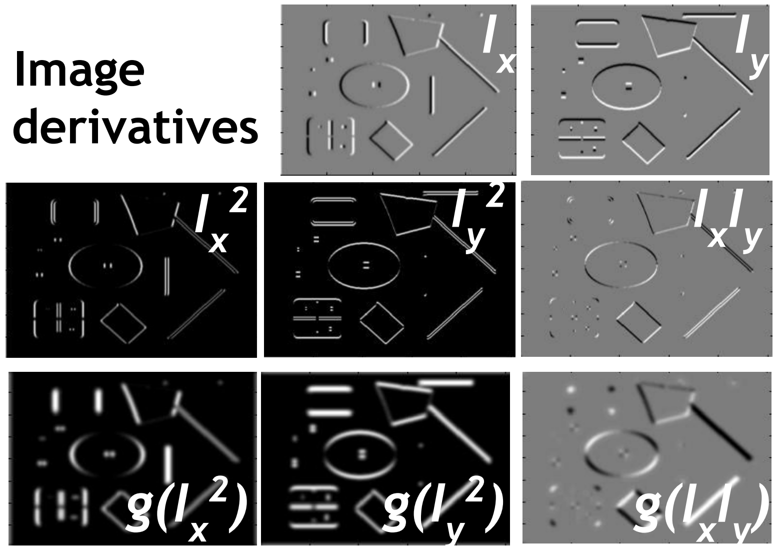
σ_D: for Gaussian in the derivative calculation

σ_I: for Gaussian in the windowing function

1. Image derivatives

2. Square of derivatives

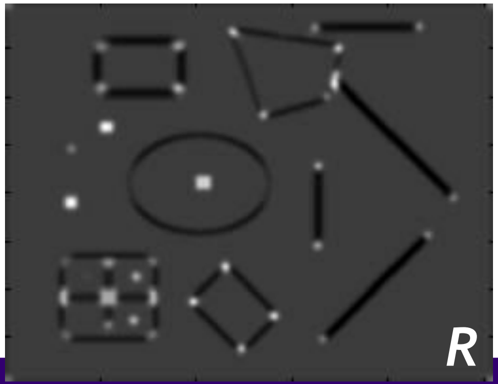
3. Gaussian filter $g(\sigma_I)$



4. Cornerness function – two strong eigenvalues

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

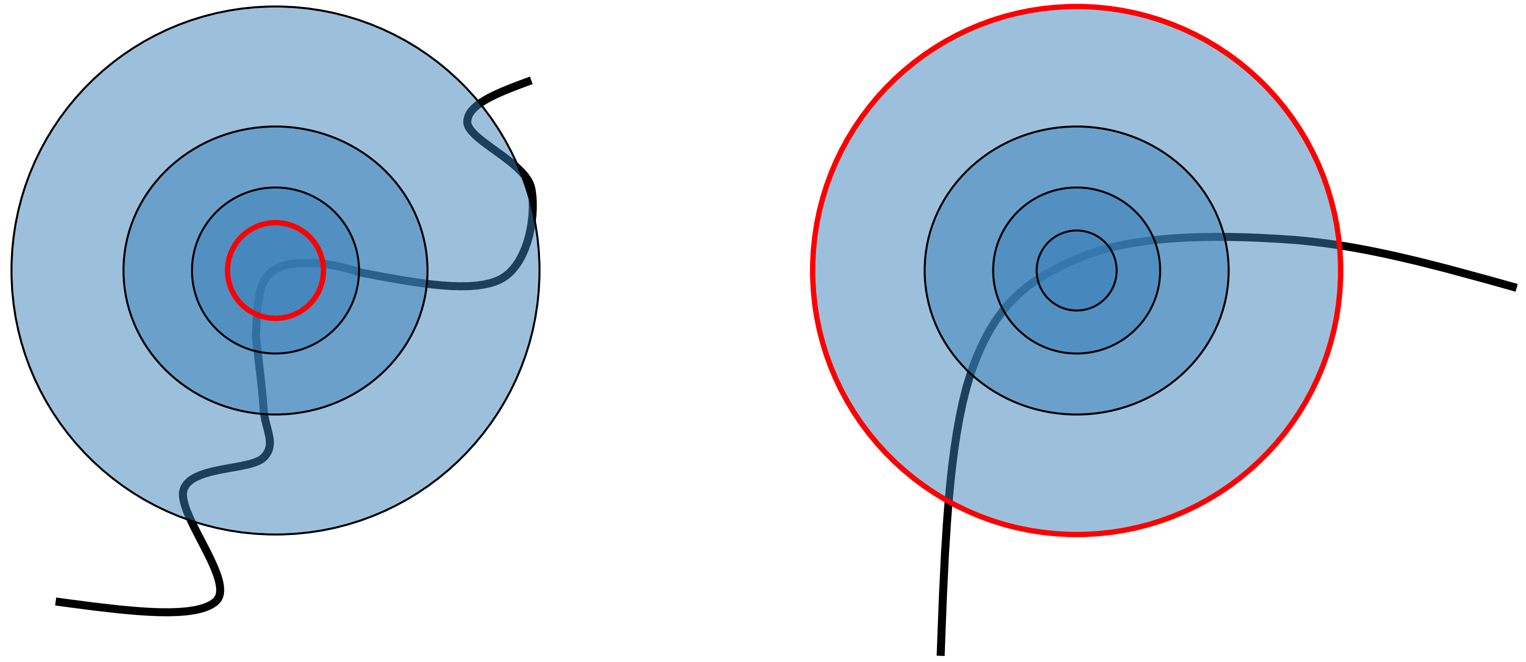
5. Perform non-maximum suppression



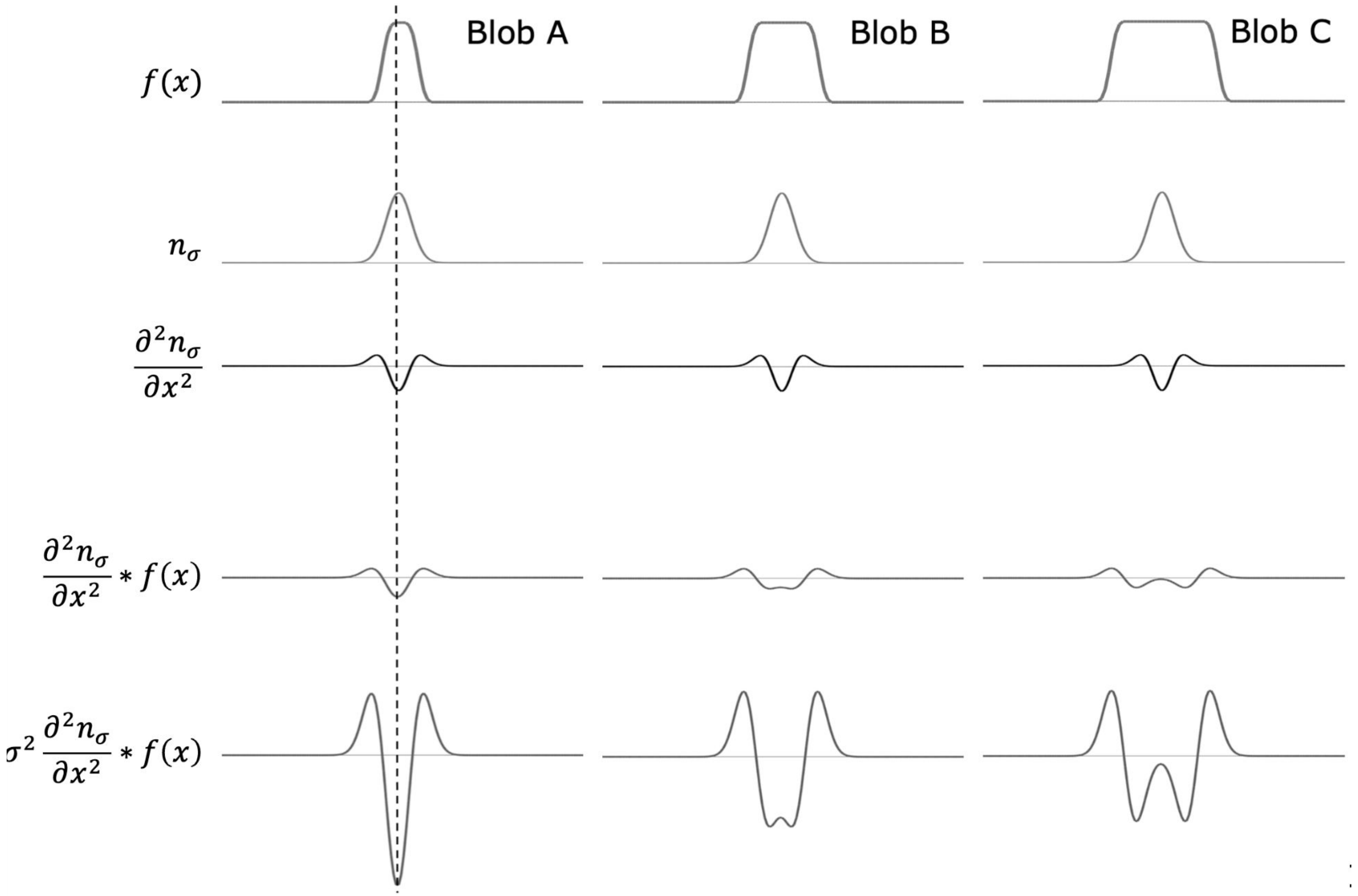
Slide credit: Krystian Mikolajczyk

So far: Harris is not a Scale Invariant Detection

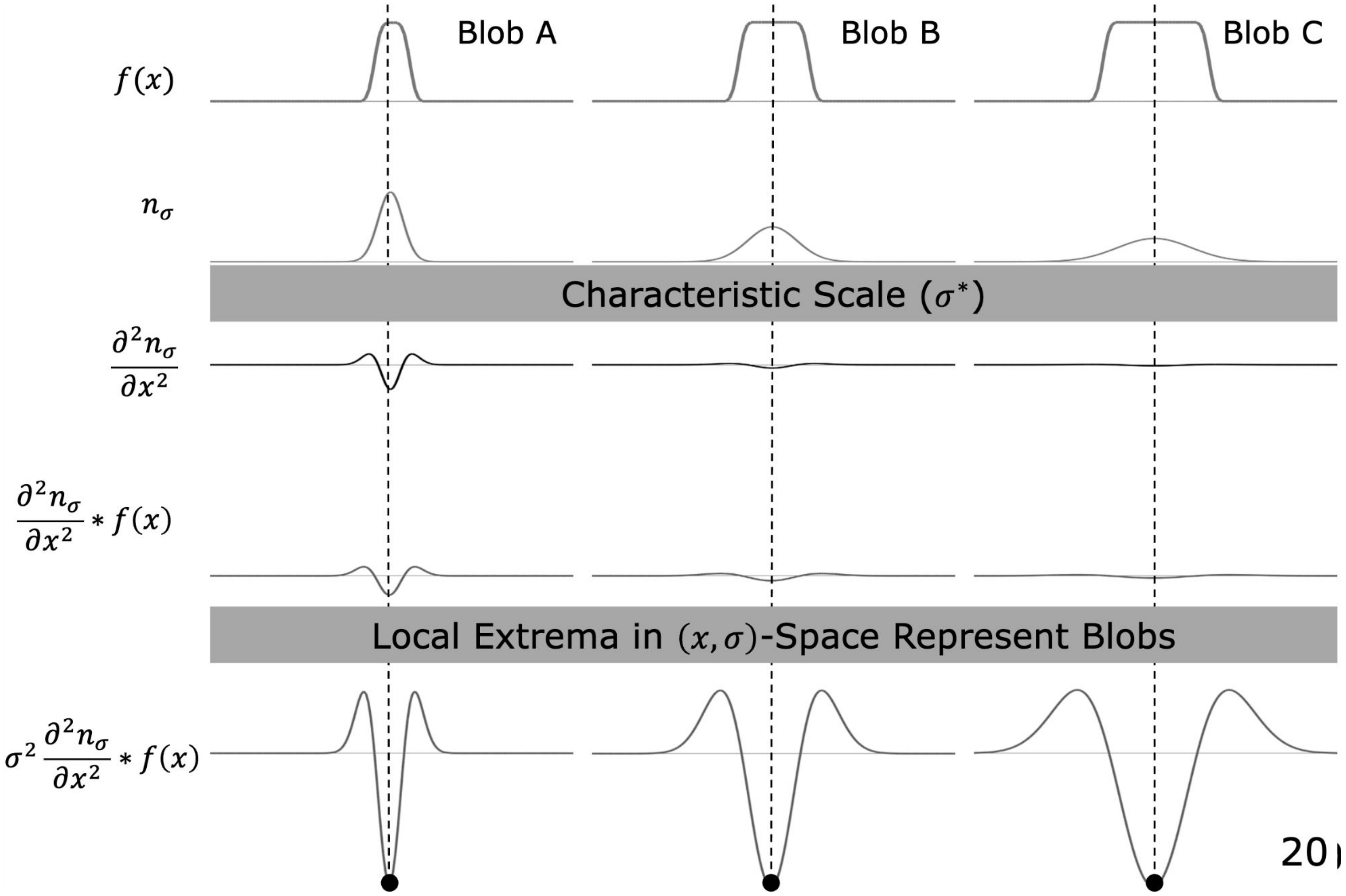
- Consider regions (e.g. circles) of different sizes around a point
- What region size do we choose, so that the regions look the same in both images?



So far: Laplacians can detect blobs of different sizes



So far: By increasing sigma, we can detect blobs of different sizes


Blob A
Blob B
Blob C
$f(x)$
n_σ
Characteristic Scale (σ^*)
$\frac{\partial^2 n_\sigma}{\partial x^2}$
$\frac{\partial^2 n_\sigma}{\partial x^2} * f(x)$
Local Extrema in (x, σ)-Space Represent Blobs
$\sigma^2 \frac{\partial^2 n_\sigma}{\partial x^2} * f(x)$
20

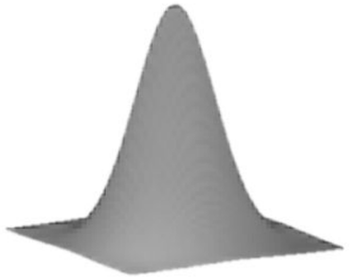
So far: Laplacians in 2D

Normalized LoG (NLoG) is used to find blobs in images

Laplacian

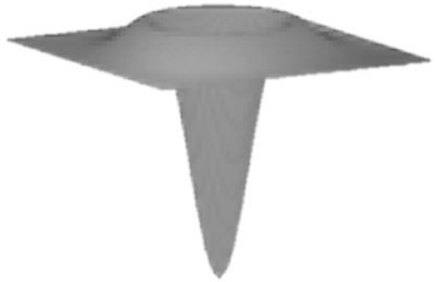
$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

Gaussian



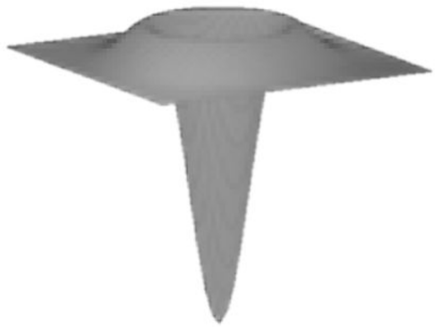
n_σ

LoG



$\nabla^2 n_\sigma$

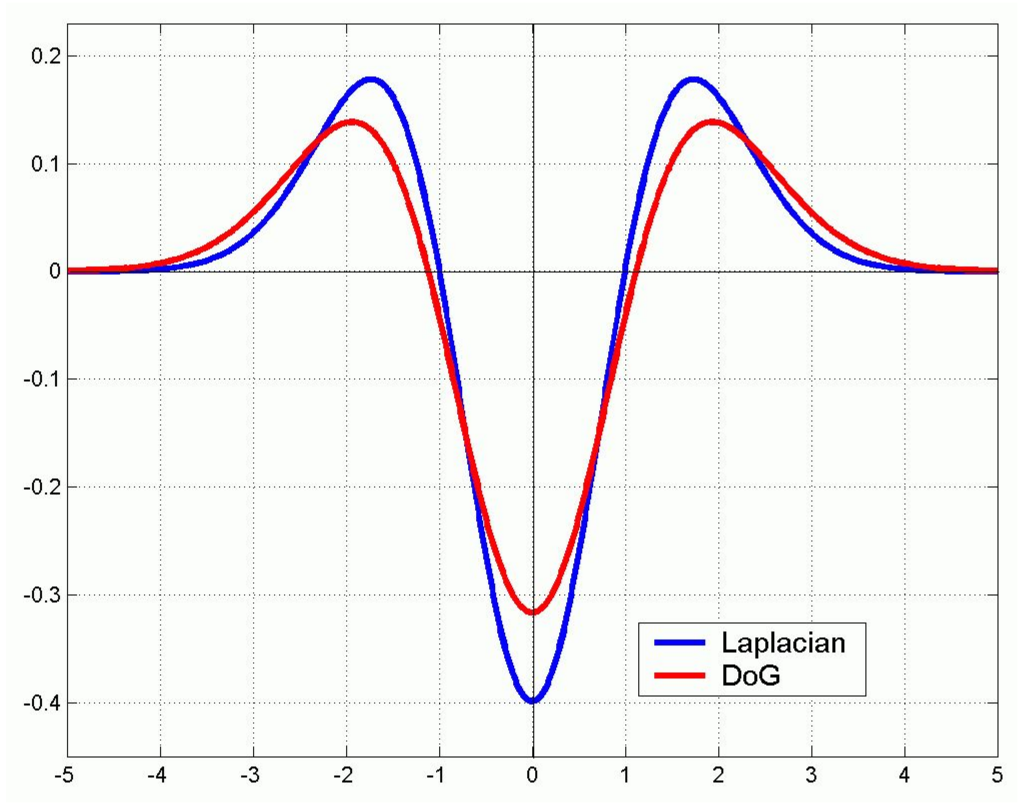
NLoG



$\sigma^2 \nabla^2 n_\sigma$

Location of Blobs identified by Local maxima after applying NLoG at many scales.

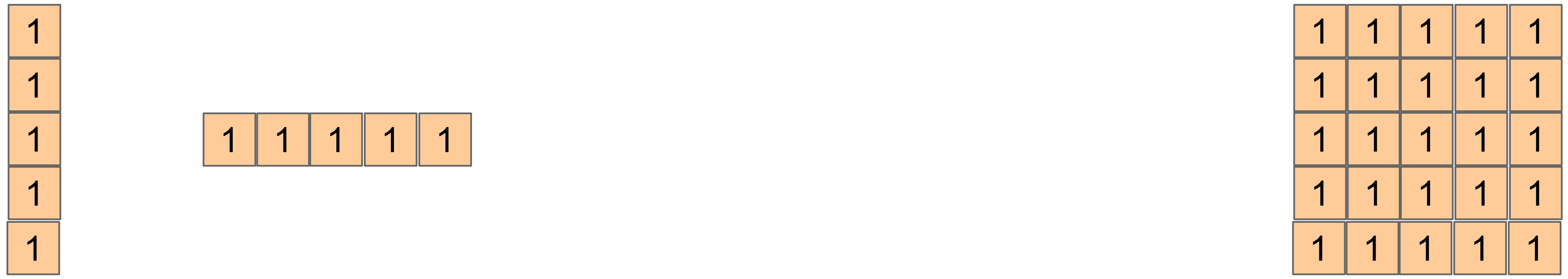
So far: SIFT detectors approximated Laplacians with difference of Gaussians (DoG)



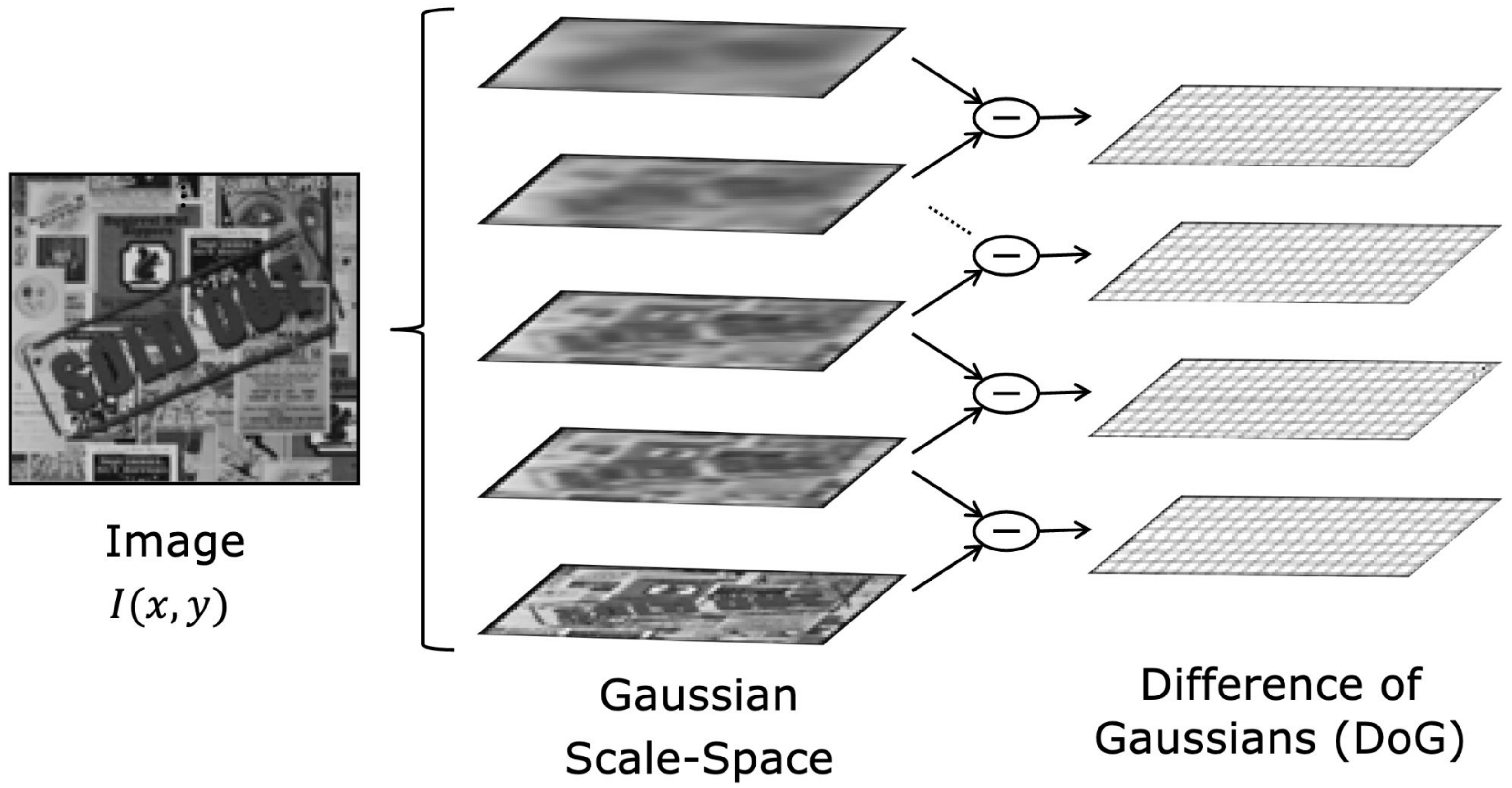
Note: both filters are invariant to *scale* and *rotation*

So far: More efficient because of separate filters

Convolving with two 1D convolution filters = convolving with a large 2D filter

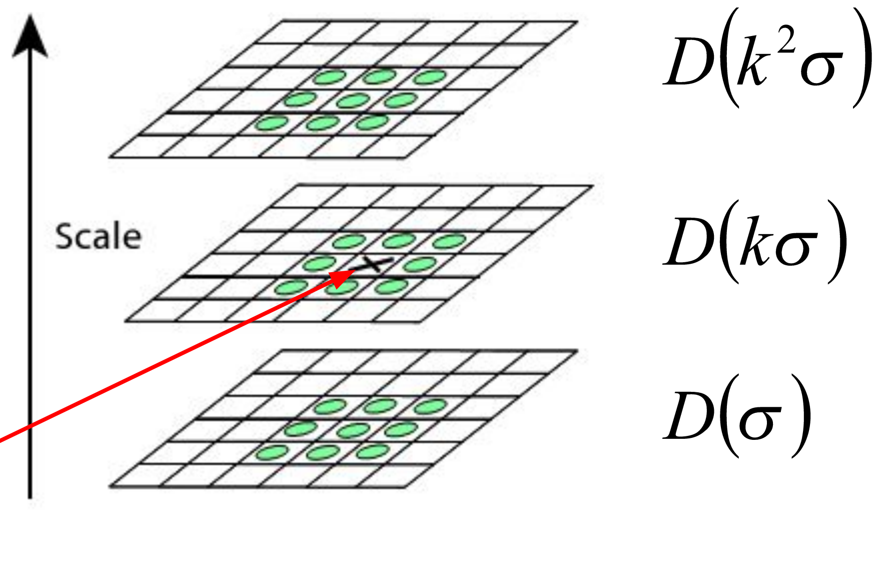


So far: Overall SIFT detector algorithm



So far: Extracting SIFT keypoints and scales

- Choose the maxima within 3x3x3 neighborhood.

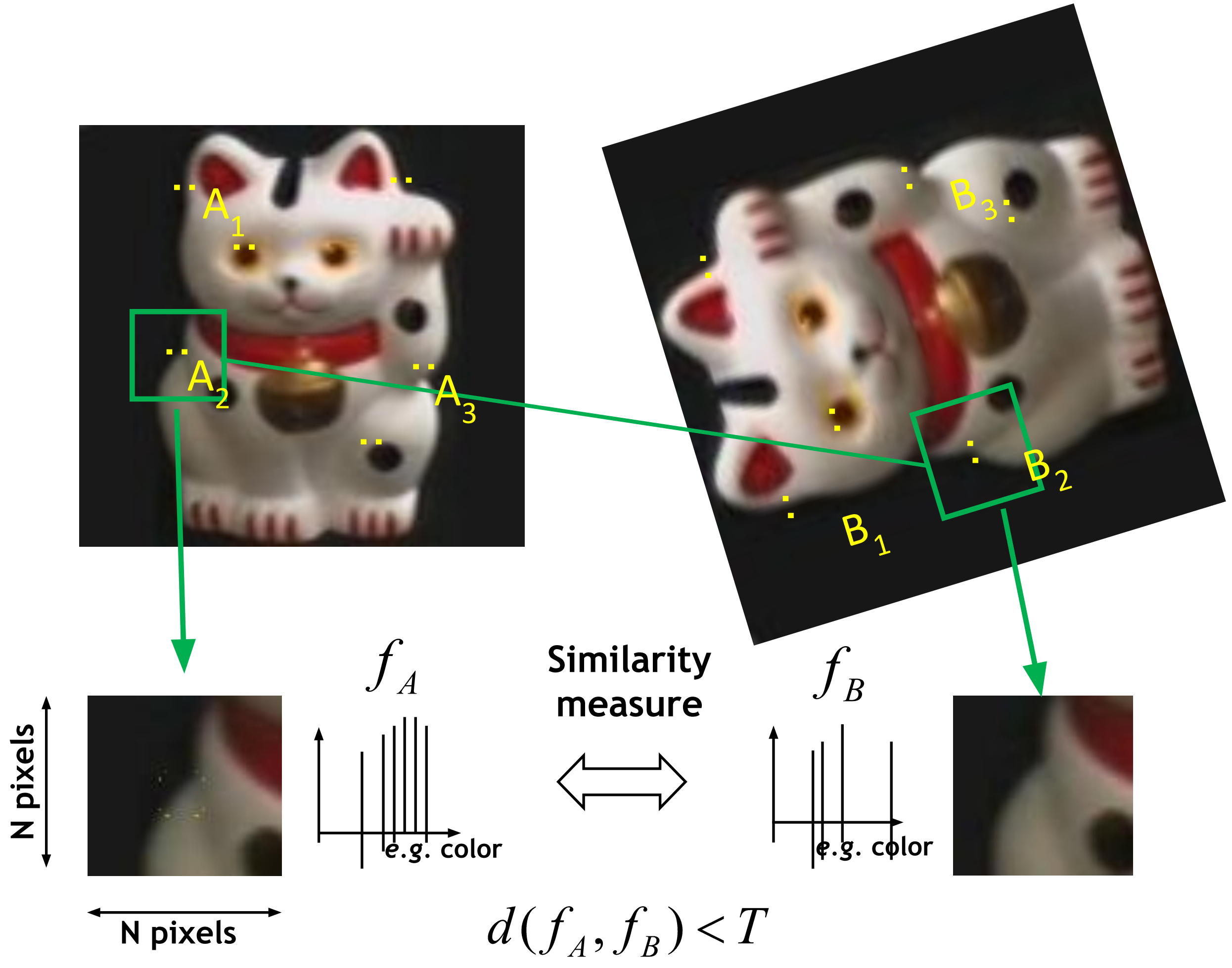


X is selected if it is larger or smaller than all 26 neighbors

Today's agenda

- Local descriptors (SIFT)
  - Making keypoints rotation invariant
  - Designing a descriptor
  - Designing a matching function
- Image Homography
- Global descriptors (HoG)

Why we care about knowing the keypoint patch size??



1. Find a set of distinctive key-points

2. Define a region/patch around each keypoint

3. Normalize the region content

4. Compute a local descriptor from the normalized region

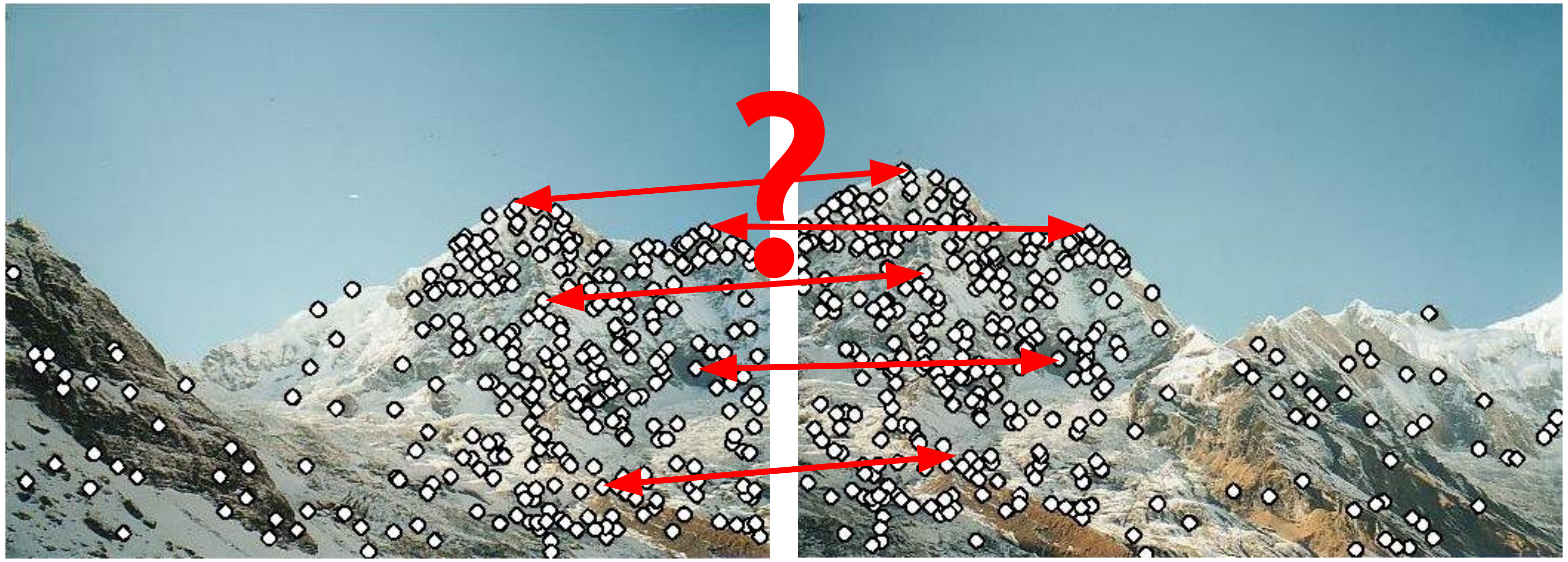
5. Match local descriptors

Today’s agenda

- Local descriptors (SIFT)
  - Making keypoints rotation invariant
  - Designing a descriptor
  - Designing a matching function
- Image Homography
- Global descriptors (HoG)

Local Descriptors are vectors

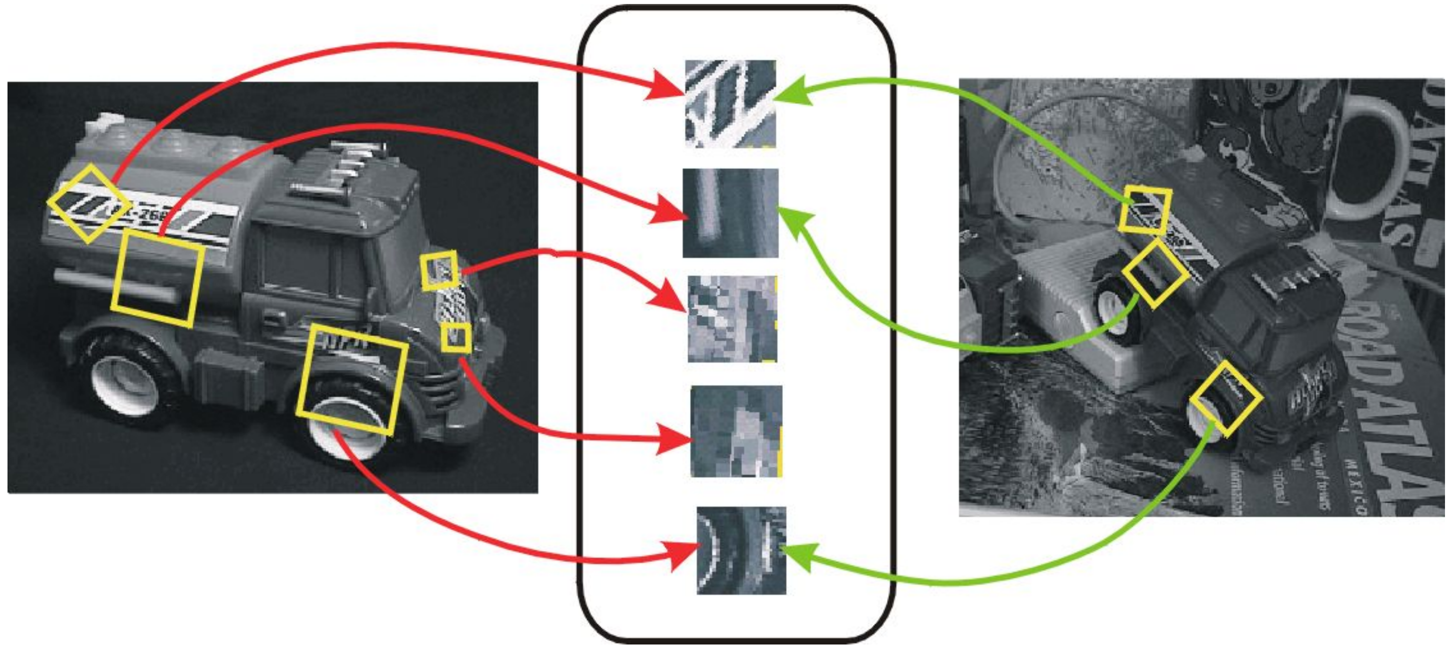
- We know how to detect points
- Next question: How to describe them for matching?
- Descriptor: Vector that summarizes the content of the keypoint neighborhood.



