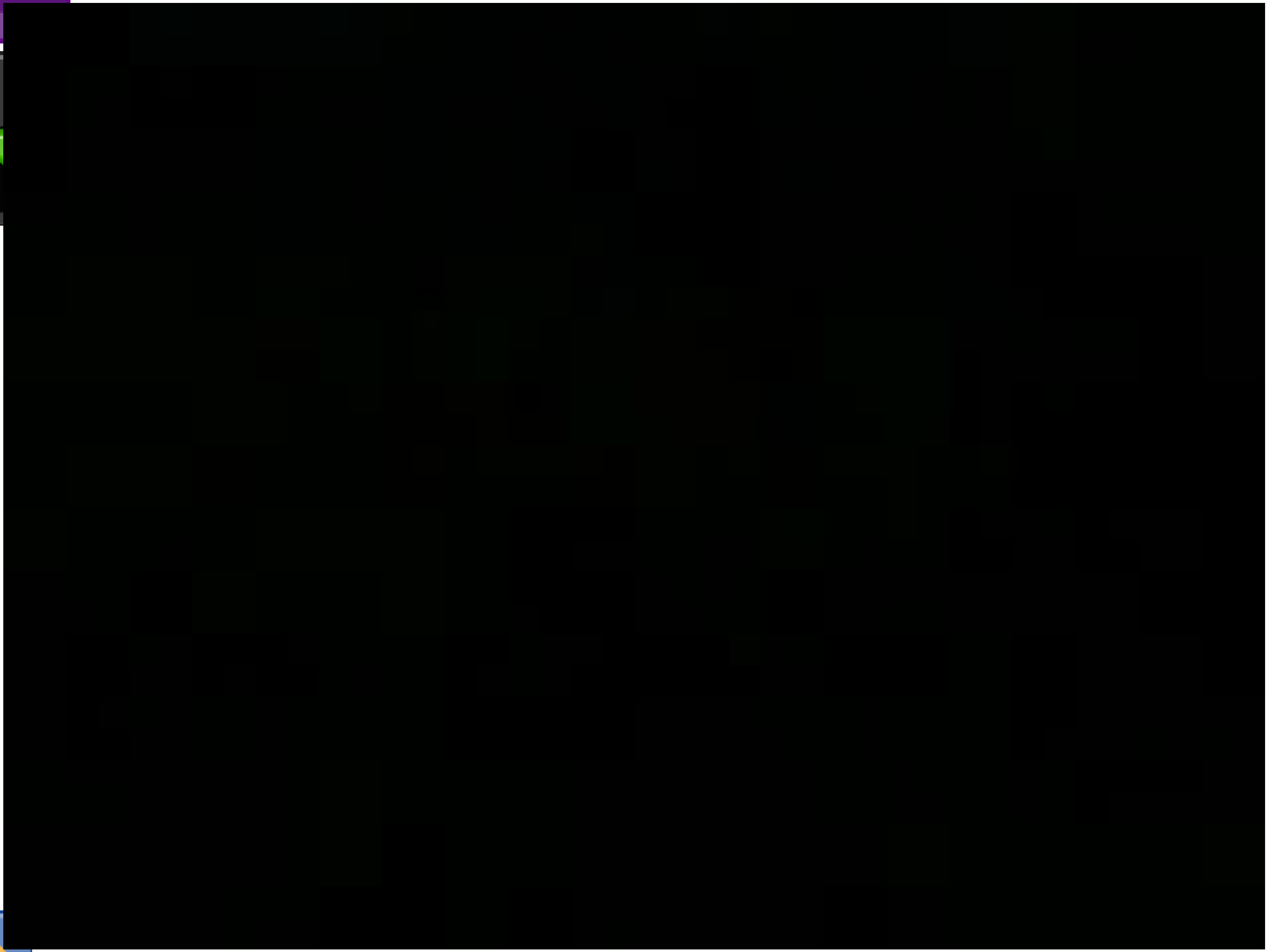
It's the features!

Nikhil Rasiwasia
Scientist
Yahoo! Labs Bangalore

Slides contributed by Gaurav Aggarwal, Yahoo! Labs



Features, Features, Features

In almost every case:

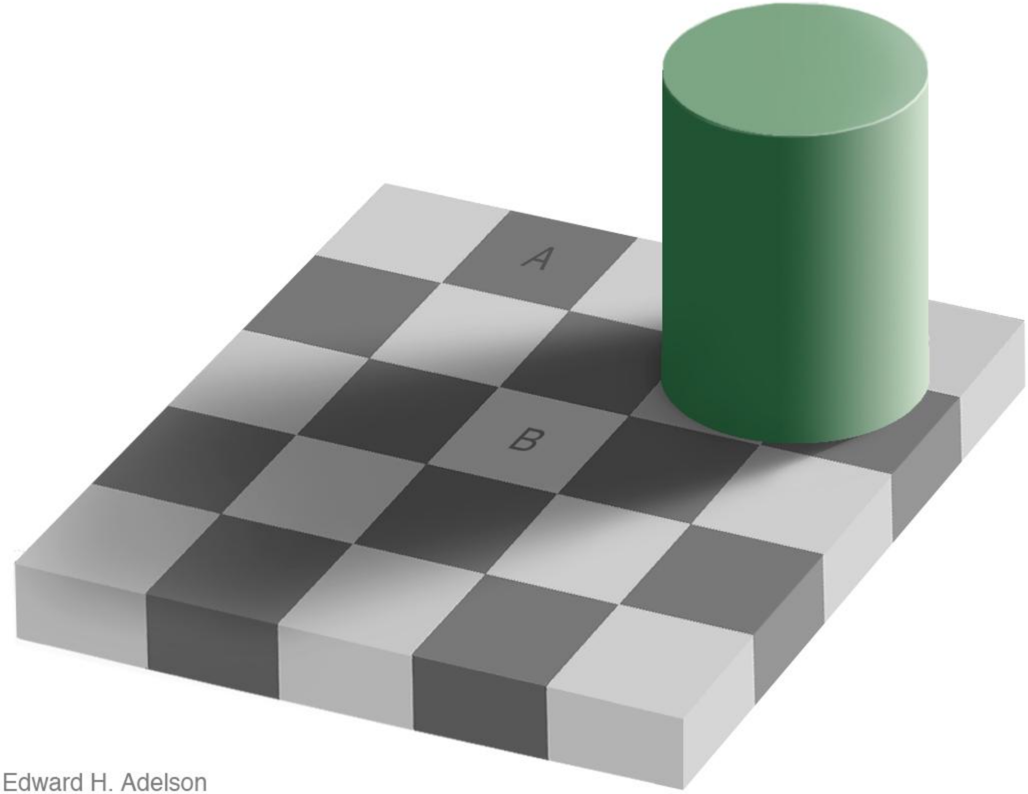
Good Features beat Good Learning

Learning beats No Learning

(Viola 2003)

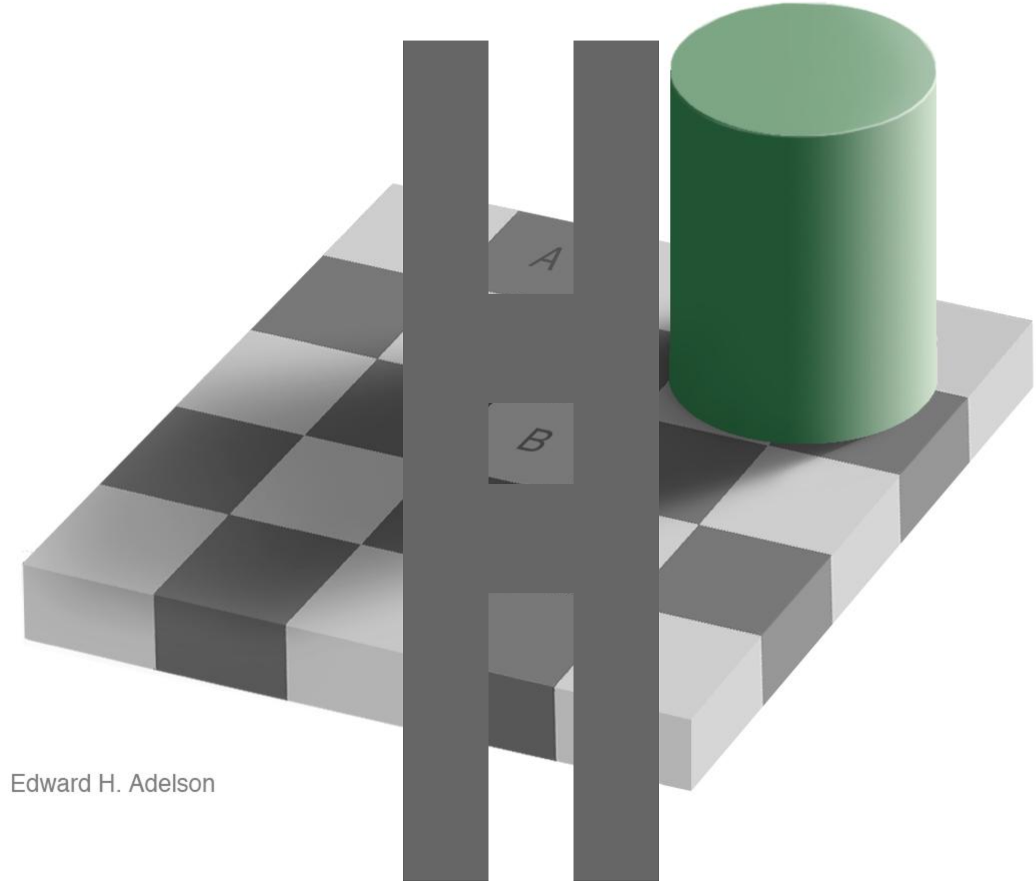
Why do we need features?

Brightness: Measurement vs. Perception

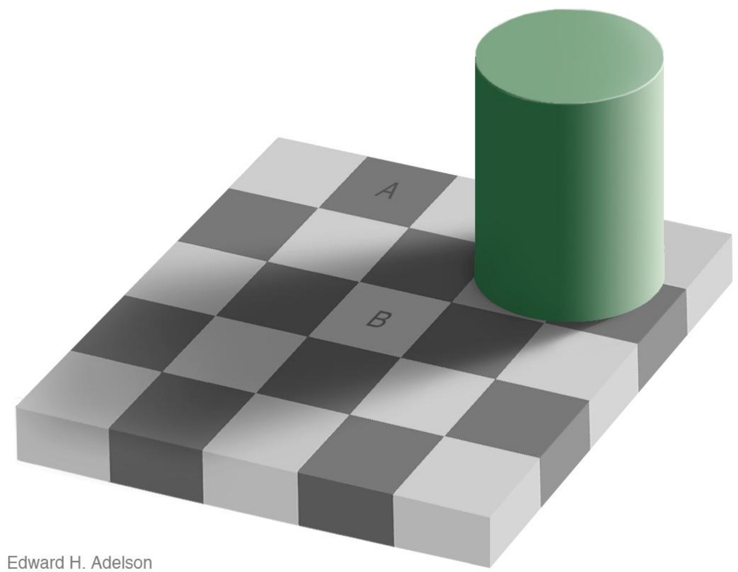


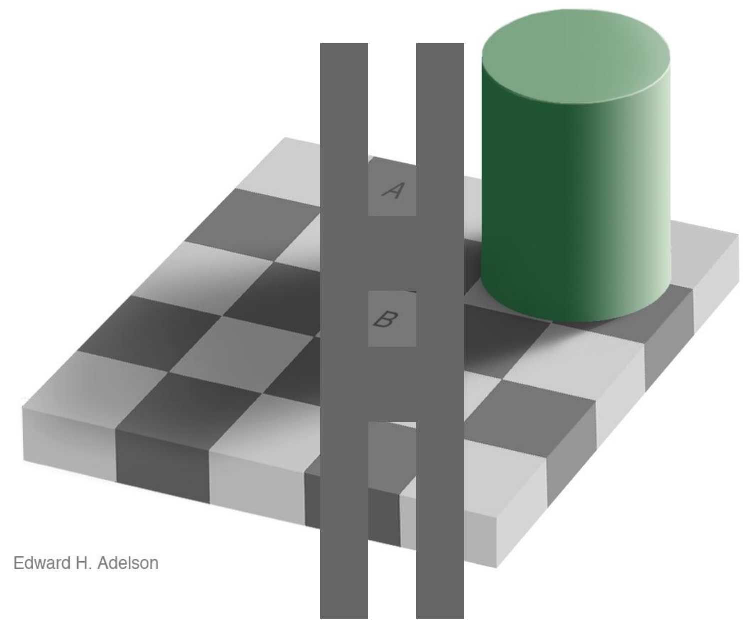
Edward H. Adelson

Brightness: Measurement vs. Perception



Brightness: Measurement vs. Perception





Proof!

A little story about Computer Vision (Recognition)

Founder, MIT AI project

In 1966, **Marvin Minsky** at MIT asked his undergraduate student Gerald Jay Sussman to "spend the summer linking a camera to a computer and getting the computer to **describe** what it saw".

Recognize

We now know that the problem is more difficult than that.

(Szeliski, 2009, pg 33)

