

UNIVERSIDAD  
**NACIONAL**  
DE COLOMBIA

# **CLASIFICACIÓN Y RECONOCIMIENTO DE PATRONES**

## **Extracción de Características y Reducción de la Dimensionalidad**

**Sesión 2**

**Jorge E. Espinosa**

**Profesor**

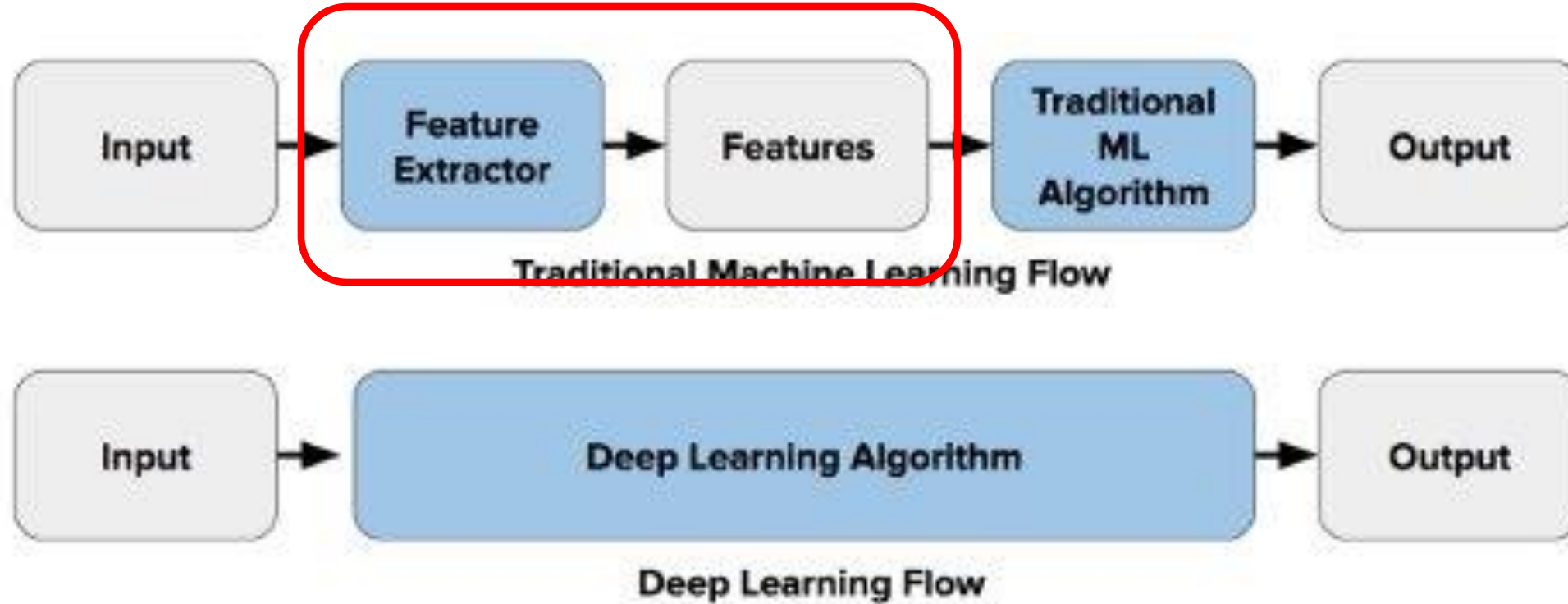
**Departamento de Ciencias de la Computación y de la Decisión**

**Investigador del Grupo de I+D en Inteligencia Artificial – GIDIA**

**[jeespinosao@unal.edu.co](mailto:jeespinosao@unal.edu.co)**

# Computer Vision Workflow

---



# Copy Right

---

This presentation is developed by Steve Seitz from  
Washington University

# Image matching

---



by [Diva Sian](#)



by [swashford](#)



# Harder case

---



by [Diva Sian](#)



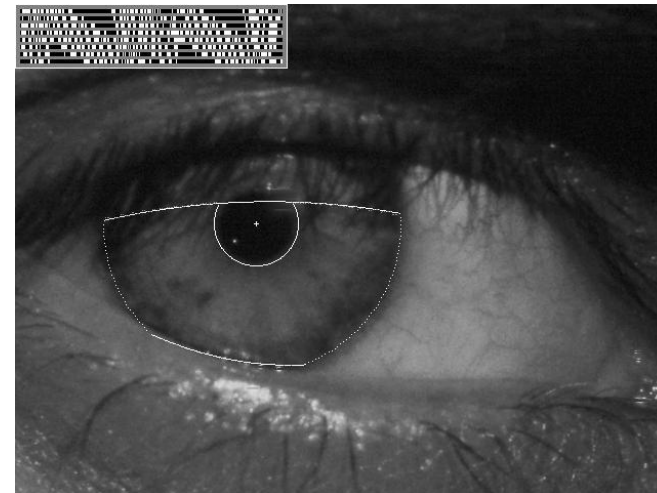
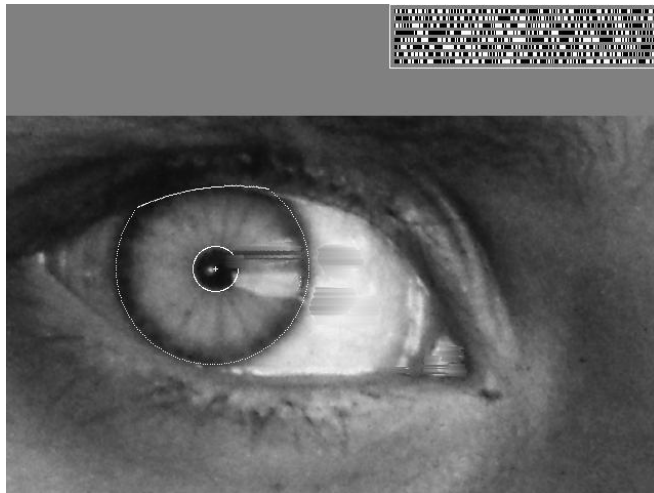
by [scgbt](#)