Point descriptor should be:
1. **Invariant**
2. **Distinctive**

Invariant Local Descriptors

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

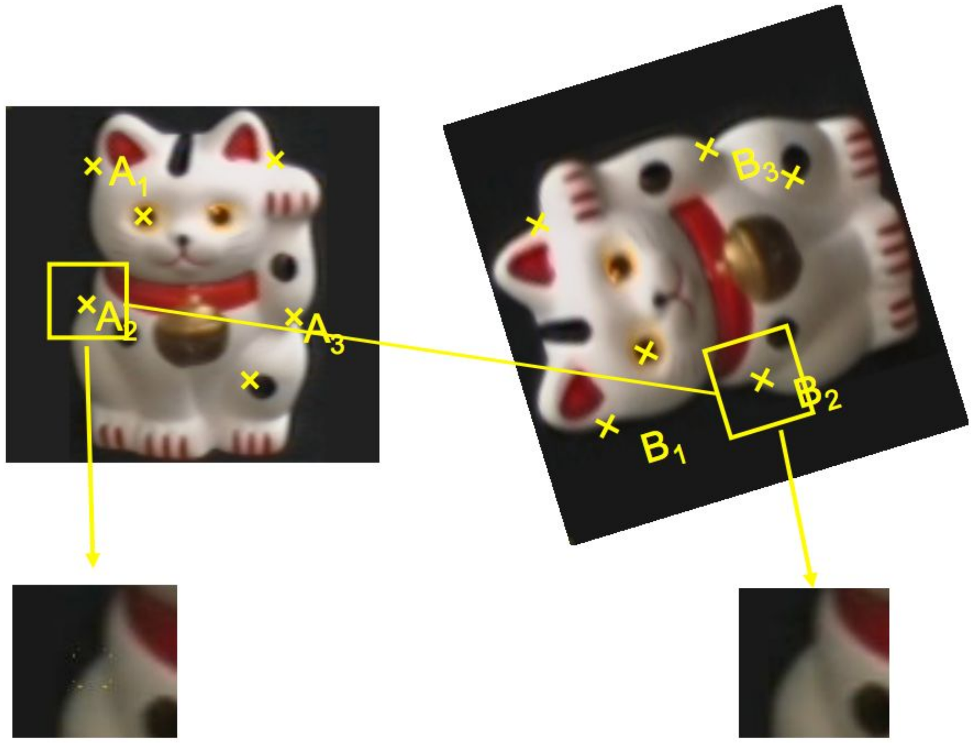


Rotation invariant descriptors

So far, we have figured out the scale of the keypoints.

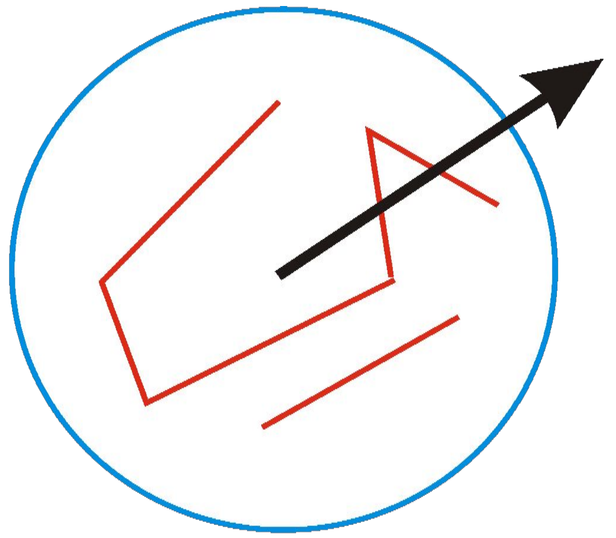
- So we can normalize them to be the same size.

Q. How do we re-orient the patches so that they are rotation invariant?



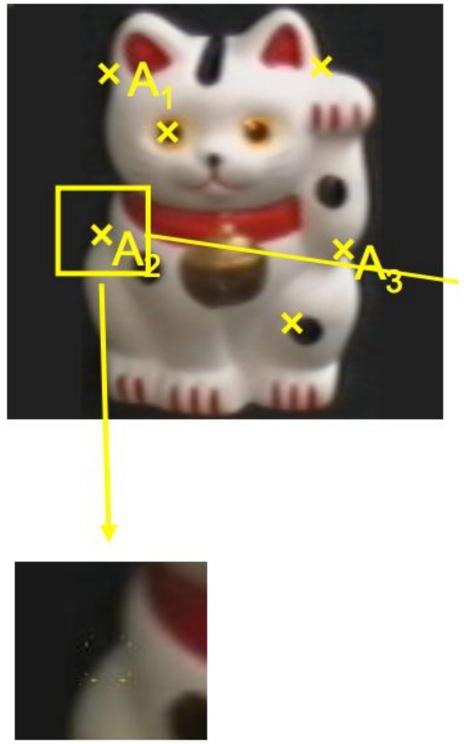
Constructing a rotation invariant descriptor

- We are given a keypoint and its scale from **DoG**
- We will select the direction of maximum gradient as the orientation for the keypoint
- We will describe all features ***relative*** to this orientation

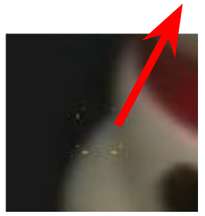


Visualizing what that looks like

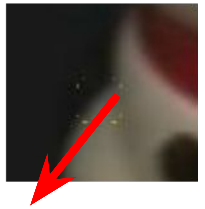
Q. Which one is the direction of the maximum gradient for this ketpoint patch?



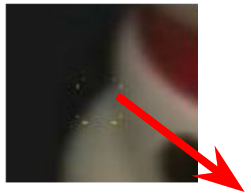
A)



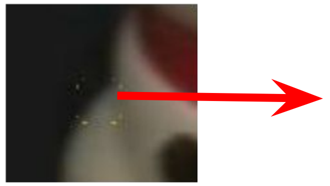
B)

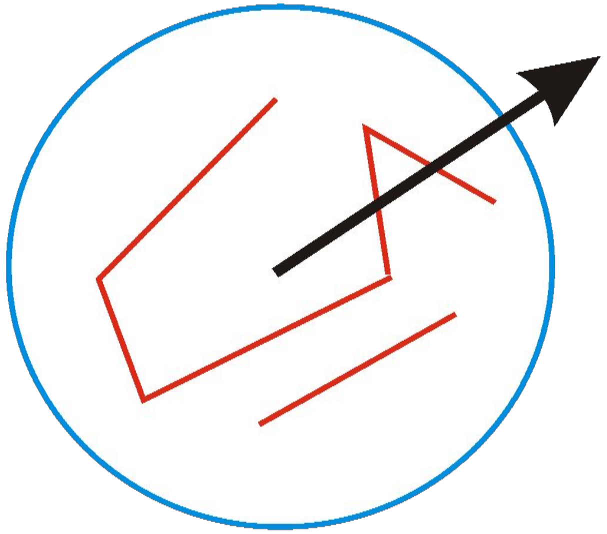


C)



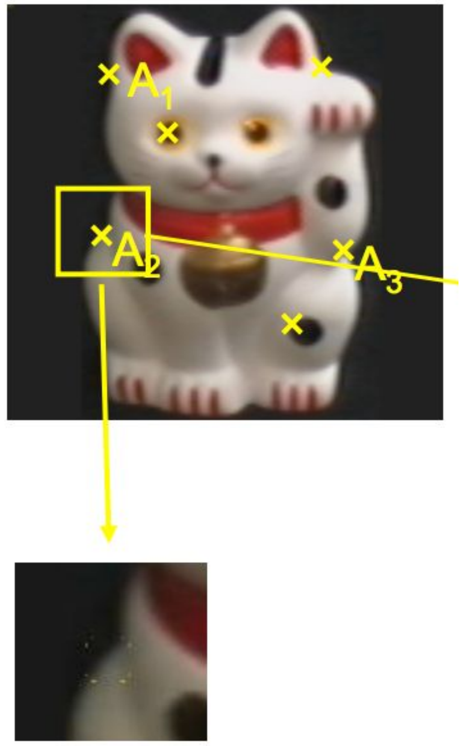
D)



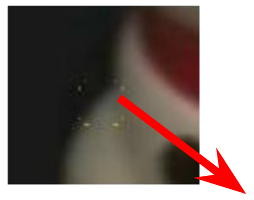


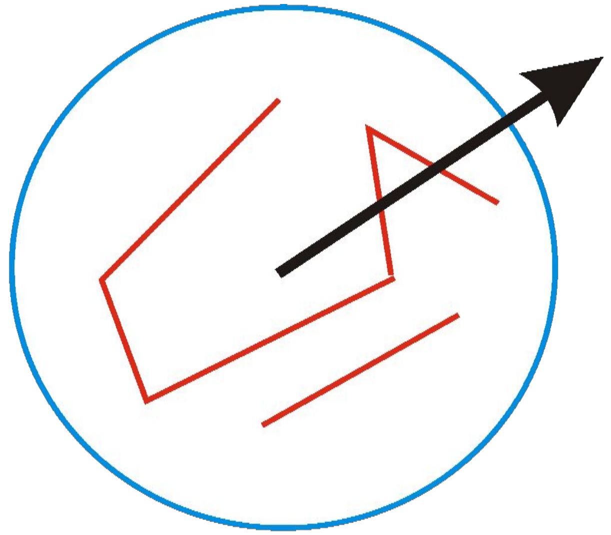
Visualizing what that looks like

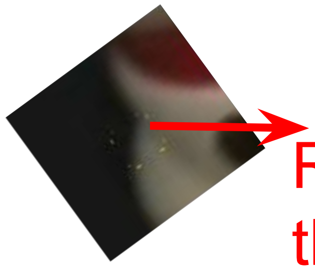
Q. Which one is the direction of the maximum gradient for this ketpoint patch?



C)



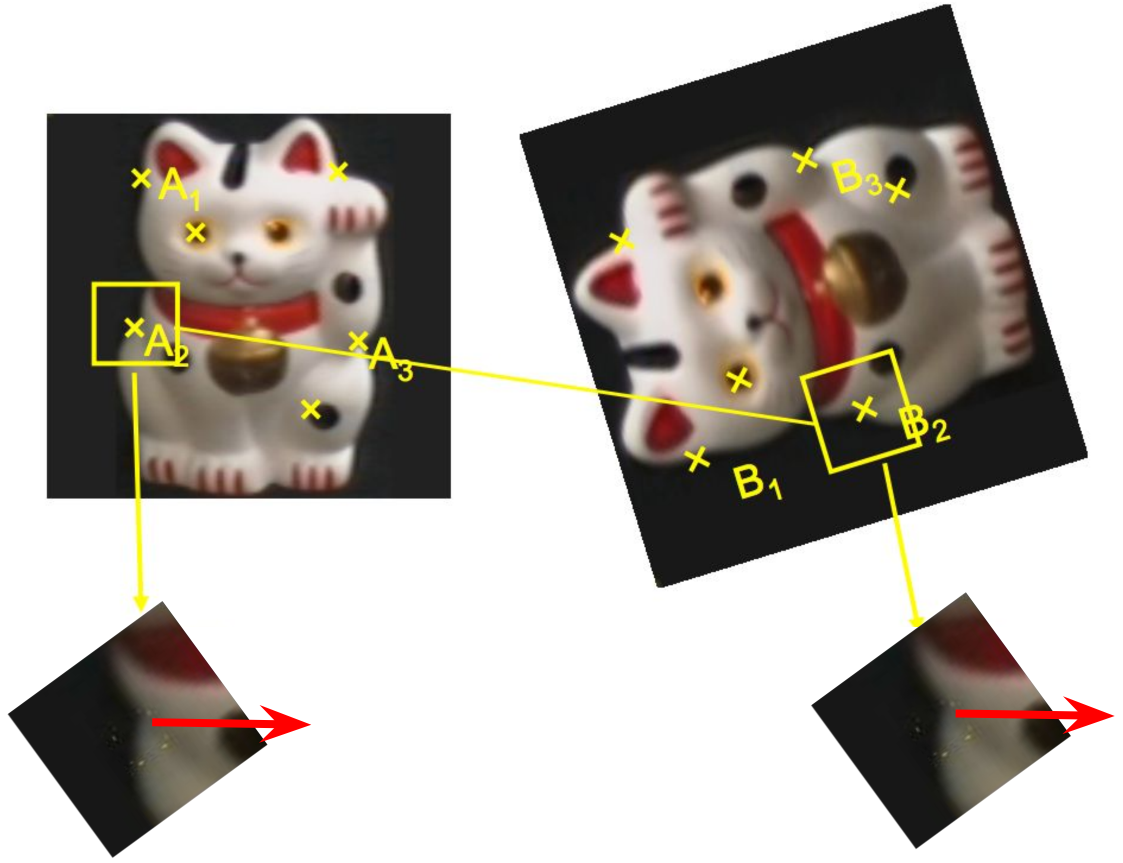


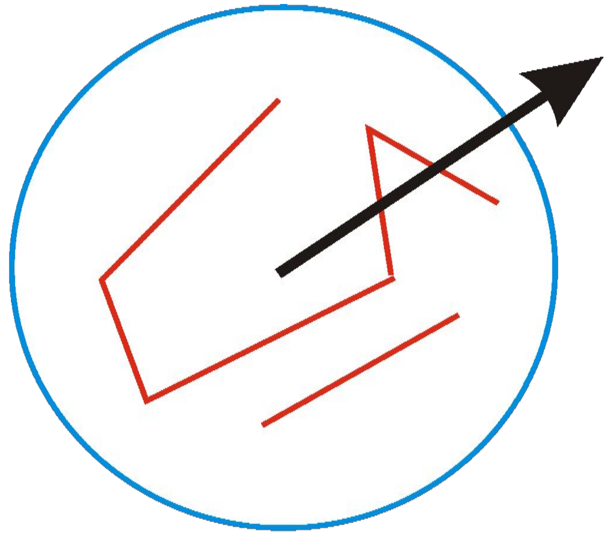


Rotated patch to make sure the gradient θ = 0

Feature descriptors become rotation invariant

- If the keypoint appears rotated in another image, the features will be the same, because they're **relative** to the characteristic orientation



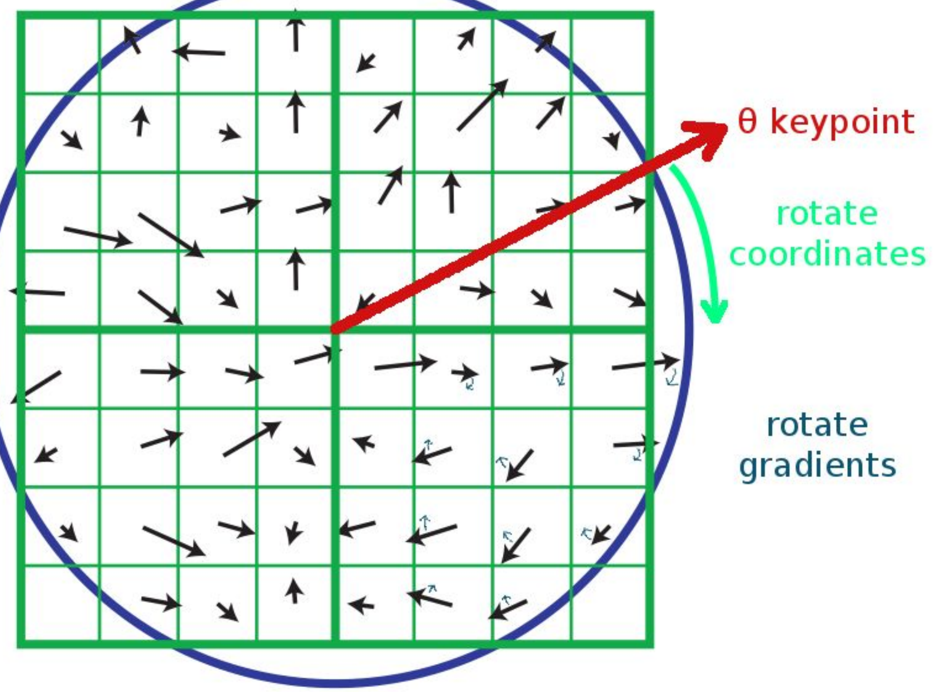


SIFT descriptor (Scale-Invariant Feature Transform)

Gradient-based descriptor to capture texture in the keypoint neighborhood

1. Blur the keypoint's image patch to remove noise
2. Calculate image **gradients** over the neighborhood patch.
3. To become rotation invariant, rotate the gradients by -θ (- maximum direction)
   - Now we've cancelled out rotation and have gradients expressed at locations relative to maximum direction θ
4. Generate a descriptor

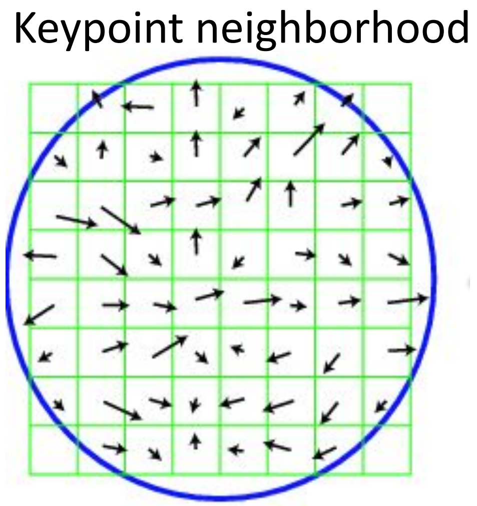
Keypoint neighborhood



Today's agenda

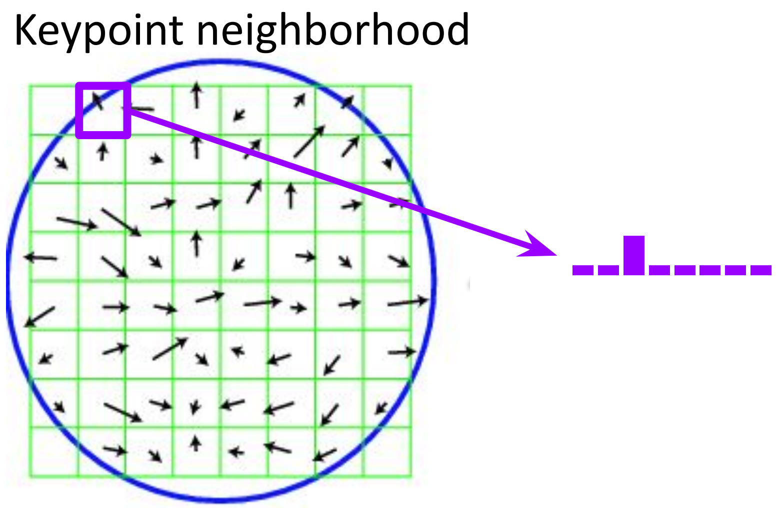
- Local descriptors (SIFT)
  - Making keypoints rotation invariant
  - Designing a descriptor
  - Designing a matching function
- Image Homography
- Global descriptors (HoG)

Generating the descriptor from rotated patch



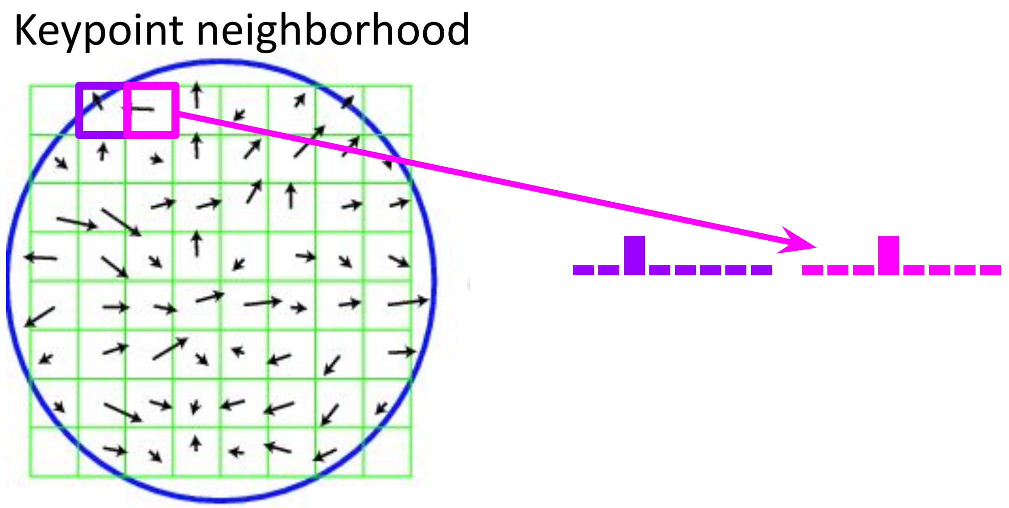
- Q. How do we turn this into a vector?

Generating the descriptor from rotated patch



- We can turn every pixel into a histogram
- Histogram contains 8 buckets, all of them zero except for one.
- Make the bucket of the direction of the gradient equal to 1

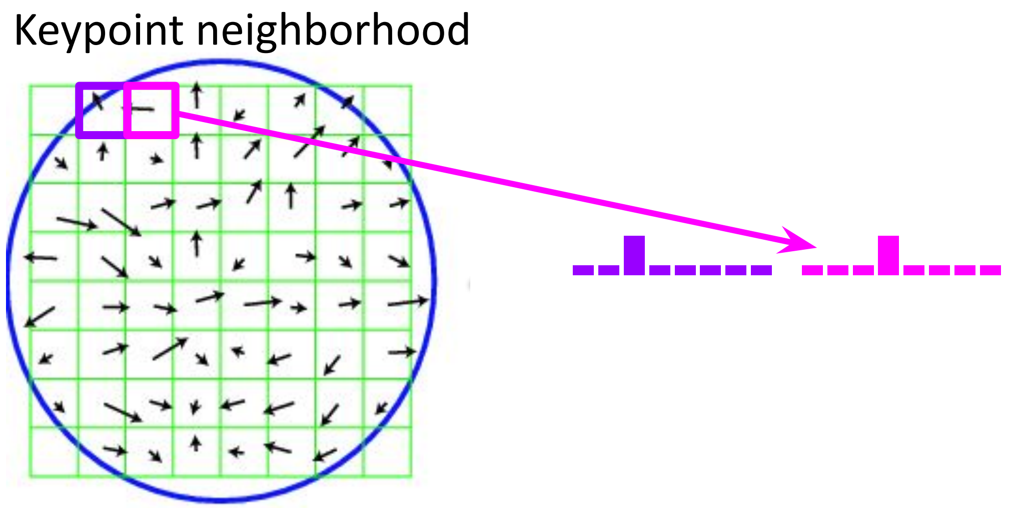
Generating the descriptor from rotated patch

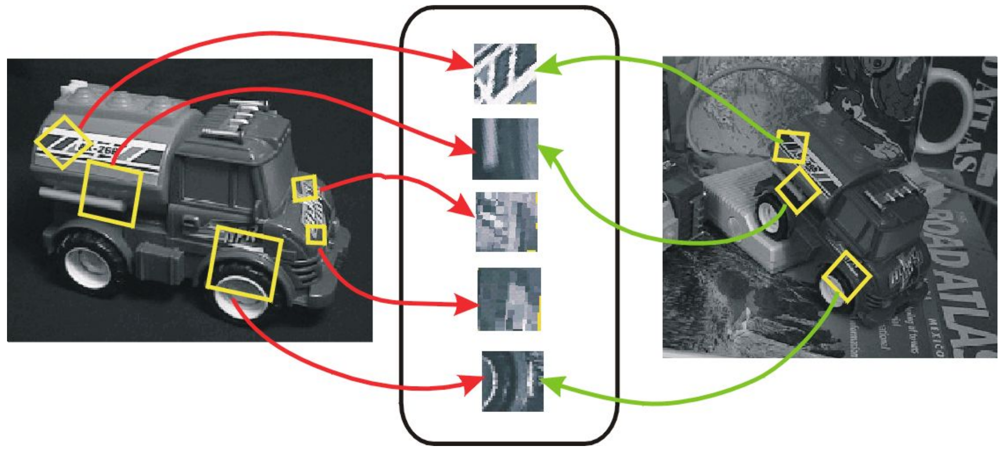


- Do this for every single pixel

Q. What would the size of the keypoint vector be?

Generating the descriptor from rotated patch



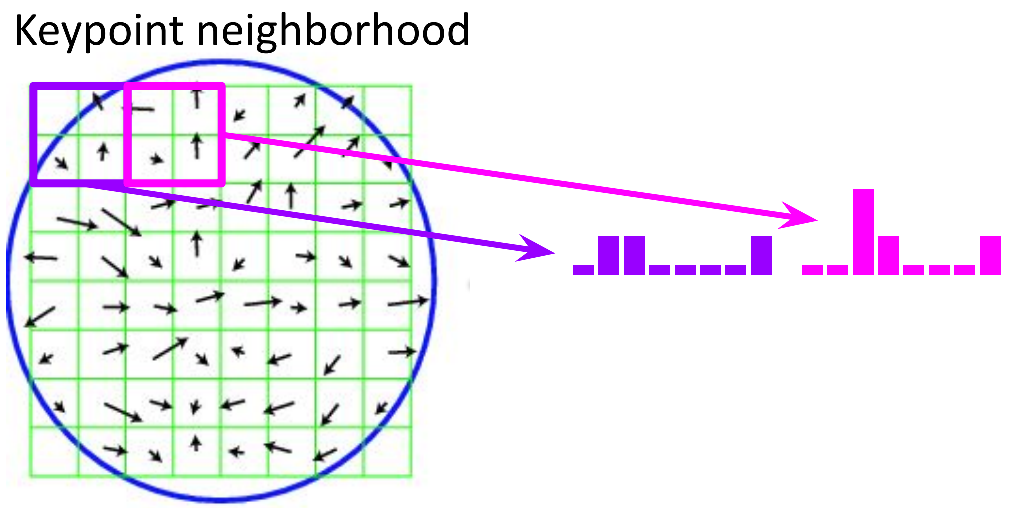


- Do this for every single pixel

Q. Why might this be a bad strategy? What could go wrong?

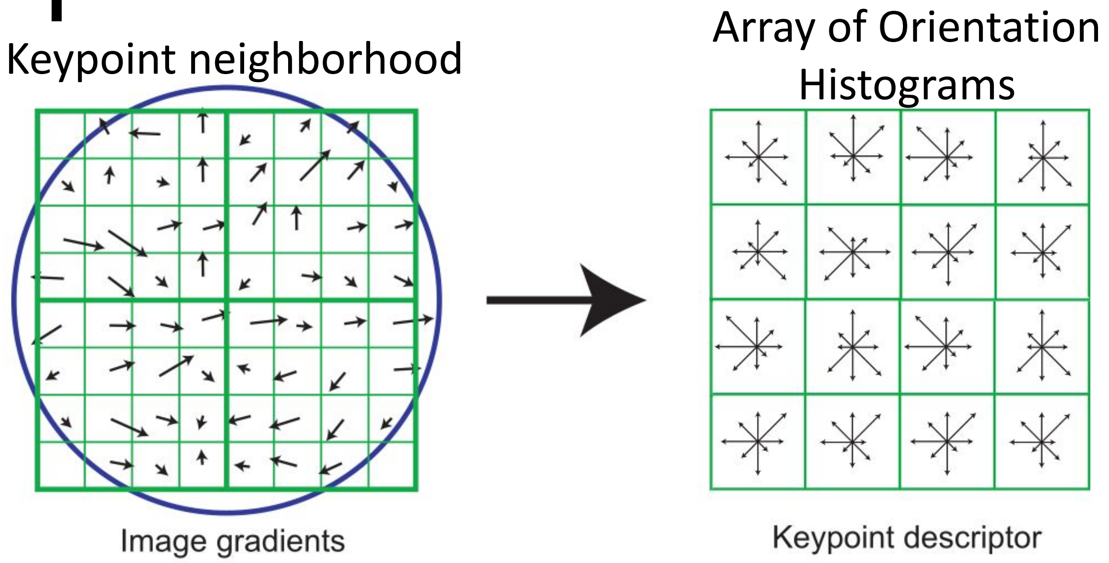
Hint: think about how matching might fail

Generating the descriptor from rotated patch



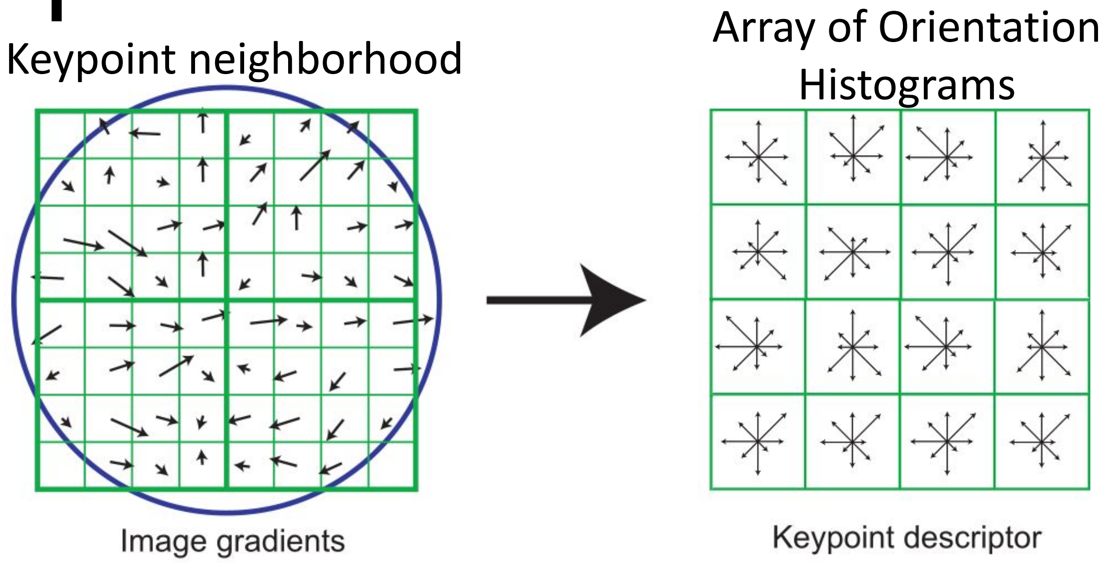
- **Solution**: divide keypoint up into 4x4 “cells”
- Calculate a histogram per cell and sum them together

SIFT descriptor formation



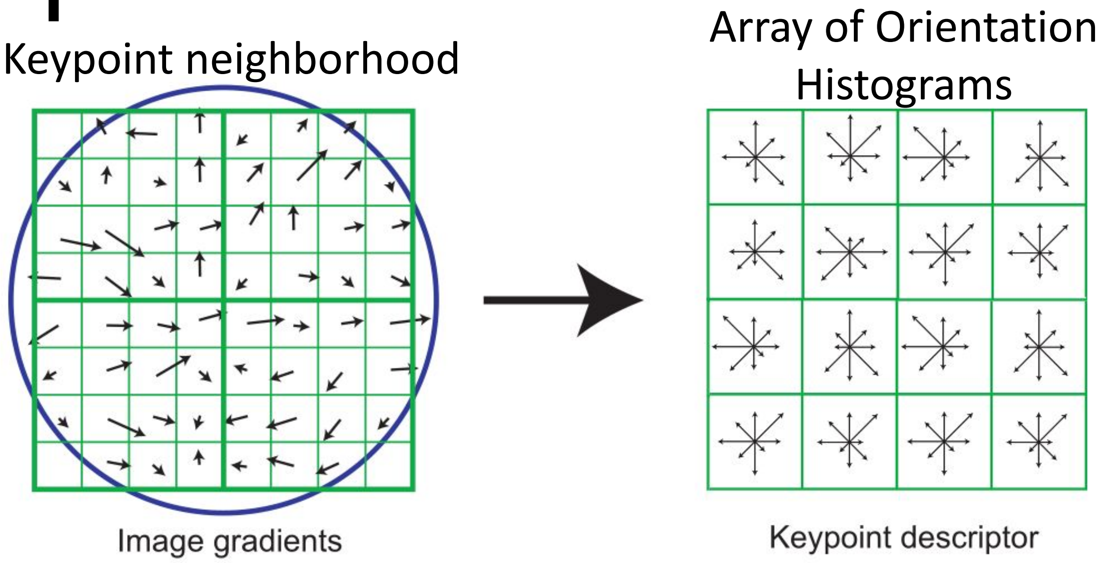
- Each cell gives us a histogram vector. We have a total of 4x4 vectors
- Calculate the overall gradients in each patch into their local orientated histograms
  - Also, scale down gradient contributions for gradients far from the center
  - Each histogram is quantized into 8 directions (each 45 degrees)

SIFT descriptor formation

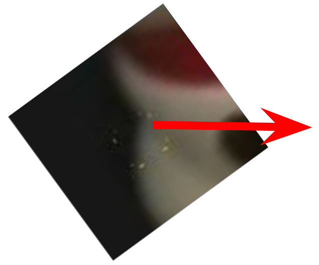


- Q. What is the size of the descriptor?