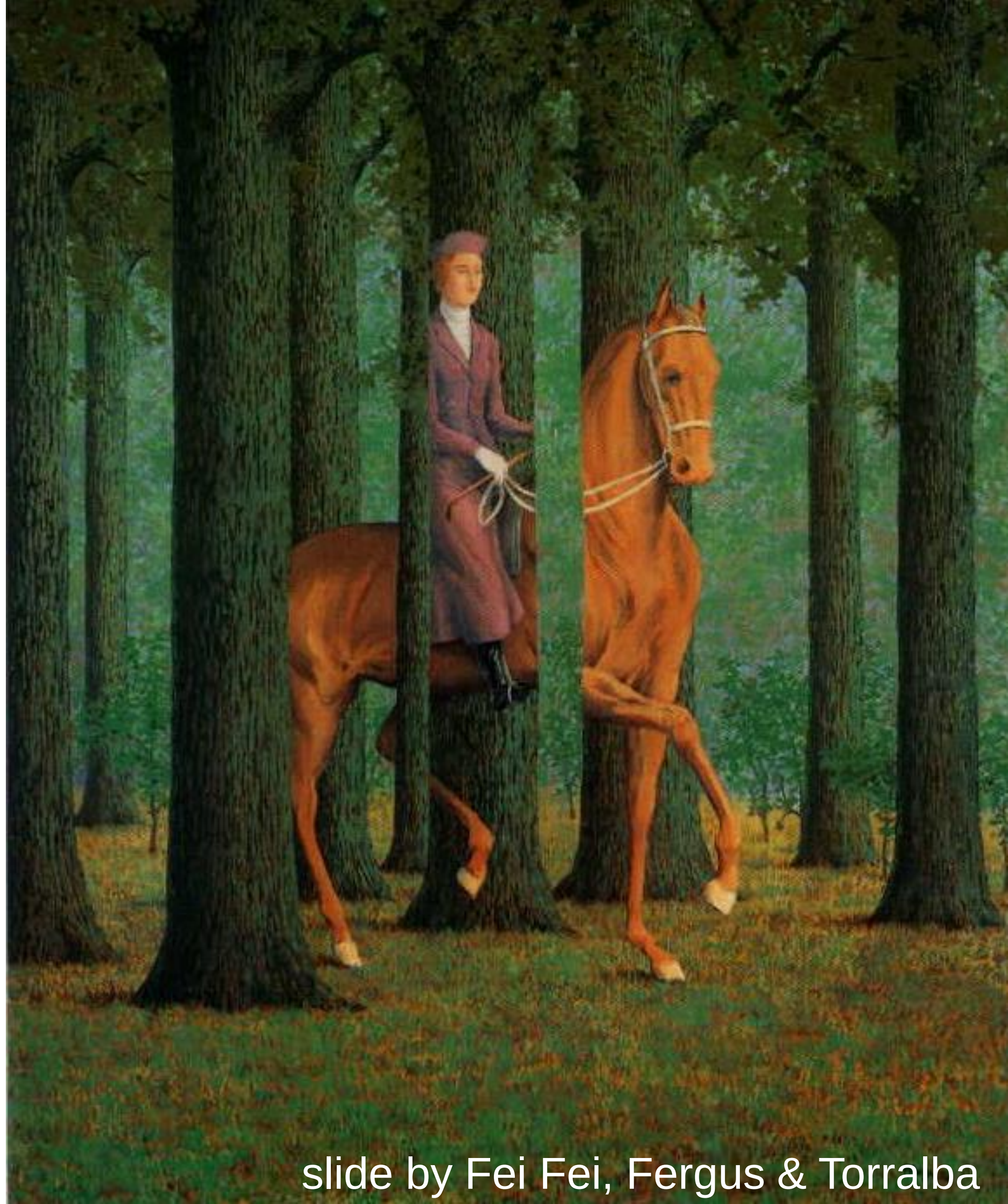
Challenges: view point variation

Peita 1498-1499

Challenges: illumination

slide credit: S. Ullman

Challenges: occlusion



Magritte, 1957

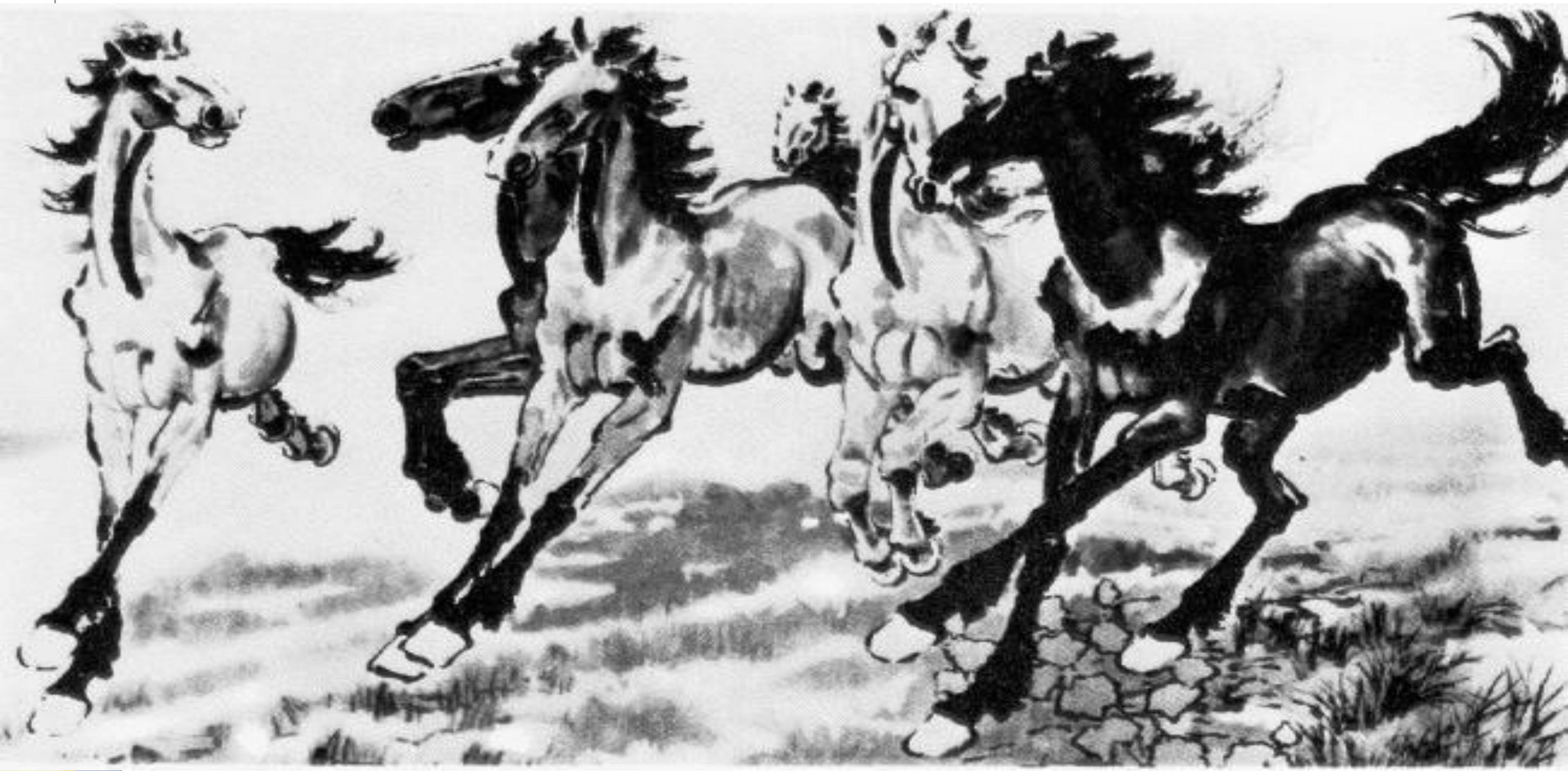
slide by Fei Fei, Fergus & Torralba

Challenges: Texture grouping and segmentation

Challenges: scale

slide by Fei Fei, Fergus & Torralba

Challenges: deformation

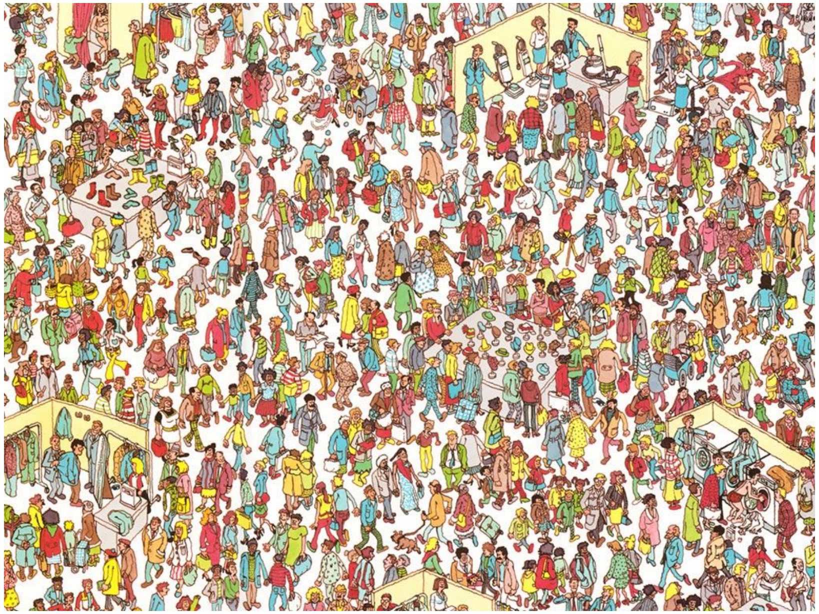


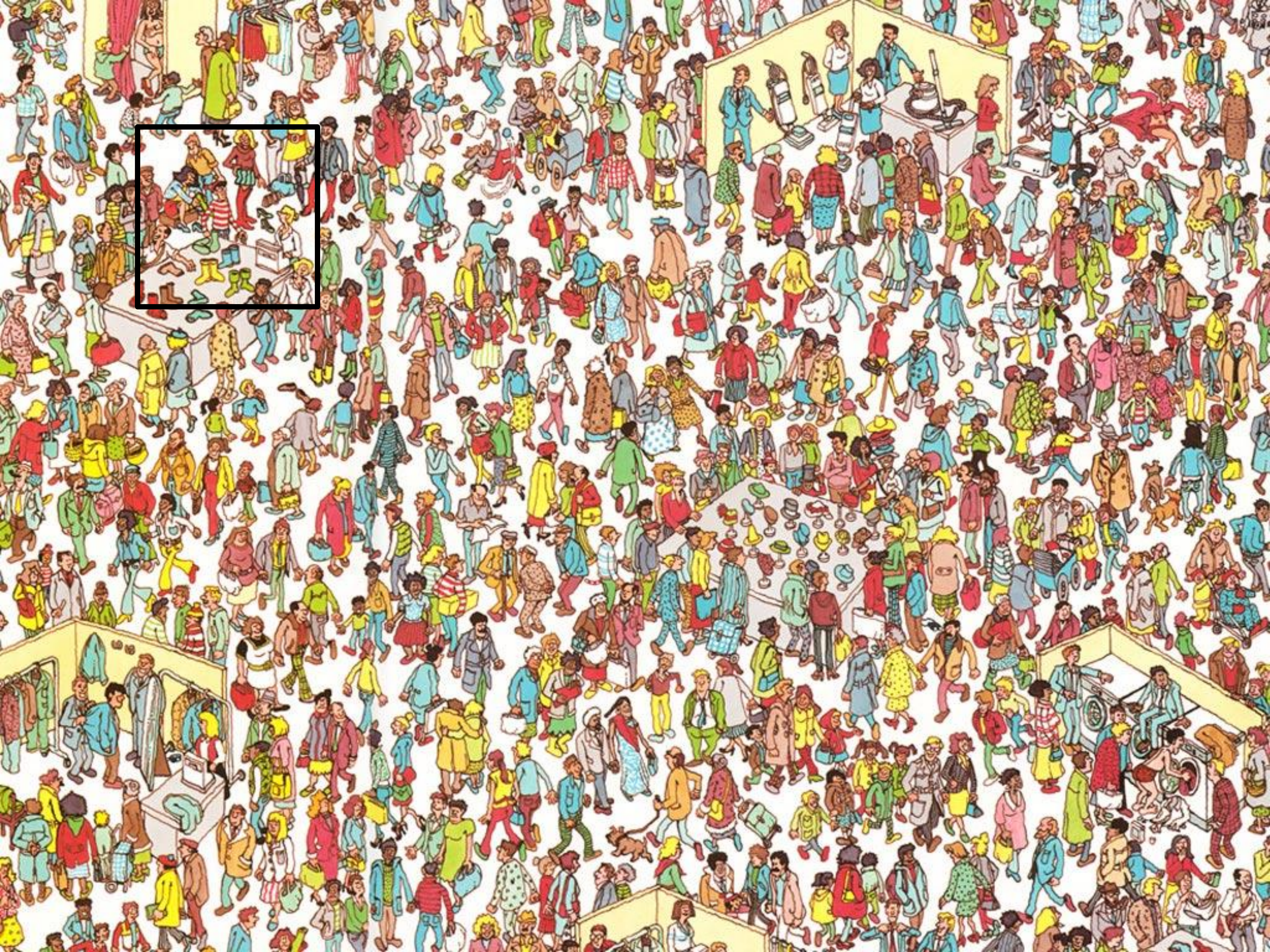
Xu, Beihong 1943

slide

Challenges: background clutter

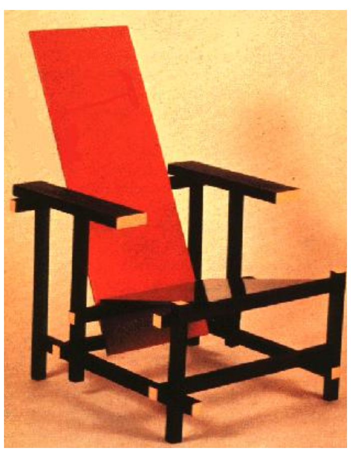
Where is Waldo?





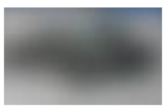
Challenges: object intra-class variation

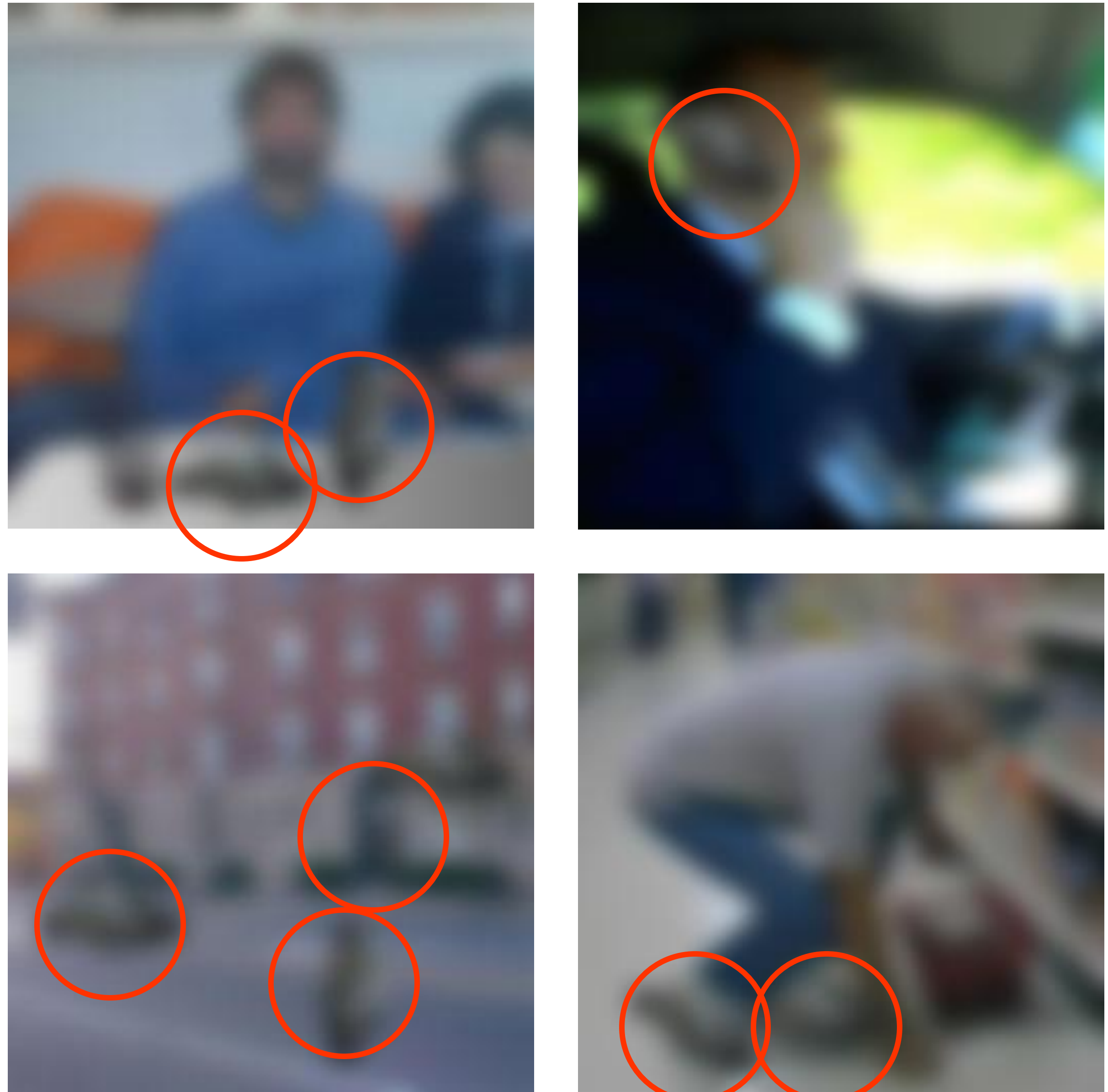


slide by Fei-Fei, Fergus & Torralba

Challenges: local ambiguity





slide by Fei-Fei, Fergus & Torralba

Challenges: Context

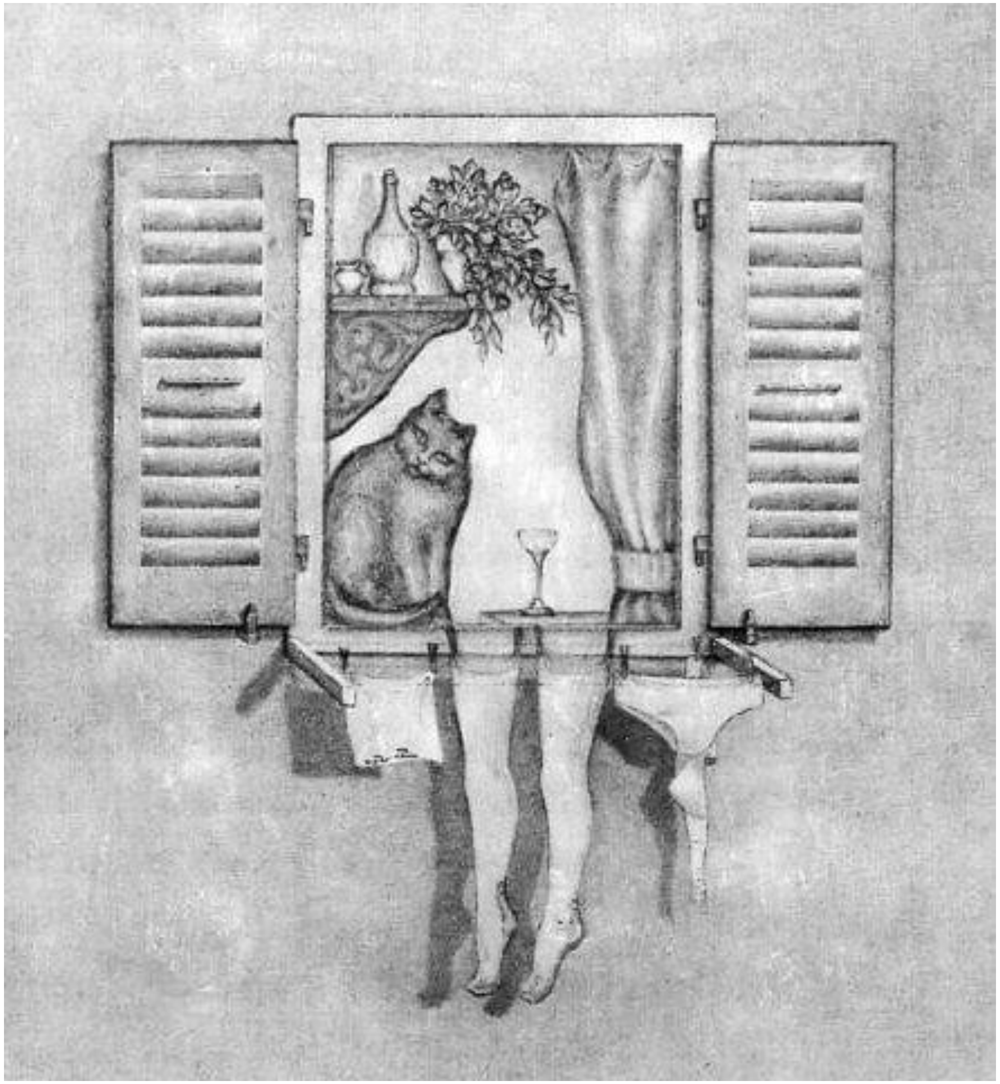
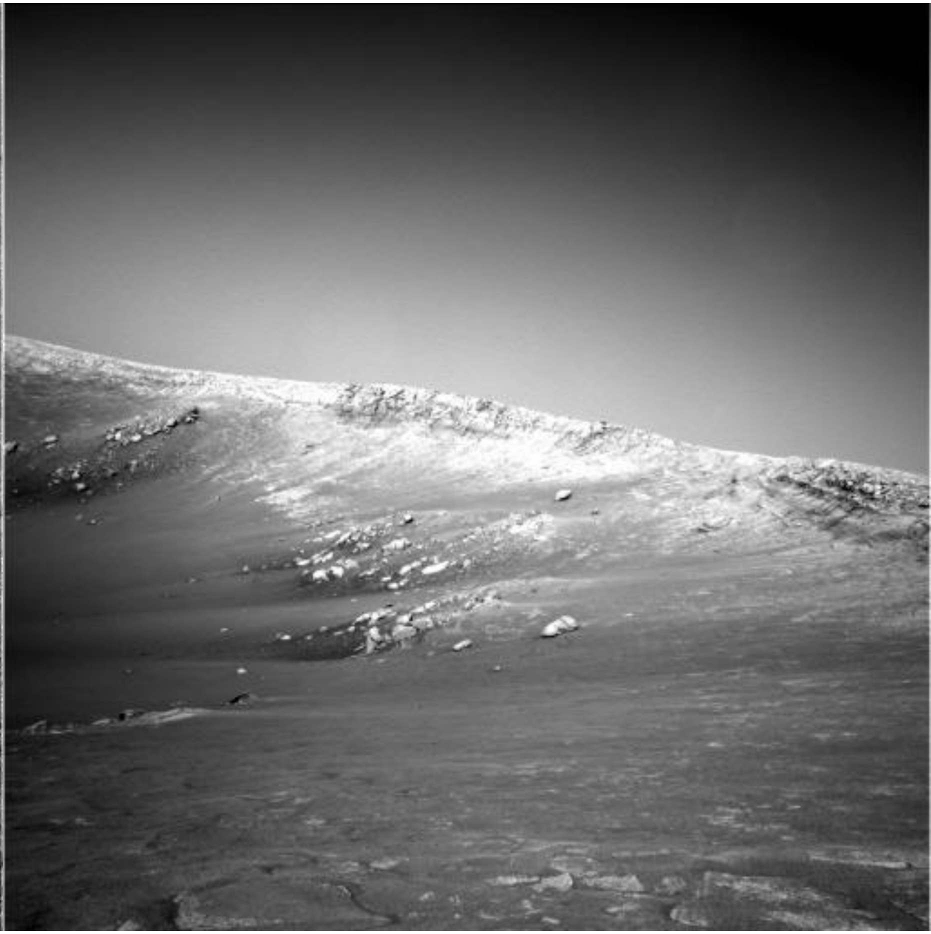


Image matching

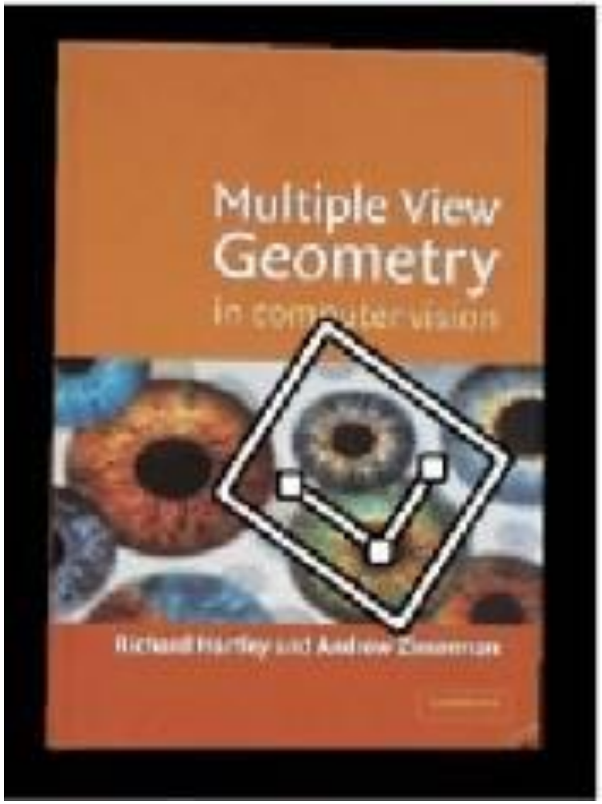
Harder case

Harder!