# Even harder case

---

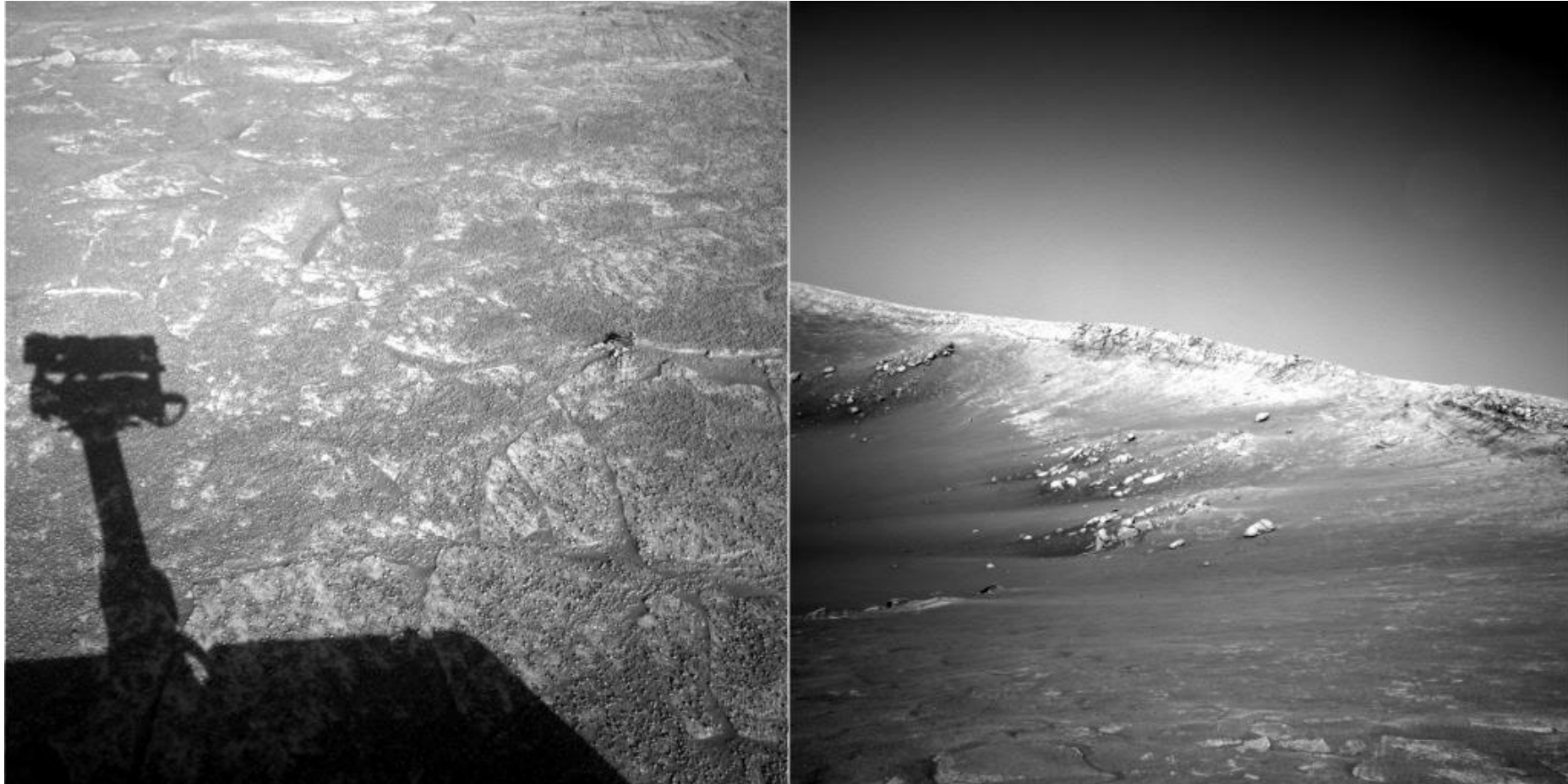


***“How the Afghan Girl was Identified by Her Iris Patterns”*** Read the [story](#)



# Harder still?

---

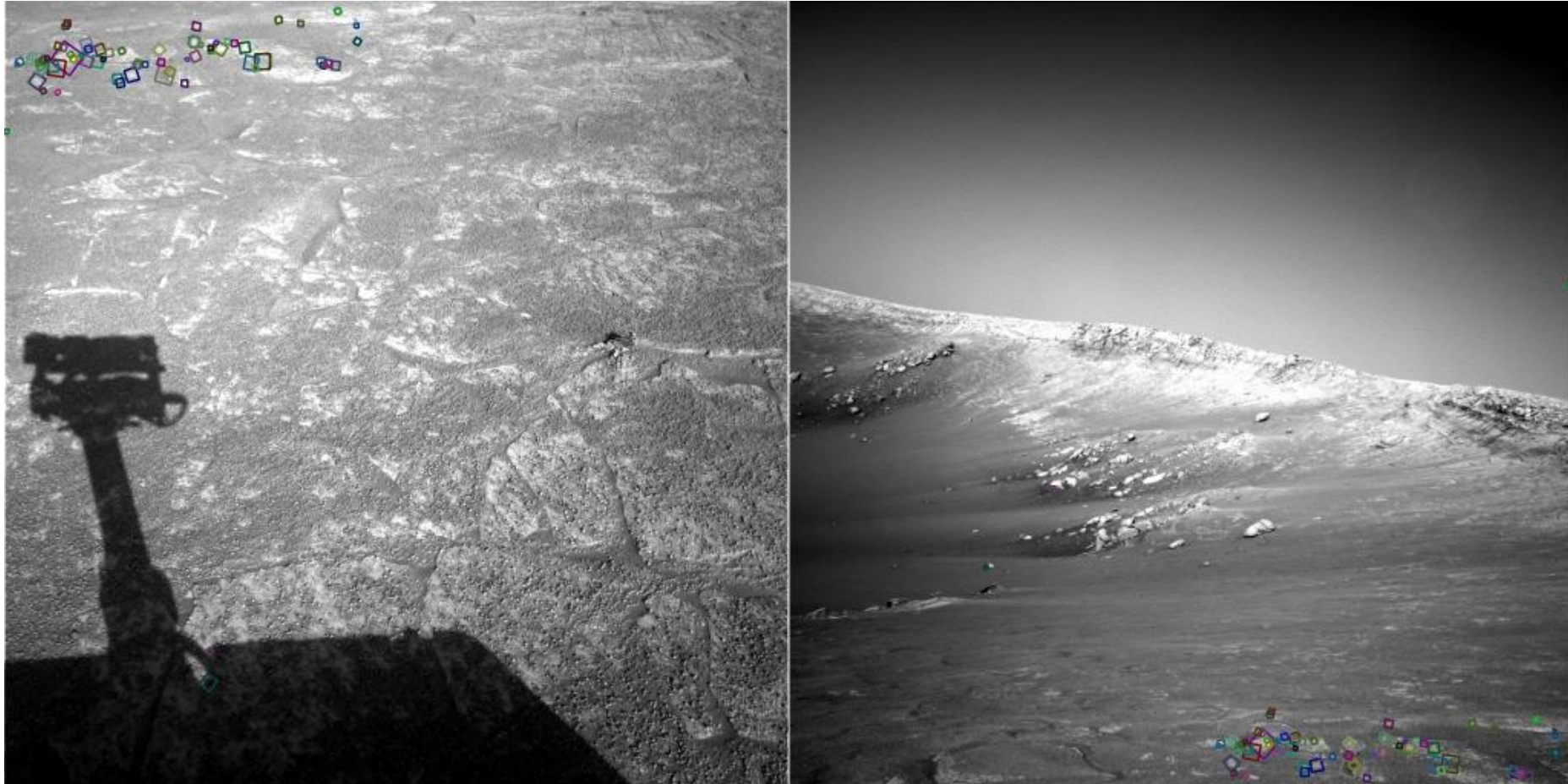


NASA Mars Rover images



# Answer below (look for tiny colored squares...)

---



NASA Mars Rover images  
with SIFT feature matches  
Figure by Noah Snavely

# Features

---



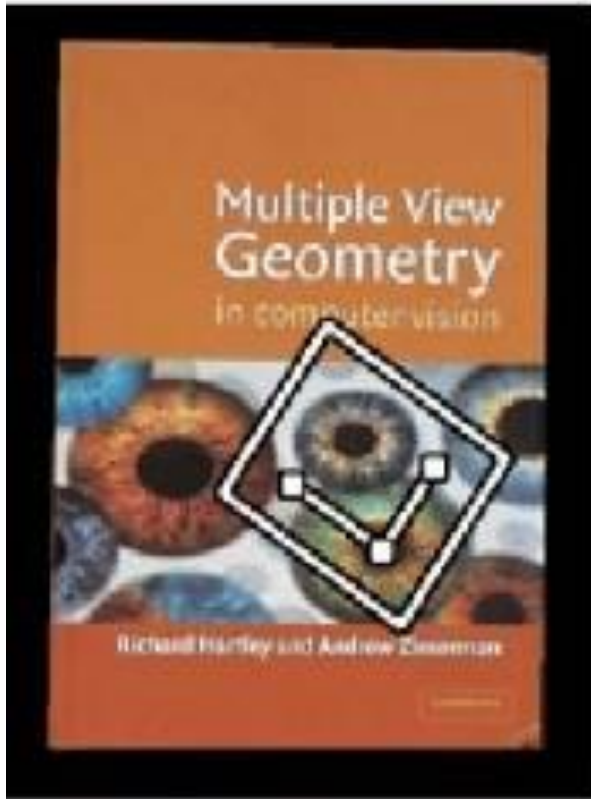