SIFT descriptor formation

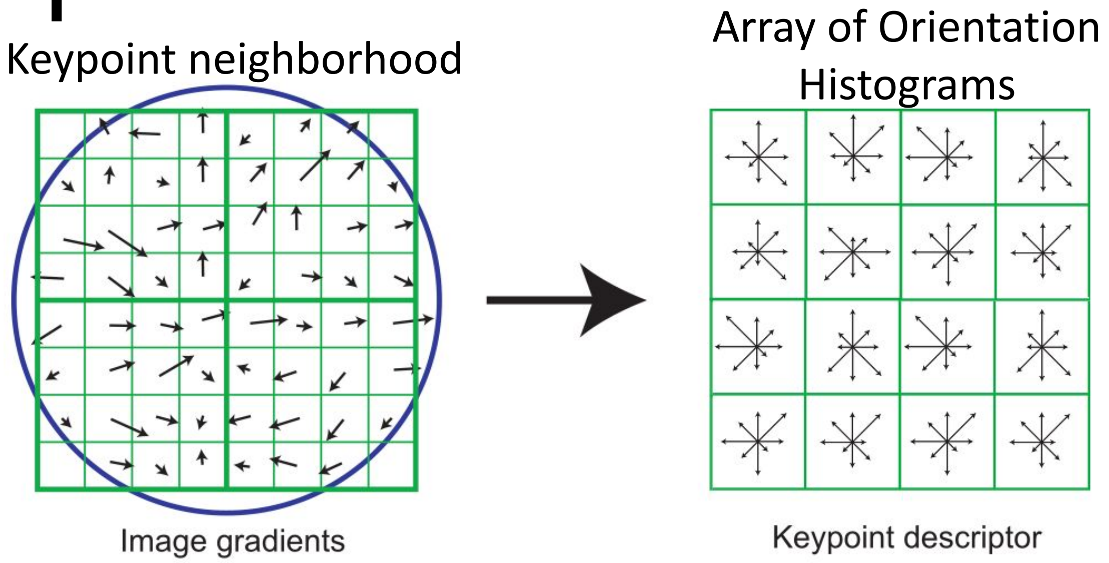


- 8 orientation bins per histogram,
- 4x4 histogram vectors,
- total is 8 x 4x4 = 128 numbers.
- So a SIFT descriptor is a length 128 vector



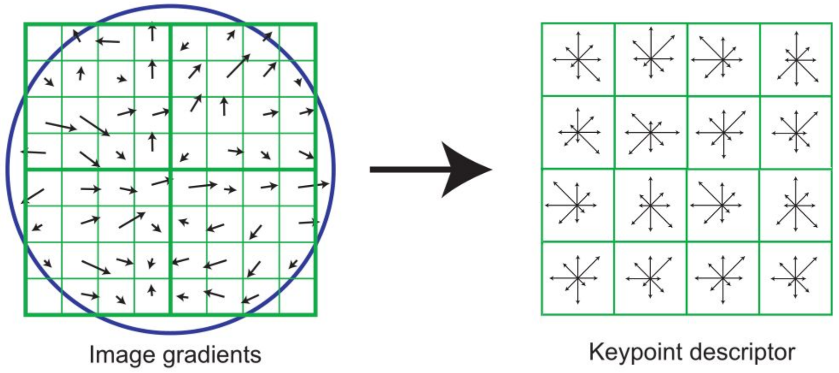
$$\mathcal{HoG}(k) = \begin{bmatrix} g_1 \\ g_2 \\ \dots \\ g_{128} \end{bmatrix}$$

SIFT descriptor formation



- SIFT descriptor is invariant to rotation (because we rotated the patch) and scale (because we worked with the scaled image from DoG)
- We can compare each vector from image A to each vector from image B to find matching keypoints!
    - How do we match distances?

Making descriptors robust



- Adding robustness to illumination changes:
- Each descriptor is made of gradients (differences between pixels),
  - It's already invariant to changes in brightness
  - (e.g. adding 10 to all image pixels yields the exact same descriptor)
- A sharpening filter applied to the image will increase the magnitude of gradients linearly.
  - To correct for contrast changes, normalize the histogram (scale to magnitude=1.0)
- Very large image gradients are usually from unreliable 3D illumination effects (glare, etc).
  - To reduce their effect, clamp all values in the vector to be ≤ 0.2 (an experimentally tuned value). Then normalize the vector again.
- Result is a vector which is fairly invariant to illumination changes.

Today's agenda

- Local descriptors (SIFT)
  - Making keypoints rotation invariant
  - Designing a descriptor
  - Designing a matching function
- Image Homography
- Global descriptors (HoG)

SIFT descriptor distances

Given keypoints k_1 and k_2, we can calculate their HoG features:

HoG(k_1)

HoG(k_2)

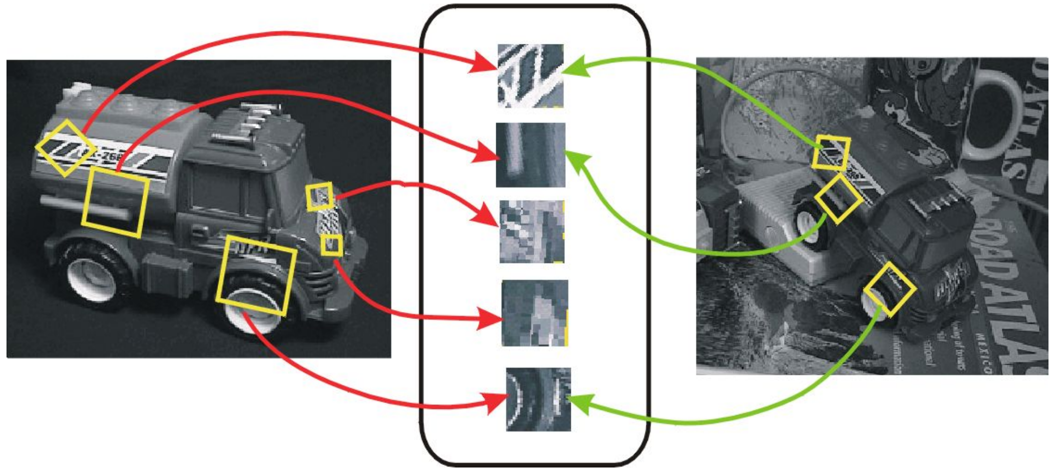
We can calculate their matching score as:

$$d_{\mathcal{HoG}}(k_1, k_2) = \sqrt{\sum_i (\mathcal{HoG}(k_1)_i - \mathcal{HoG}(k_2)_i)^2}$$

Find nearest neighbor for each keypoint in image A in image B

Given keypoints k_1 and k_2, we can calculate their HoG features:

HoG(k_1)
HoG(k_2)



We can calculate their matching score as:

$$d_{\mathcal{HoG}}(k_1, k_2) = \sqrt{\sum_i (\mathcal{HoG}(k_1)_i - \mathcal{HoG}(k_2)_i)^2}$$

Sensitivity to number of histogram orientations

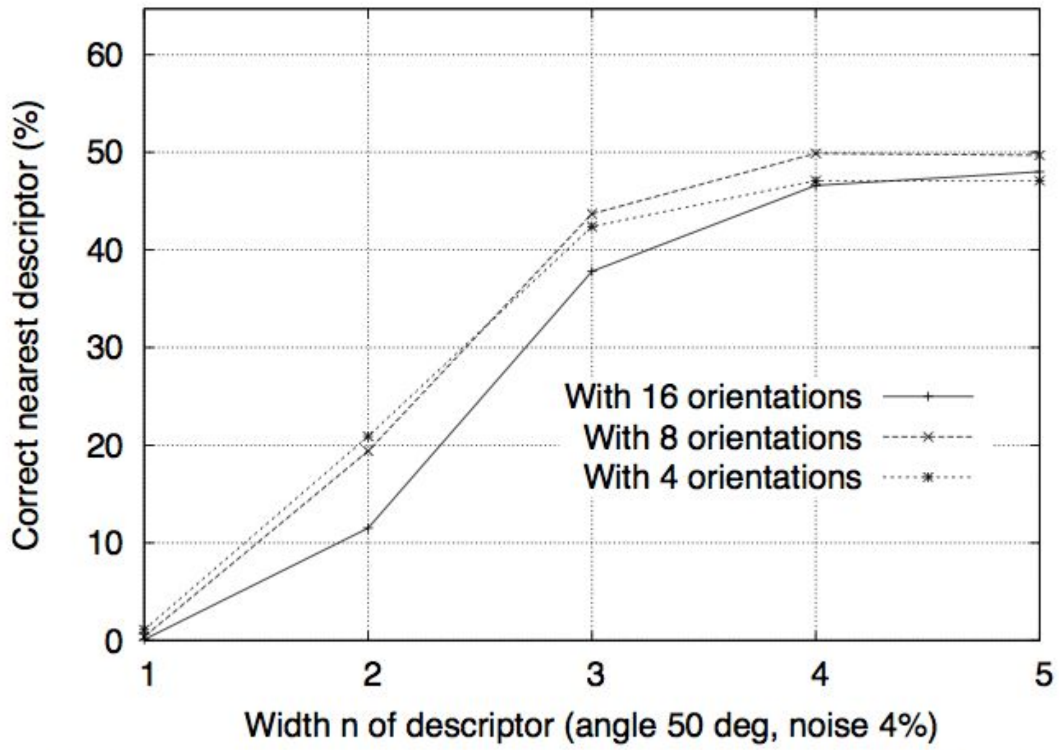
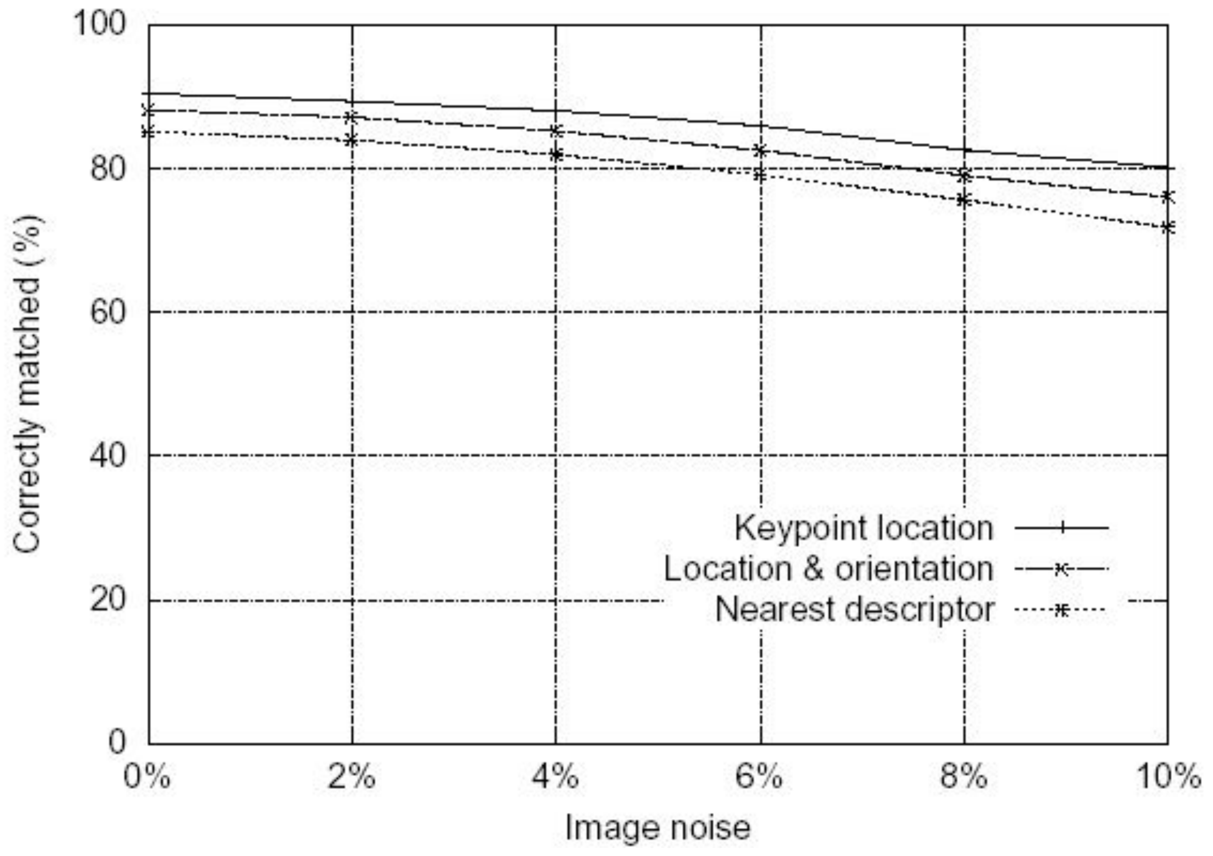


Figure 8: This graph shows the percent of keypoints giving the correct match to a database of 40,000 keypoints as a function of width of the $n \times n$ keypoint descriptor and the number of orientations in each histogram. The graph is computed for images with affine viewpoint change of 50 degrees and addition of 4% noise.

David G. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, 60, 2 (2004), pp. 91-110

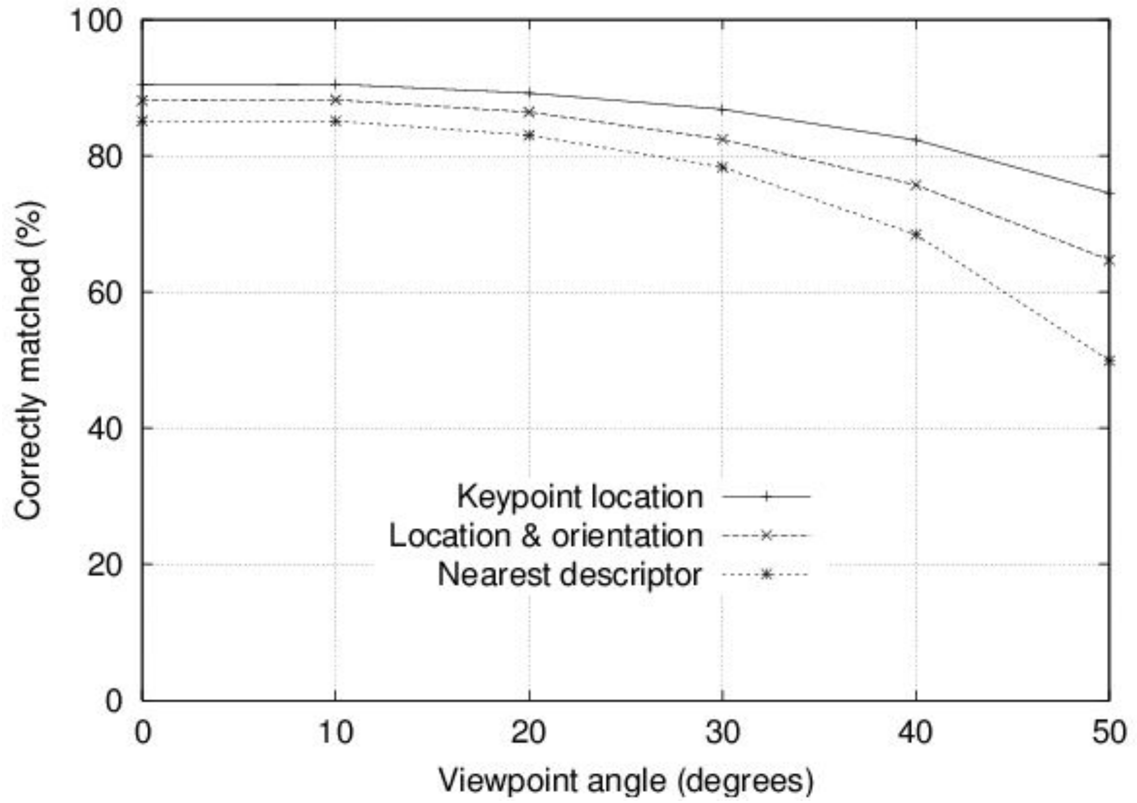
Feature stability to noise

- Match features after random change in image scale & orientation, with differing levels of image noise
- Find nearest neighbor in database of **30,000** features



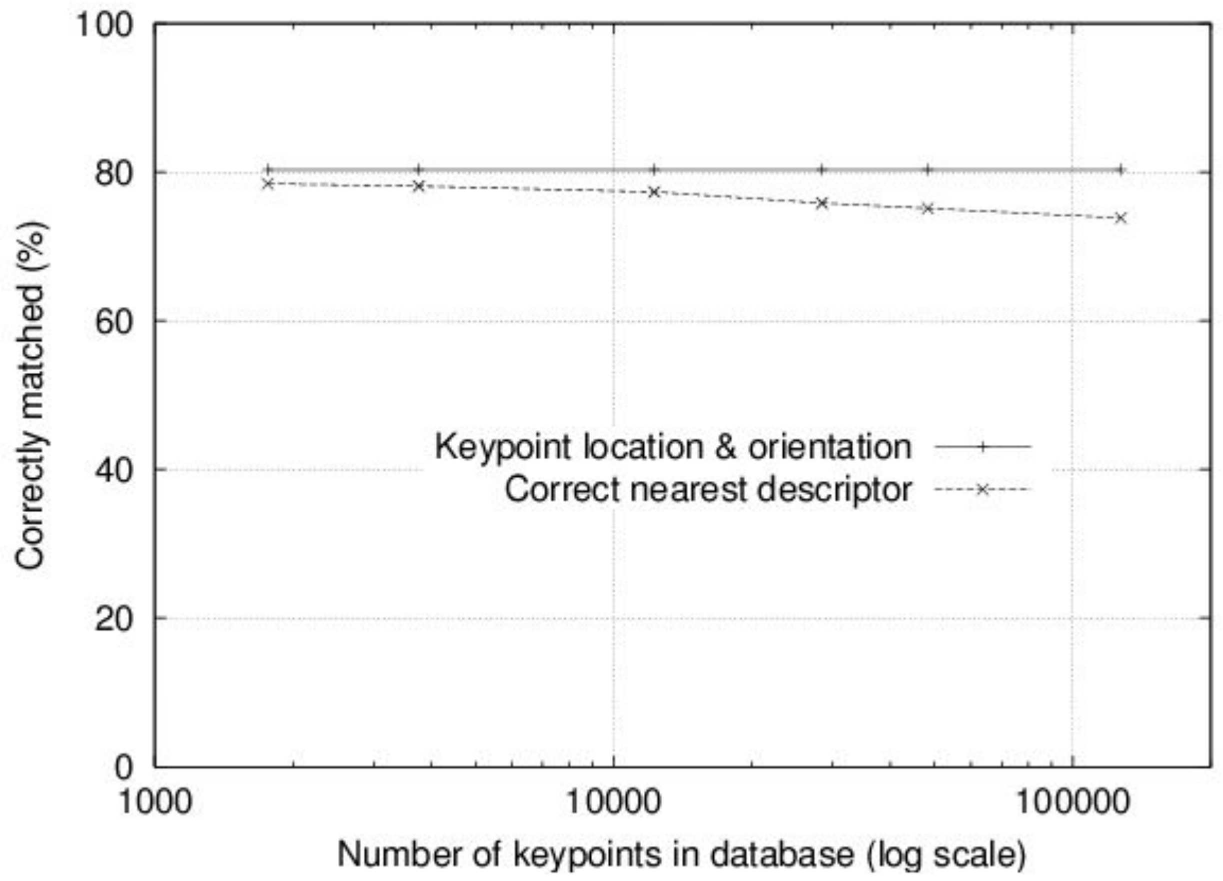
Feature stability to affine changes

- Match features after random change in image scale & orientation, with 2% image **noise**, and affine distortion
- Find nearest neighbor in database of **30,000** features



Distinctiveness of features

- Vary size of database of features, with **30 degree affine change**, **2% image noise**
- Measure % correct for single nearest neighbor match



Useful SIFT resources

- An online tutorial: http://www.aishack.in/2010/05/sift-scale-invariant-feature-transform/
- Wikipedia: http://en.wikipedia.org/wiki/Scale-invariant_feature_transform

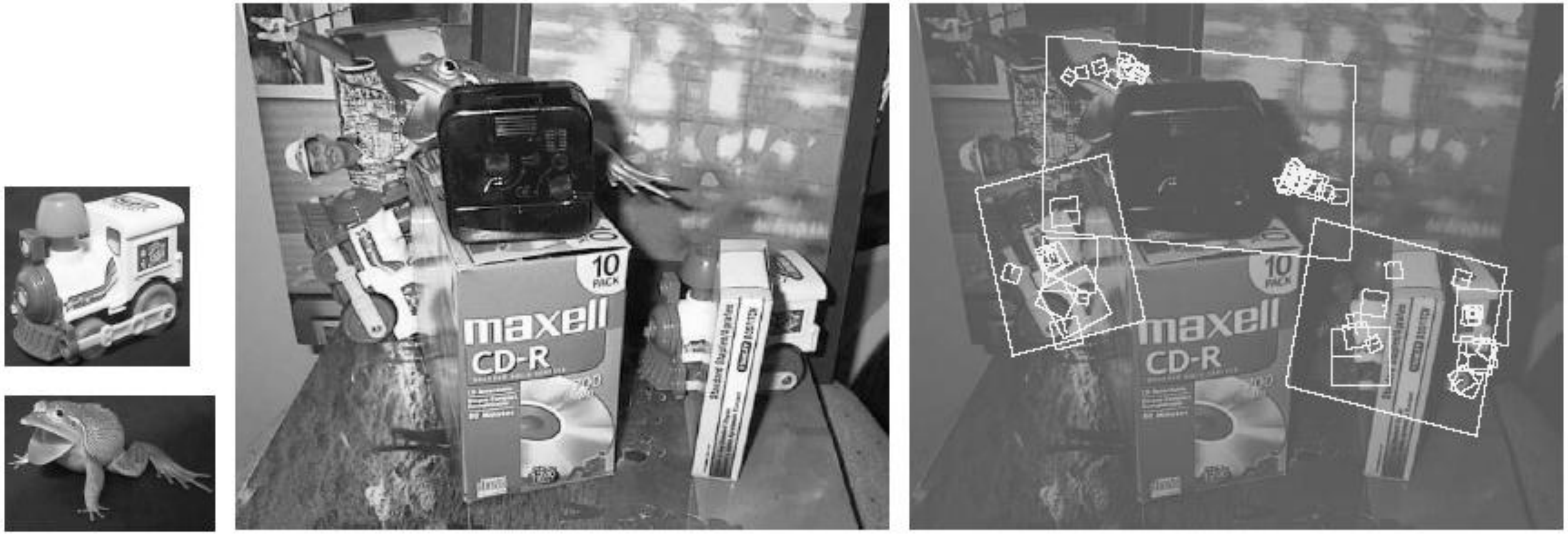


Figure 12: The training images for two objects are shown on the left. These can be recognized in a cluttered image with extensive occlusion, shown in the middle. The results of recognition are shown on the right. A parallelogram is drawn around each recognized object showing the boundaries of the original training image under the affi ne transformation solved for during recognition. Smaller squares indicate the keypoints that were used for recognition.

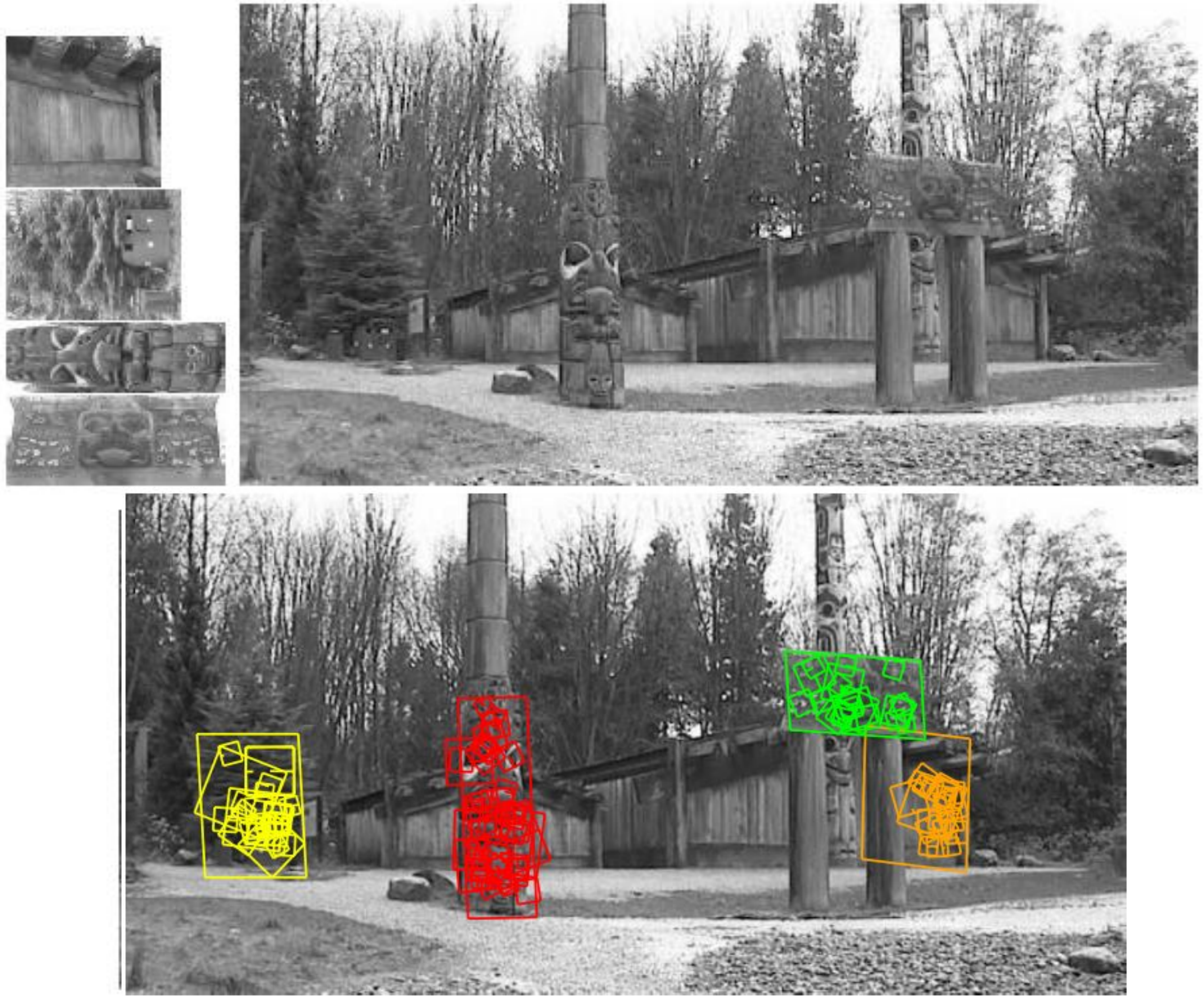


Figure 13: This example shows location recognition within a complex scene. The training images for locations are shown at the upper left and the 640x315 pixel test image taken from a different viewpoint is on the upper right. The recognized regions are shown on the lower image, with keypoints shown as squares and an outer parallelogram showing the boundaries of the training images under the affi ne transform used for recognition.