CV works! (Look for tiny colored squares)

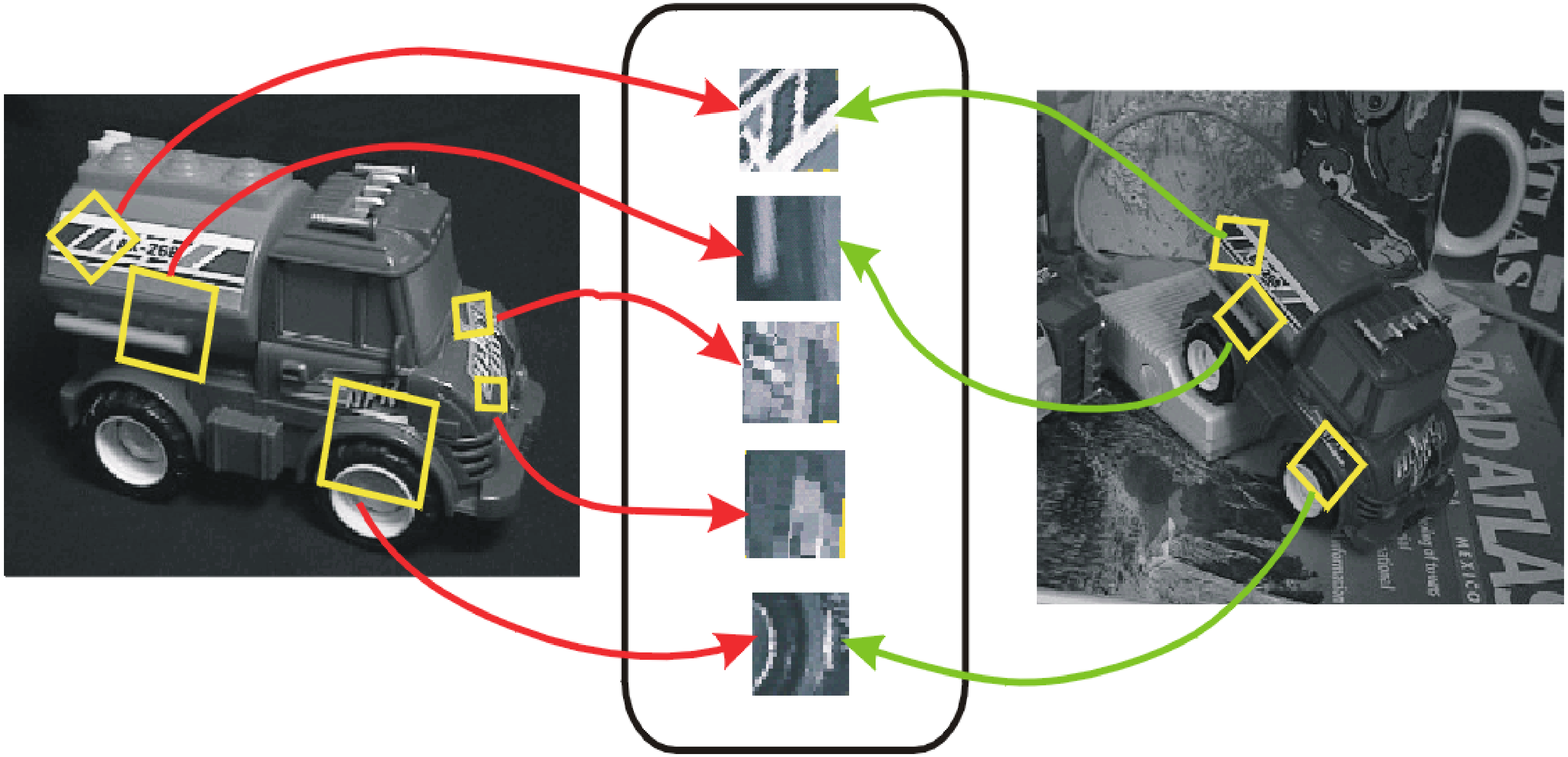
Image Matching



1) At an interesting point, let's define a coordinate system (x,y axis)
2) Use the coordinate system to pull out a patch at that point

Image Matching

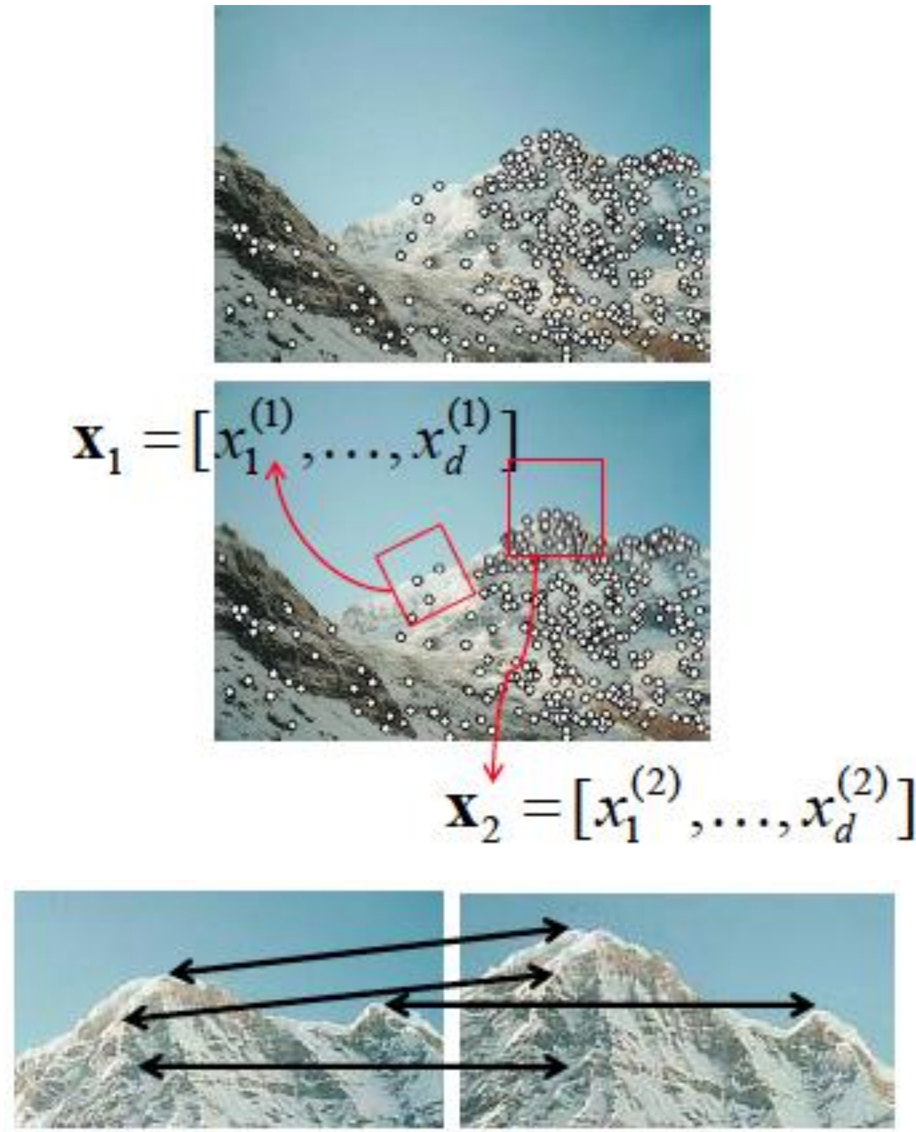
Invariant Local Features



Buzzword is invariance!

Local Features: main components

- Detection
  - Identify the interest points
- Description
  - Extract feature descriptor surrounding each point
- Matching
  - Determine correspondence between descriptors
  - (we will not cover this)
- Global Description
  - Bag-of-Words



Local Features: desired properties

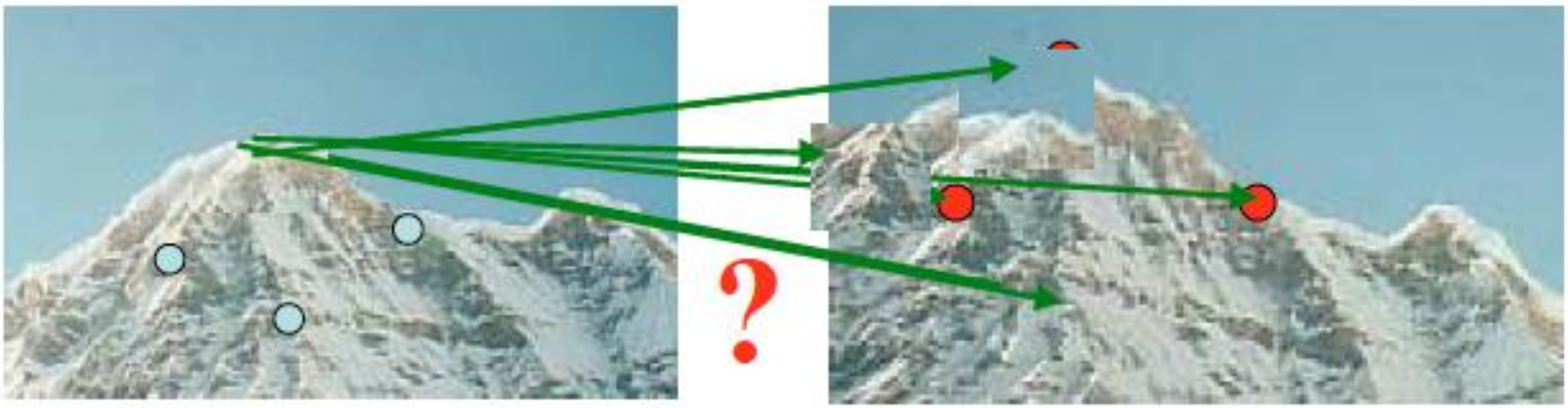
- Repeatability
  - Can be found despite geometric and photometric transformations

No chance to find true matches!

We have to be able to run the detection procedure *independently* per image.

Local Features: desired properties

- Saliency
  - Distinctive description



- Reliably determine which point goes with which.
- Must provide some invariance to geometric and photometric differences

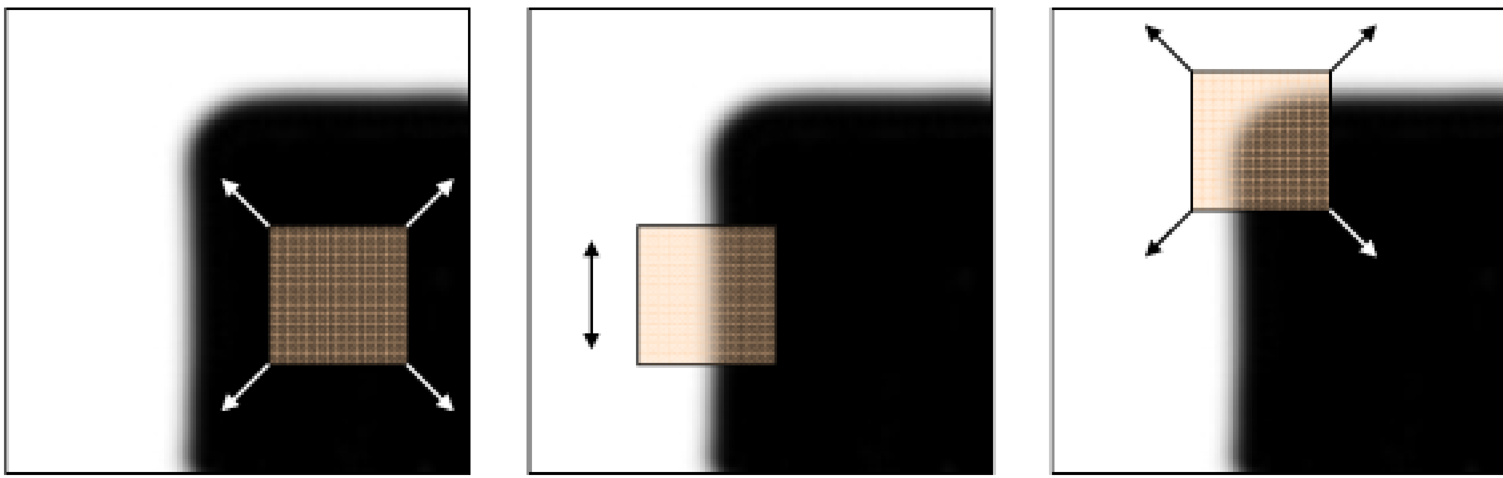
Local Features: desired properties

- Compactness and Efficiency
  - Many fewer features than image pixels
- Locality
  - Occupies relatively a small area in the image
  - Robust to clutter and occlusion

What points would you choose?

Uniqueness

- How to define unique/unusual?
- Local measure of uniqueness
  - Corners as distinctive interest points



“flat” region:
no change in
all directions

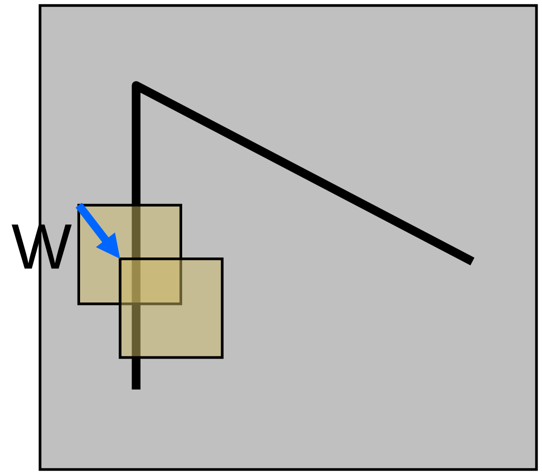
“edge”:
no change
along the edge
direction

“corner”:
significant
change in all
directions

Feature detection: the math

Consider shifting the window W by (u,v)

- how do the pixels in W change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD "error" of *E(u,v)*:



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

Small motion assumption

Taylor Series expansion of I:

$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \text{higher order terms}$$

If the motion (u,v) is small, then first order approx is good

$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v$$

$$\approx I(x,y) + [I_x \;\; I_y] \begin{bmatrix} u \\ v \end{bmatrix}$$

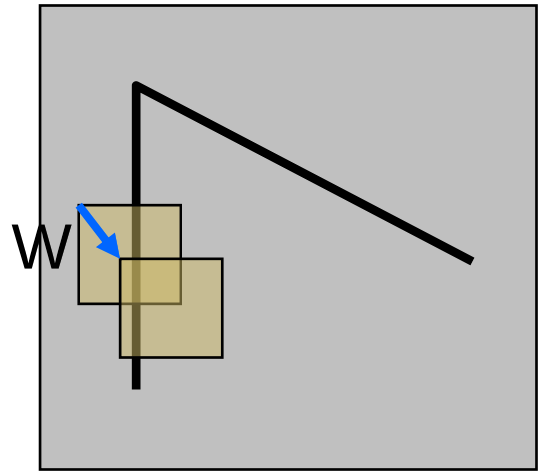
shorthand: $I_x = \frac{\partial I}{\partial x}$

Plugging this into the formula on the previous slide…

Feature detection: the math

Consider shifting the window W by (u,v)

- how do the pixels in W change?
- compare each pixel before and after by summing up the squared differences
- this defines an “error” of E(u,v):

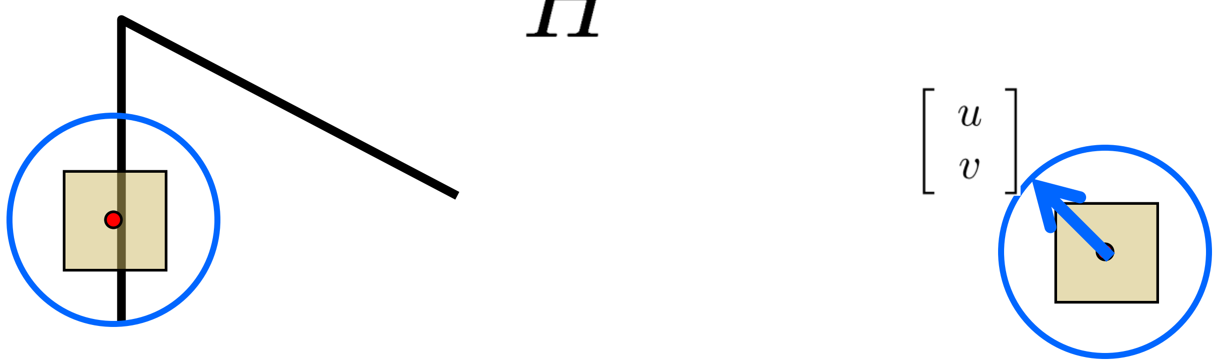


$$
\begin{aligned}
E(u,v) &= \sum_{(x,y)\in W} \left[I(x+u,y+v) - I(x,y)\right]^2 \\
&\approx \sum_{(x,y)\in W} \left[I(x,y) + [I_x \; I_y]\begin{bmatrix} u \\ v \end{bmatrix} - I(x,y)\right]^2 \\
&\approx \sum_{(x,y)\in W} \left[[I_x \; I_y]\begin{bmatrix} u \\ v \end{bmatrix}\right]^2
\end{aligned}
$$

Feature detection: the math

This can be rewritten:

$$E(u,v) = \sum_{(x,y)\in W} [u \ v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

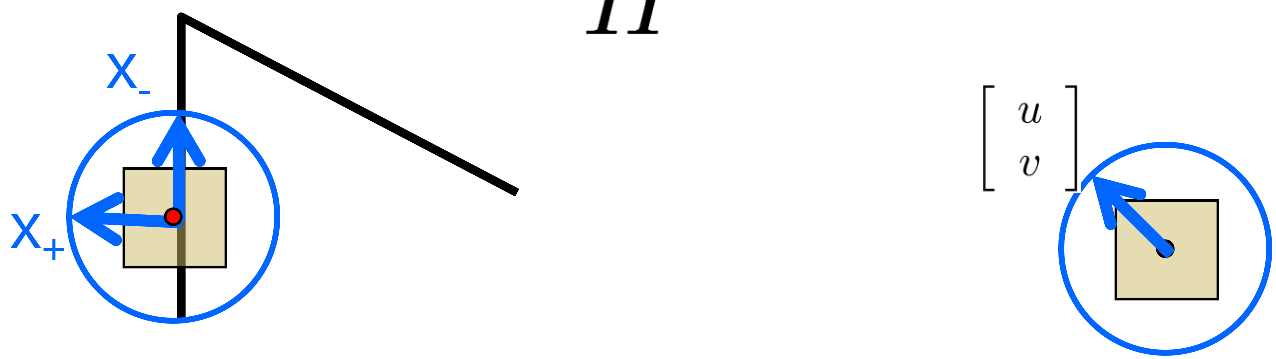


For the example above

- You can move the center of the green window to anywhere on the blue unit circle
- Which directions will result in the largest and smallest E values?
- We can find these directions by looking at the eigenvectors of *H*

Feature detection: the math

$$E(u,v) = \sum_{(x,y)\in W} [u\ v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$



Eigenvalues and eigenvectors of H

- Define shifts with the smallest and largest change (E value)
- x_+ = direction of largest increase in E.
- λ_+ = amount of increase in direction x_+
- x_- = direction of smallest increase in E.
- λ- = amount of increase in direction x_-

$$\begin{aligned} Hx_+ &= \lambda_+ x_+ \\ Hx_- &= \lambda_- x_- \end{aligned}$$

Feature detection: the math

How are λ_+, x_+, λ_-, and x_+ relevant for feature detection?