*All is Vanity*, by C. Allan Gilbert, 1873-1929

## Readings

- Szeliski, Ch 4.1
- (optional) K. Mikolajczyk, C. Schmid, A performance evaluation of local descriptors. In PAMI 27(10):1615-1630
  - [http://www.robots.ox.ac.uk/~vgg/research/affine/det\\_eval\\_files/mikolajczyk](http://www.robots.ox.ac.uk/~vgg/research/affine/det_eval_files/mikolajczyk)

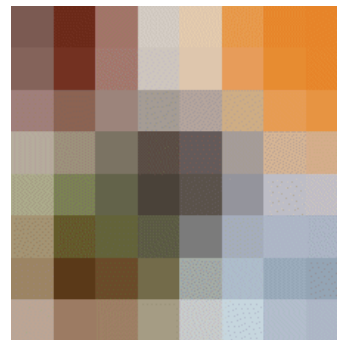
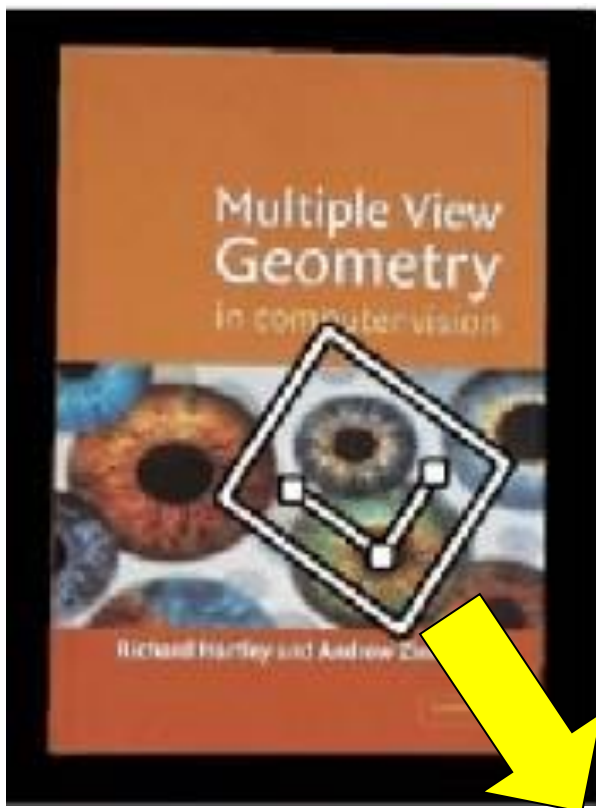
# Image Matching