Recognition of specific objects, scenes

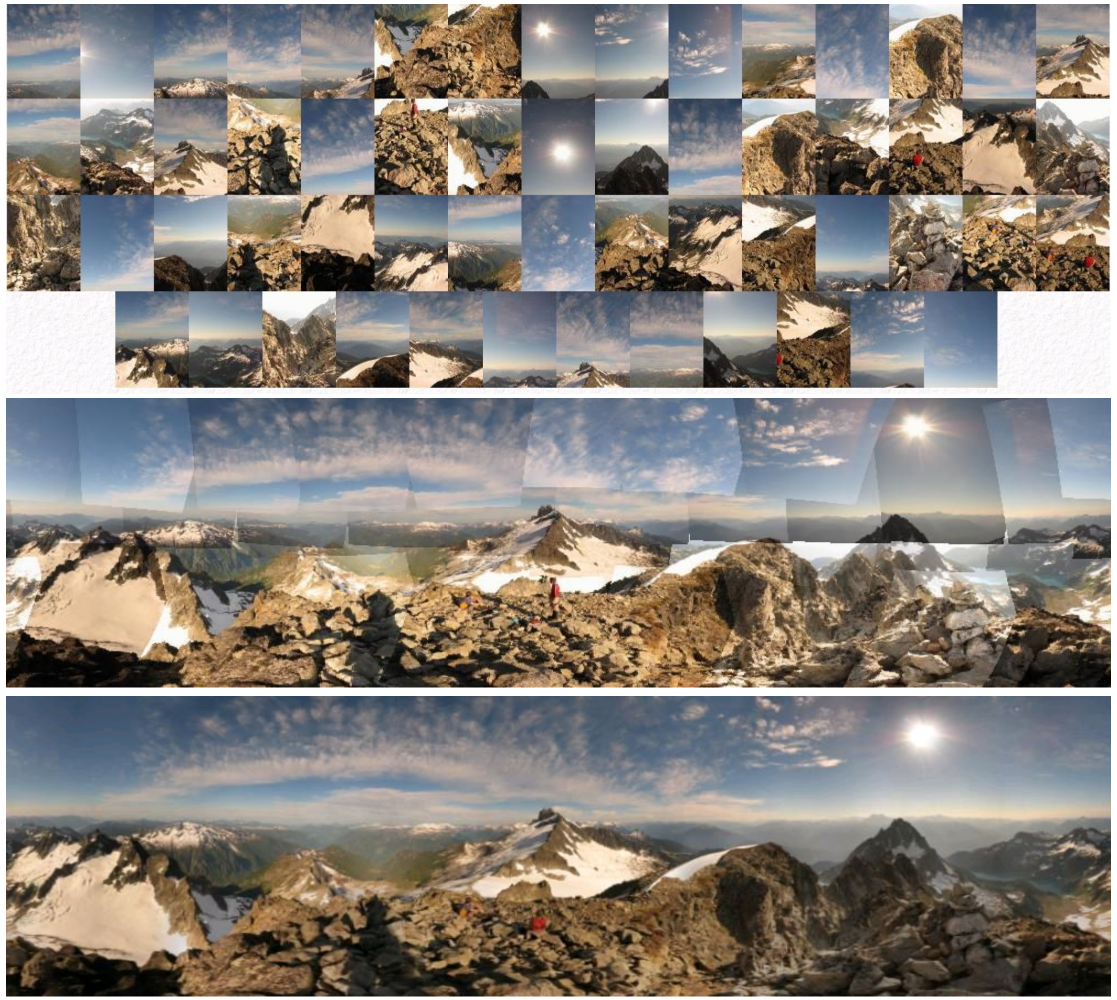
Schmid and Mohr 1997

Sivic and Zisserman, 2003

Rothganger et al. 2003

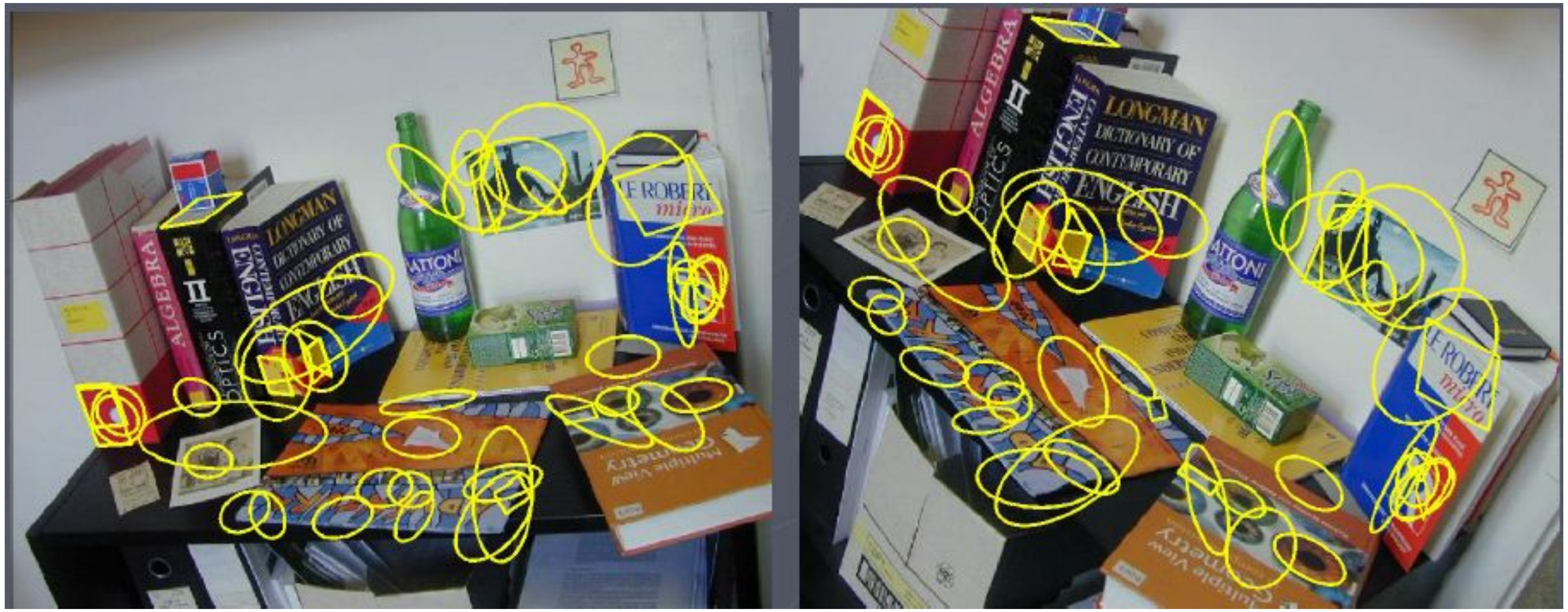
Lowe 2002

Panorama stitching/Automatic image mosaic

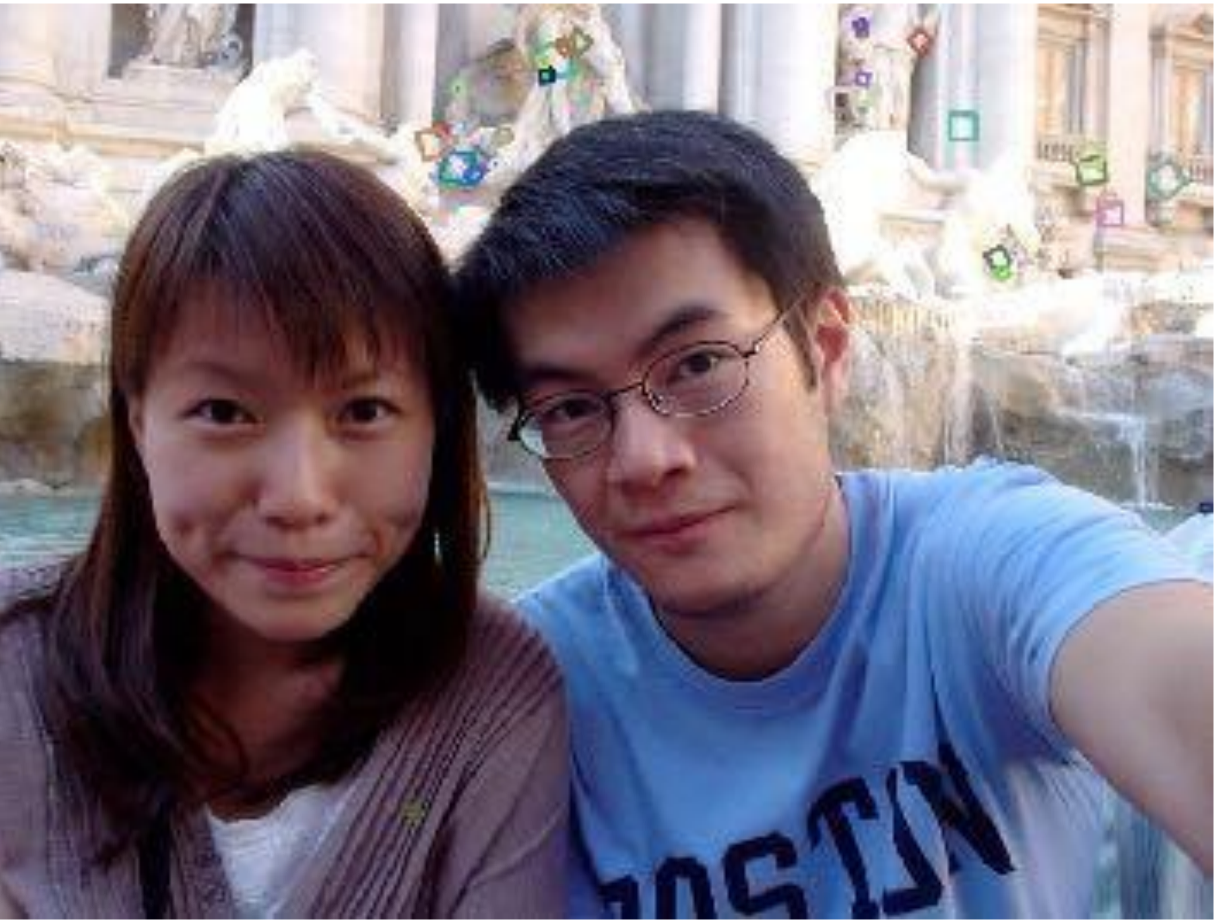


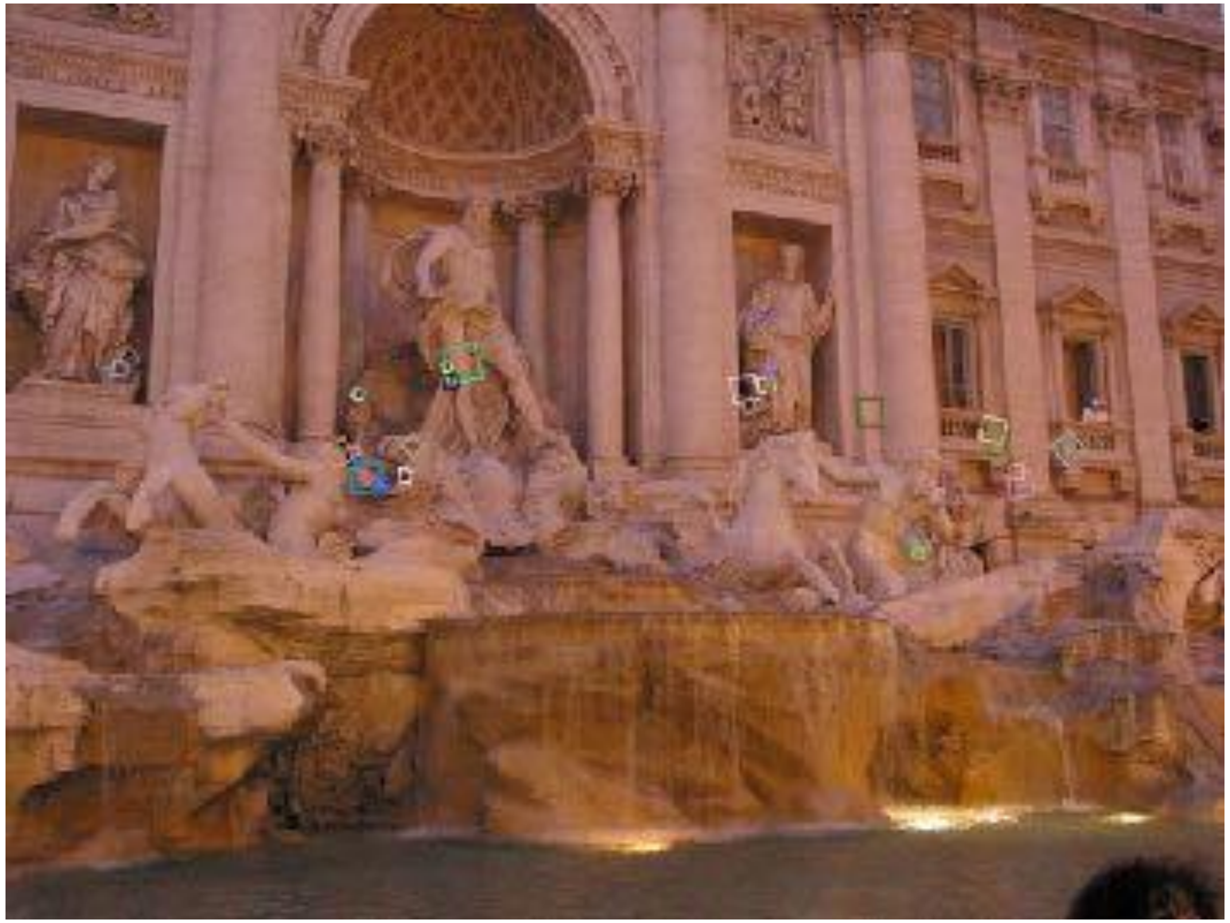
http://matthewalunbrown.com/autostitch/autostitch.html

Wide baseline stereo



Even robust to extreme occlusions





Applications of local invariant features

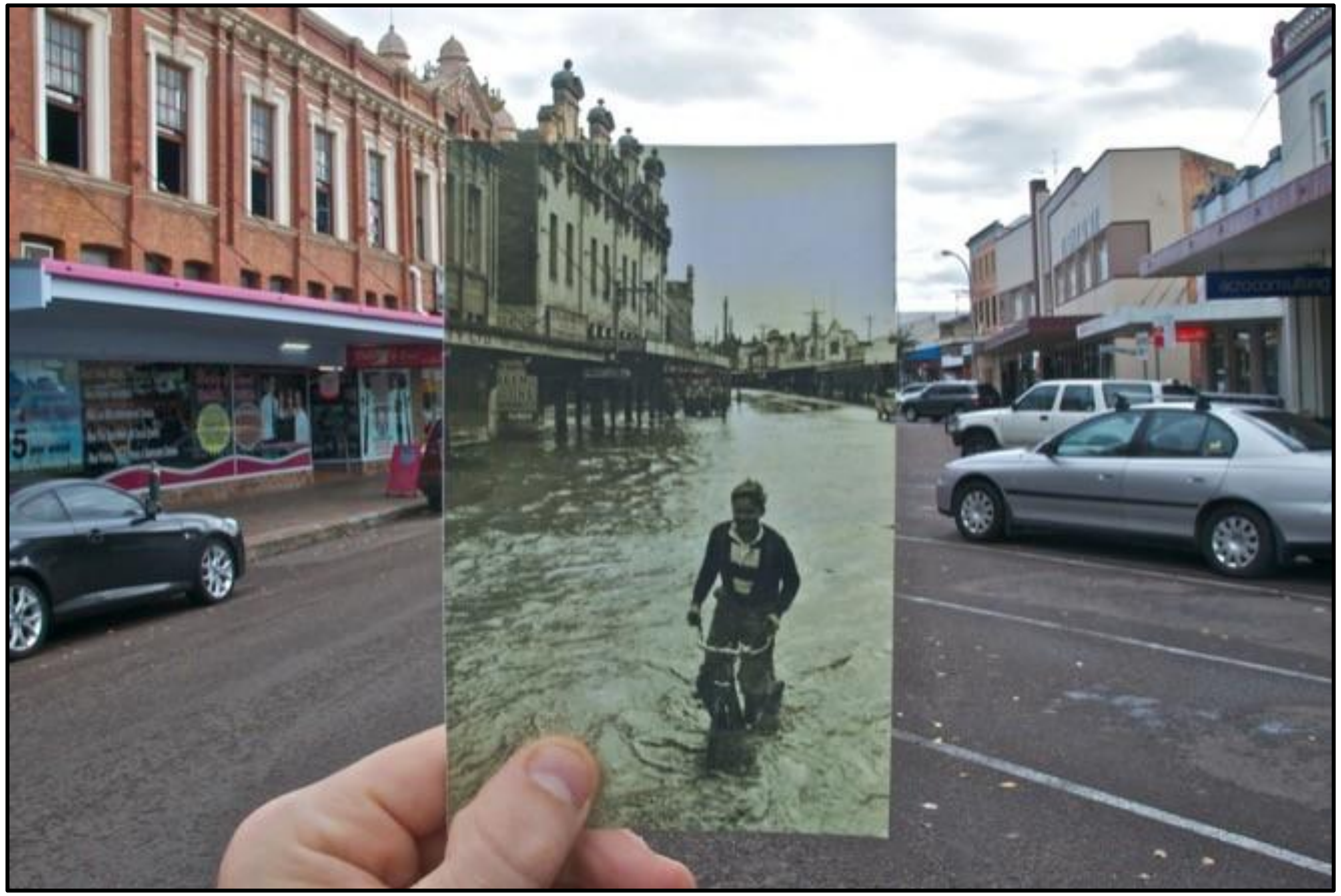
- Recognition
- Wide baseline stereo
- Panorama stitching
- Mobile robot navigation
- Motion tracking
- 3D reconstruction
- …

Today's agenda

- Local descriptors (SIFT)
  - Making keypoints rotation invariant
  - Designing a descriptor
  - Designing a matching function
- Image Homography
- Global descriptors (HoG)

Image homographies

a geometric transformation that maps points from one image plane to another



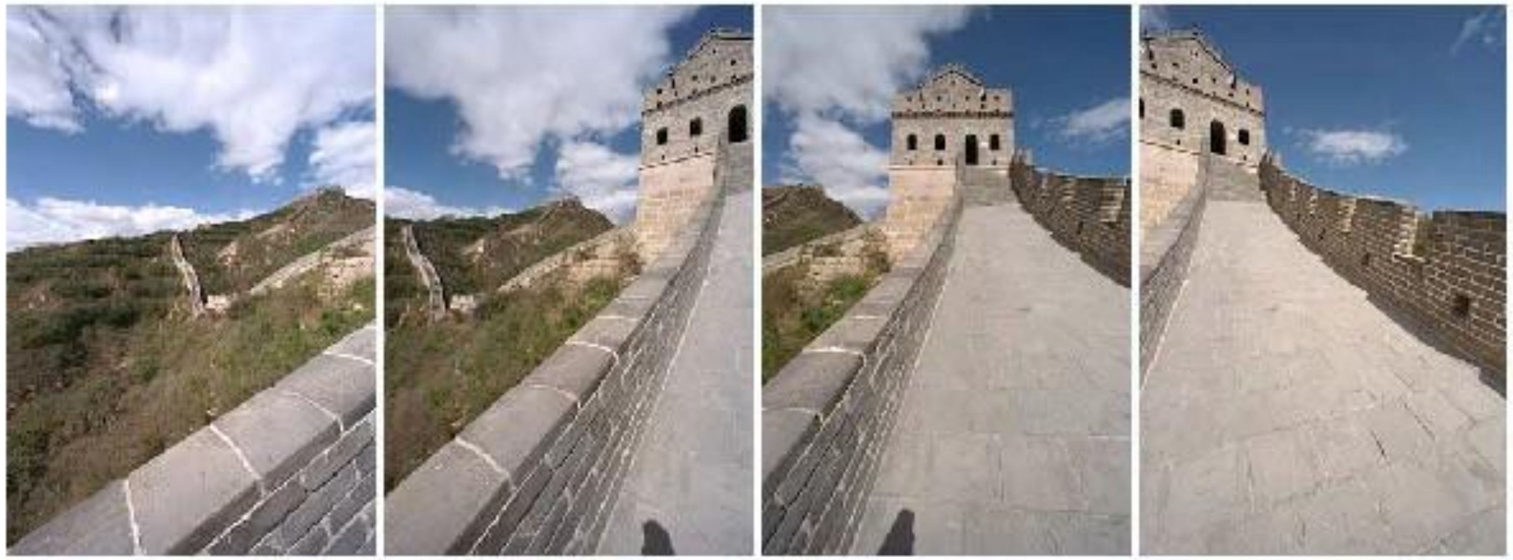
How do you create a panorama?

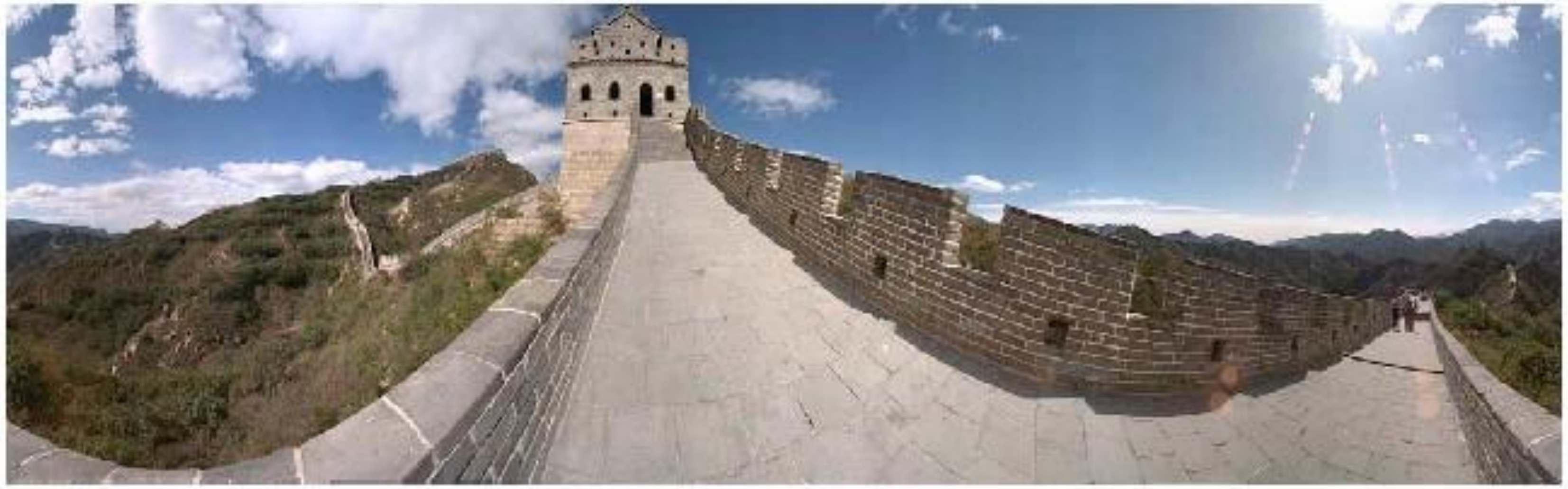
Panorama: an image of (near) 360o field of view.

How do you create a panorama?

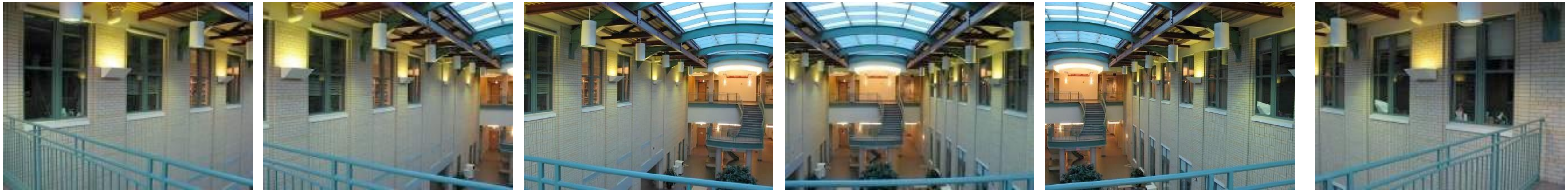
Could Use a very wide-angle lens.
Pros: Everything is done optically, single capture.
Cons: Lens is super expensive and bulky, lots of distortion (can be dealt-with in post).

Or you can capture multiple photos and combine them

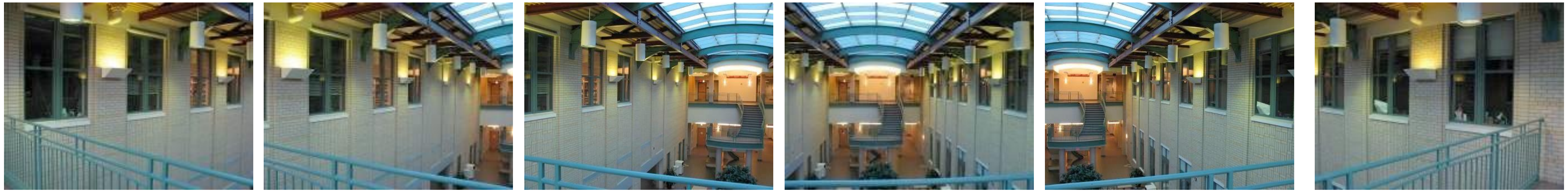




How do we stitch images from different viewpoints?

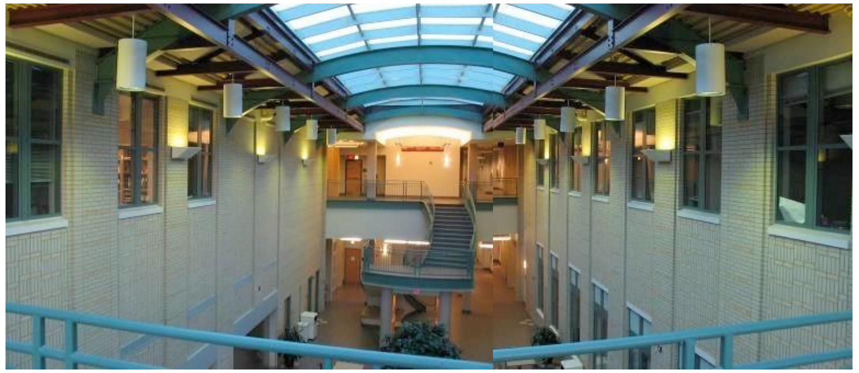


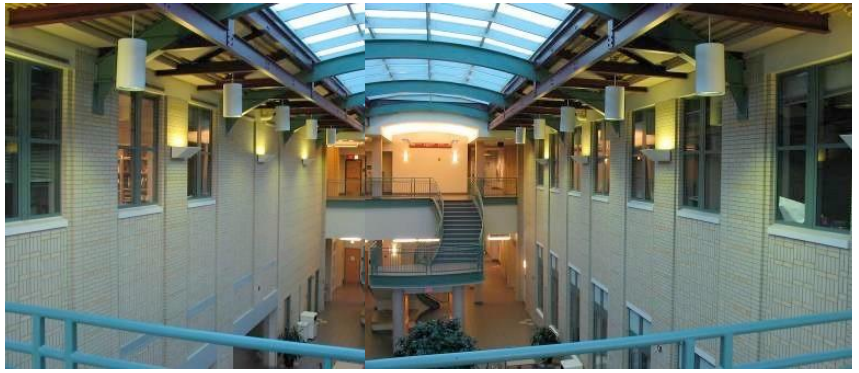
How do we stitch images from different viewpoints?



We can't simply place on on top of another.

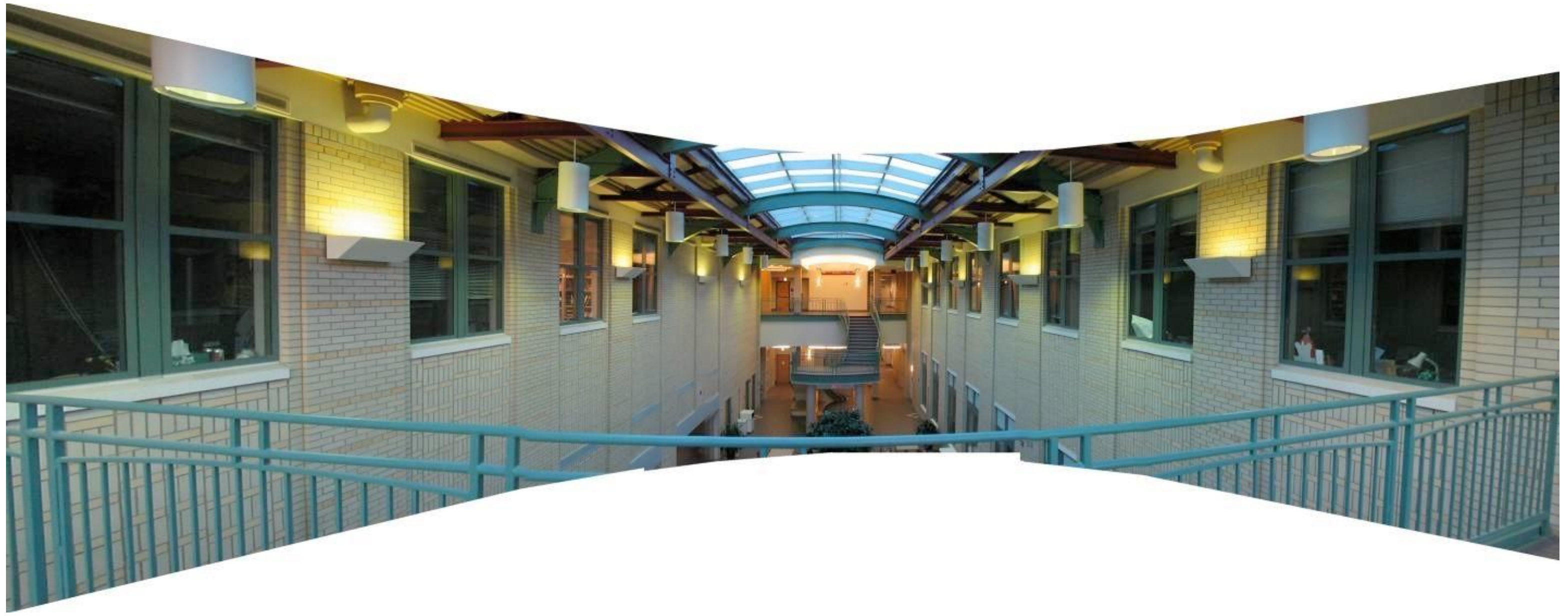
left on top



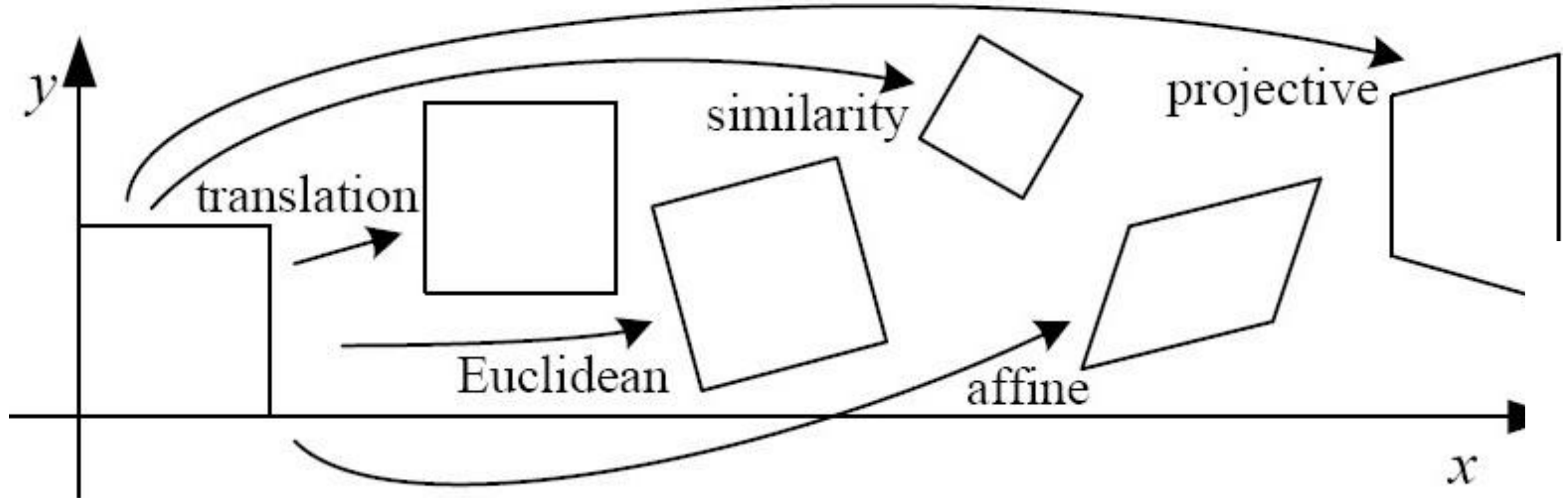


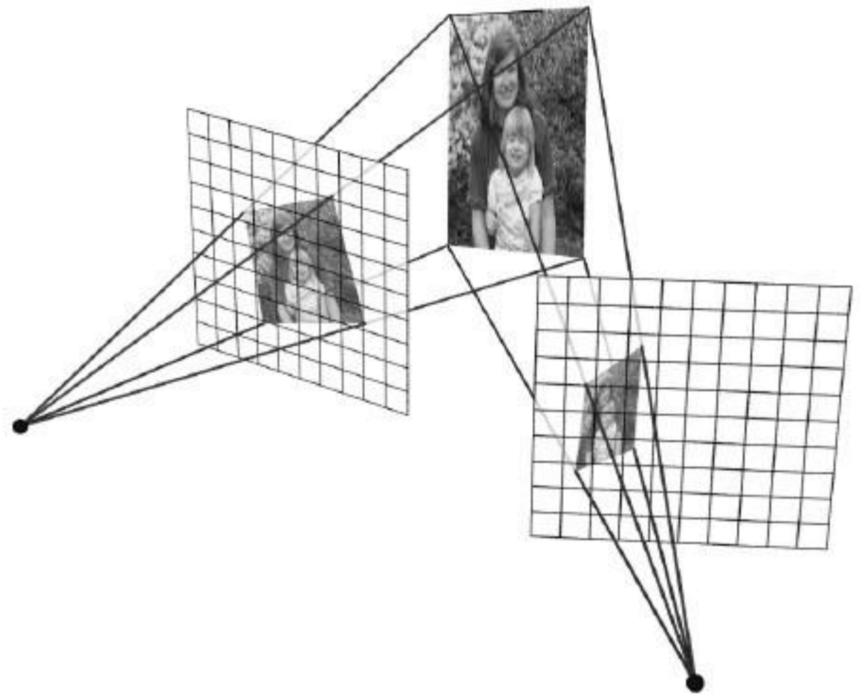
right on top

This is where homographies come in



Homographies explain how one image needs to be transformers

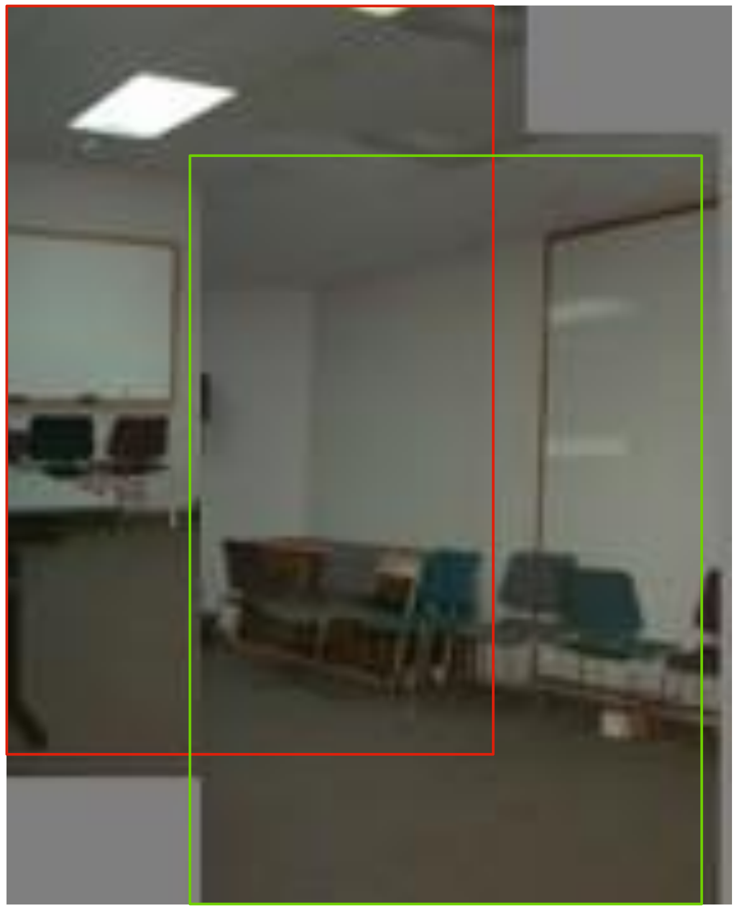




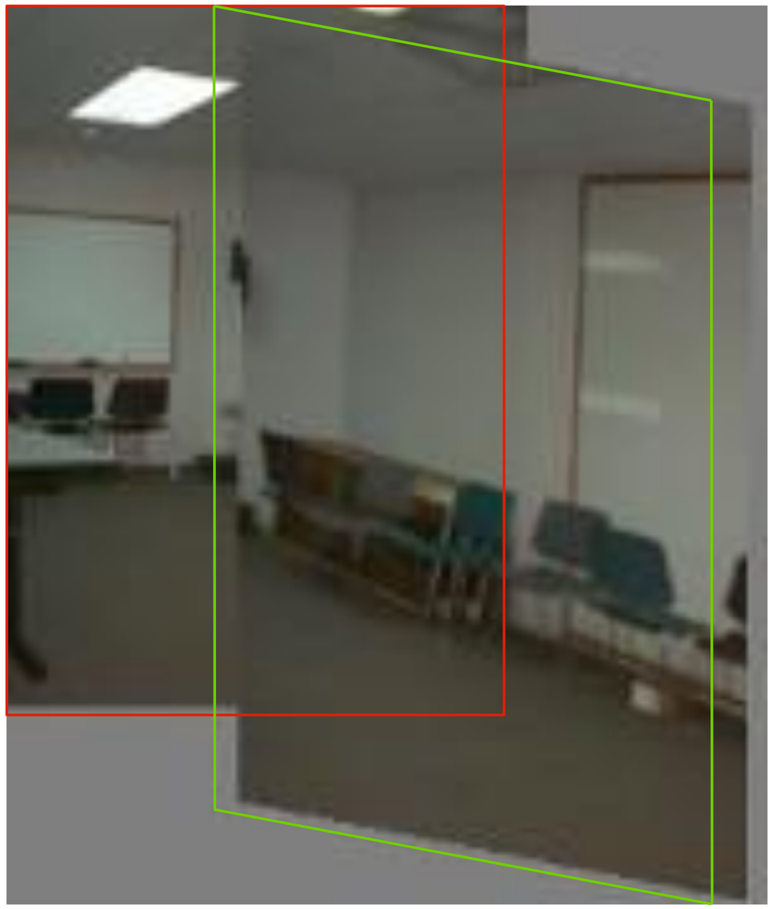
Q. Which kind of transformation is needed to warp projective plane 1 into projective plane 2?