- What's our feature scoring function?

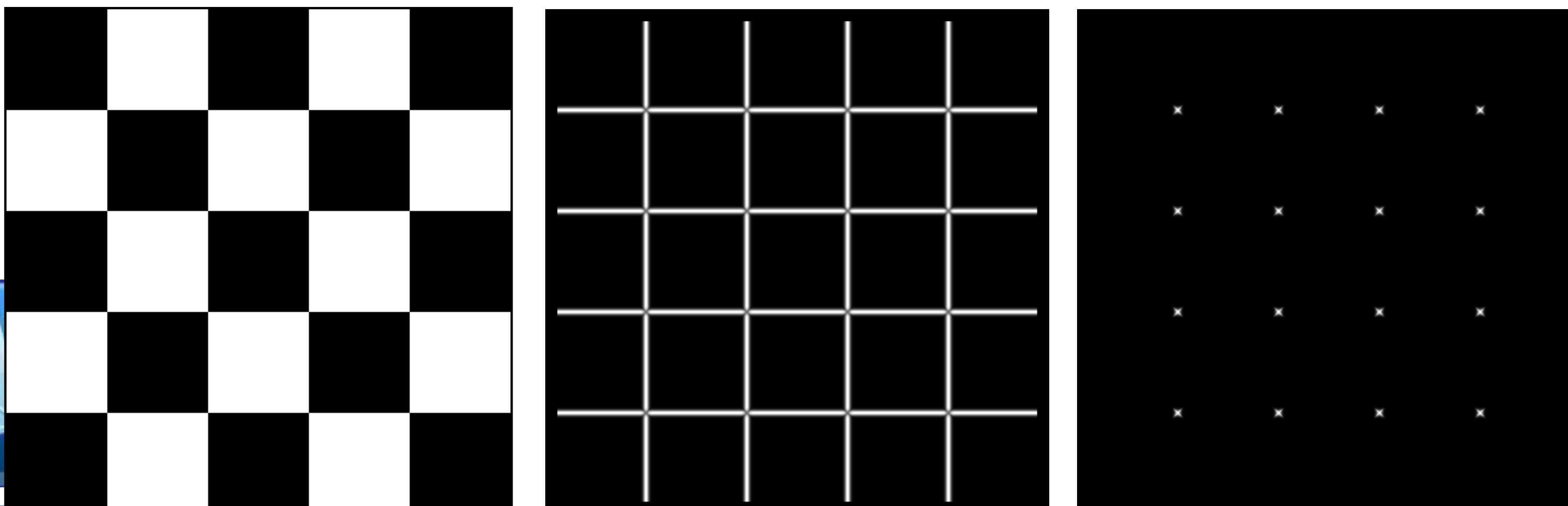
Feature detection: the math

How are λ_+, x_+, λ_-, and x_+ relevant for feature detection?

- What's our feature scoring function?

Want *E(u,v)* to be *large* for small shifts in *all* directions

- the *minimum* of *E(u,v)* should be large, over all unit vectors [u v]
- this minimum is given by the smaller eigenvalue (λ_-) of *H*

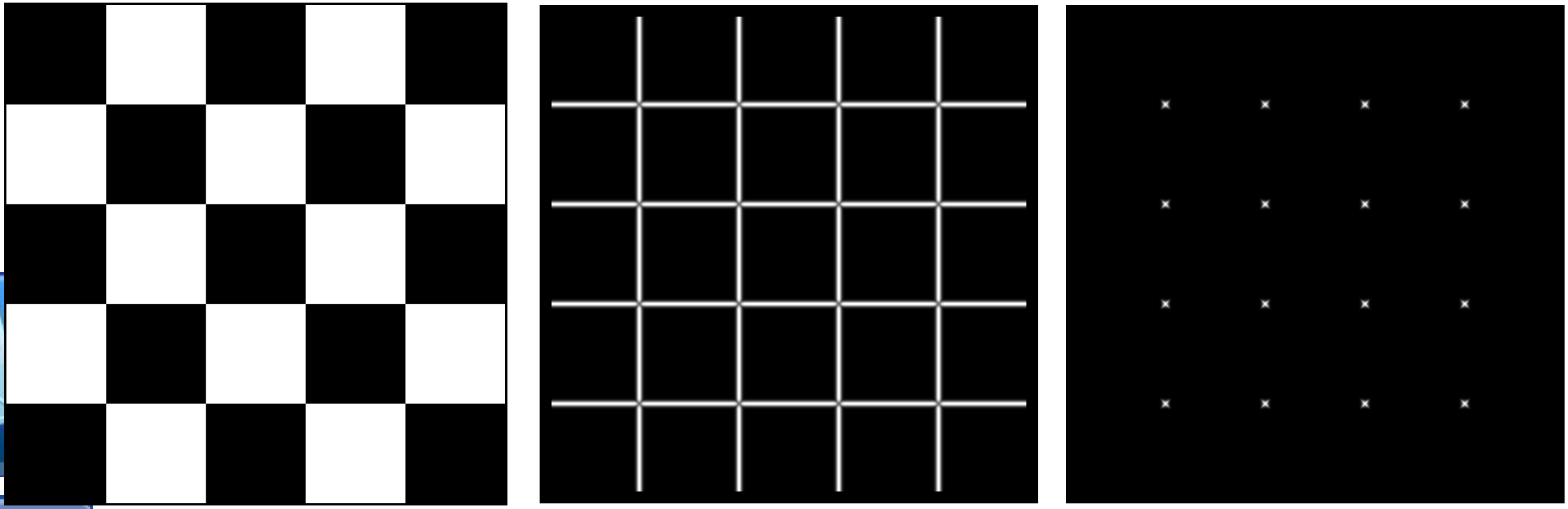


I λ_+ λ_-

Feature detection summary

Here's what you do

- Compute the gradient at each point in the image
- Create the *H* matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response (λ_- > threshold)
- Choose those points where λ_- is a local maximum as features

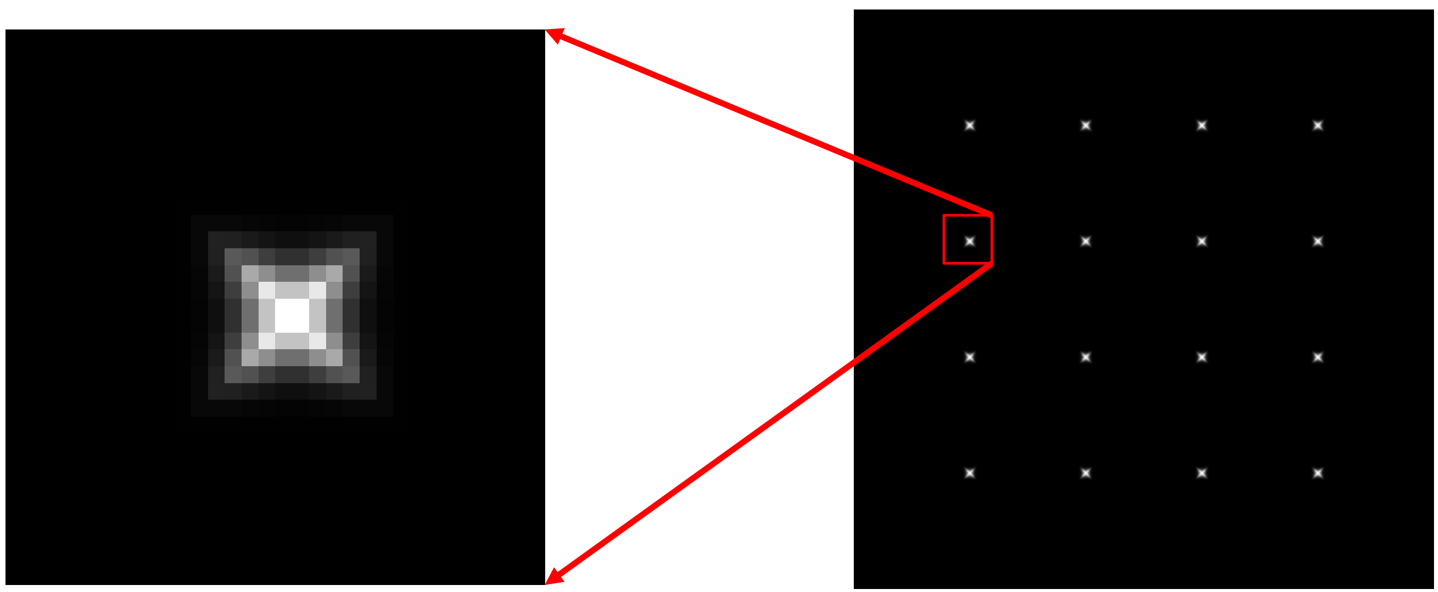


I λ_+ λ_-

Feature detection summary

Here's what you do

- Compute the gradient at each point in the image
- Create the *H* matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response (λ_- > threshold)
- Choose those points where λ_- is a local maximum as features



λ_-

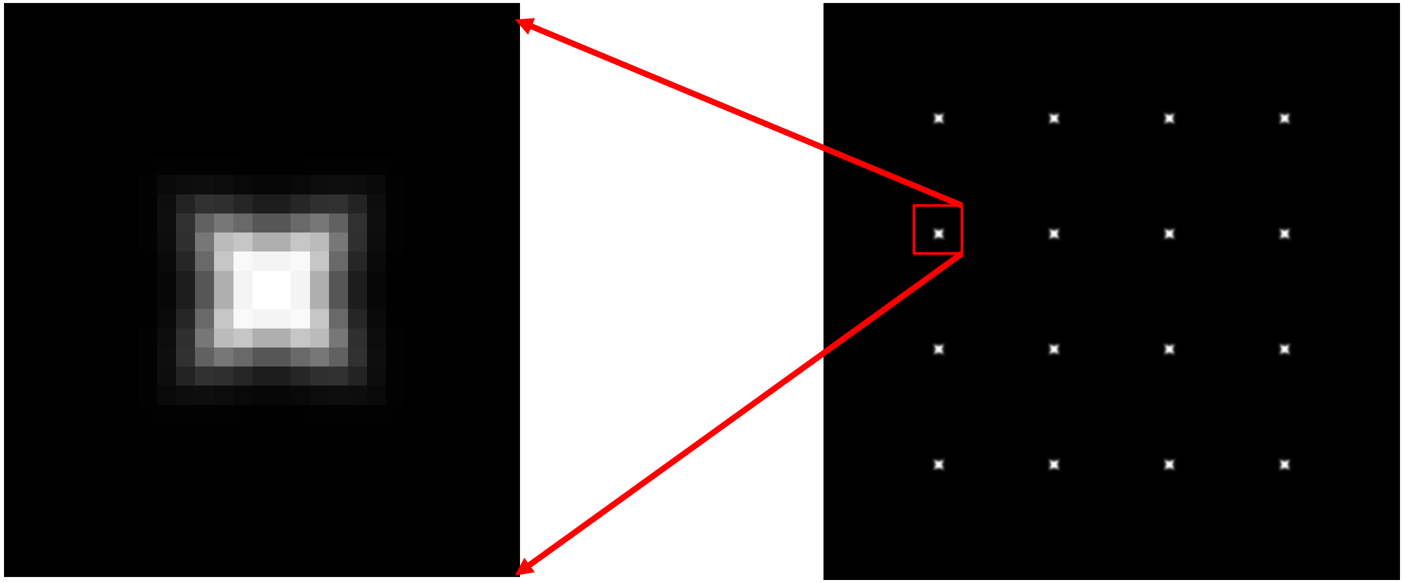
The Harris operator

λ_- is a variant of the "Harris operator" for feature detection

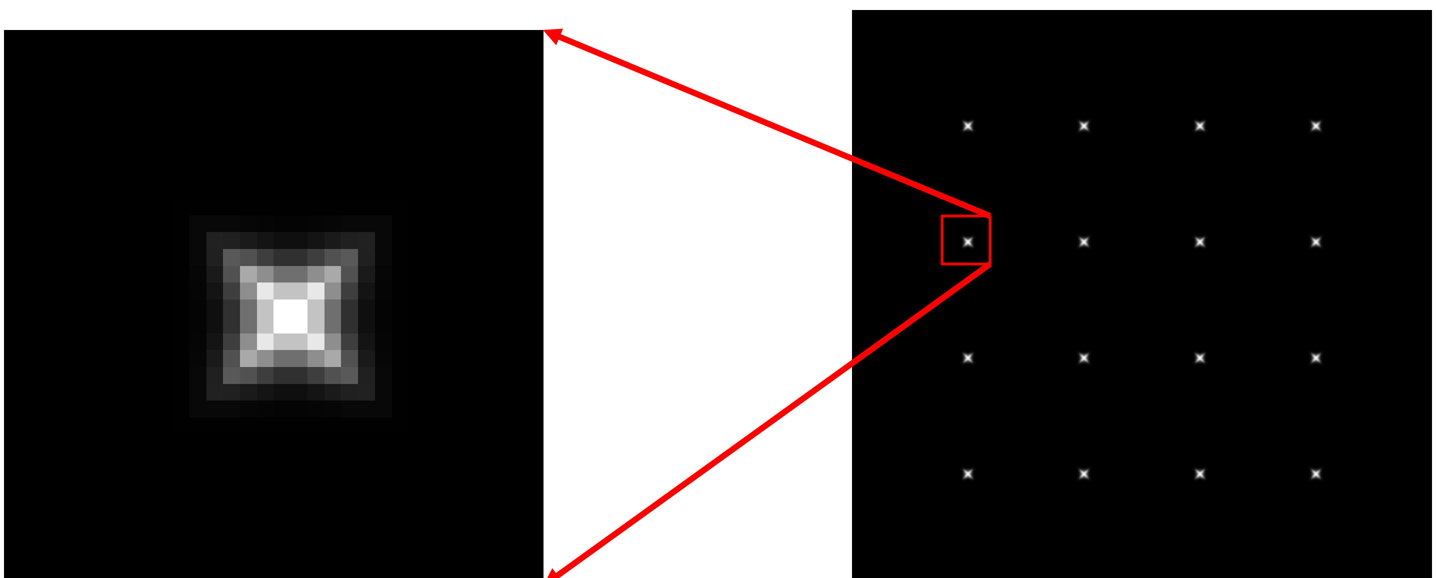
$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{determinant(H)}{trace(H)}$$

- The *trace* is the sum of the diagonals, i.e., $trace(H) = h_{11} + h_{22}$
- Called the "Harris Corner Detector" or "Harris Operator"
- Lots of other detectors, this is one of the most popular