---



# Image Matching

---



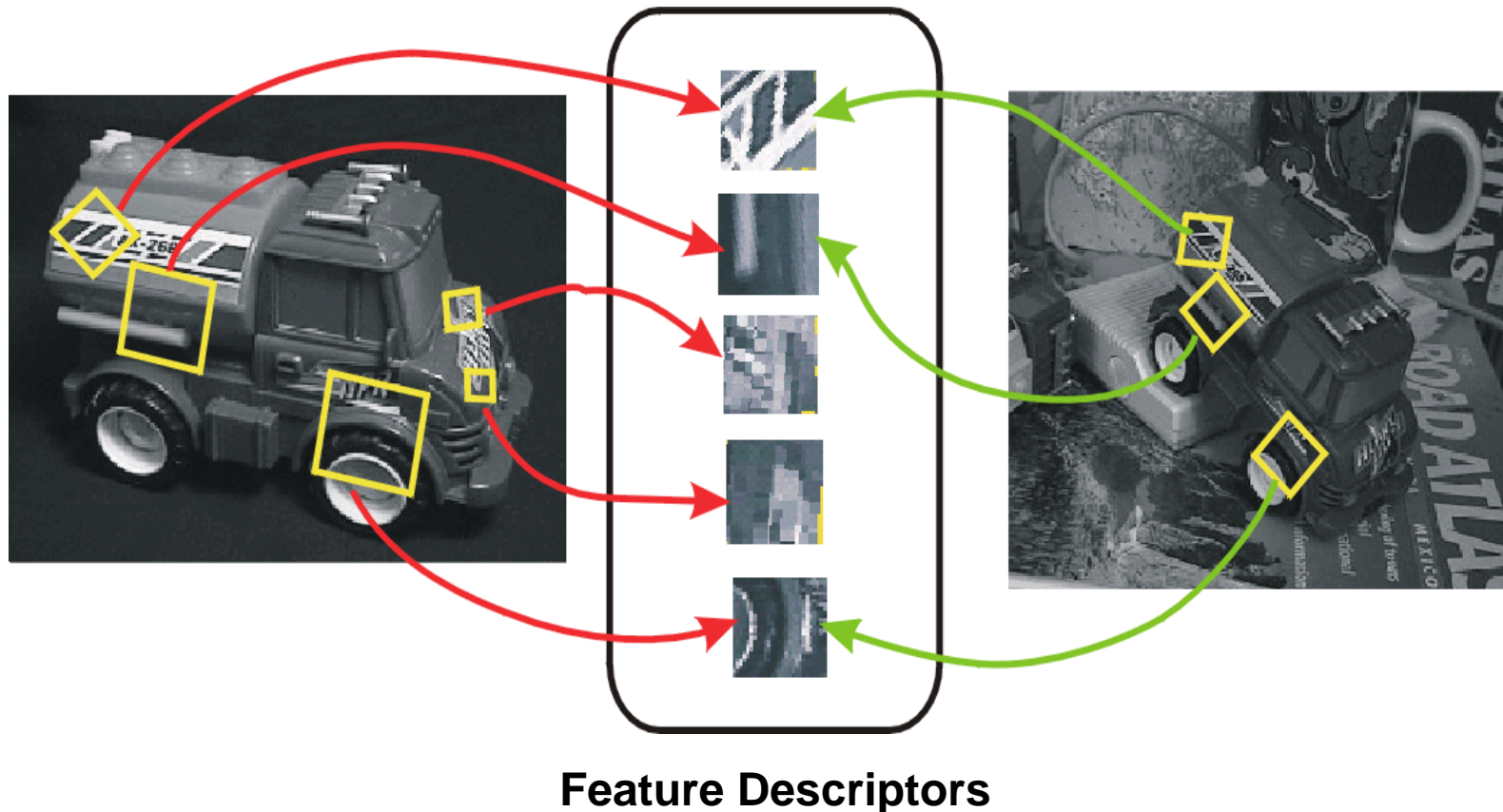


# Invariant local features

---

Find features that are invariant to transformations

- geometric invariance: translation, rotation, scale
- photometric invariance: brightness, exposure, ...