Warping with different transformations

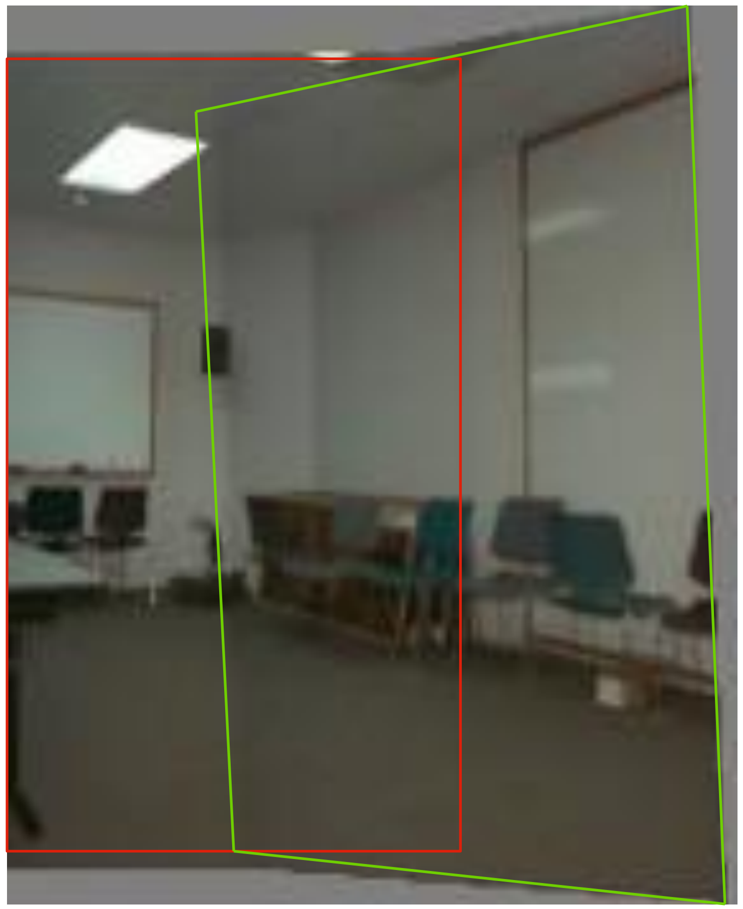
translation



affine

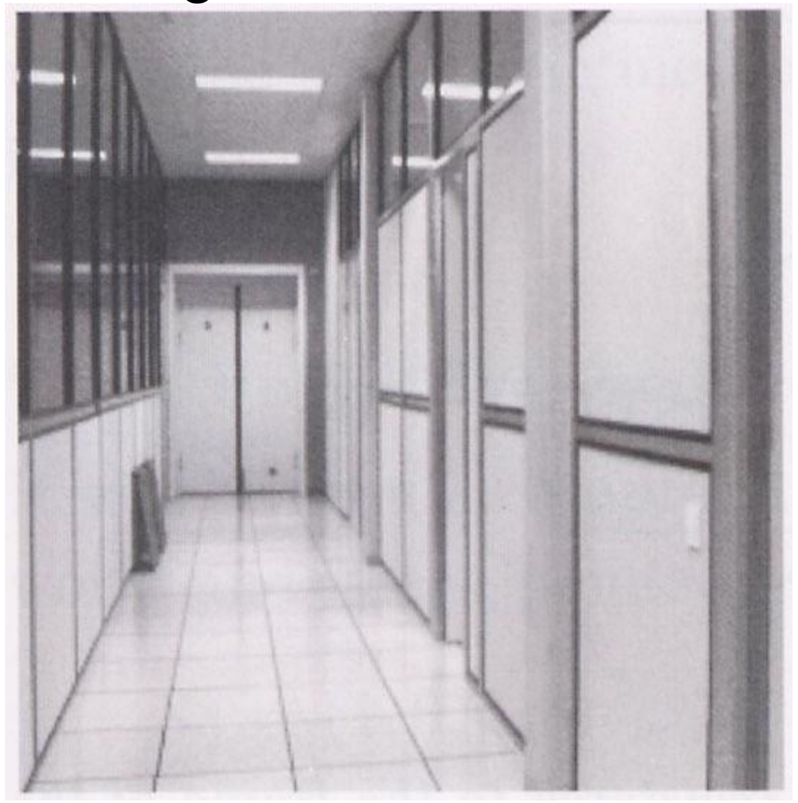


projective (homography)

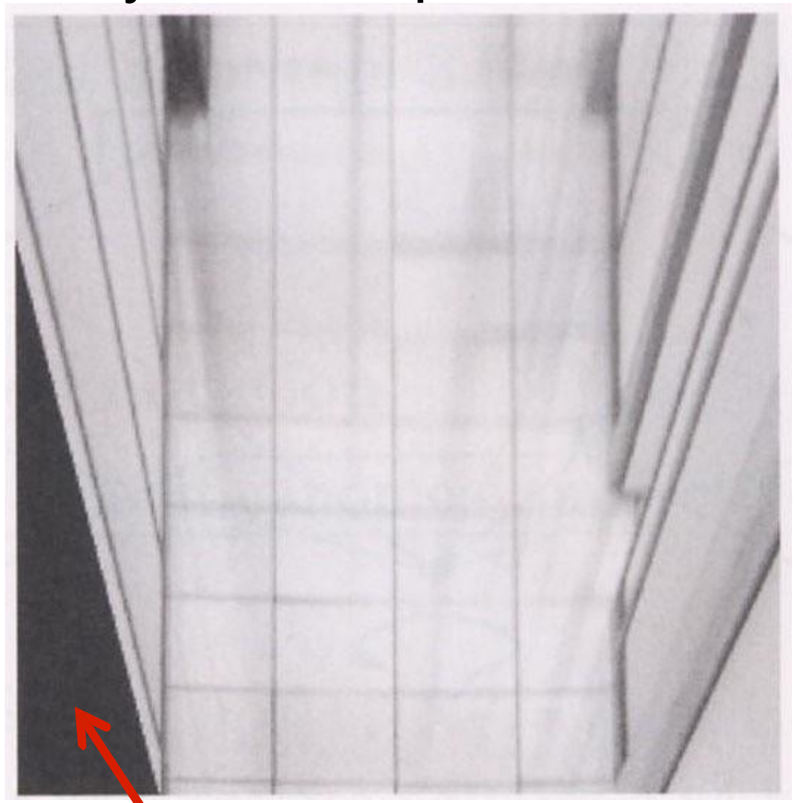


What happens when you transform one image to another view?

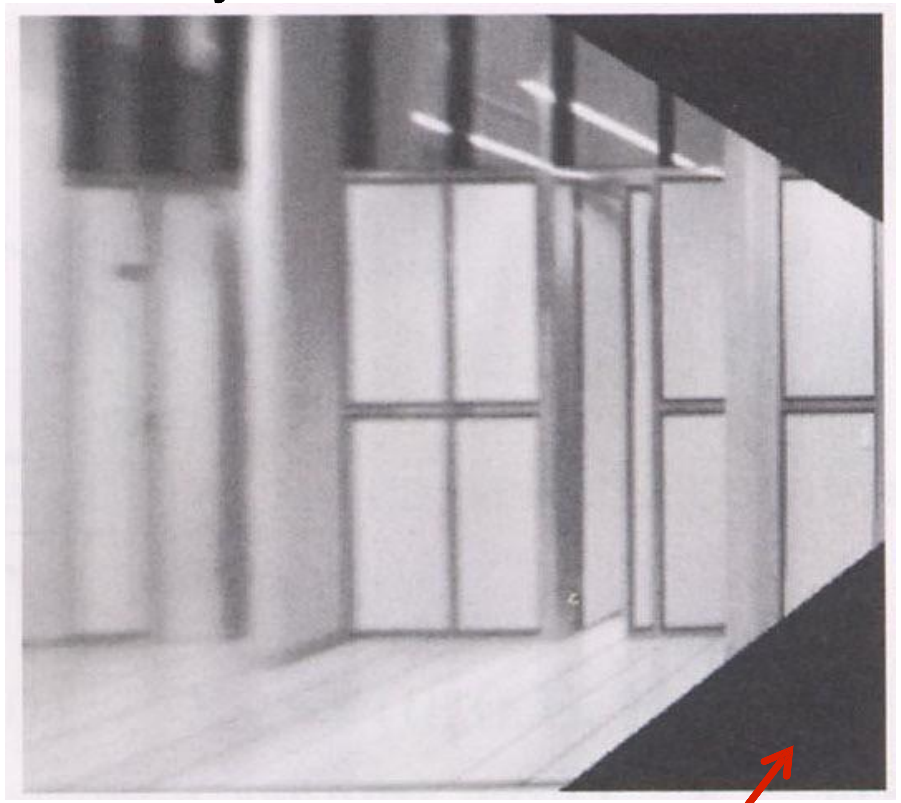
original view



synthetic top view

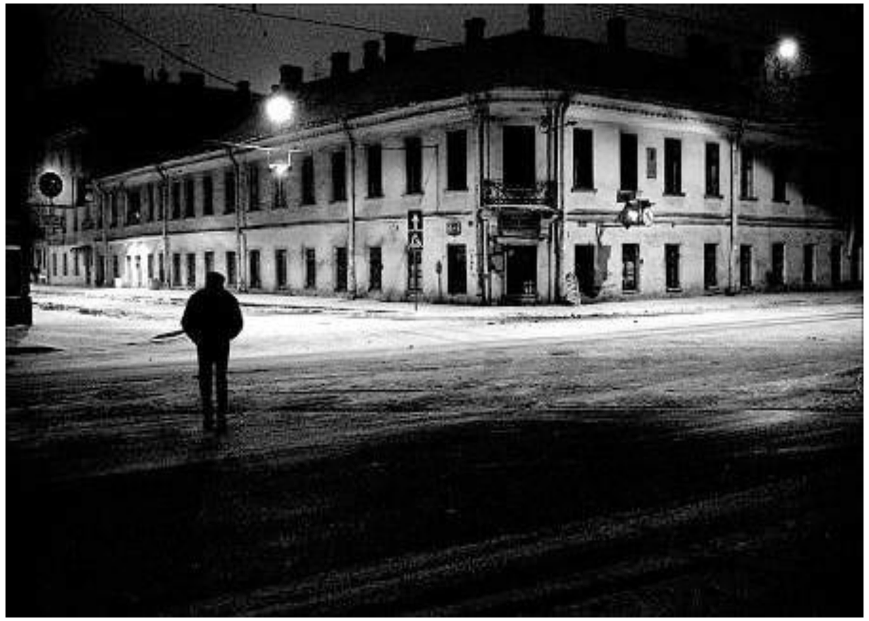


synthetic side view



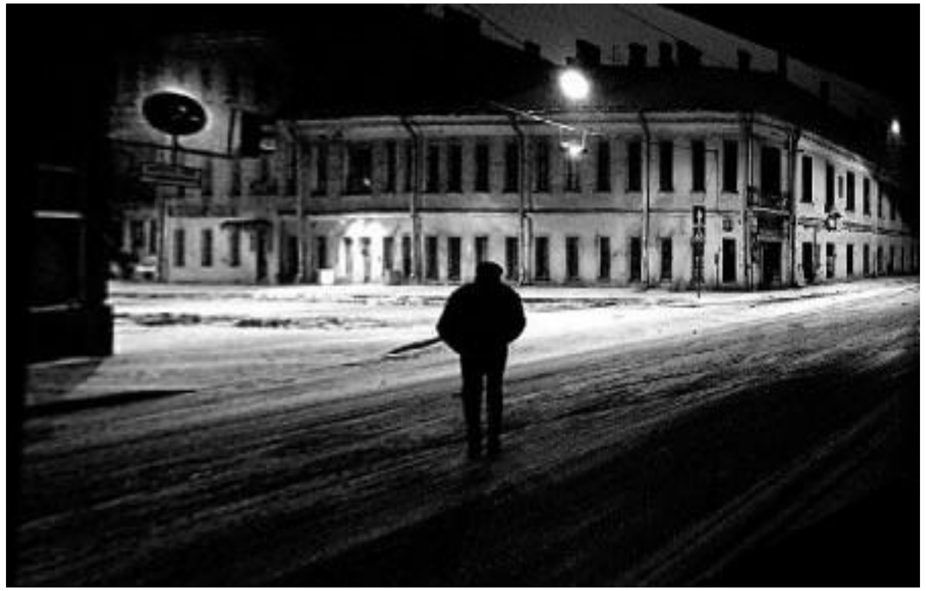
What are these black areas near the boundaries?

Virtual camera rotations



original view

synthetic rotations



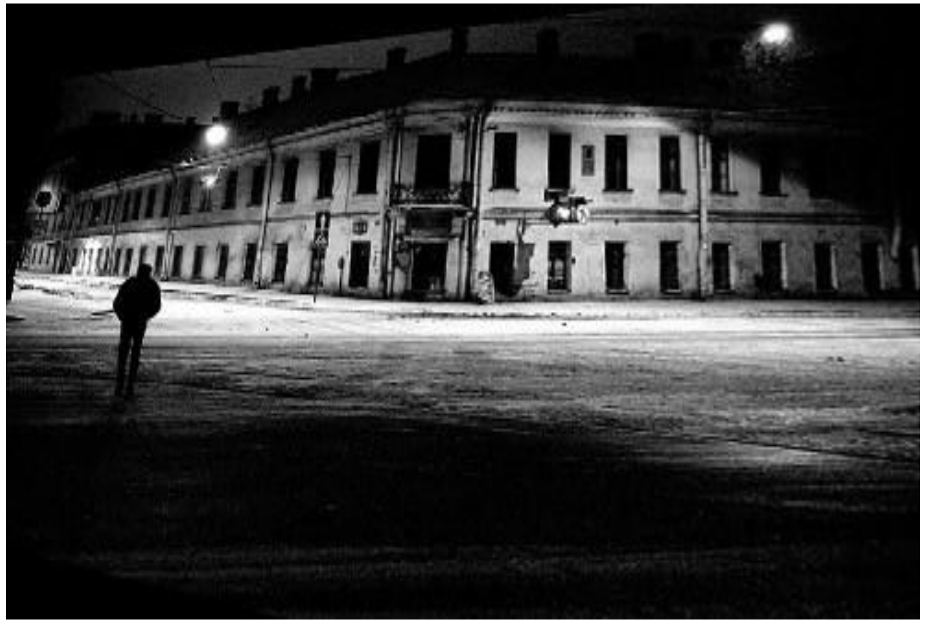
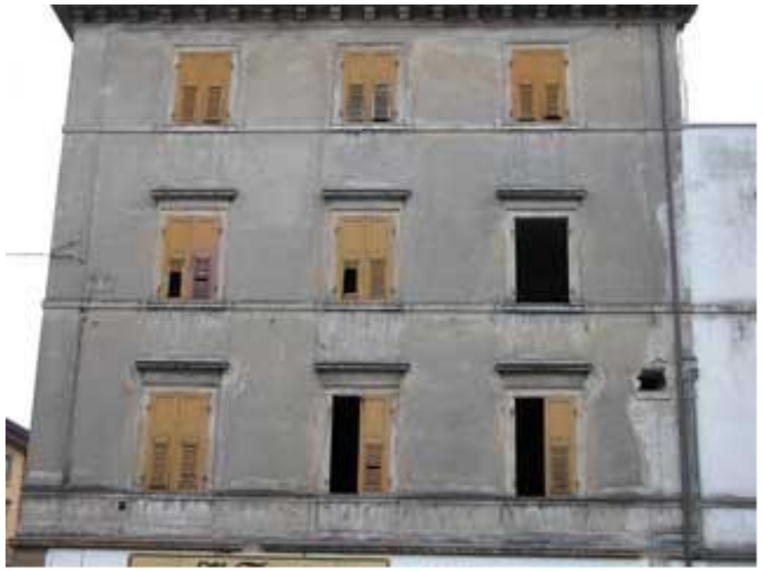
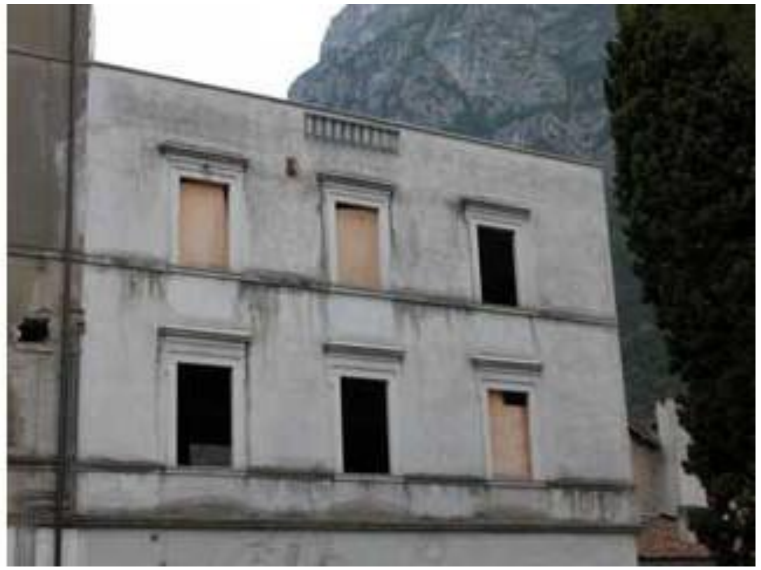
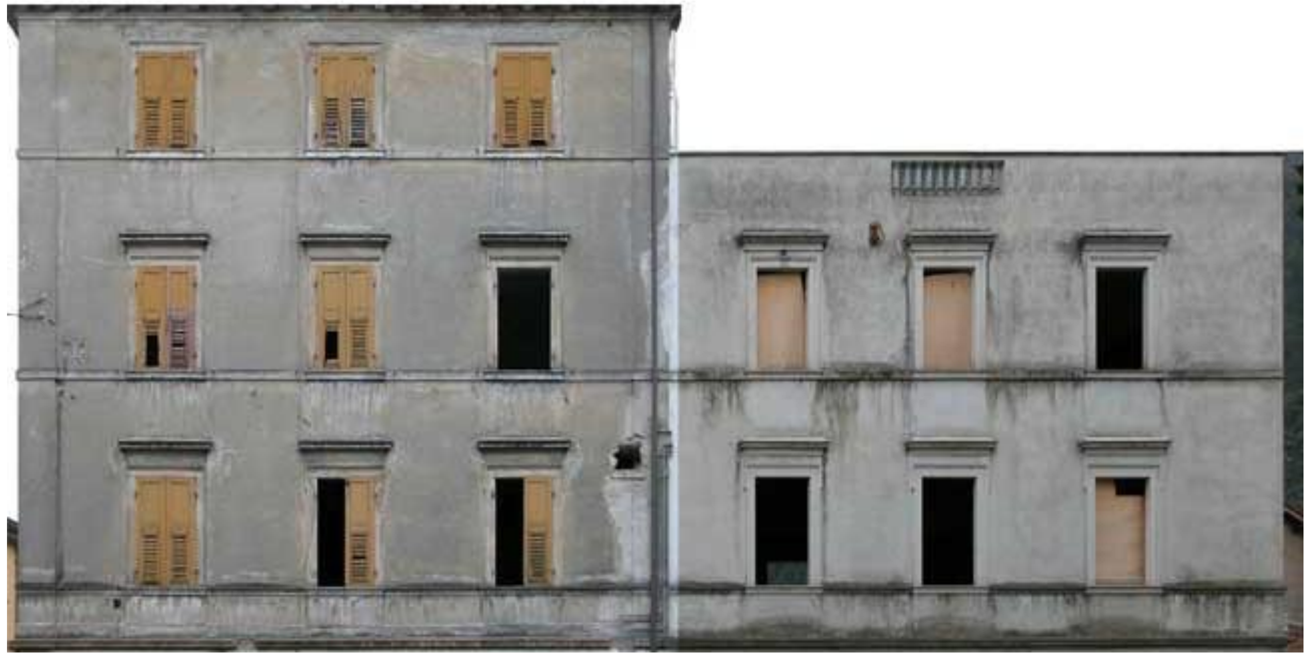


Image rectification

o
nal
ges

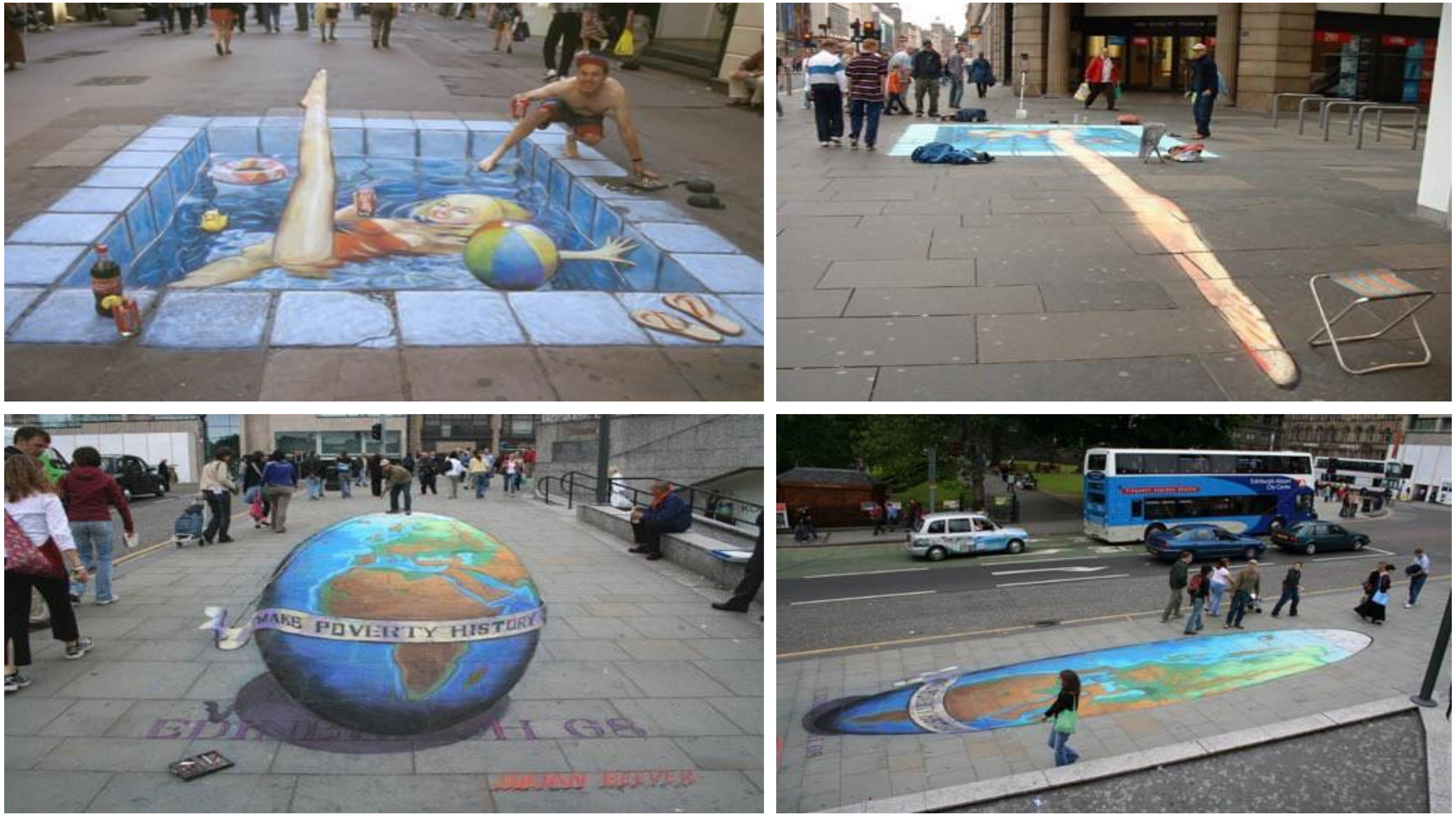




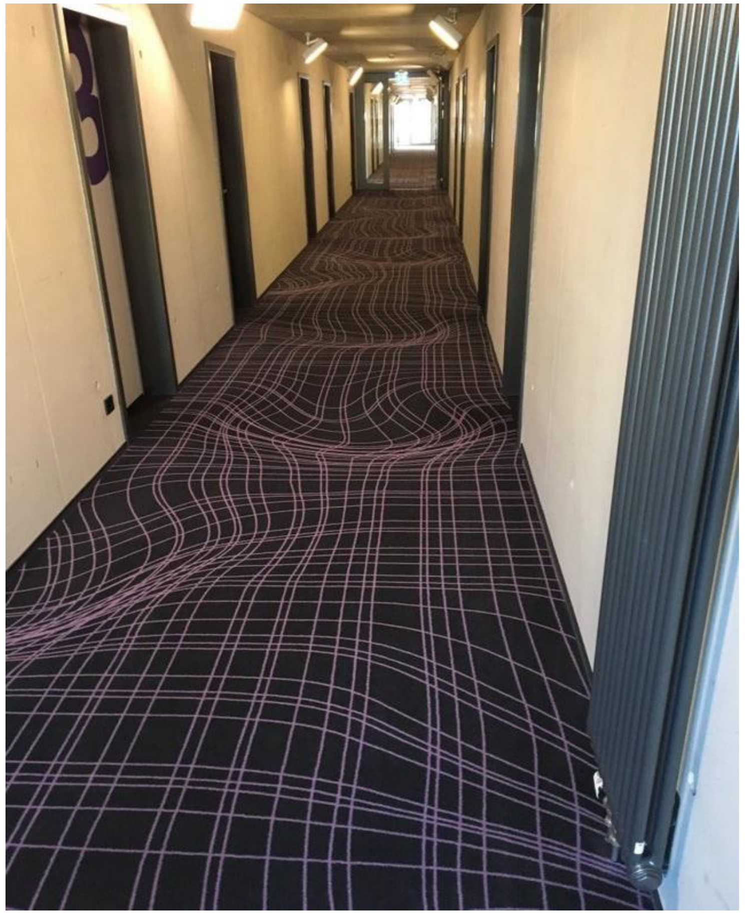


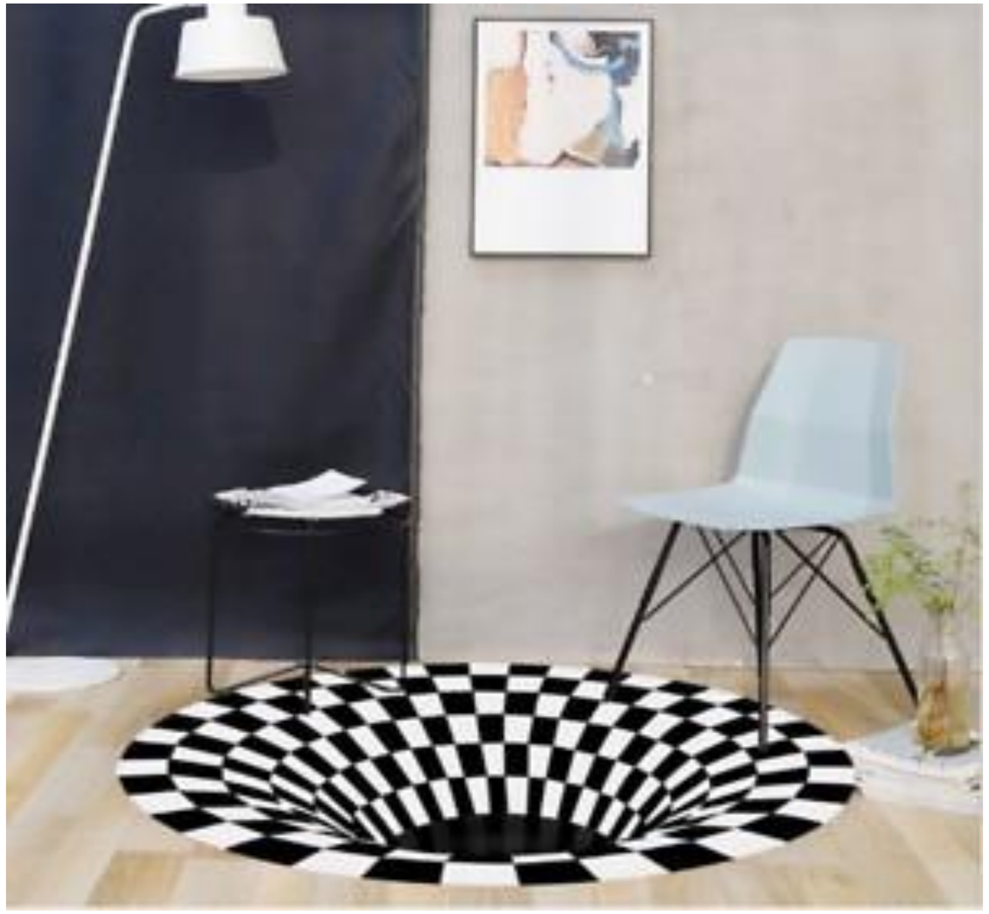
rectified and stitched

Street art



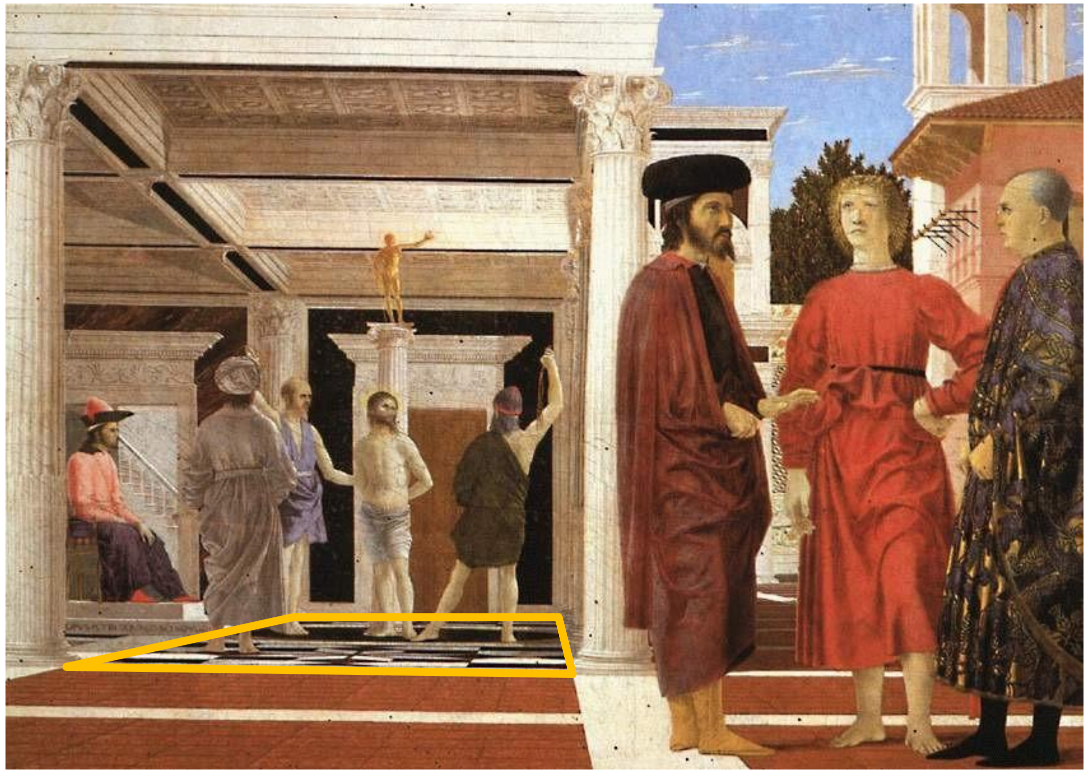
Carpet illusion





Understanding geometric patterns

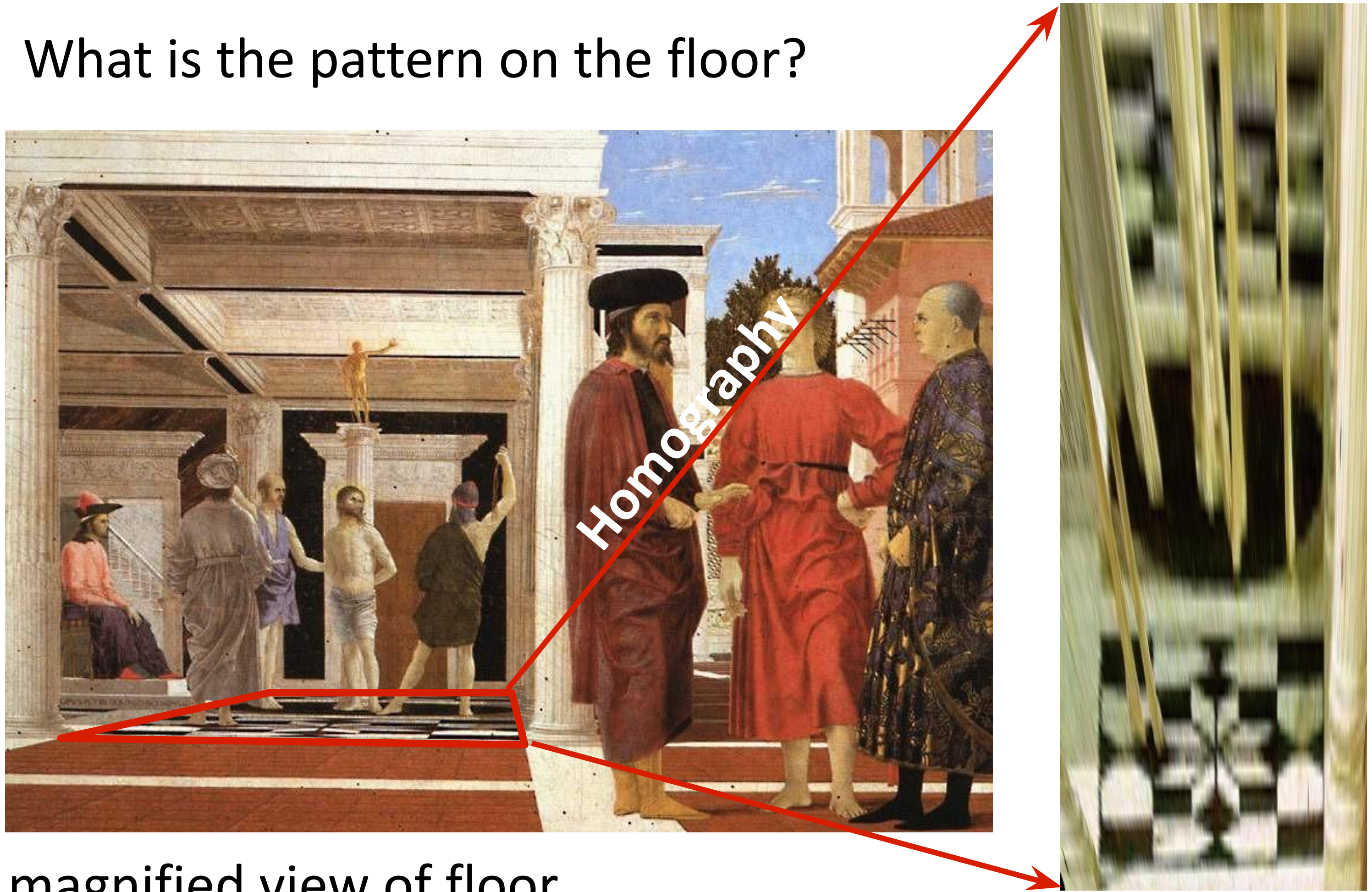
What is the pattern on the floor?