The Harris operator



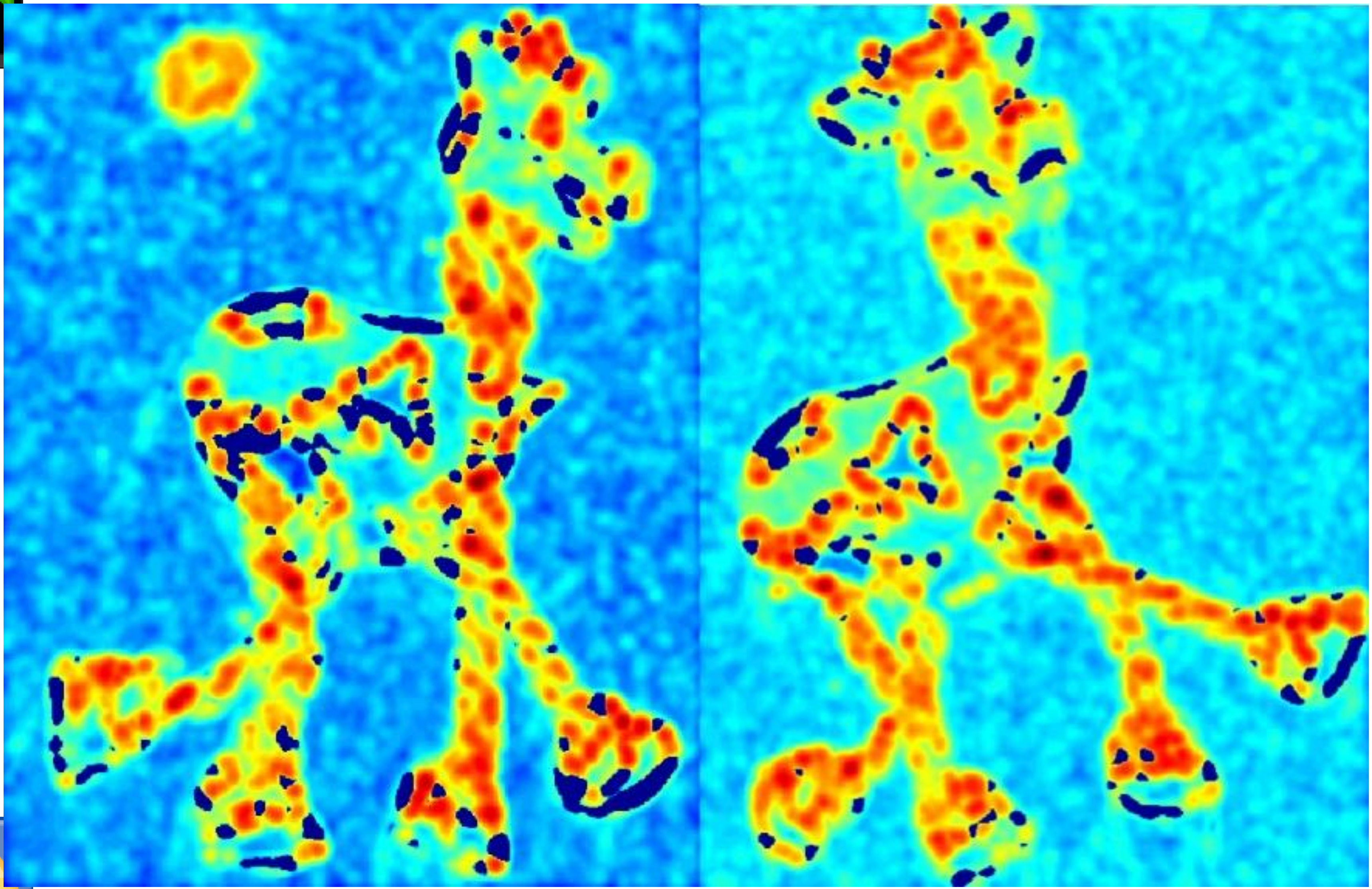
Harris operator



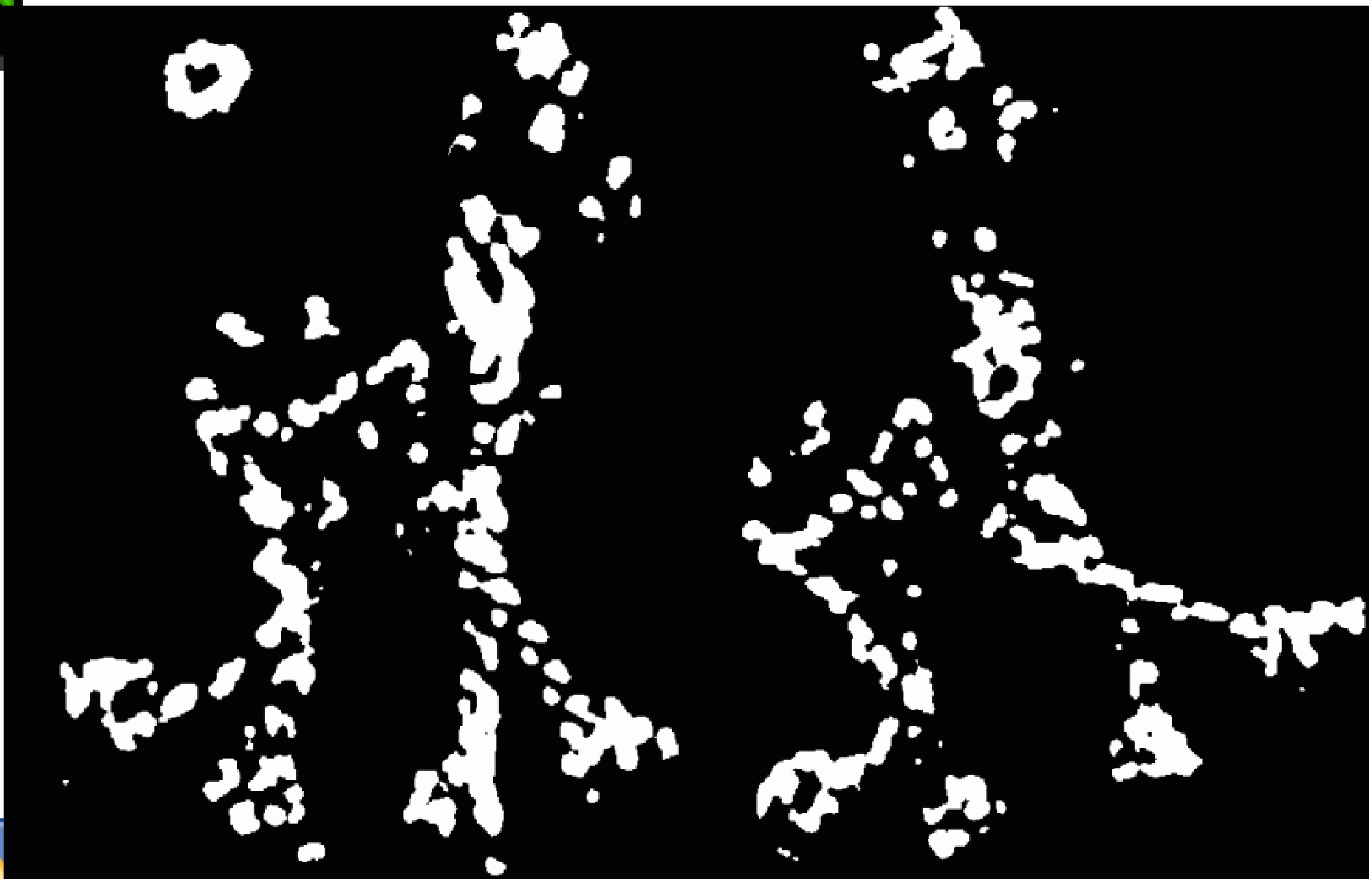
λ_-

Harris detector example

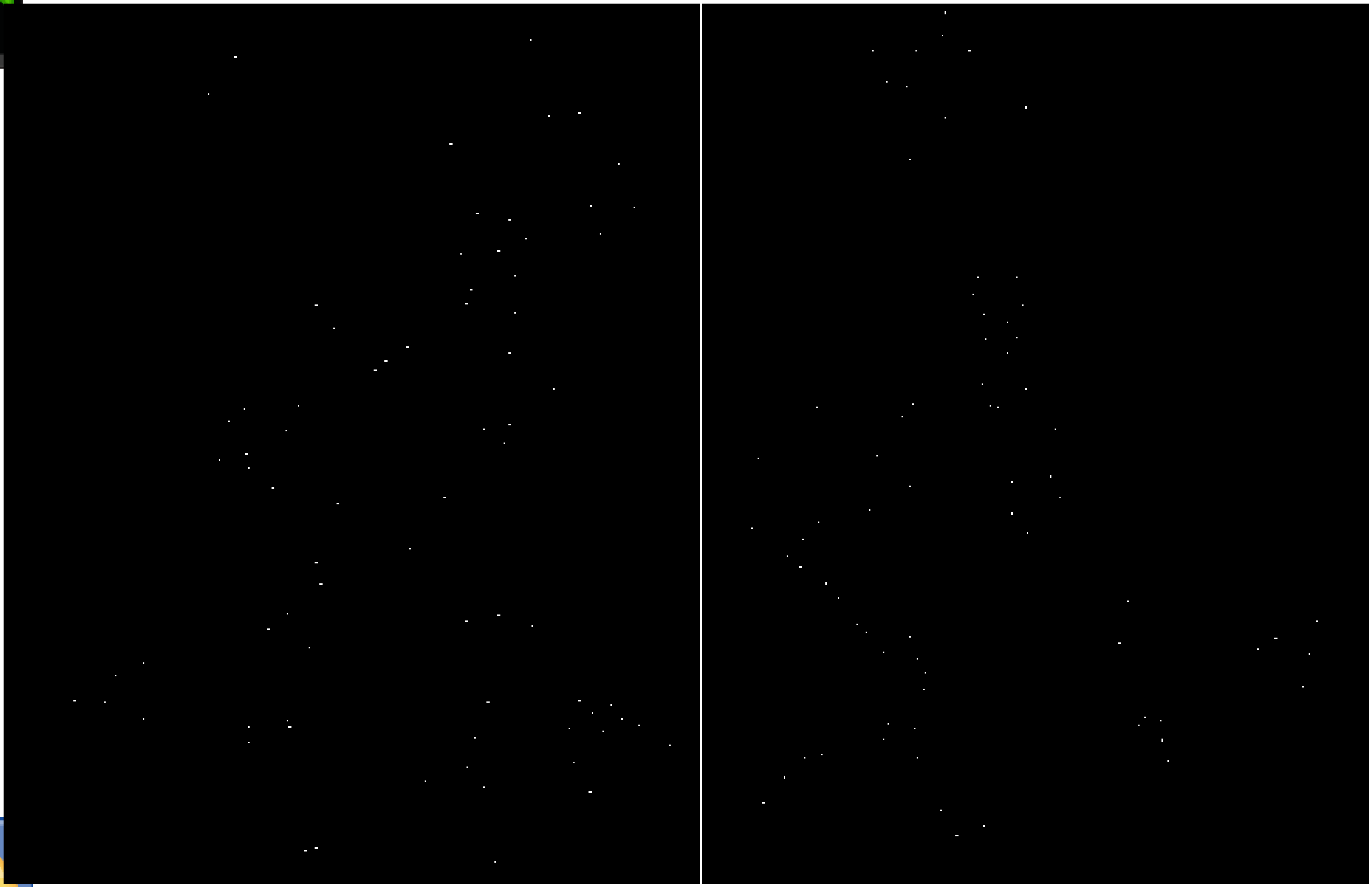
f value (red high, blue low)



Threshold (f > value)



Find local maxima of f



Harris features (in red)

The tops of the horns are detected in both images

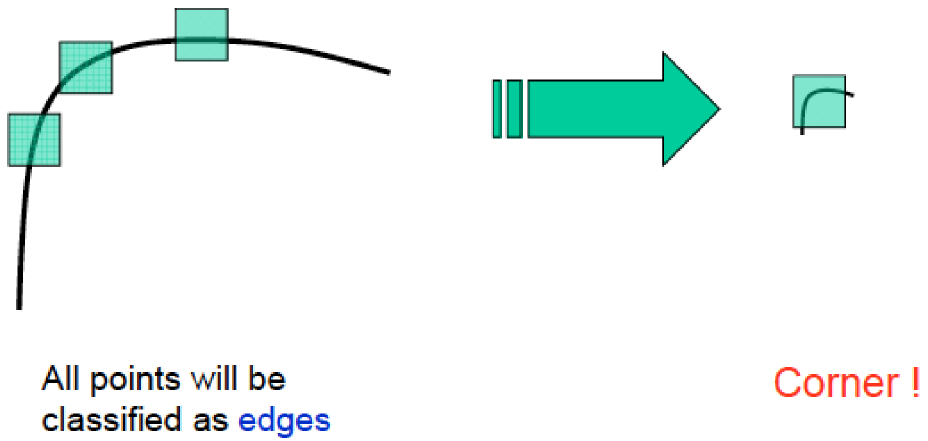
Invariance

Suppose you **rotate** the image by some angle

- Will you still pick up the same features?

What if you change the brightness?

Scale?



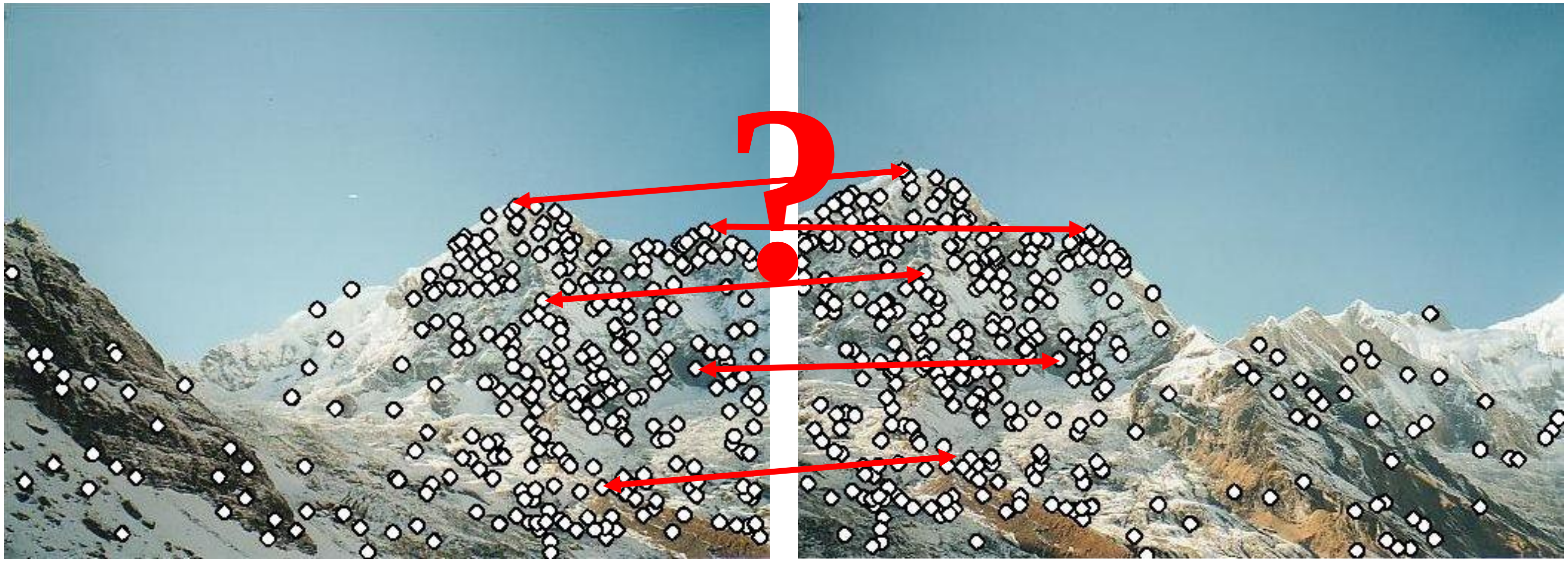
Scale-invariant Interest Points

- How can we independently select interest points in each image, such that the detections are repeatable across different scales?

Will not be covered in this tutorial

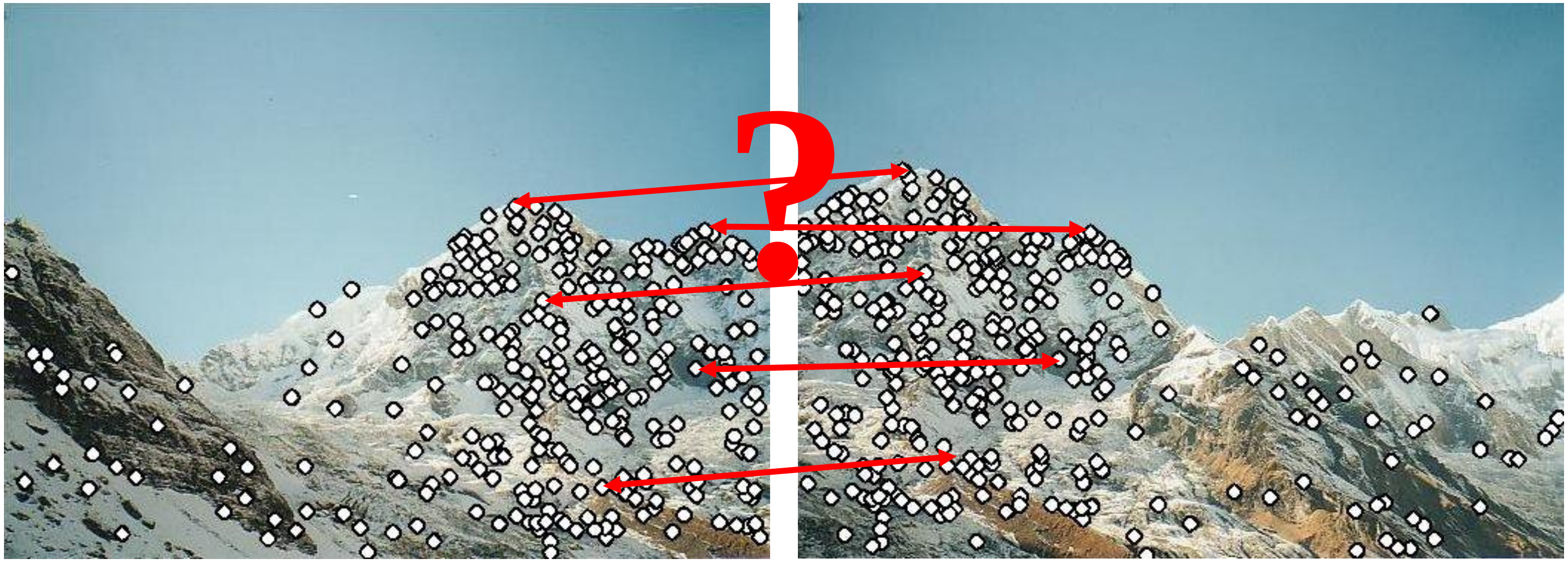
Feature descriptors

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



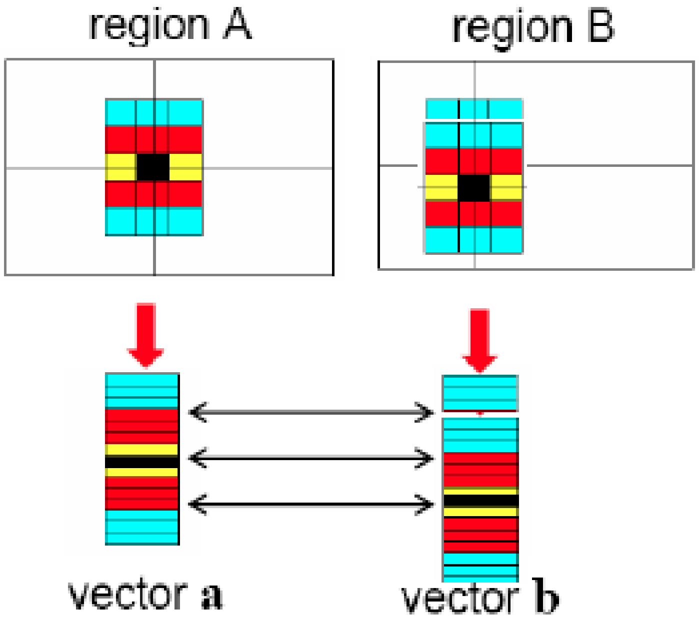
Feature descriptors

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



- Simple option: match square windows around the point
  - Better approach: SIFT
    - – David Lowe, UBC http://www.cs.ubc.ca/~lowe/keypoints/

Raw image patches as descriptors



The simplest way to describe the neighborhood around an interest point is to write down the list of intensities to form a feature vector.

But this is very sensitive to even small shifts, rotations.

Scale Invariant Feature Transform

Basic idea:

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations

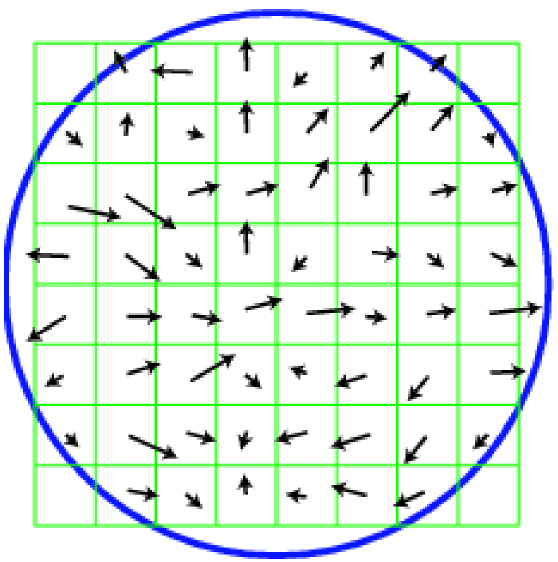
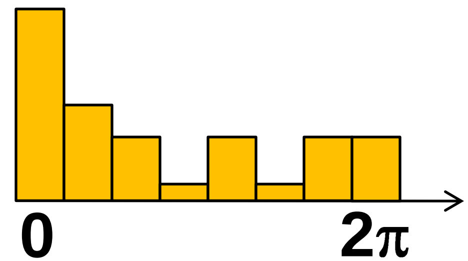
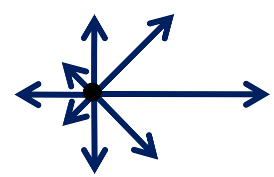