# Advantages of local features

---

## Locality

- features are local, so robust to occlusion and clutter

## Distinctiveness:

- can differentiate a large database of objects

## Quantity

- hundreds or thousands in a single image

## Efficiency

- real-time performance achievable

## Generality

- exploit different types of features in different situations

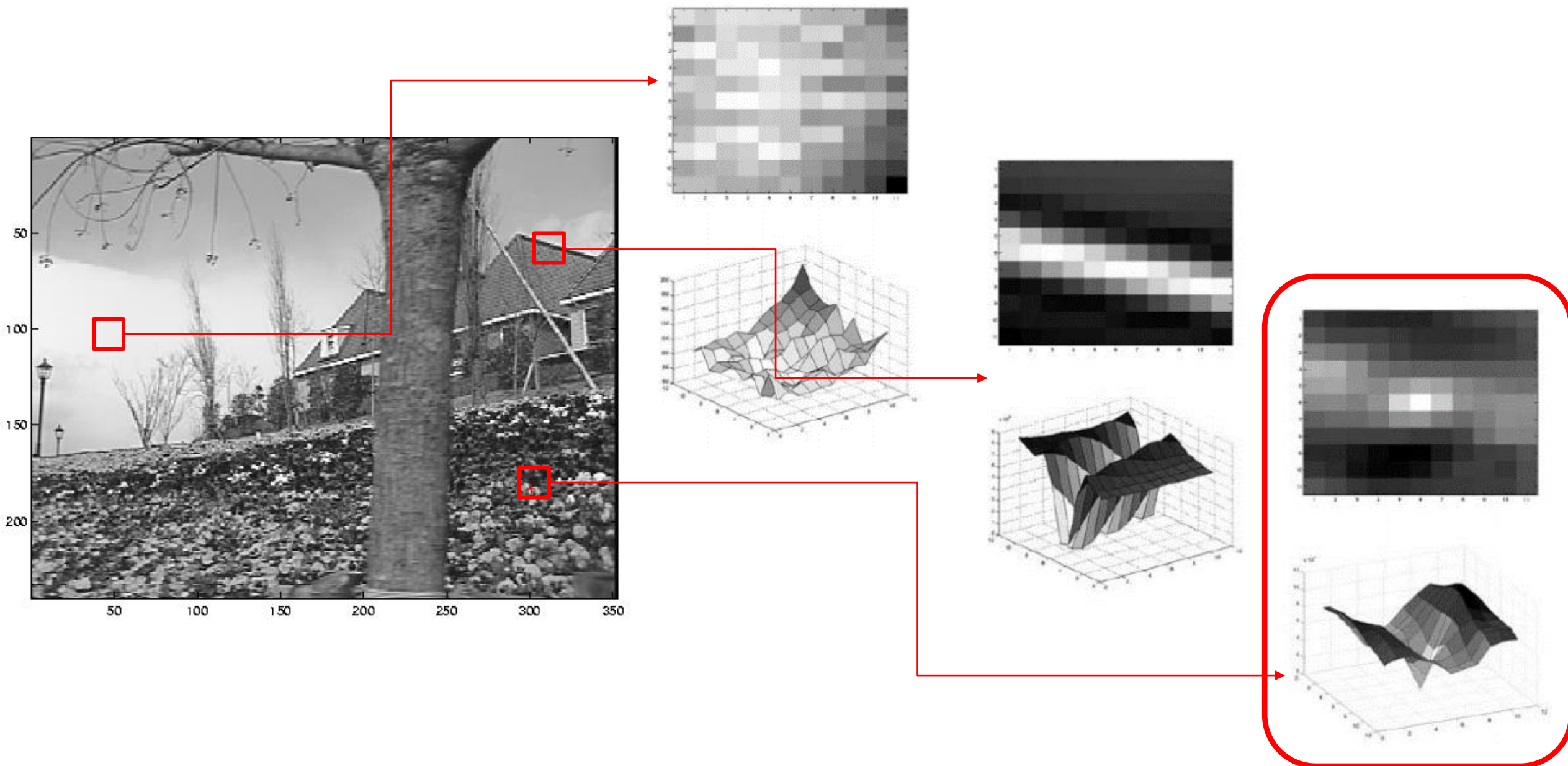
# More motivation...

---

Feature points are used for:

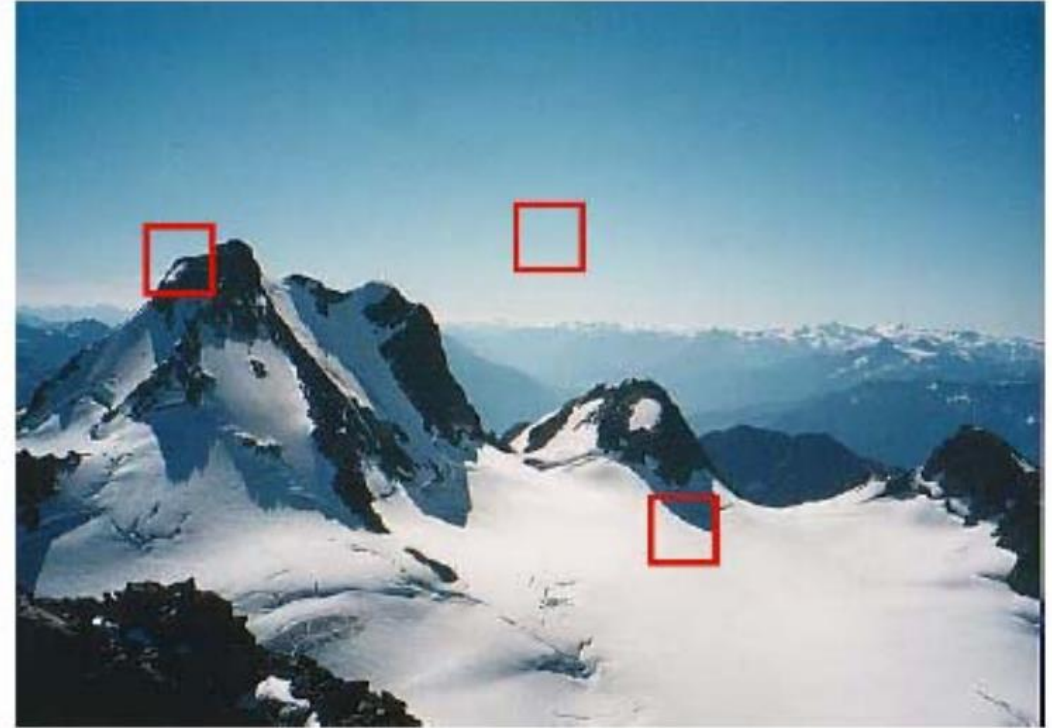
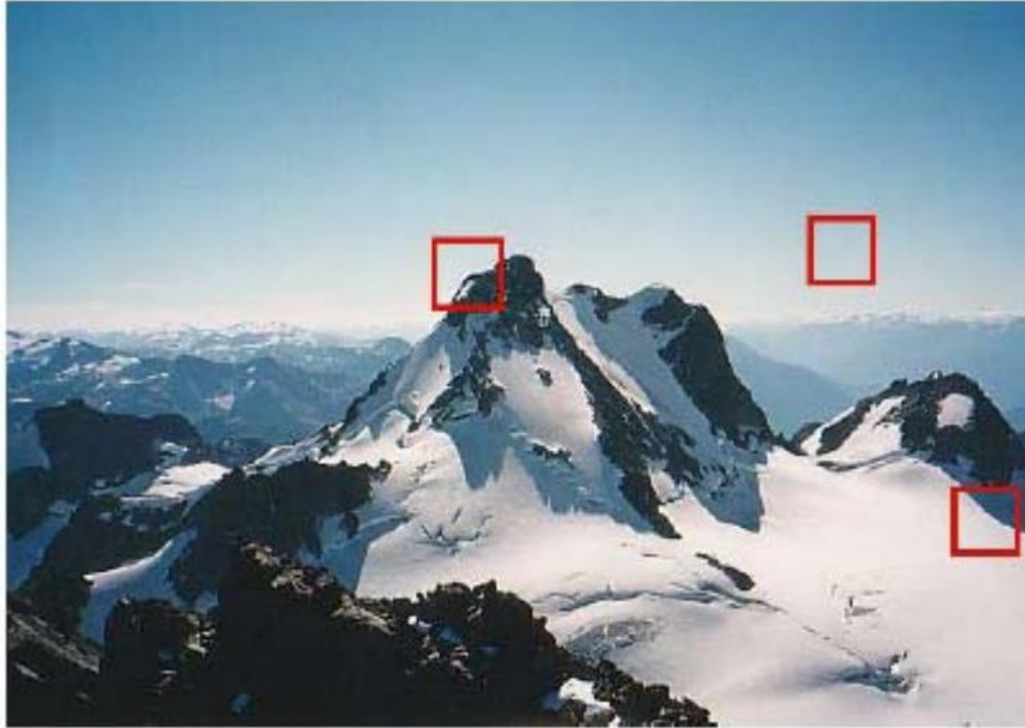
- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation
- ... other