magnified view of floor

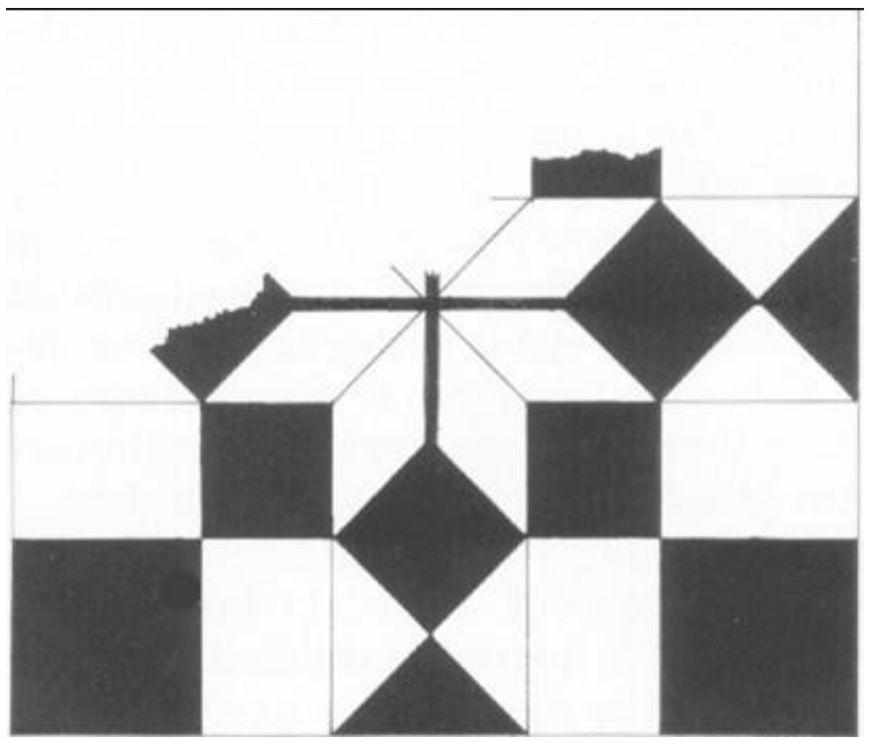
Understanding geometric patterns

What is the pattern on the floor?



magnified view of floor

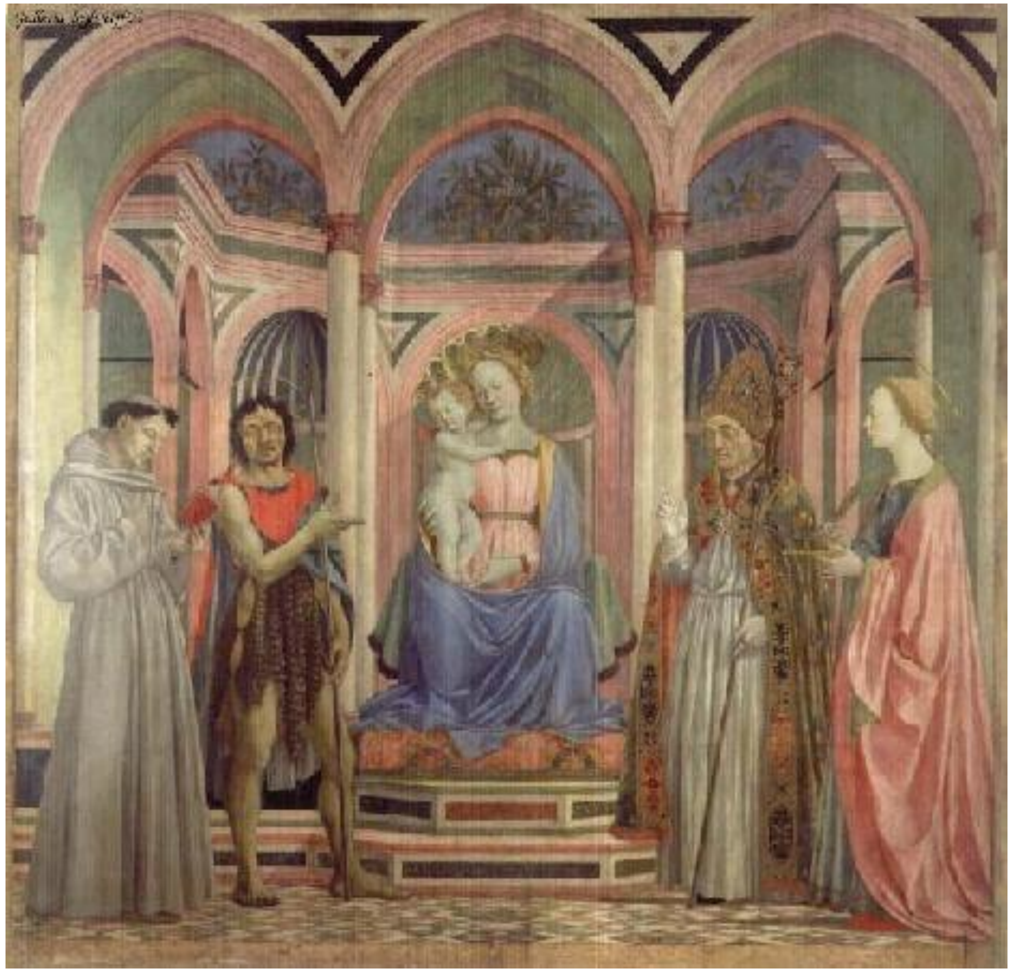
rectified view

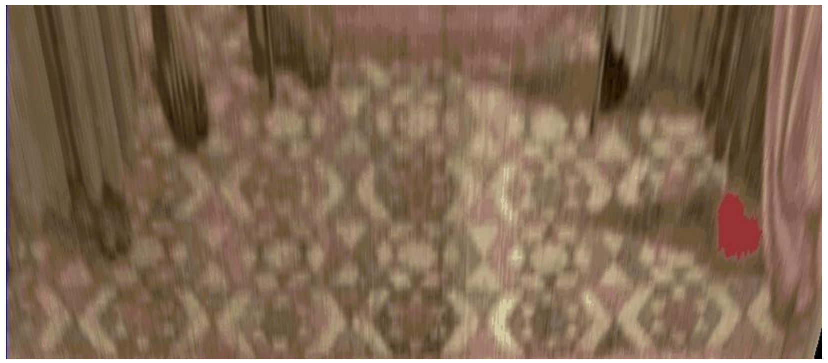


reconstruction from rectified view

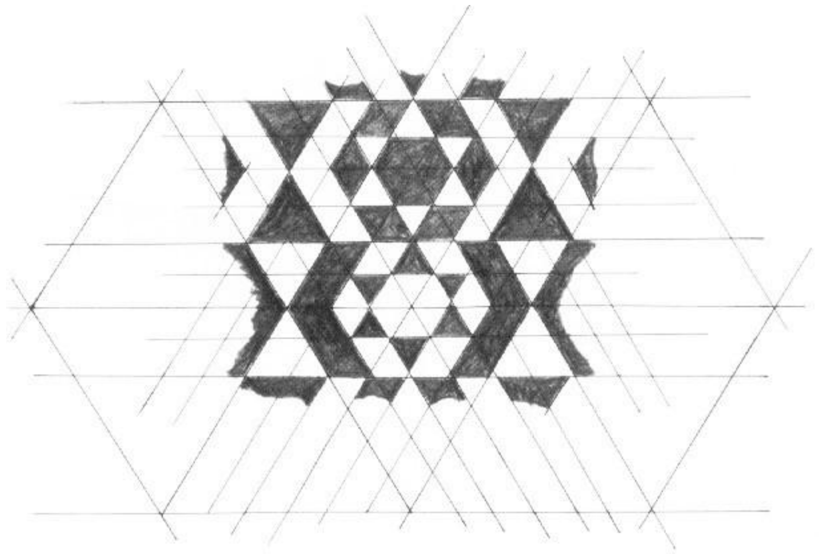
Understanding geometric patterns

What is the pattern on the floor?



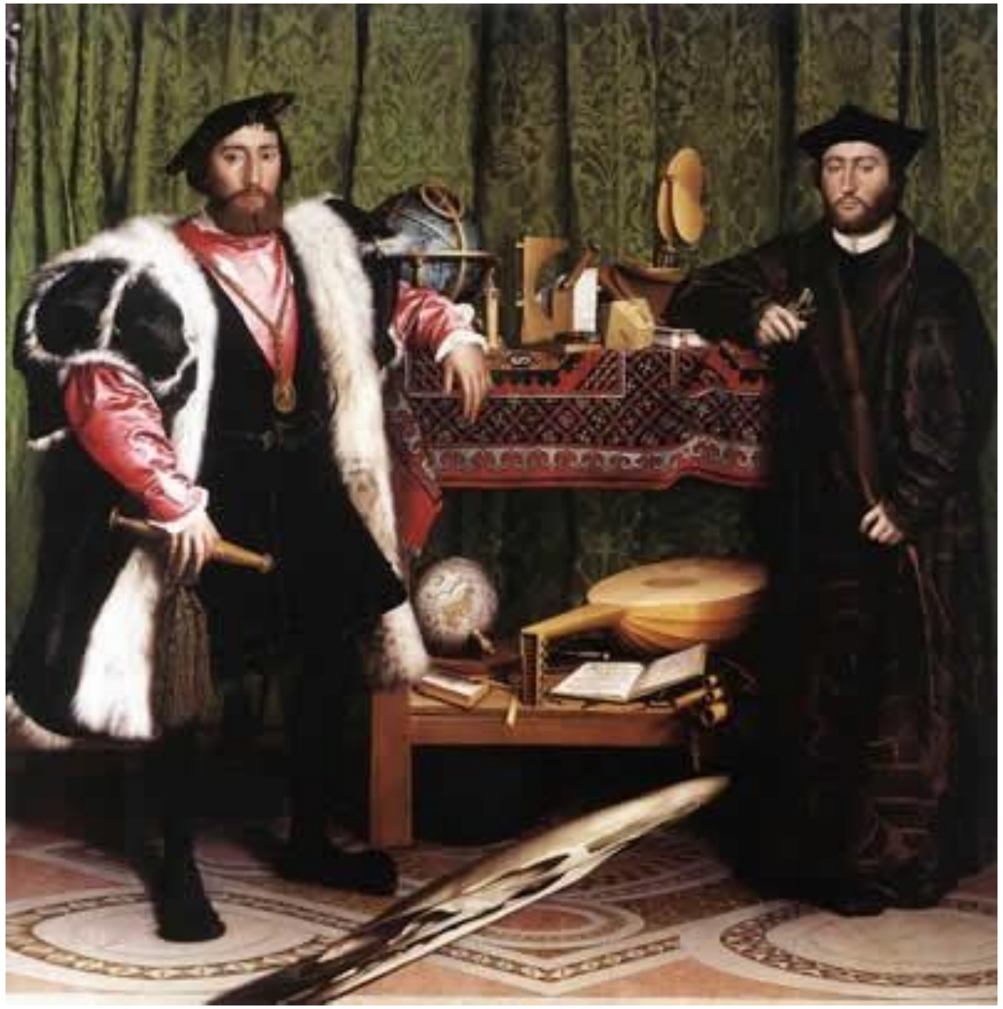


rectified view of floor



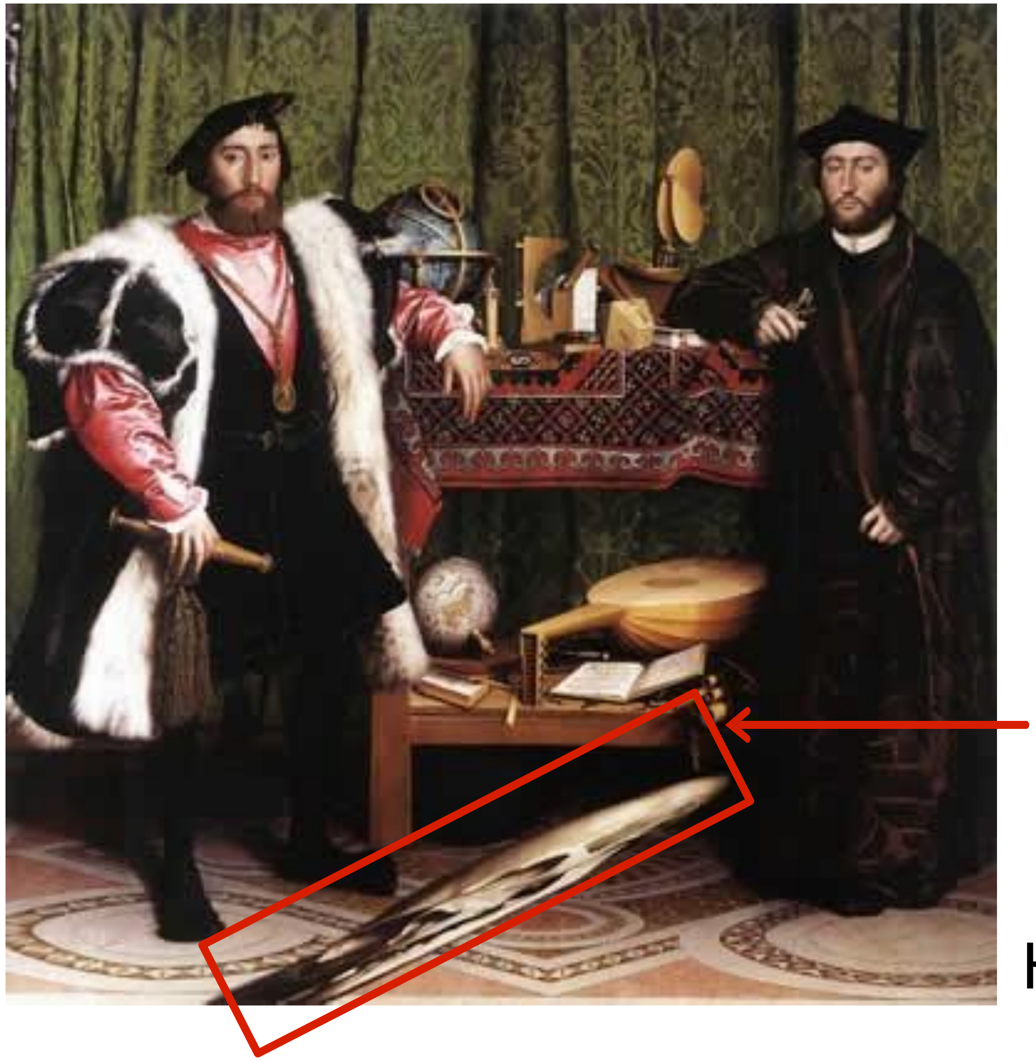
reconstruction

A weird painting



Holbein, “The Ambassadors”

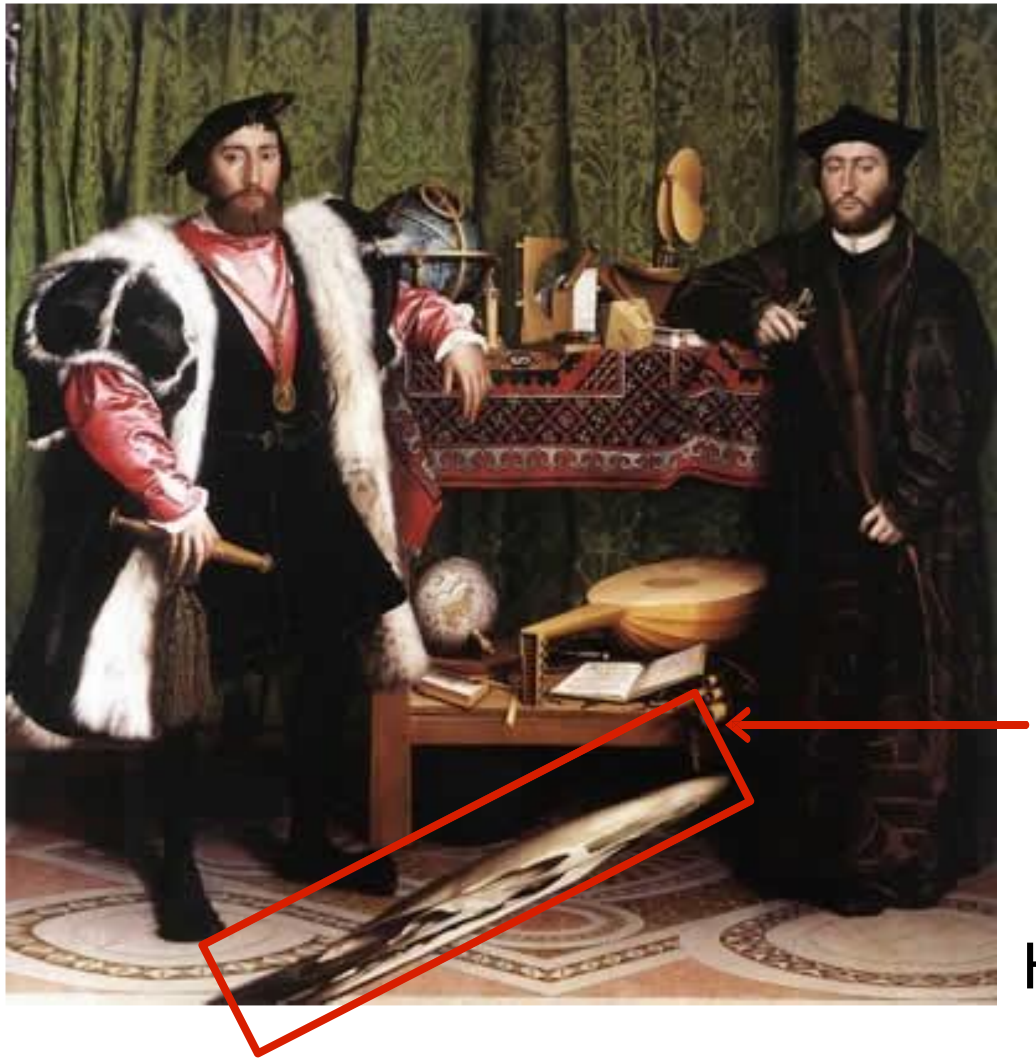
A weird painting



What's this???

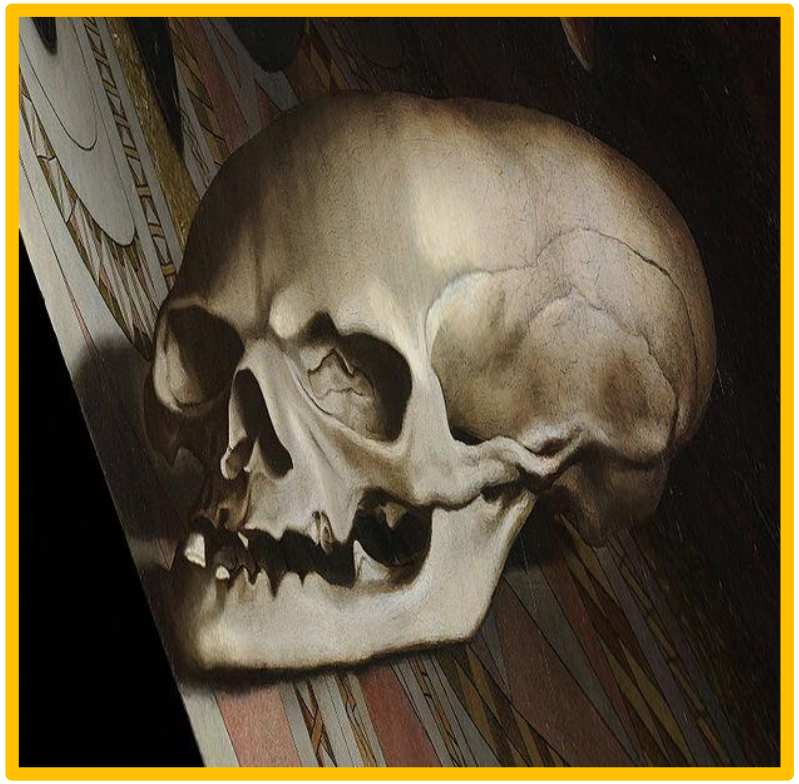
Holbein, "The Ambassadors"

A weird painting



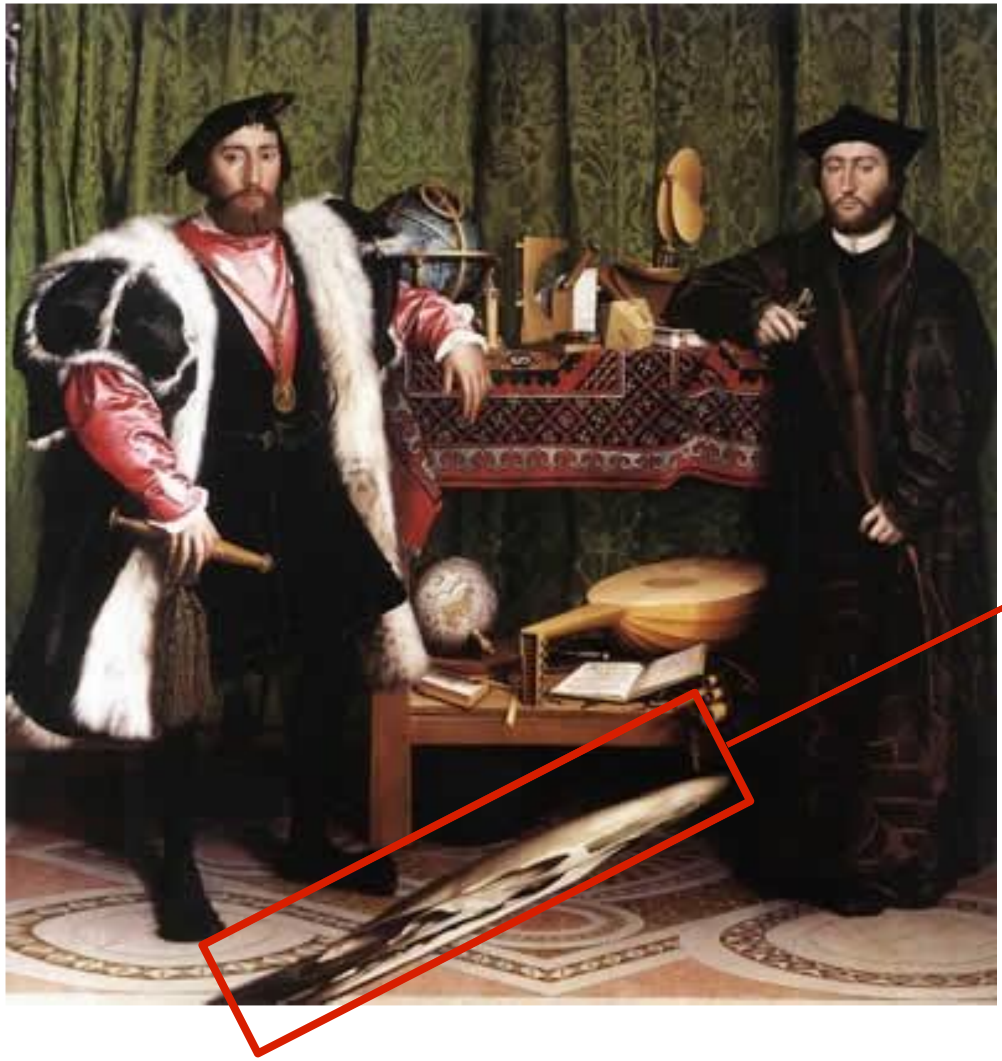
rectified viewskull under anamorphic perspective

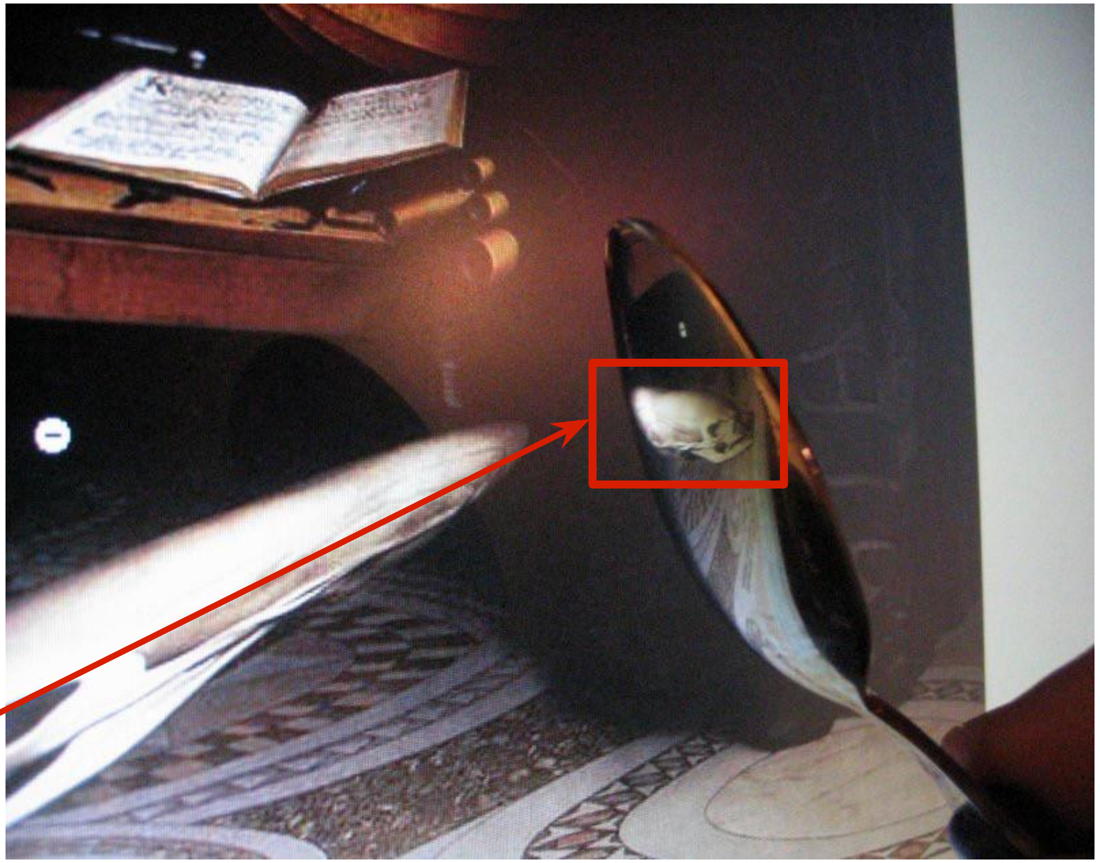
Holbein, “The Ambassadors”



rectified view

A weird painting





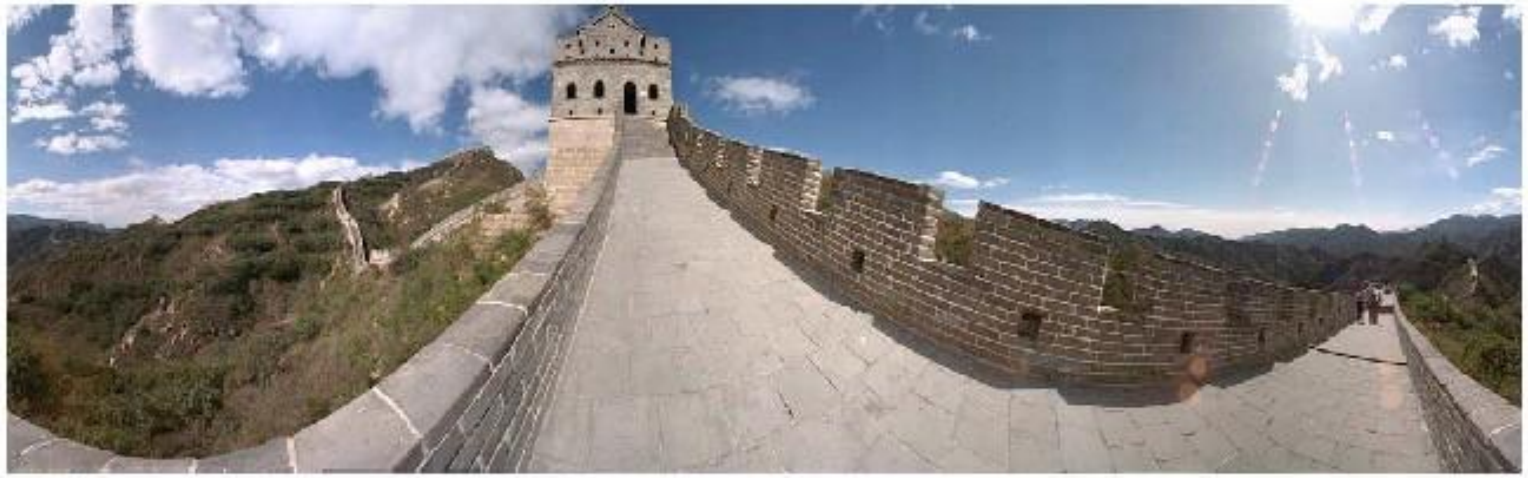
DIY: use a polished spoon to see the skull

Holbein, “The Ambassadors”

What we will focus on: Panoramas

1. Capture multiple images from different viewpoints.

2. Stitch them together into a virtual wide-angle image.



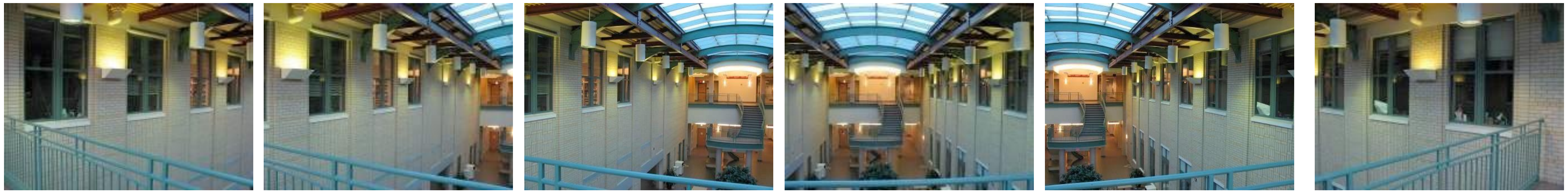
When can we calculate homographies?

when the scene is planar;
or

when the scene is very far or has small (relative) depth variation
→ scene is approximately planar

When can we calculate homographies?

when the scene is captured under camera rotation only (no translation or pose change)

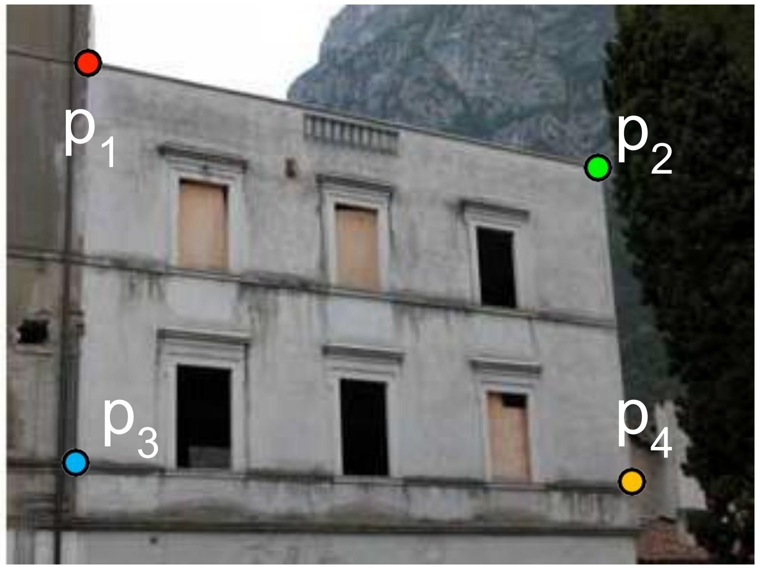


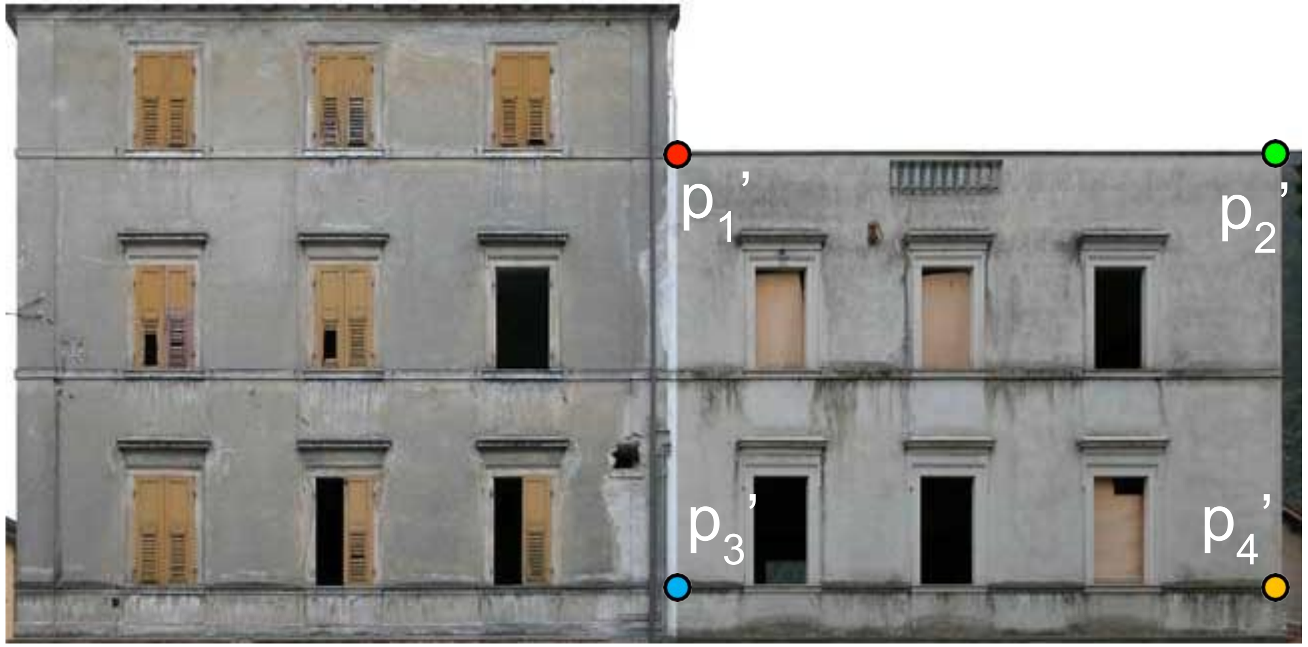
More on why this is the case in a later lecture.