Image gradients



angle histogram

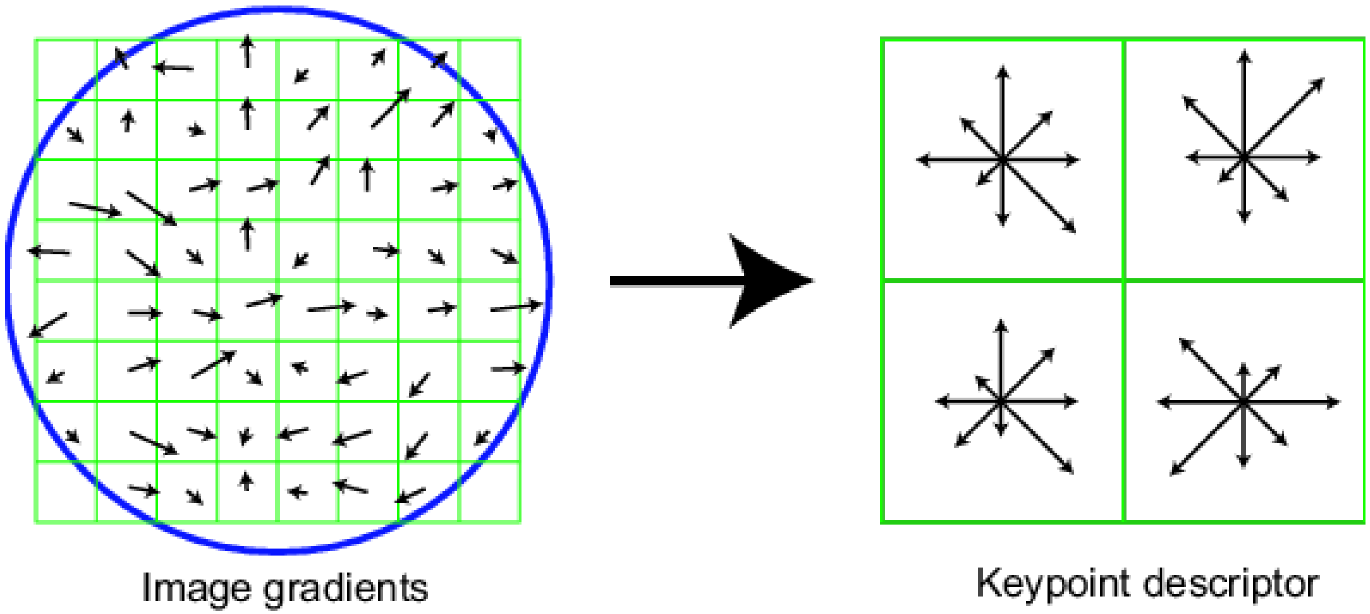


Adapted from slide by David Lowe

SIFT descriptor

Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor

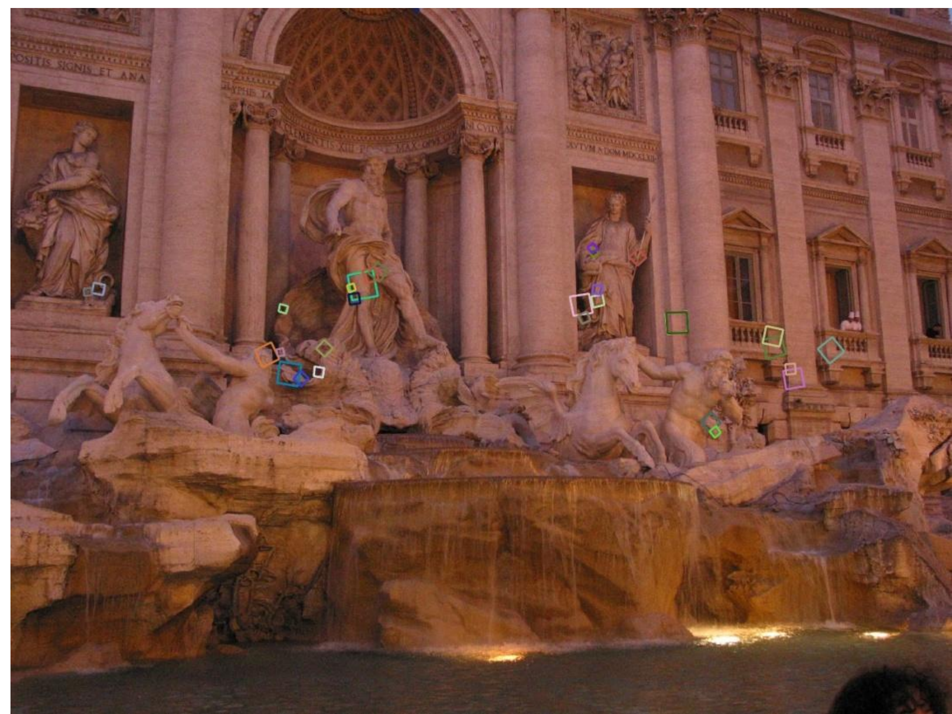


Adapted from slide by David Lowe

Properties of SIFT

Extraordinarily robust matching technique

- Can handle changes in viewpoint
  - Up to about 60 degree out of plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night (below)
- Fast and efficient—can run in real time
- Lots of code available
  - http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT



Feature matching

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

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

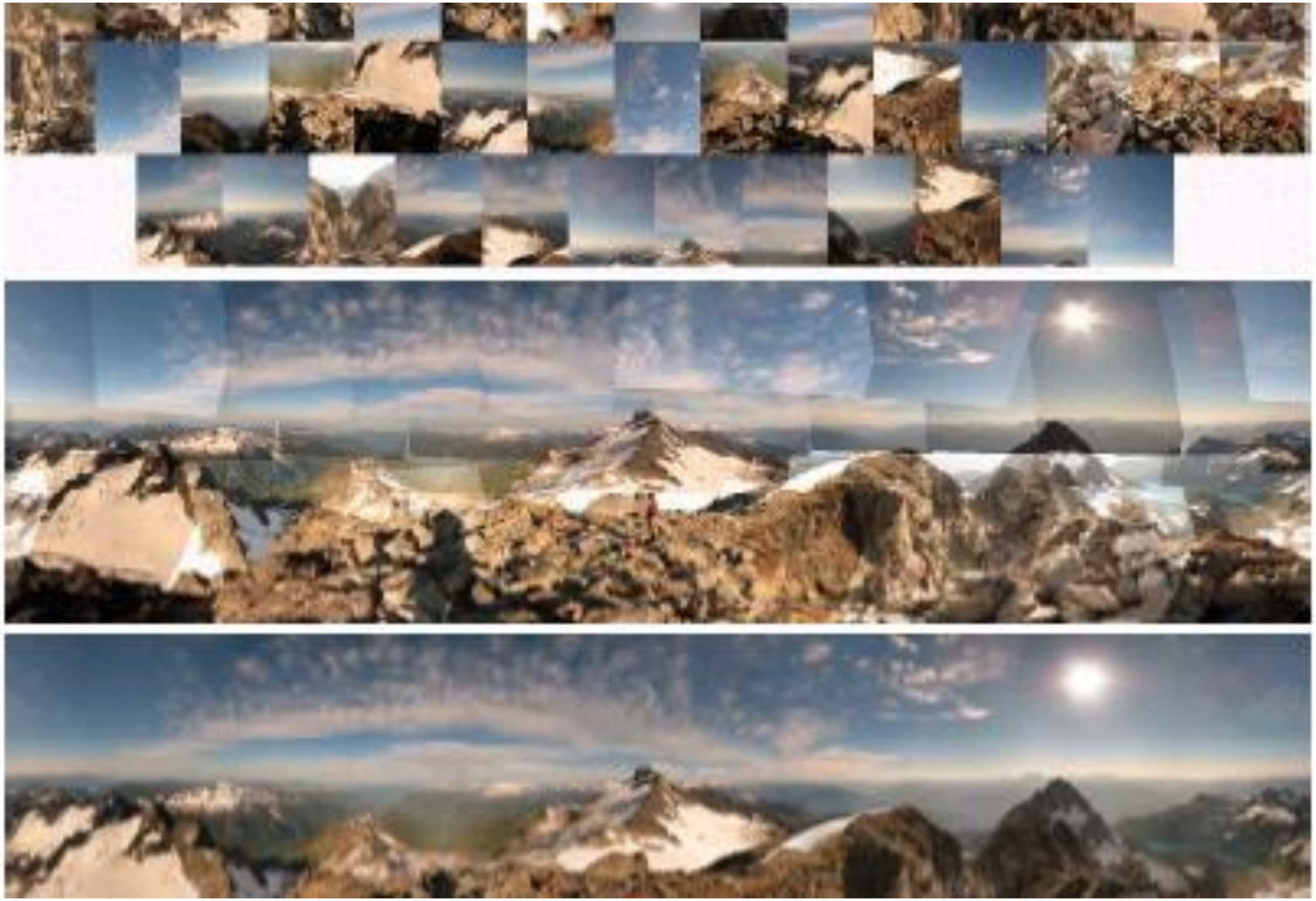
Will not be covered in this tutorial

Lots of applications

Features are used for:

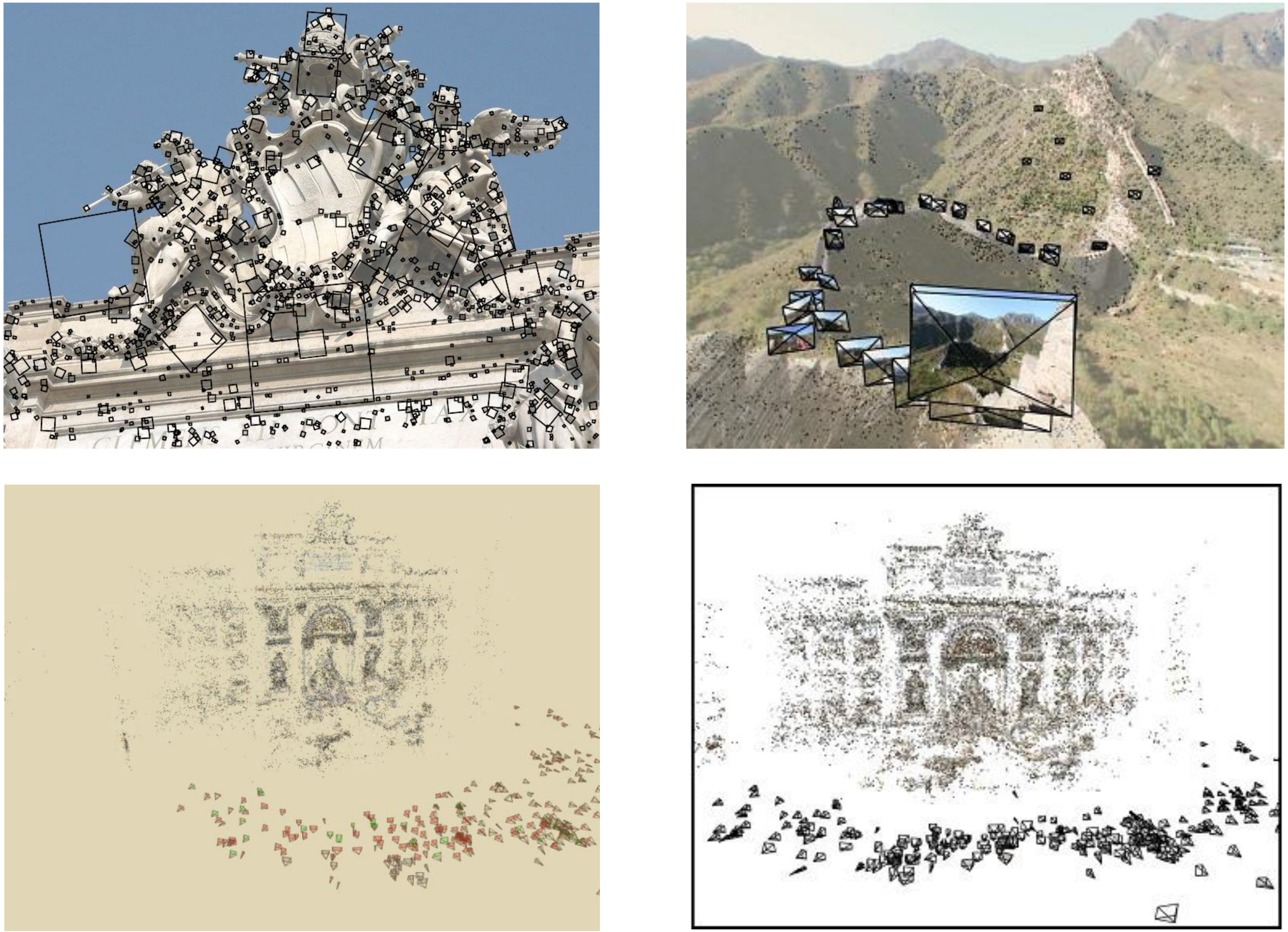
- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation
- … other

Automatic Mosaicing



Wide baseline stereo

Geometry Estimation



Snavely, Seitz, & Szeliski 2006

Object Recognition

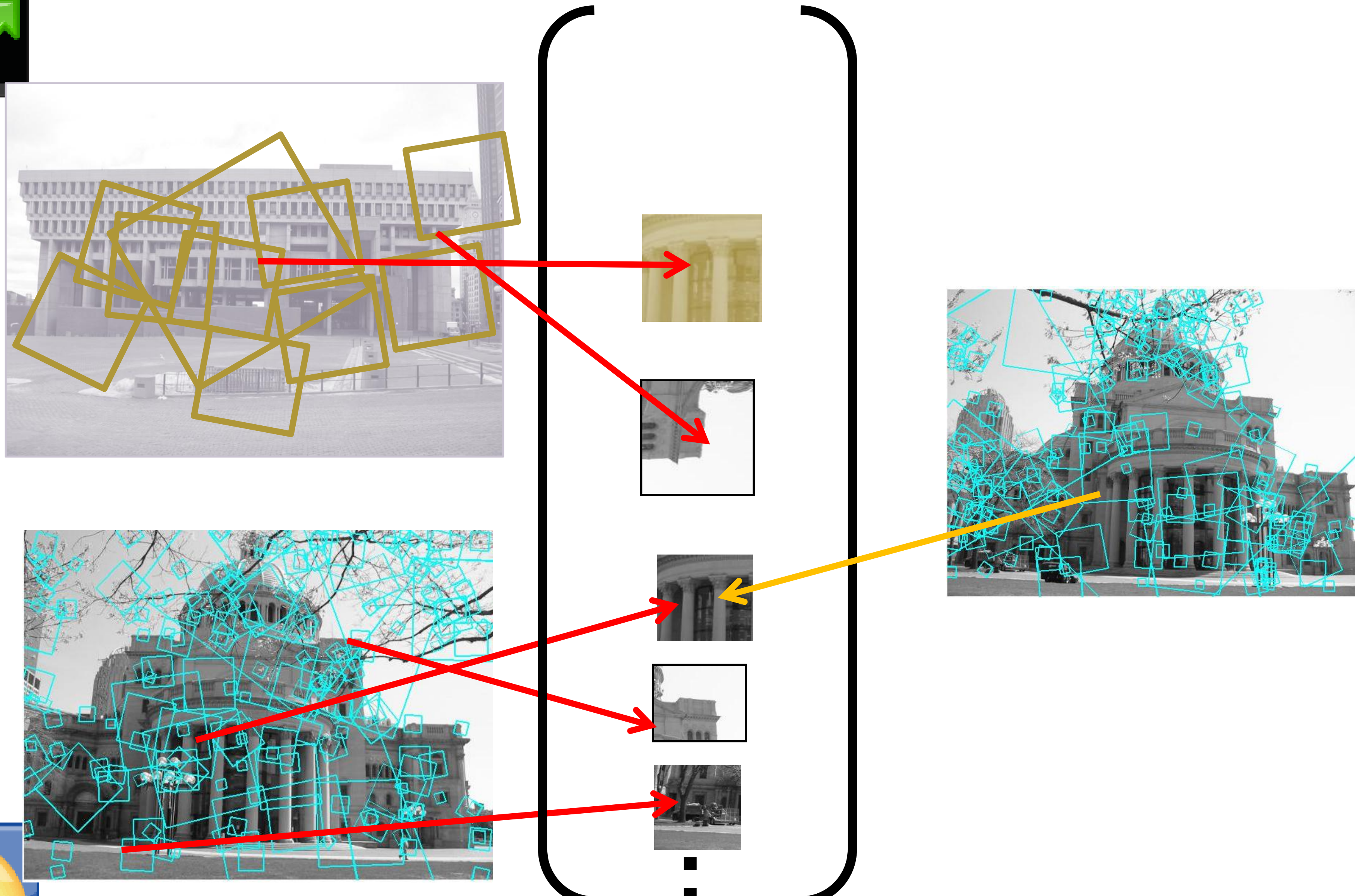
Schmid and Mohr 1997

Sivic and Zisserman, 2003

Rothganger et al. 2003

Lowe 2002

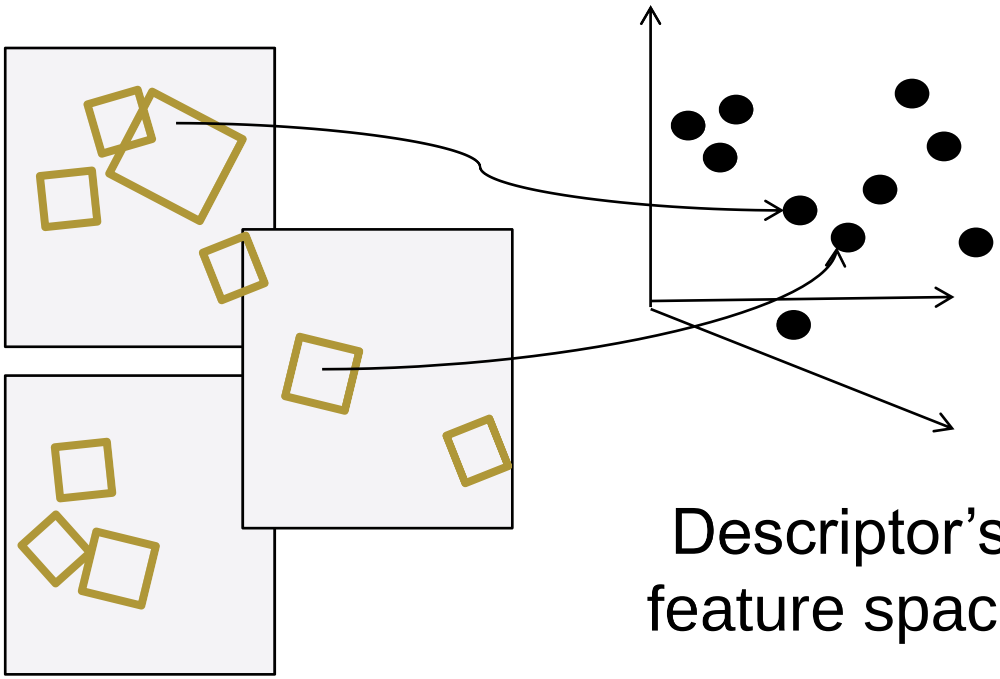
Indexing local features



Kristen Grauman

Indexing local features

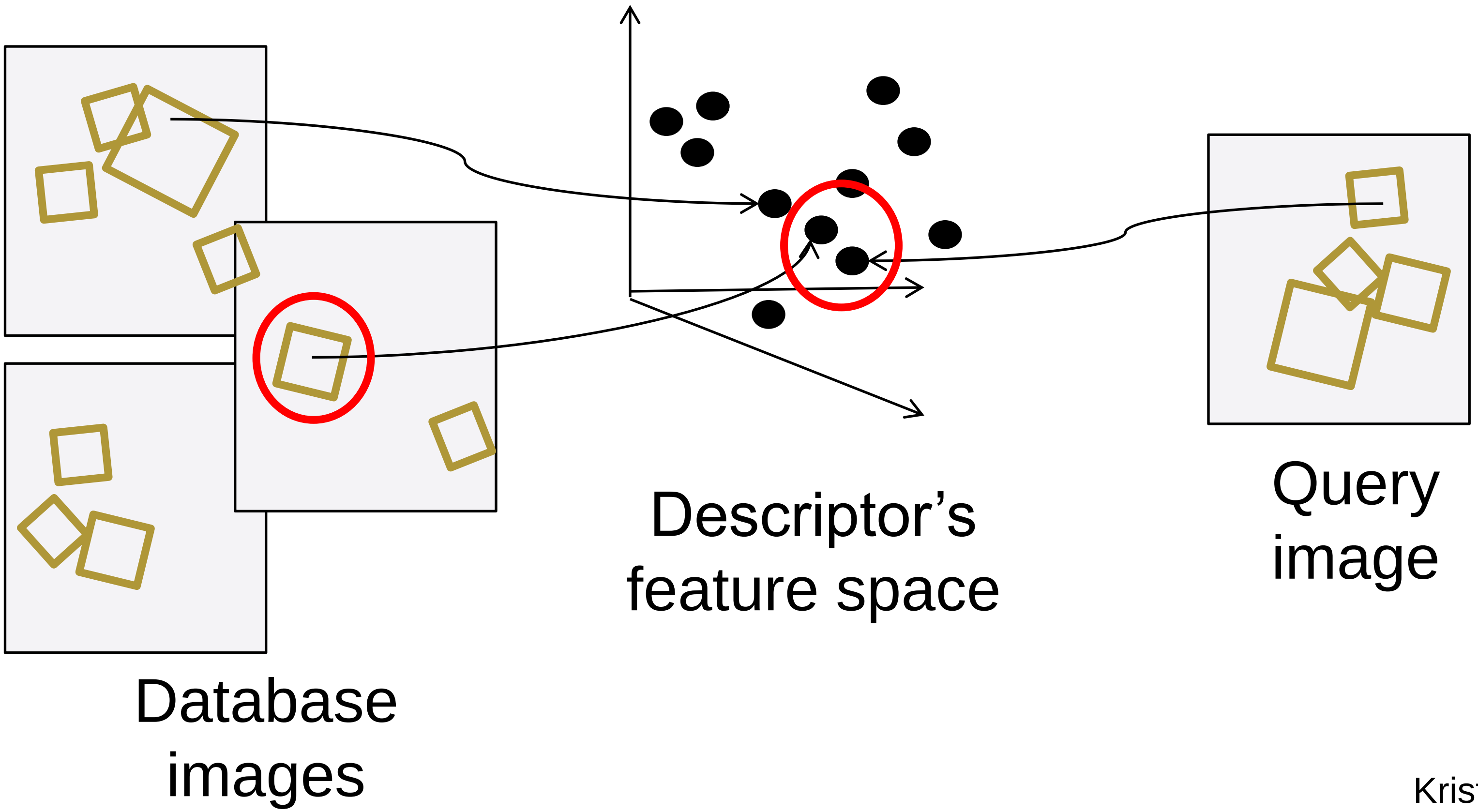
- Each patch / region has a descriptor, which is a point in some high-dimensional feature space (e.g., SIFT)



Descriptor's feature space

Indexing local features

When we see close points in feature space, we have similar descriptors, which indicates similar local content.