# What makes a good feature?



# Which is the best patch here??

---



# Want uniqueness

---

Look for image regions that are unusual

- Lead to unambiguous matches in other images

How to define “unusual”?

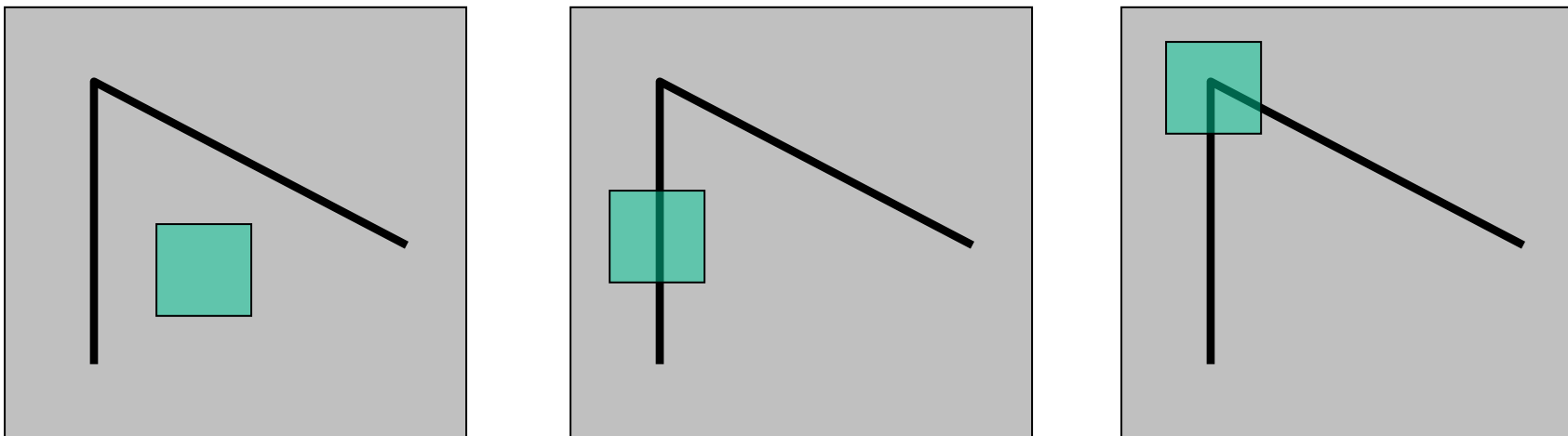


# Local measures of uniqueness

---

Suppose we only consider a small window of pixels

- What defines whether a feature is a good or bad candidate?

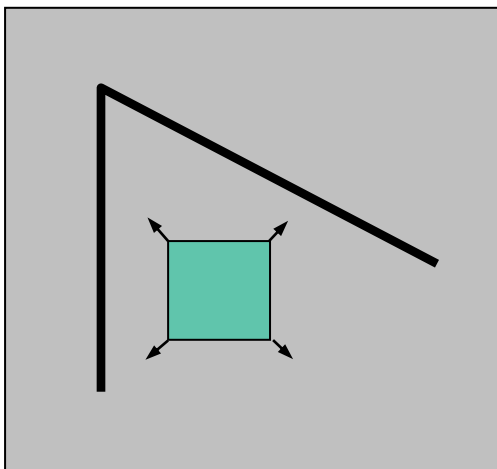


# Feature detection

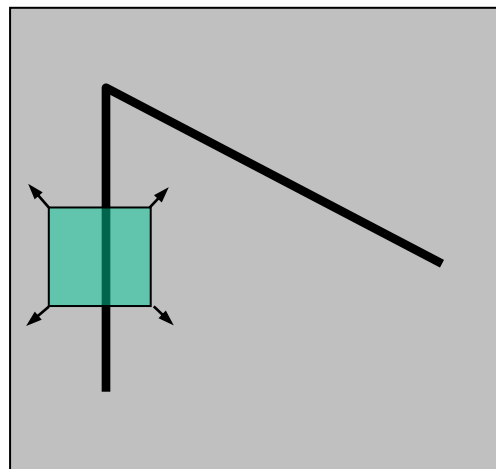
---

## Local measure of feature uniqueness

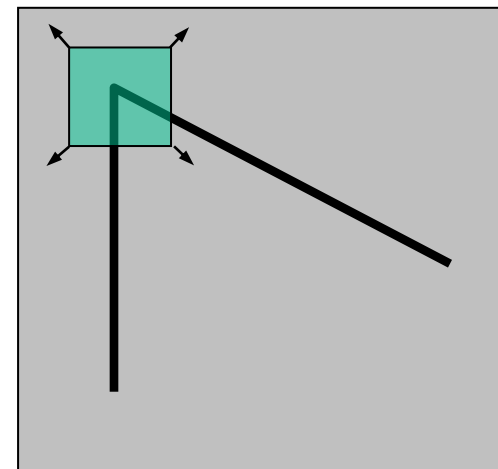
- How does the window change when you shift it?
- Shifting the window in *any direction* causes a *big change*