How do we do it? Keypoint matching!

The homography matrix H!

$$P' = H \cdot P$$





original image target image

How many correspondences do we need?

Determining the homography matrix

$$P^{!} = H \cdot P$$

Determining the homography matrix

Write out linear equation for each correspondence:

$$P^! = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Determining the homography matrix

Write out linear equation for each correspondence:

$$P^{!} = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ \boxed{1} \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ \boxed{1} \end{bmatrix}$$

Q. Why is there a 1 here?

Determining the homography matrix

Write out linear equation for each correspondence:

$$P^! = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ \boxed{1} \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ \boxed{1} \end{bmatrix}$$

Q. Why is there a 1 here?

Homogenous coordinates:

More common to use w.

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Determining the homography matrix

Write out linear equation for each correspondence:

$$P^{!} = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ \boxed{1} \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ \boxed{1} \end{bmatrix}$$

Q. Why is there a 1 here?

Homogenous coordinates:

More common to use w.

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

The output x' and y' in image space is found by: $x' = \frac{x'}{w'}, \quad y' = \frac{y'}{w'}$

Determining the homography matrix

Write out linear equation for each correspondence:

$$P^! = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ \boxed{1} \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ \boxed{1} \end{bmatrix}$$

Q. Why is there a 1 here?

Homogenous coordinates:

More common to use w’

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Q. What can you say about points where w’ = 0?

Determining the homography matrix

Write out linear equation for each correspondence:

$$P^! = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Q. Why is there an alpha there?

Determining the homography matrix

Write out linear equation for each correspondence:

$$P^{!} = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Expand matrix multiplication:

$$x' = \alpha(h_1 x + h_2 y + h_3)$$
$$y' = \alpha(h_4 x + h_5 y + h_6)$$
$$1 = \alpha(h_7 x + h_8 y + h_9)$$

Determining the homography matrix

Write out linear equation for each correspondence:

$$P^! = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Expand matrix multiplication:

$$x' = \alpha(h_1 x + h_2 y + h_3)$$
$$y' = \alpha(h_4 x + h_5 y + h_6)$$
$$1 = \alpha(h_7 x + h_8 y + h_9)$$

Divide out unknown scale factor:

$$x'(h_7 x + h_8 y + h_9) = (h_1 x + h_2 y + h_3)$$
$$y'(h_7 x + h_8 y + h_9) = (h_4 x + h_5 y + h_6)$$

Ok so we have 2 equations and 9 unknowns!

$$x'(h_7x + h_8y + h_9) = (h_1x + h_2y + h_3)$$
$$y'(h_7x + h_8y + h_9) = (h_4x + h_5y + h_6)$$

Let’s rearrange the terms:

$$h_7xx' + h_8yx' + h_9x' - h_1x - h_2y - h_3 = 0$$
$$h_7xy' + h_8yy' + h_9y' - h_4x - h_5y - h_6 = 0$$

Same equations from previous slide:

$$h_7xx' + h_8yx' + h_9x' - h_1x - h_2y - h_3 = 0$$
$$h_7xy' + h_8yy' + h_9y' - h_4x - h_5y - h_6 = 0$$

Re-write in matrix form:

$$\mathbf{A}_i \boldsymbol{h} = \mathbf{0}$$

Same equations from previous slide:

$$h_7 x x' + h_8 y x' + h_9 x' - h_1 x - h_2 y - h_3 = 0$$
$$h_7 x y' + h_8 y y' + h_9 y' - h_4 x - h_5 y - h_6 = 0$$

Re-write in matrix form:

$$\mathbf{A}_i \boldsymbol{h} = \mathbf{0}$$

$$\mathbf{A}_i = \left[\begin{array}{ccccccccc} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{array} \right]$$

$$\boldsymbol{h} = \left[\begin{array}{ccccccccc} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 \end{array} \right]^\top$$

What is this form useful?

$$\mathbf{A}\boldsymbol{h} = \mathbf{0}$$

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

We get 2 rows per matching keypoint

Stack together constraints from multiple point correspondences:

$$\mathbf{A}\boldsymbol{h} = \mathbf{0}$$

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$
$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$
$$\vdots$$
$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

This is called the *Homogeneous* linear least squares problem

Q. Do you remember this equation from your linear algebra course?

$$\mathbf{A}\boldsymbol{h} = \mathbf{0}$$

$$\begin{bmatrix} \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \\ \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \\ \vdots \\ \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

This is called the *Homogeneous* linear least squares problem

We can solve this using SVD

SVD decomposition: $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$

h parameters are the eigenvector in V associated with the smallest eigenvalue in $\boldsymbol{\Sigma}$

$$\boldsymbol{h} = \boldsymbol{v}_{\hat{i}}$$

Putting it all together to create a panorama

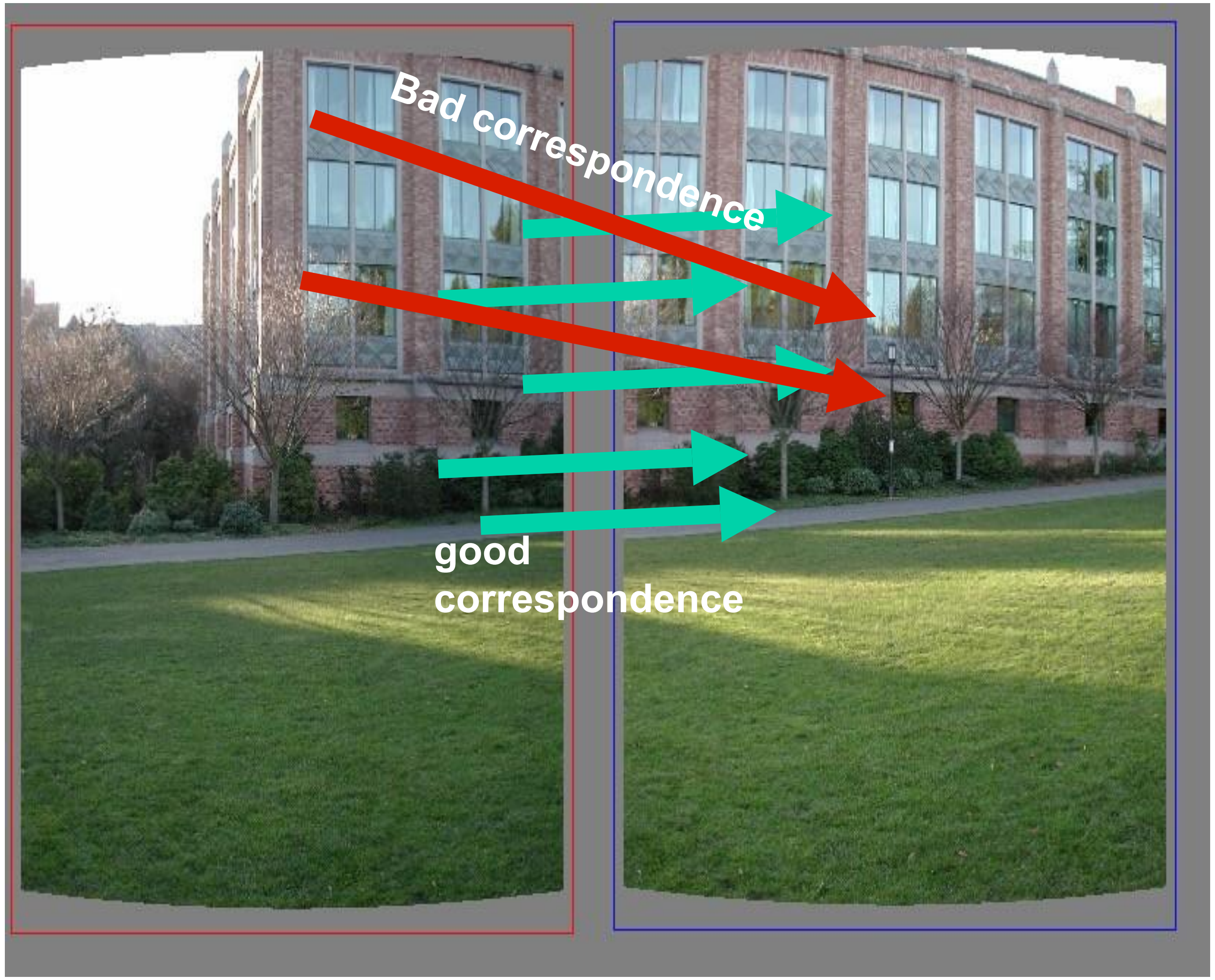
1. Find keypoints using SIFT or Harris corner
2. Find matches using local feature descriptors
3. Put all the matching points in the matrix form in the previous slide
4. Use SVD to solve for homography matrix h

Putting it all together to create a panorama

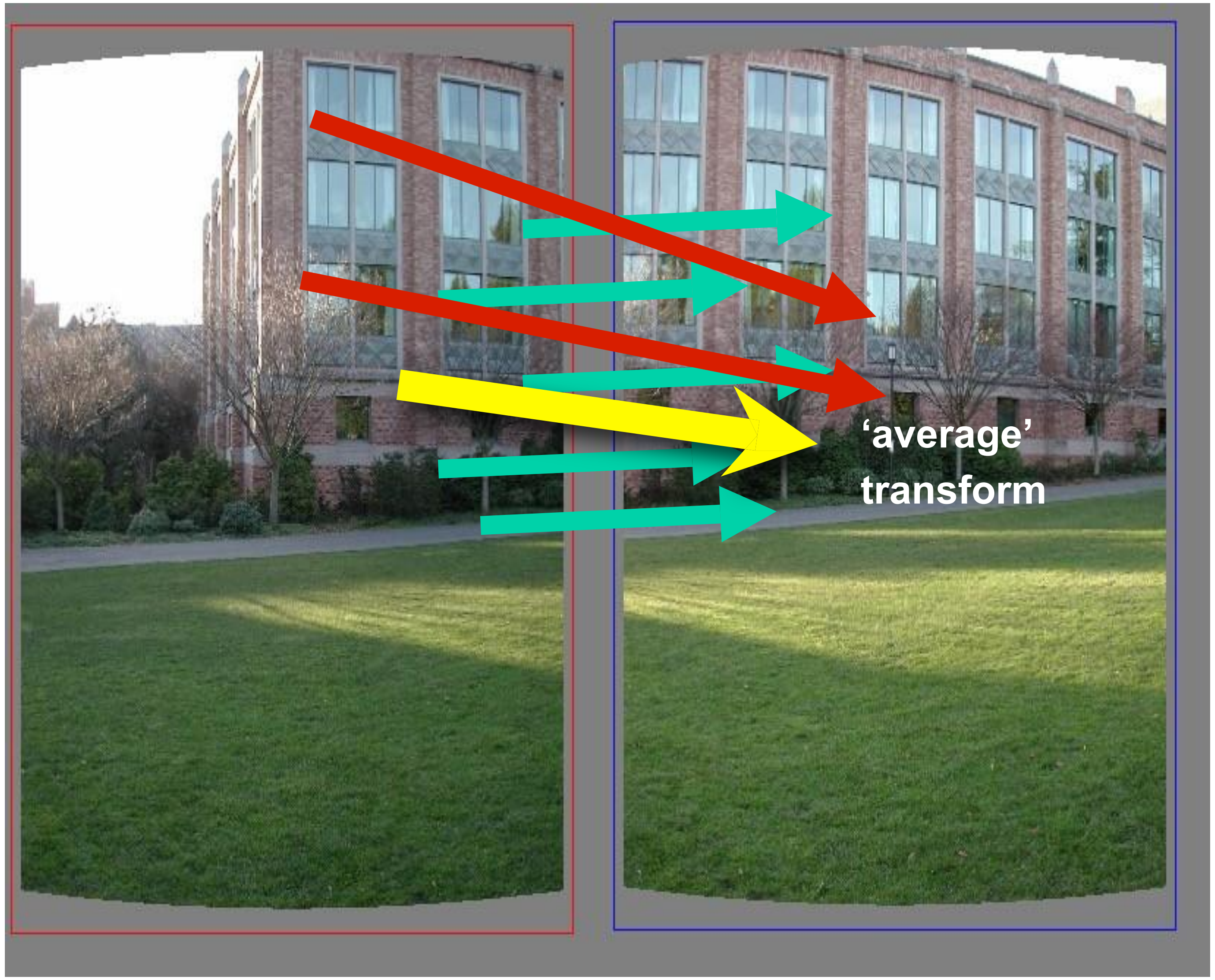
1. Find keypoints using SIFT or Harris corner
2. Find matches using local feature descriptors
3. Put all the matching points in the matrix form in the previous slide
4. Use SVD to solve for homography matrix h

Q. But wait, what if the keypoints are noisy and you have some bad matches?

Won't that give you a bad homography???

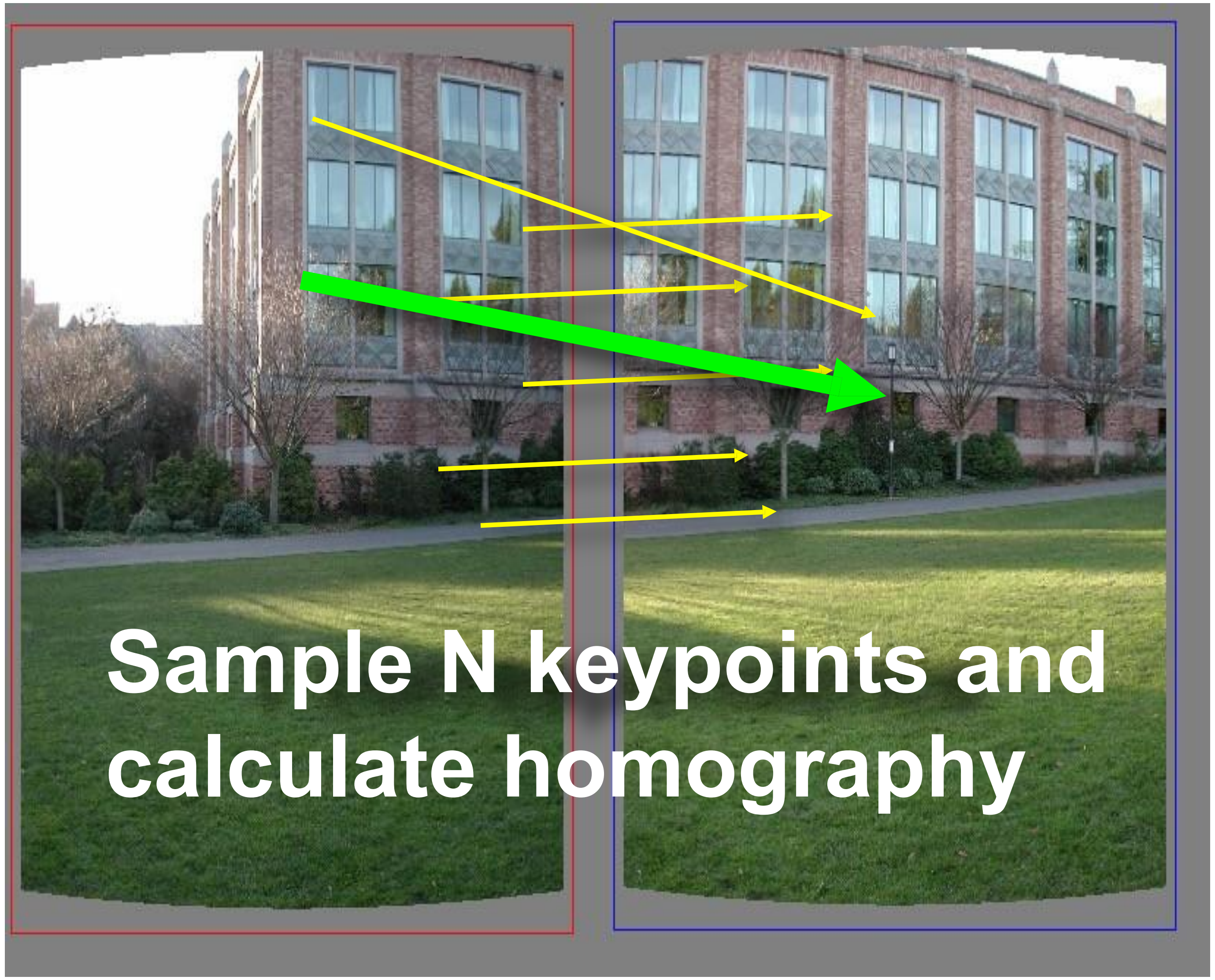

Bad correspondence
good
correspondence

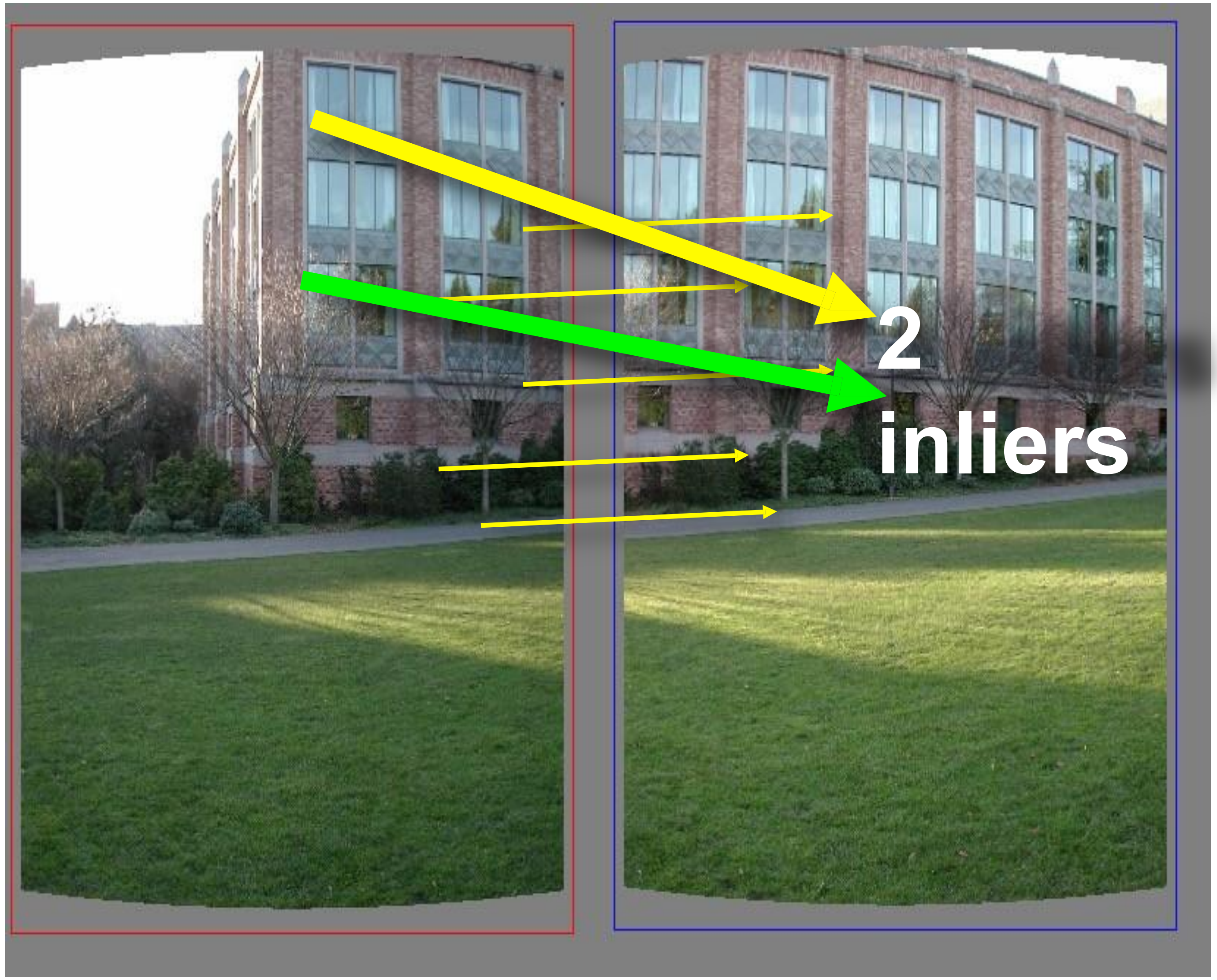
If we use noisy keypoints, we will get this bad transformation.

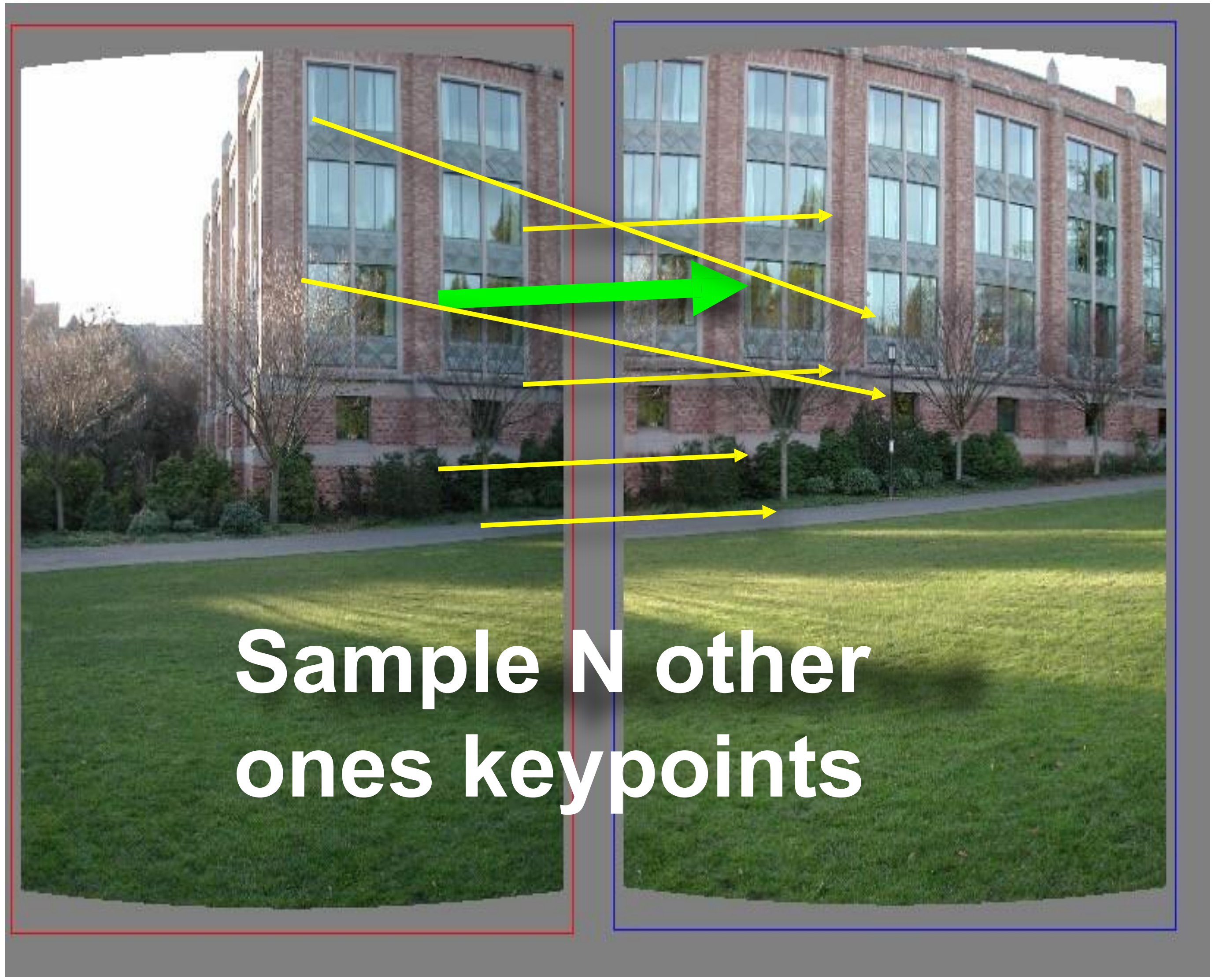


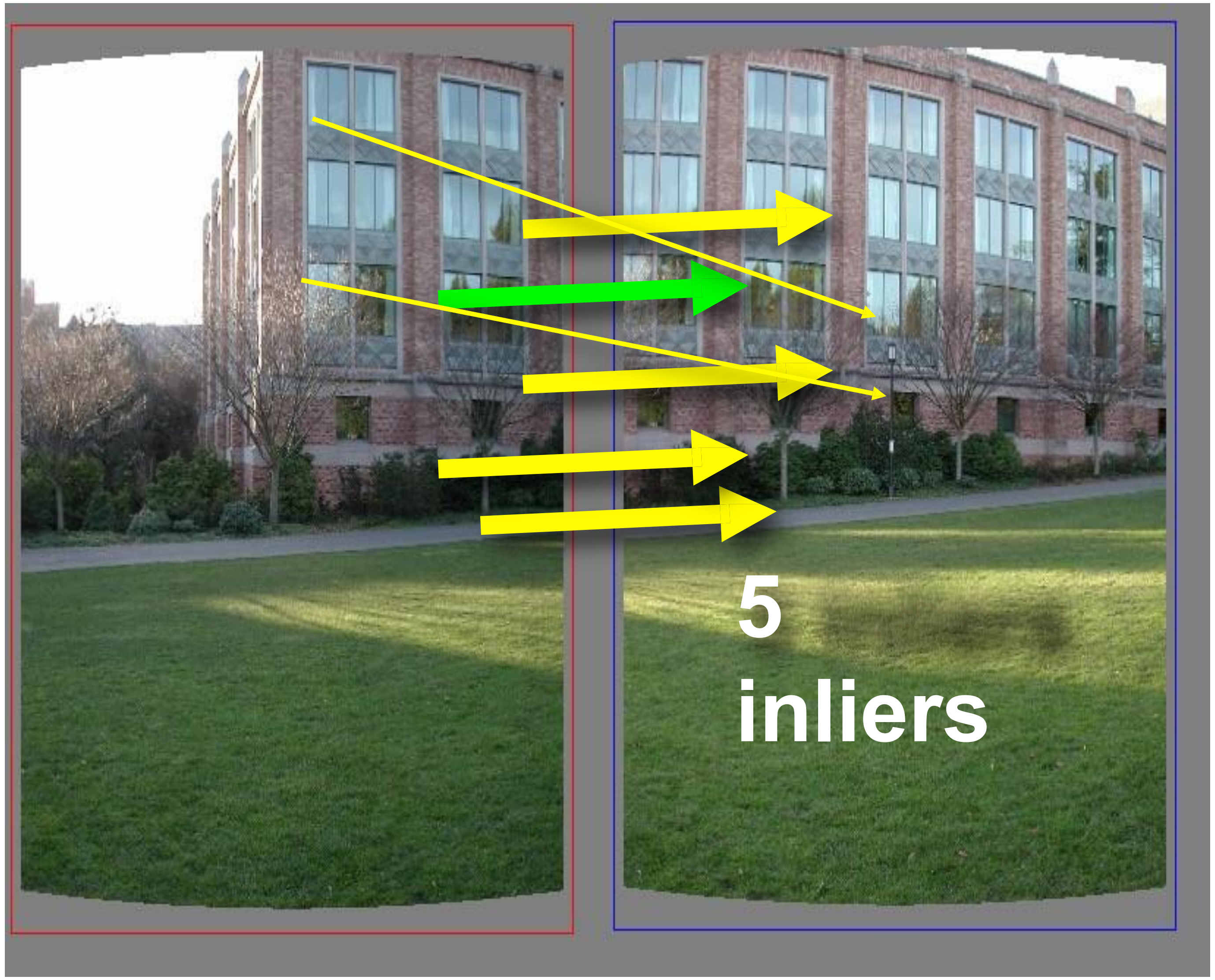
Q. Can you think of an algorithm we have learned that can fix this problem?

RANSAC!!!!