Kristen Grauman

Indexing local features

With potentially thousands of features per image, and hundreds to millions of images to search, how to efficiently find those that are relevant to a new image?

Kristen Grauman

Indexing local features: inverted file index

Index

For text documents, an efficient way to find all *pages* on which a *word* occurs is to use an index…

We want to find all *images* in which a *feature* occurs.

To use this idea, we'll need to map our features to "visual words".

Kristen Grauman

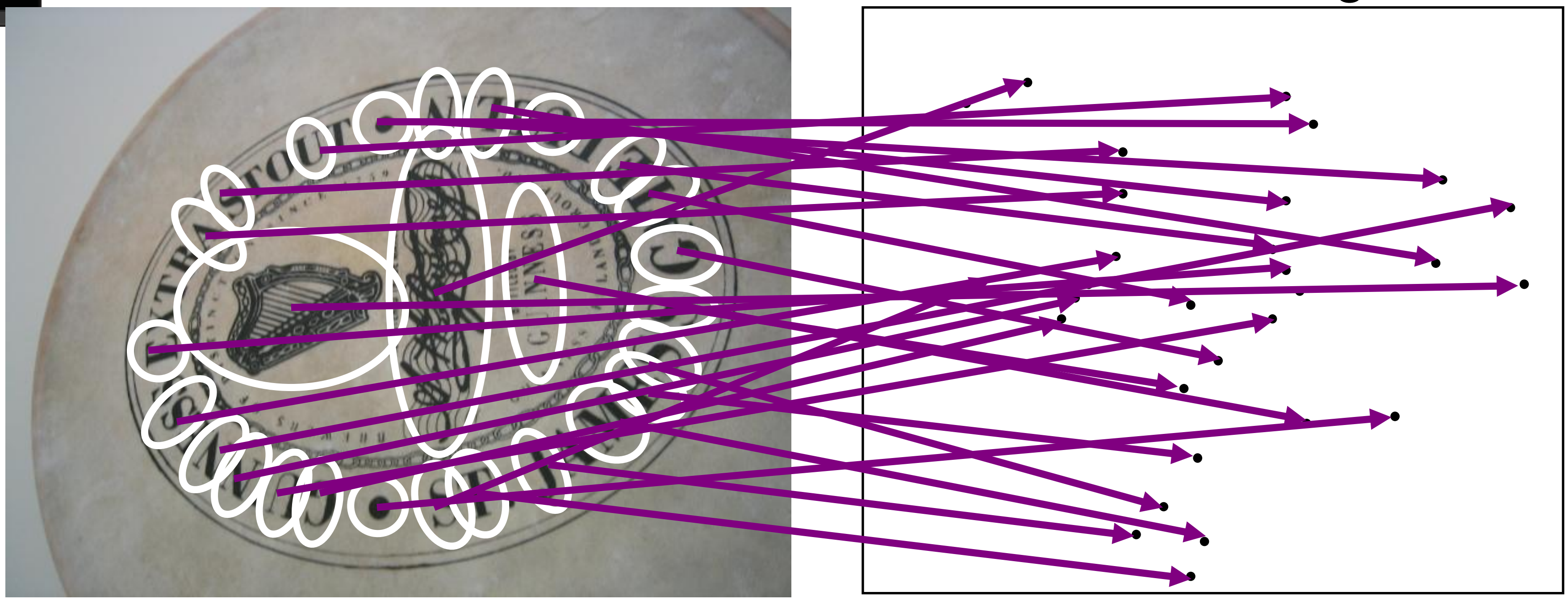
Text retrieval vs. image search

What makes the problems similar, different?