“flat” region:  
no change in all  
directions



“edge”:  
no change along  
the edge direction



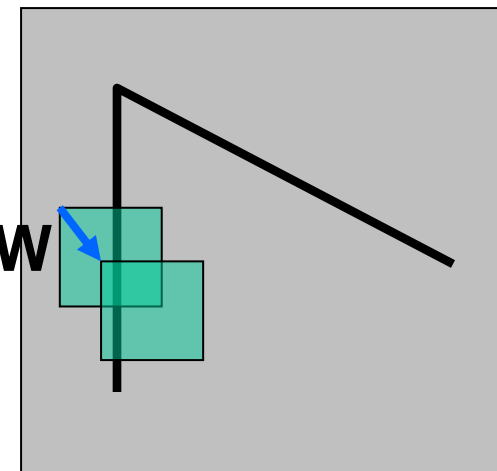
“corner”:  
significant change  
in all directions

# 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  $E(u,v)$ :



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

# 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

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

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

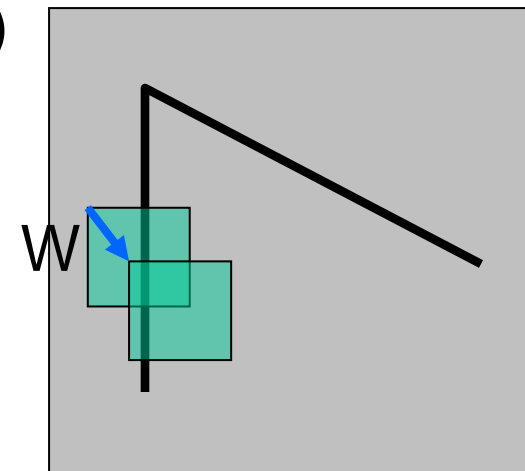
Plugging this into the formula on the previous slide...

# 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  $E(u,v)$ :



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

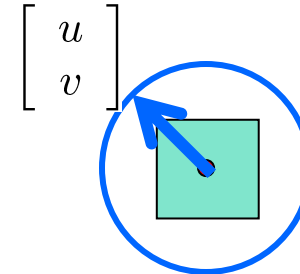
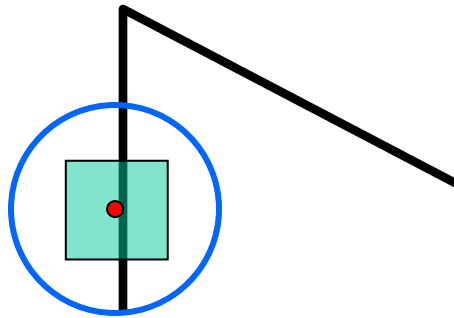


# Feature detection: the math

---

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  $H$

# Quick eigenvalue/eigenvector review

---

The **eigenvectors** of a matrix **A** are the vectors **x** that satisfy:

$$Ax = \lambda x$$

The scalar  $\lambda$  is the **eigenvalue** corresponding to **x**

- The eigenvalues are found by solving:

$$\det(A - \lambda I) = 0$$

- In our case, **A** = **H** is a 2x2 matrix, so we have

$$\det \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

- The solution:

$$\lambda_{\pm} = \frac{1}{2} \left[ (h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2} \right]$$

Once you know  $\lambda$ , you find **x** by solving

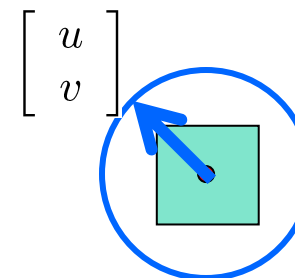
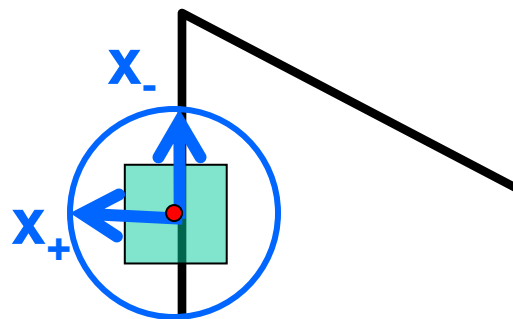
$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$

# Feature detection: the math

---

This can be rewritten:

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



## Eigenvalues and eigenvectors of H

- Define shifts with the smallest and largest change (E value)
- $x_+$  = direction of **largest** increase in E.
- $\lambda_+$  = amount of increase in direction  $x_+$
- $x_-$  = direction of **smallest** increase in E.
- $\lambda_-$  = amount of increase in direction  $x_-$

$$Hx_+ = \lambda_+ x_+$$

$$Hx_- = \lambda_- x_-$$

# Feature detection: the math

---

How are  $\lambda_+$ ,  $\mathbf{x}_+$ ,  $\lambda_-$ , and  $\mathbf{x}_-$  relevant for feature detection?

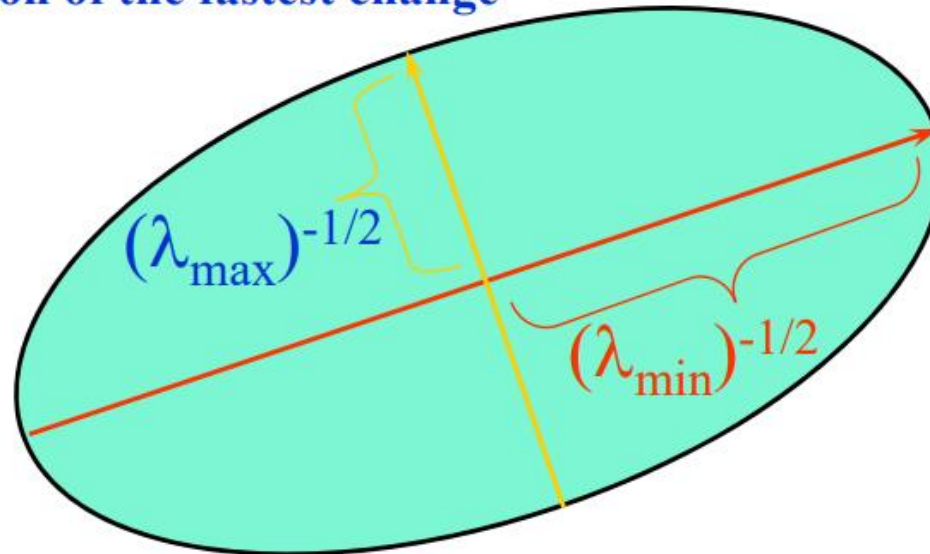
- What's our feature scoring function?