2
inliers

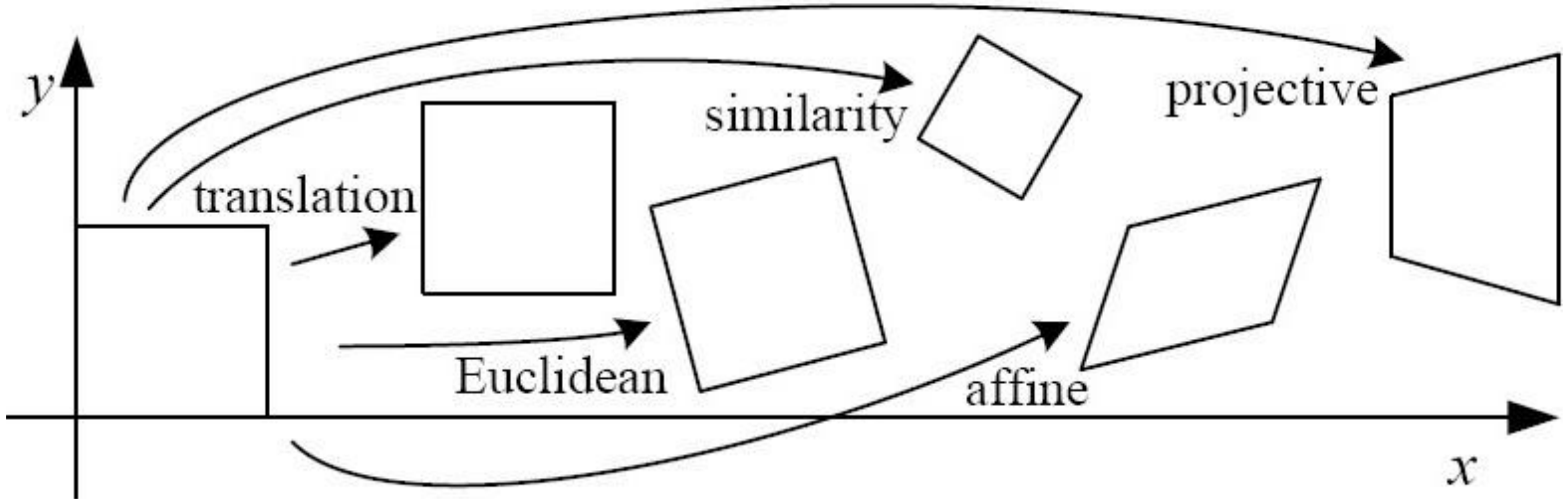

Sample N other ones keypoints


5
inliers

Putting it all together to create a panorama

1. Find keypoints using SIFT or Harris corner
2. Find matches using local feature descriptors
3. Sample N keypoints
    a. Put the sampled points in the matrix form $Ah = 0$
    b. Use SVD to solve for homography matrix h
    c. Calculate inliers
    d. Repeat
4. Re-calculate h using the inliers from best homography

Aside: Remember that we are doing projective transformations.



If the transformation was affine, the homography matrix would be simpler.
We would only have rotation, translation and scaling.

For affine transformations, the solution is simpler!

Affine transformation:

$$H_{\text{affine}} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

For affine transformations, the solution is simpler!

Affine transformation:

$$H_{\text{affine}} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

Vectorize transformation parameters:

Stack equations from point correspondences:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ x' \\ y' \\ \vdots \\ x' \\ y' \end{bmatrix} \quad \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \\ x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \\ & & & \vdots & & \\ x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Today’s agenda

- Local descriptors (SIFT)
  - Making keypoints rotation invariant
  - Designing a descriptor
  - Designing a matching function
- Image Homography
- Global descriptors (HoG)

Global Feature descriptors

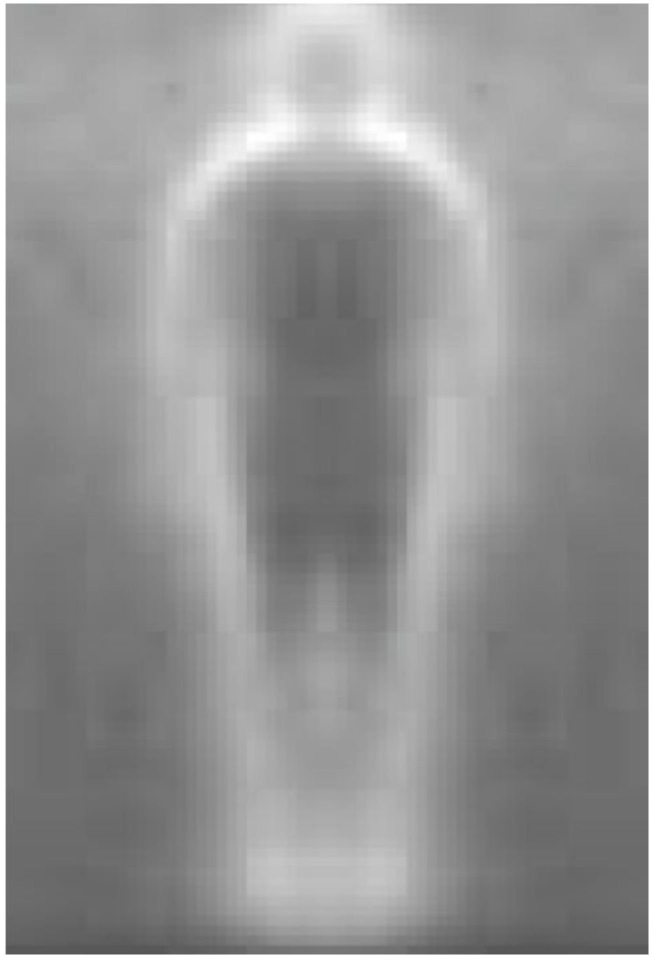
- Find robust feature set that allows object shape to be recognized.
- **Challenges**
  - Wide range of pose and large variations in appearances
  - Cluttered backgrounds under different illumination
  - Computation speed
- **Histogram of Oriented Gradients (HoG)**

[1] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In CVPR, pages 886-893, 2005

[2] Chandrasekhar et al. CHoG: Compressed Histogram of Gradients - A low bit rate feature descriptor, CVPR 2009

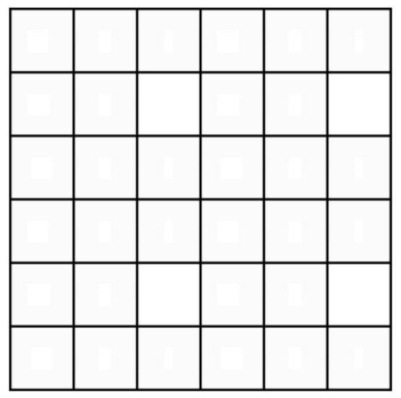
Histogram of Oriented Gradients

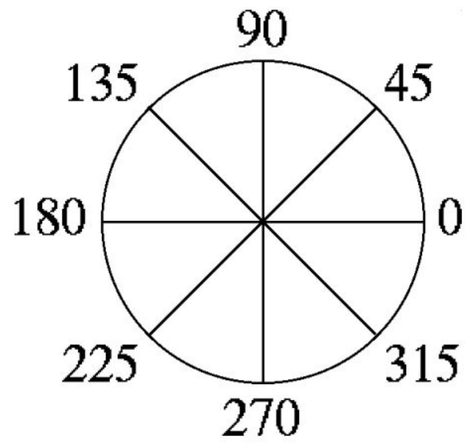
- Local object appearance and shape can often be characterized well using gradients.
- Specifically, the distribution of local intensity gradients or edge directions.



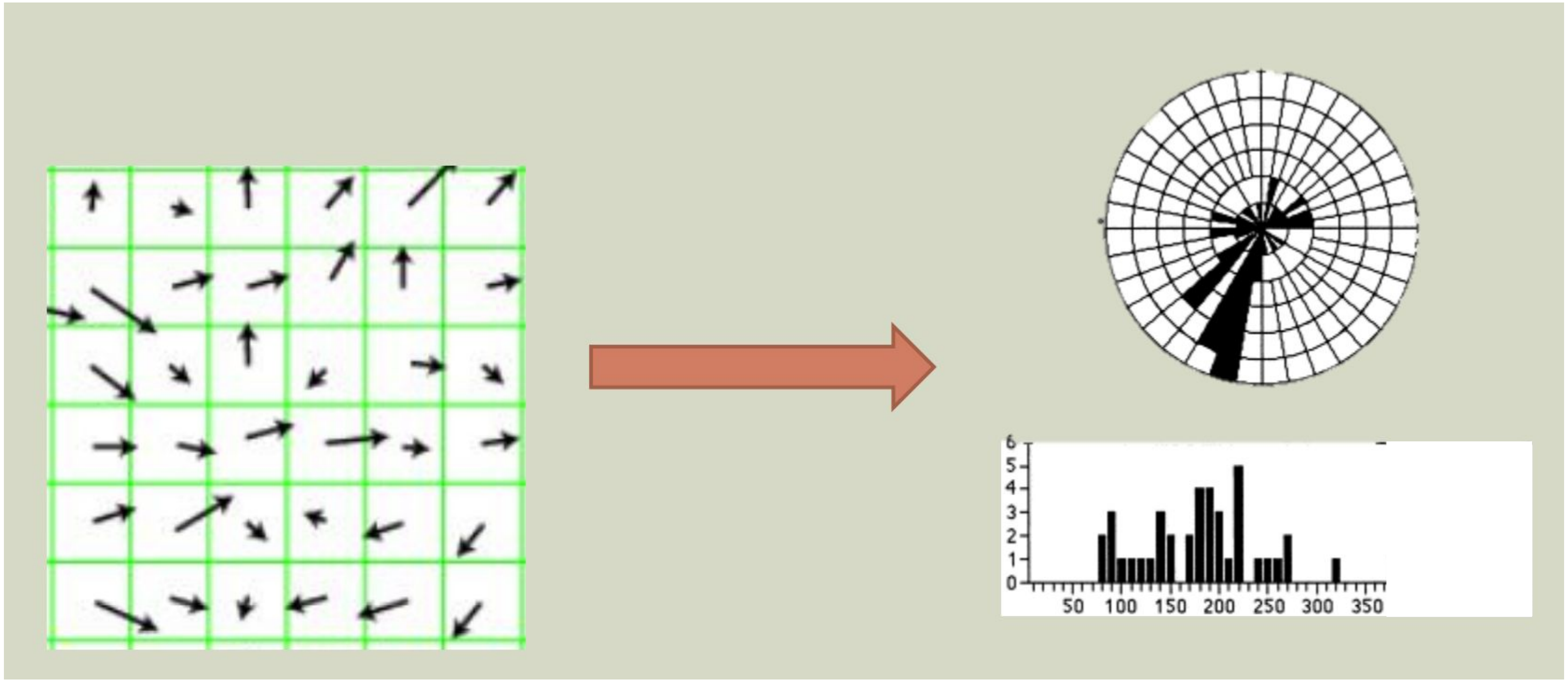
Histogram of Oriented Gradients

- Dividing the image window into small spatial regions (cells)
- Cells can be either rectangle or radial.
- Each window sums up local 1-D histogram of gradient directions over the pixels of the cell.

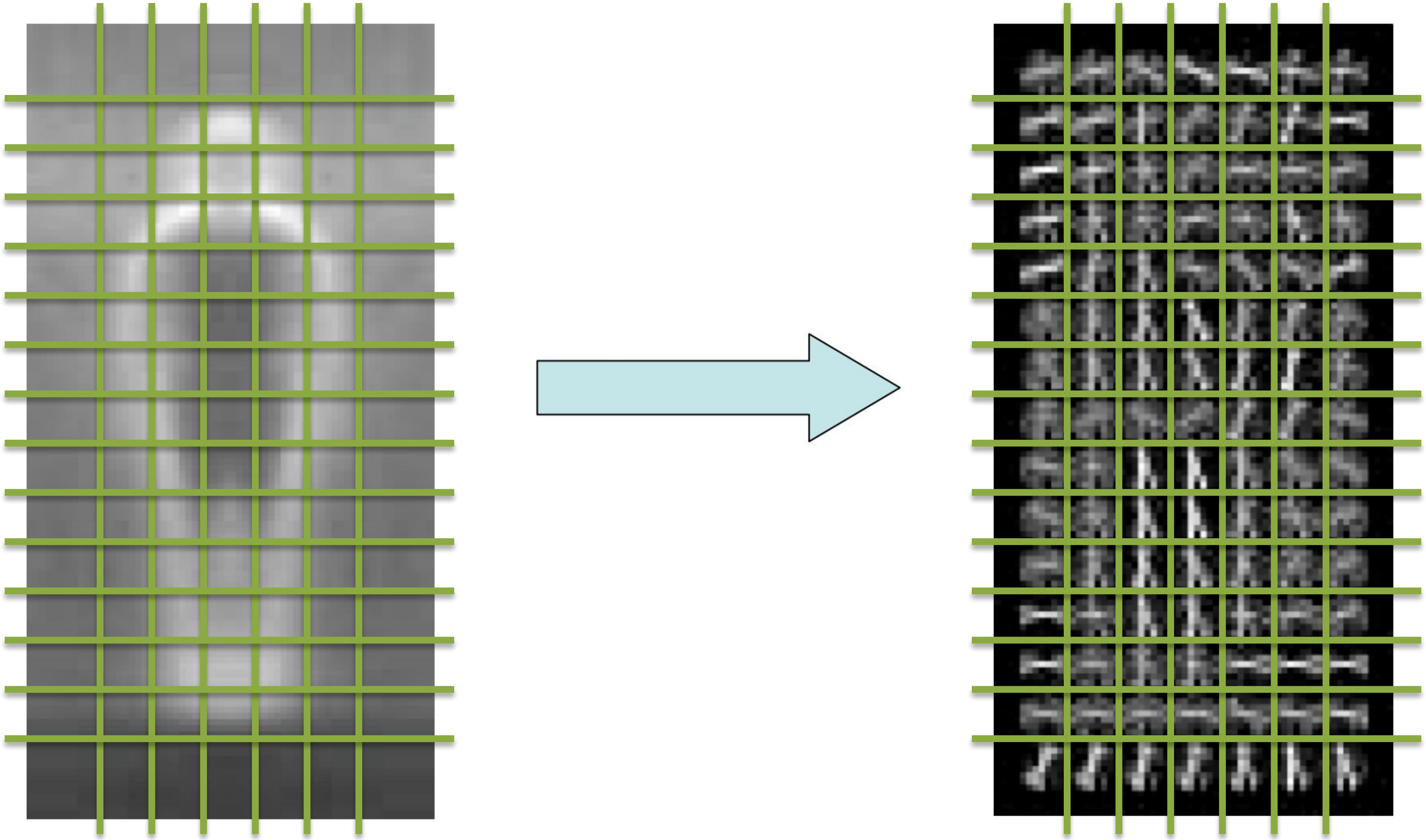




Histogram of Oriented Gradients

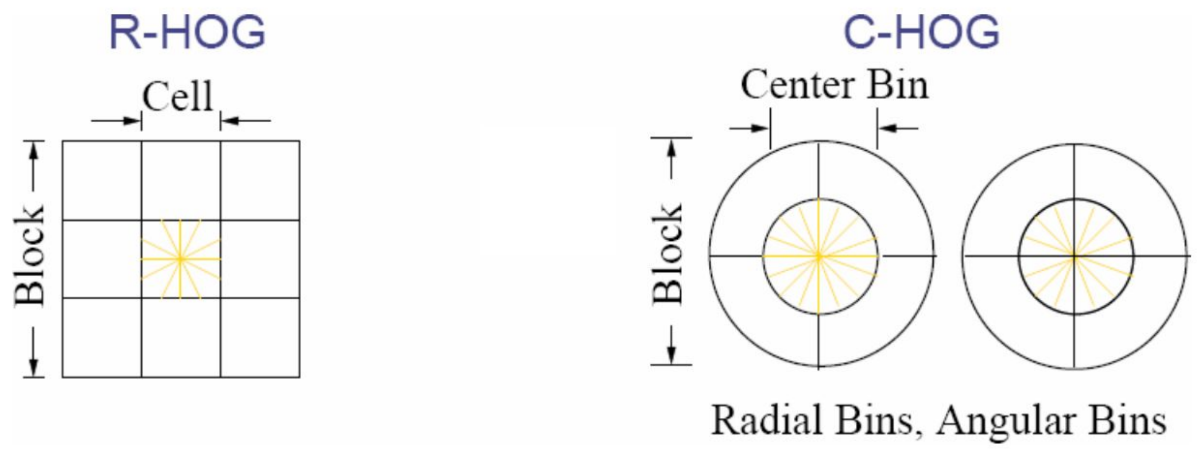


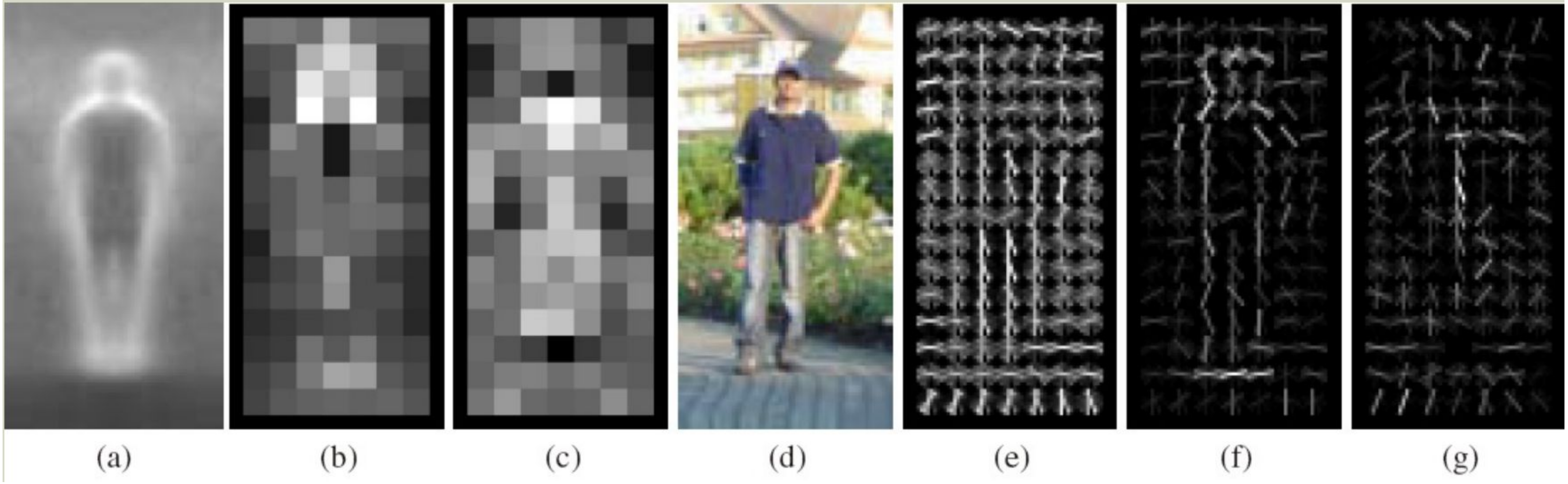
Histogram of Oriented Gradients



Normalization

- To make HoG invariant to illumination and shadows, it is useful to normalize the local responses
- Normalize each cell's histogram using histogram over a larger regions ("blocks").

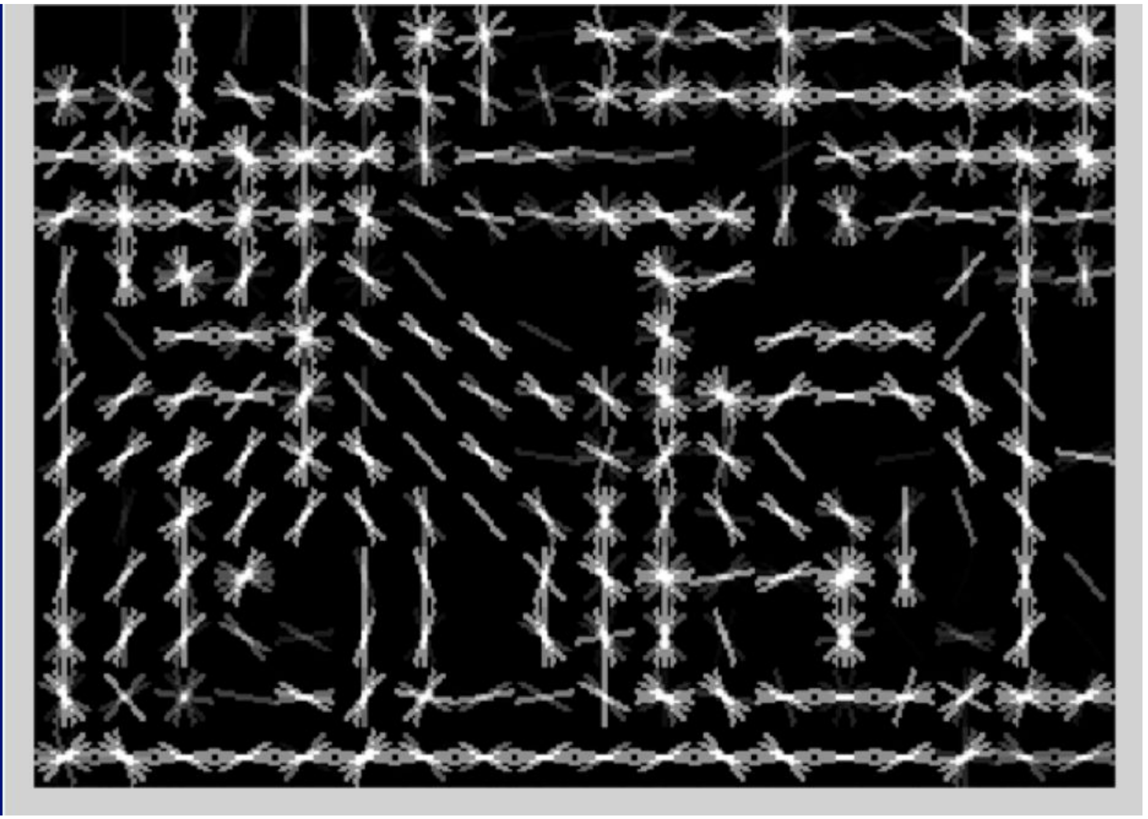




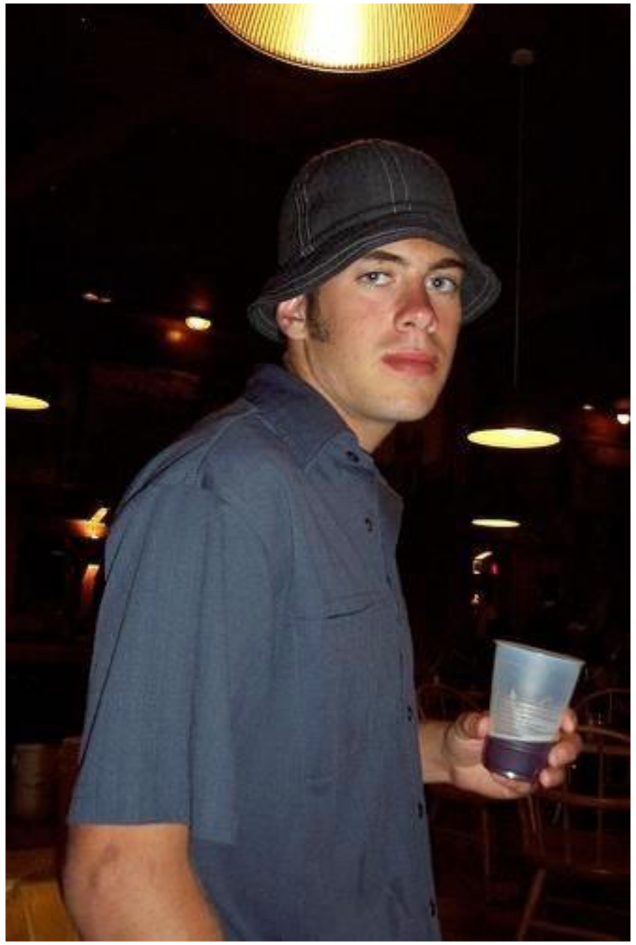
Visualizing HoG

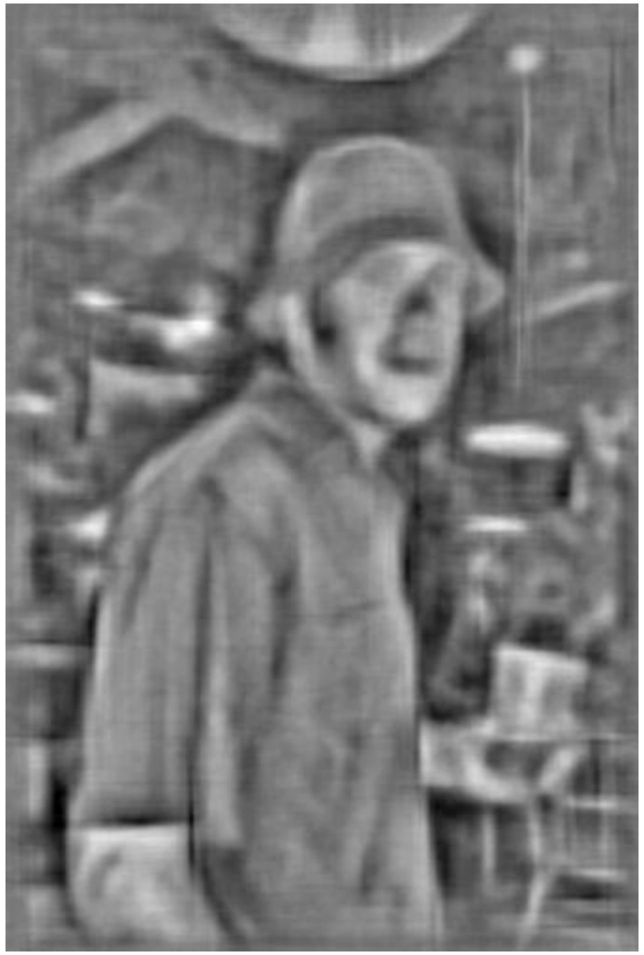
a. Average gradient over example photo of a person
b. “Positive” blocks that help match to other photos of people
c. “Negative” blocks that do not match to photos of other people
d. A test image
e. It’s HOG descriptor visualized
f. HOG descriptor weighted by positive weights
g. HOG descriptor weighted by negative weights

Visualizing HoG

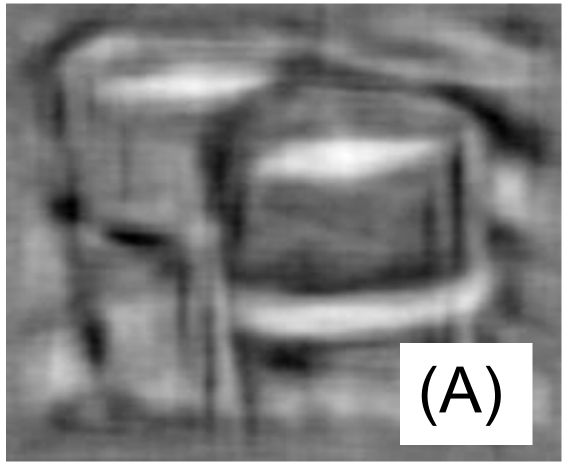


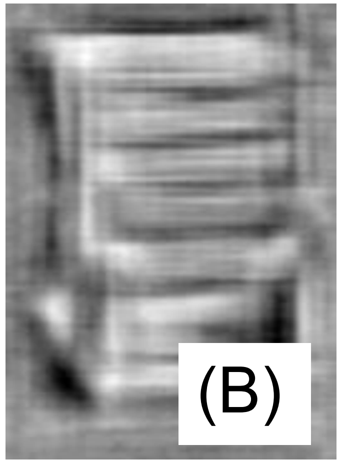
HoG features are good but gradients are insufficient sometimes

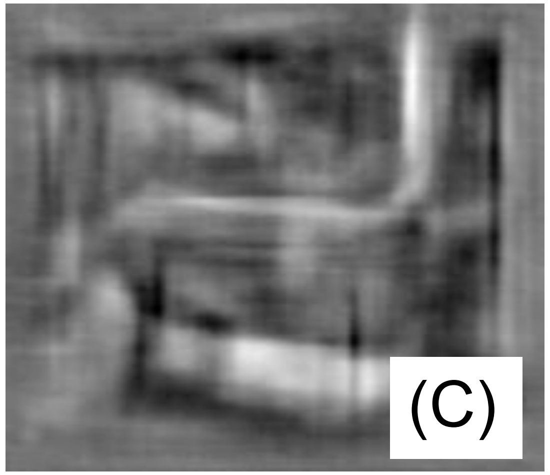


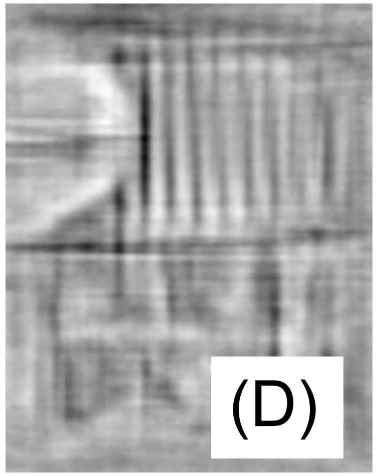


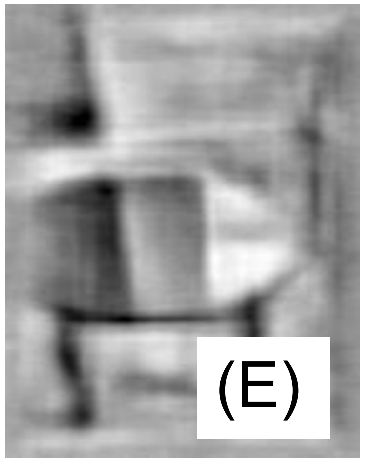
Chair Detections

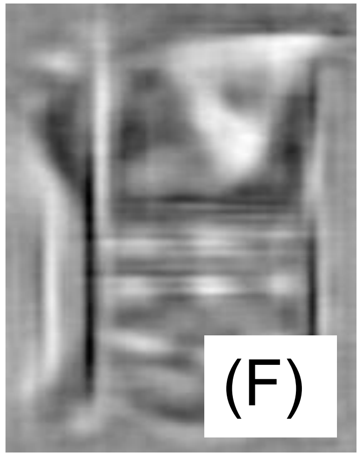


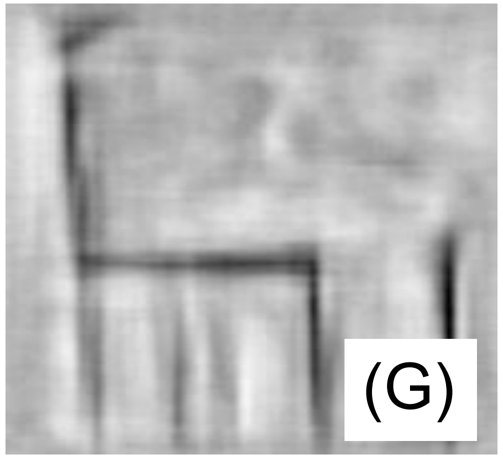






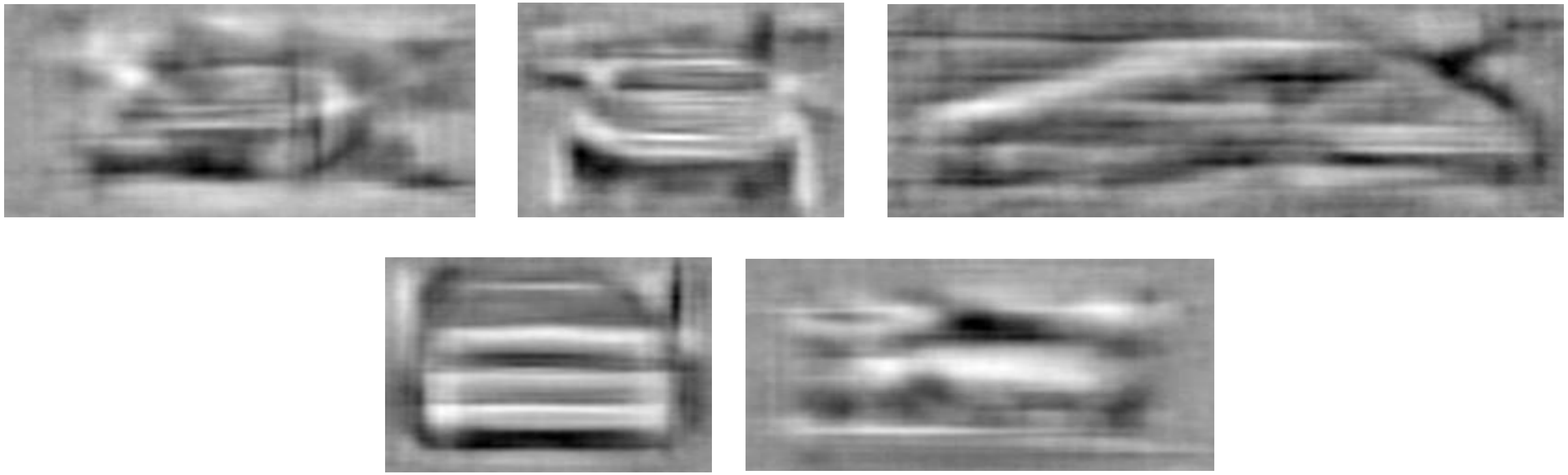






Chair Detections

Car Detections



Car Detections

Difference between HoG and SIFT

- HoG is usually used to describe larger image regions.
- SIFT is used for key point matching

- SIFT histograms are normalized with respect to the dominant gradient.
- HoG gradients are normalized using neighborhood blocks.

Today's agenda

- Local descriptors (SIFT)
  - Making keypoints rotation invariant
  - Designing a descriptor
  - Designing a matching function
- Image Homography
- Global descriptors (HoG)

Next time

Resizing image content

Extra slides

The HOGgles Challenge

Clap your hands when you see a person

The HOGgles Challenge