Kristen Grauman

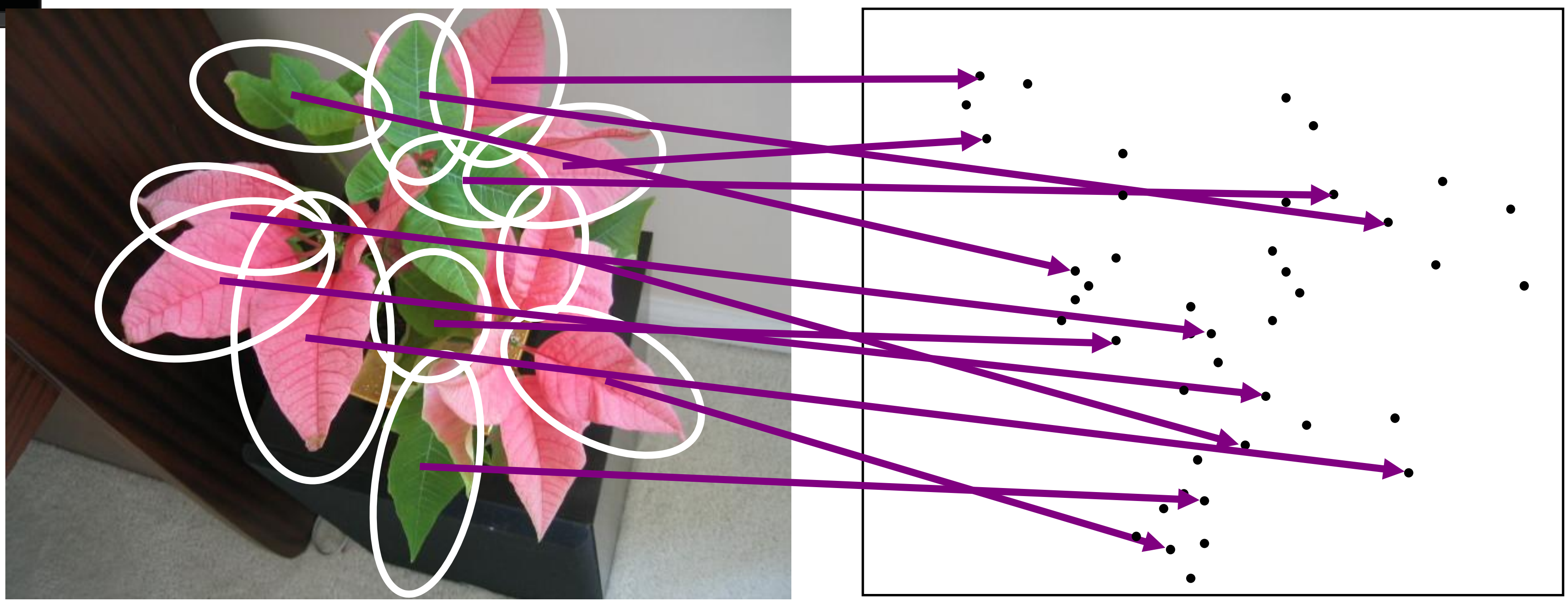
Visual words: main idea

- Extract some local features from a number of images …

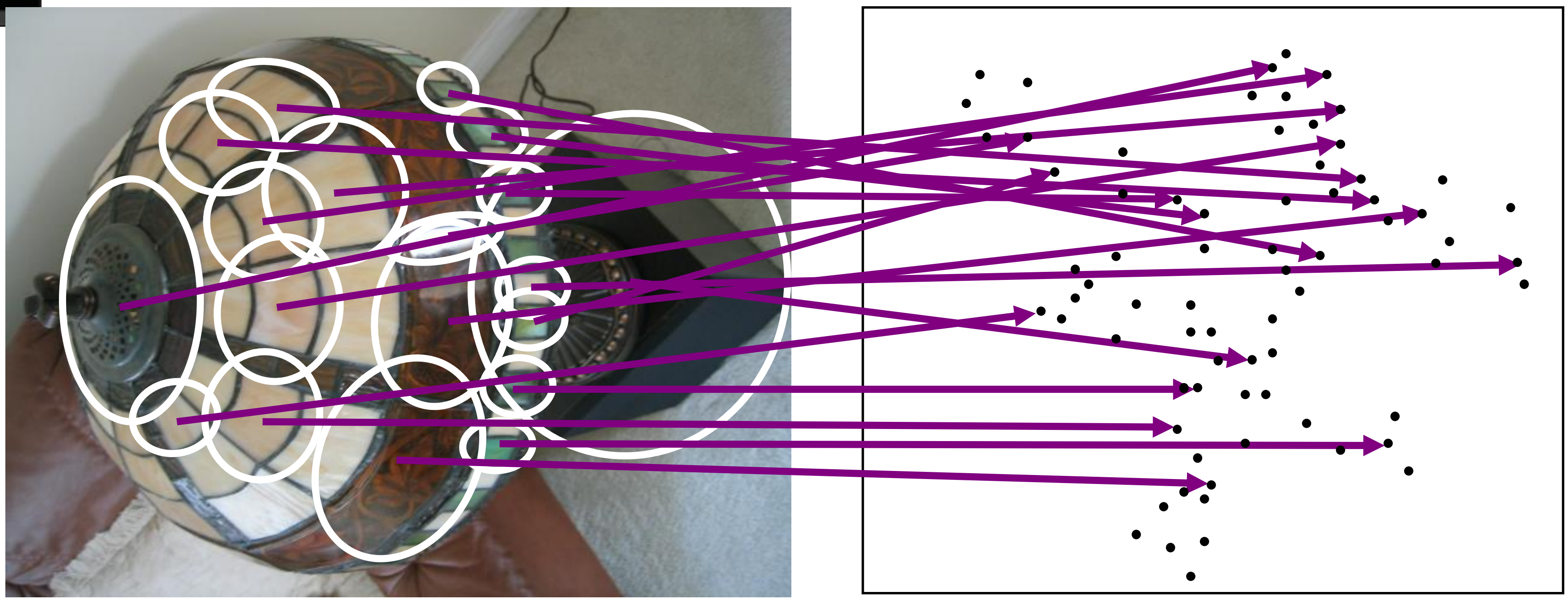


e.g., SIFT descriptor space: each point is 128-dimensional

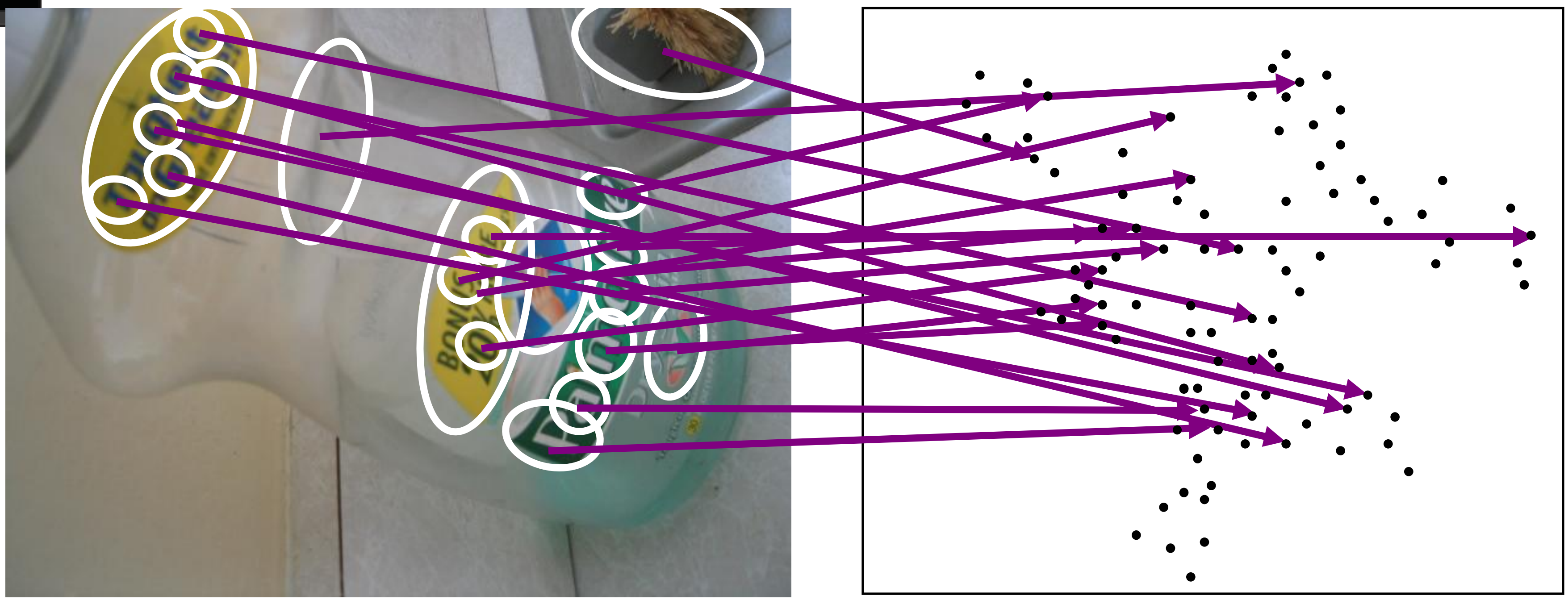
Visual words: main idea

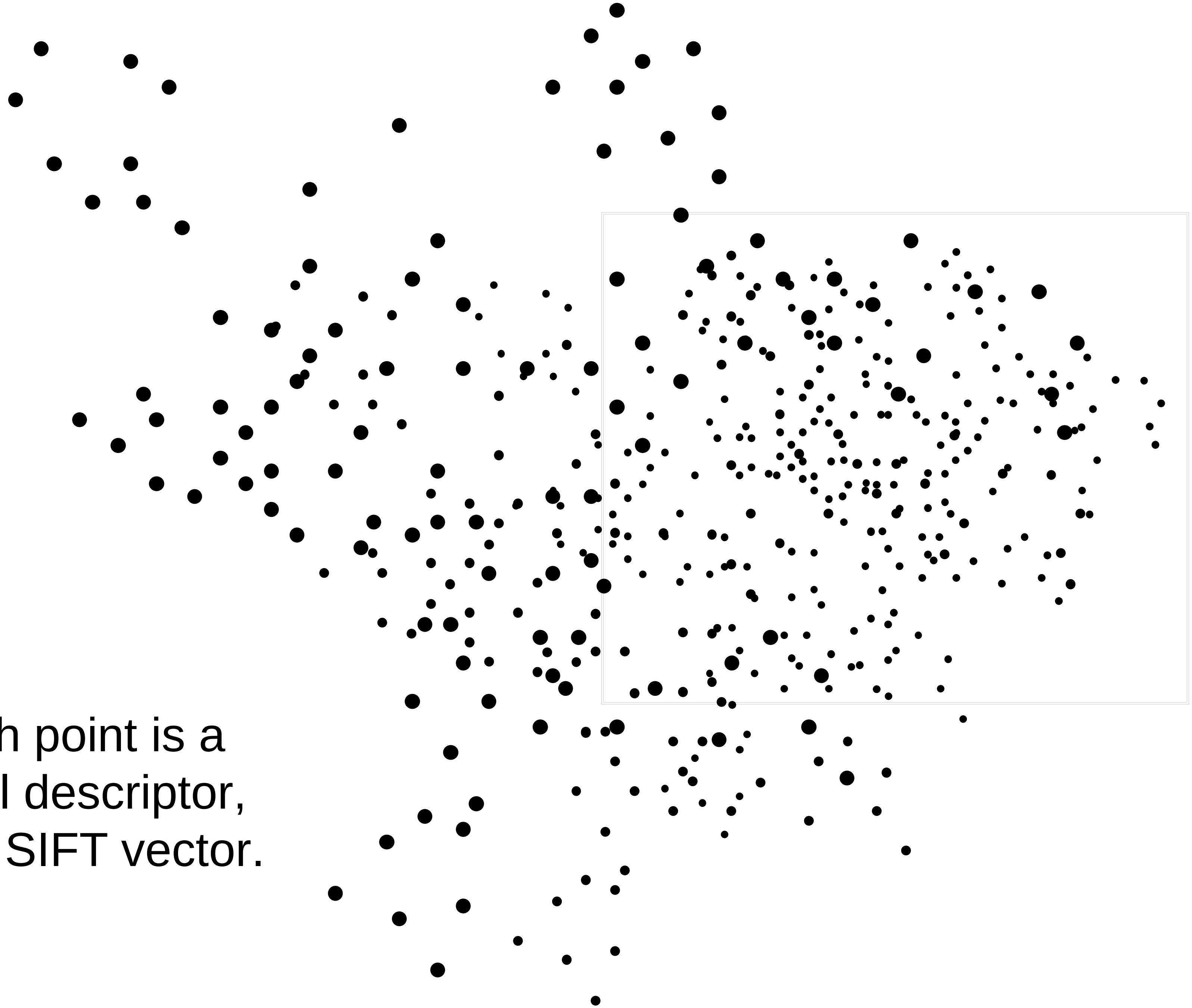


Visual words: main idea

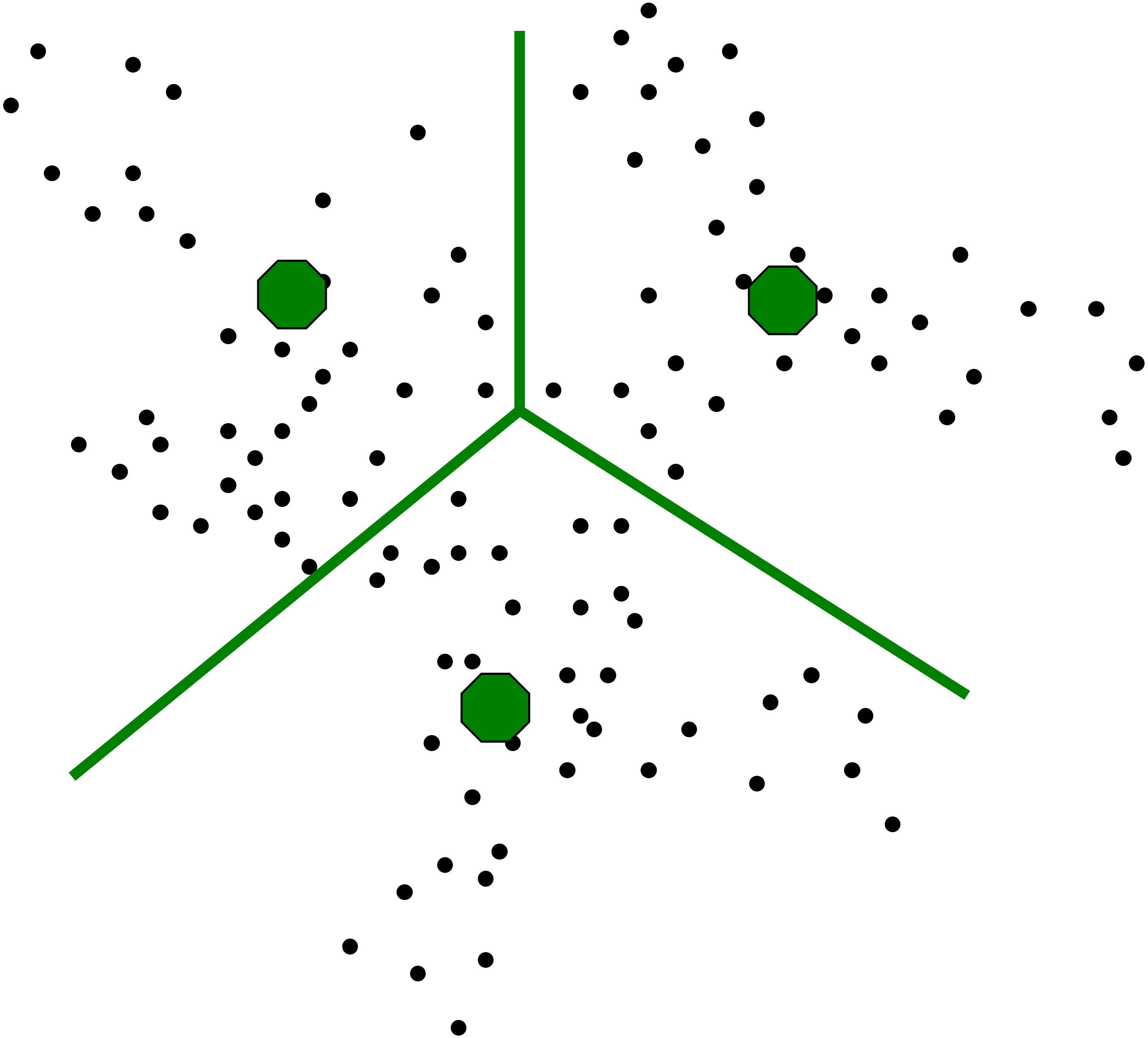


Visual words: main idea



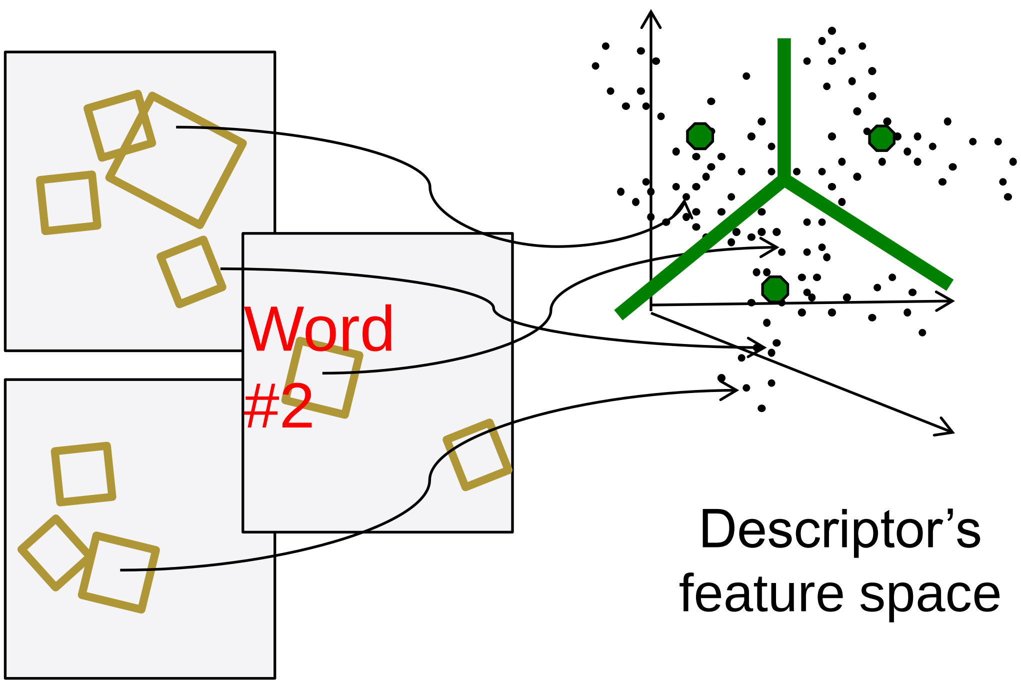


Each point is a
local descriptor,
e.g. SIFT vector.



Visual words

Map high-dimensional descriptors to tokens/words by quantizing the feature space



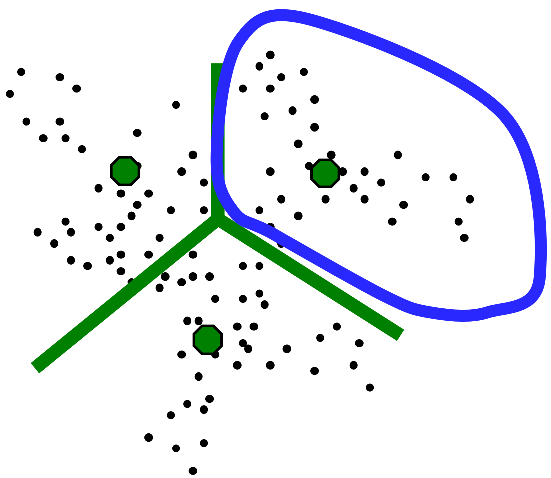
- Quantize via clustering, let cluster centers be the prototype "words"

- Determine which word to assign to each new image region by finding the closest cluster center.

Kristen Grauman

Visual words

Example: each group of patches belongs to the same visual word



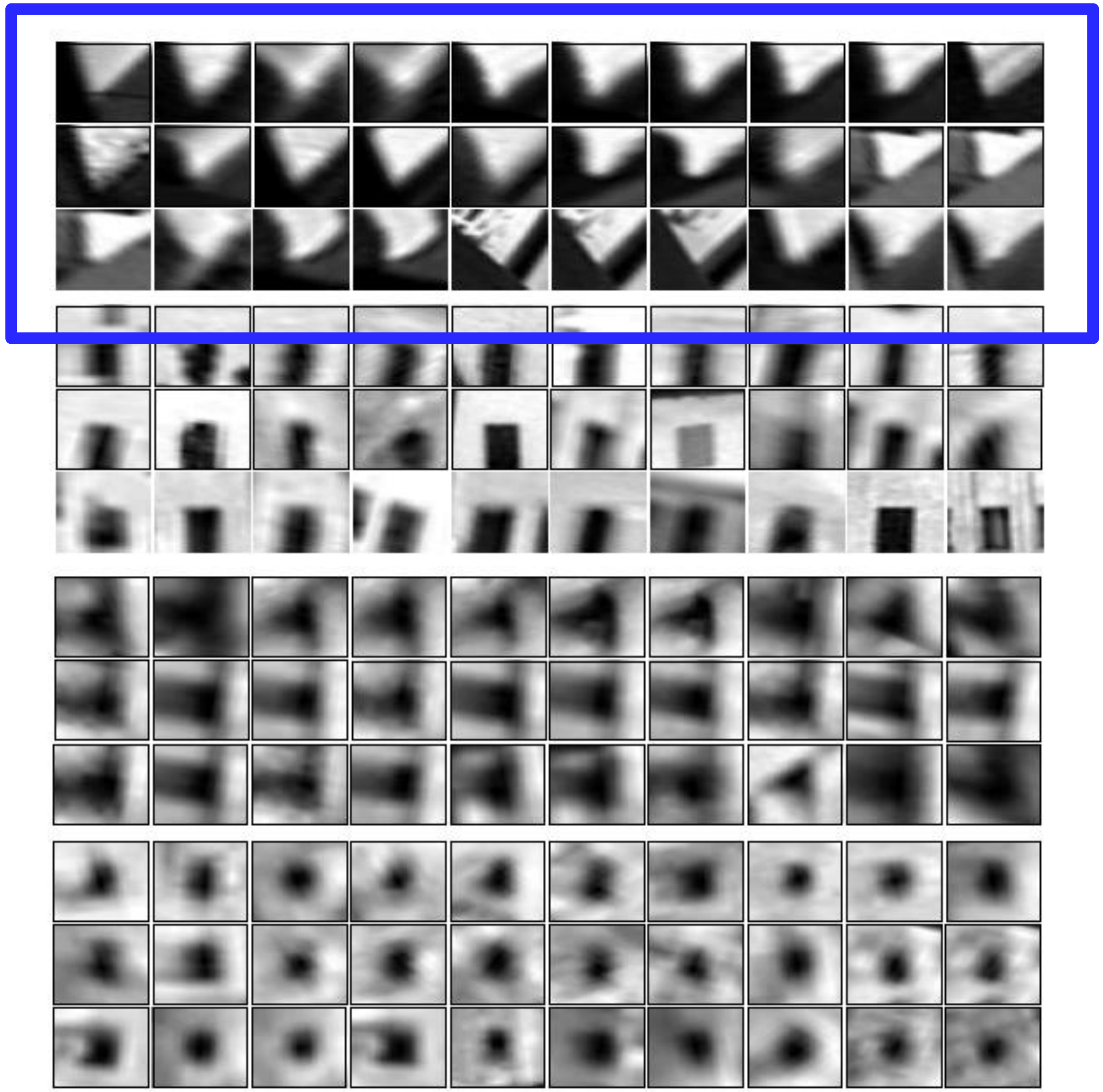


Figure from Sivic & Zisserman, ICCV 2003

Kristen Grauman

Visual vocabulary formation

Issues:

Sampling strategy: where to extract features?

Clustering / quantization algorithm

Unsupervised vs. supervised

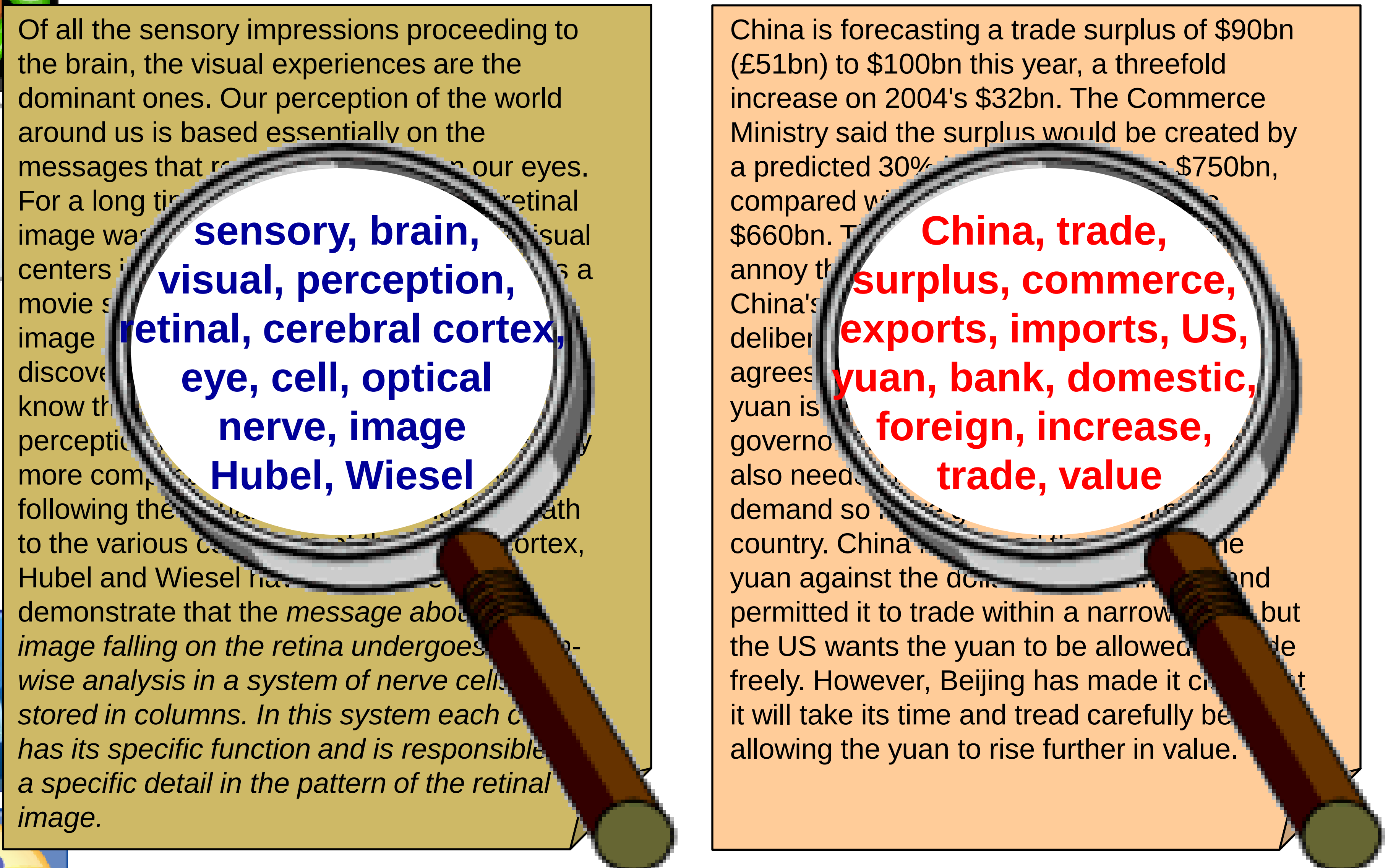
What corpus provides features (universal vocabulary?)

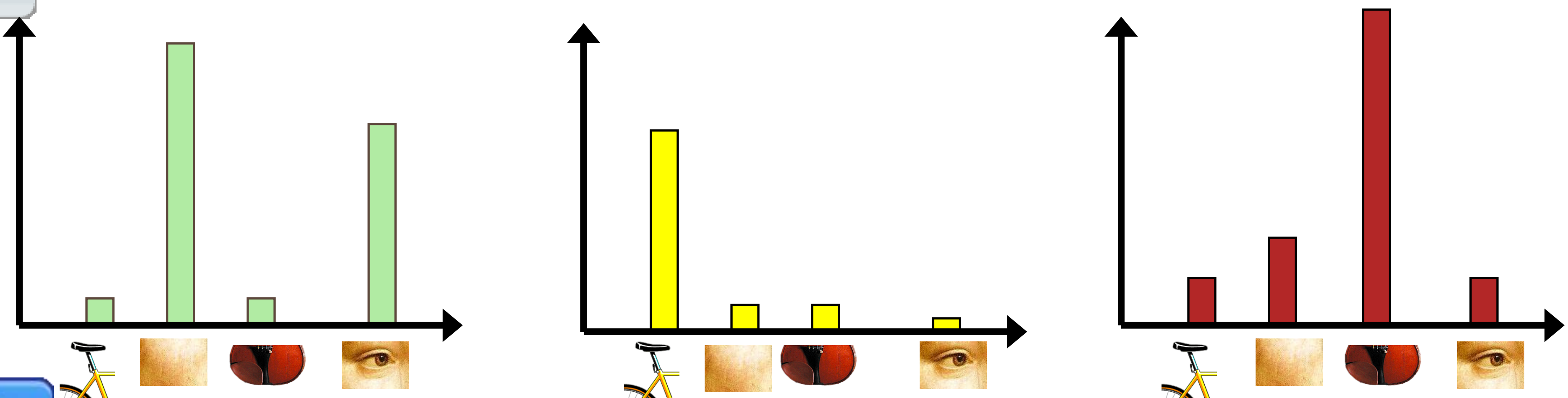
Vocabulary size, number of words

Kristen Grauman

If a local image region is a visual word, how can we summarize an image (the document)?

Analogy to documents

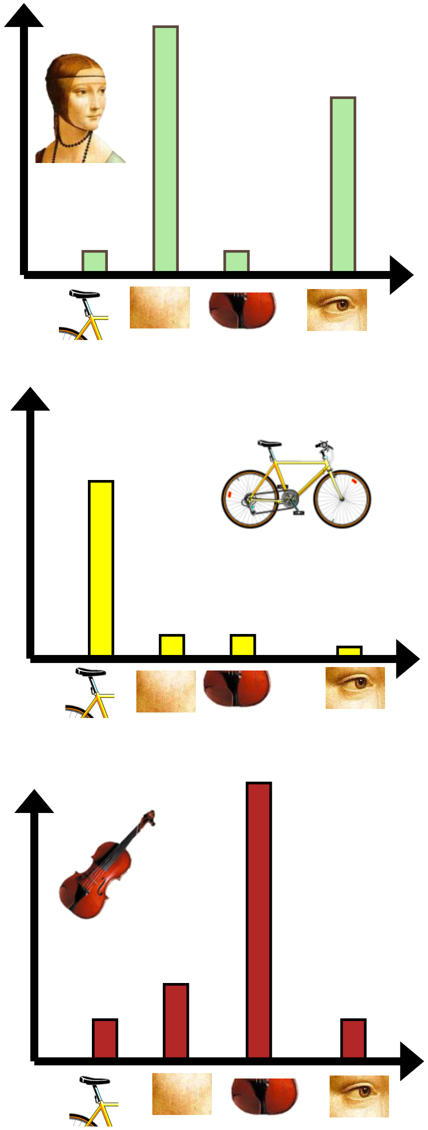




Bags of visual words

Summarize entire image based on its distribution (histogram) of word occurrences.

Analogous to bag of words representation commonly used for documents.



Bags of words for content-based image retrieval

Visually defined query

“Groundhog Day” [Rammis, 1993]

“Find this clock”

“Find this place”

Slide from Andrew Zisserman
Sivic & Zisserman, ICCV 2003

Example

retrieved shots

- Slide from Andrew Zisserman
Sivic & Zisserman, ICCV 2003

Bags of words: pros and cons

+ flexible to geometry / deformations / viewpoint
+ compact summary of image content
+ provides vector representation for sets
+ very good results in practice

- basic model ignores geometry – must verify afterwards, or encode via features
- background and foreground mixed when bag covers whole image
- optimal vocabulary formation remains unclear