---


$$E(u, v) = (u \quad v) M \begin{pmatrix} u \\ v \end{pmatrix} \qquad M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

- $E(u, v)$  is an equation of an ellipse, where  $M$  is the covariance
- Let  $\lambda_1$  and  $\lambda_2$  be eigenvalues of  $M$

direction of the fastest change



direction of the  
slowest change



# Feature detection: the math

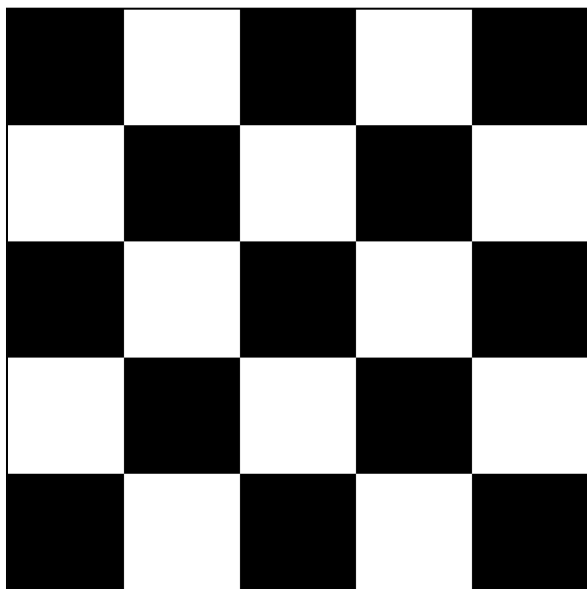
---

How are  $\lambda_+$ ,  $\mathbf{x}_+$ ,  $\lambda_-$ , and  $\mathbf{x}_-$  relevant for feature detection?

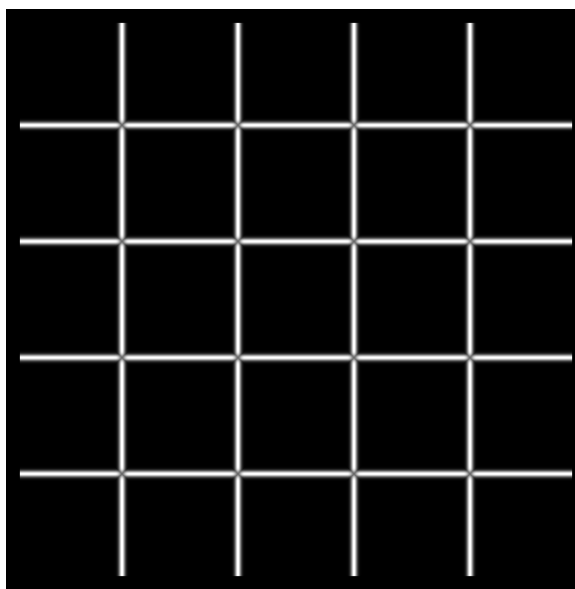
- What's our feature scoring function?

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

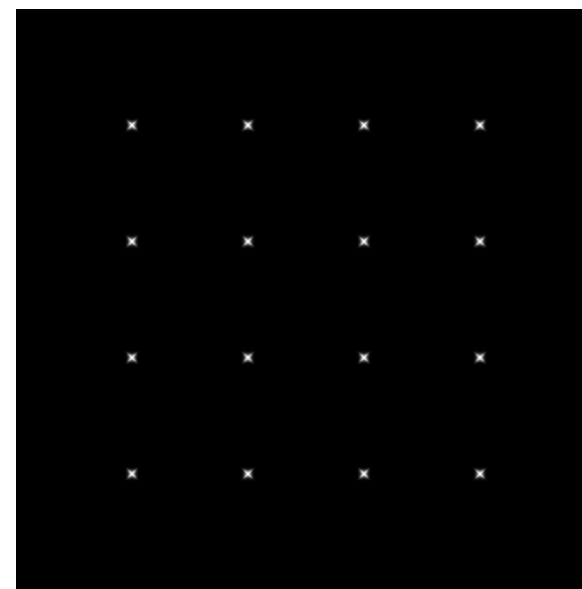
- the *minimum* of  $E(u,v)$  should be large, over all unit vectors  $[u \ v]$
- this minimum is given by the smaller eigenvalue ( $\lambda_-$ ) of  $\mathbf{H}$



$I$



$\lambda_+$



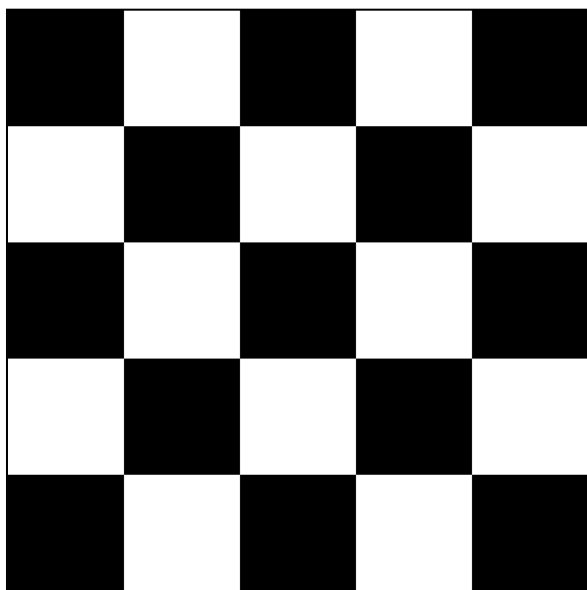
$\lambda_-$

# Feature detection summary

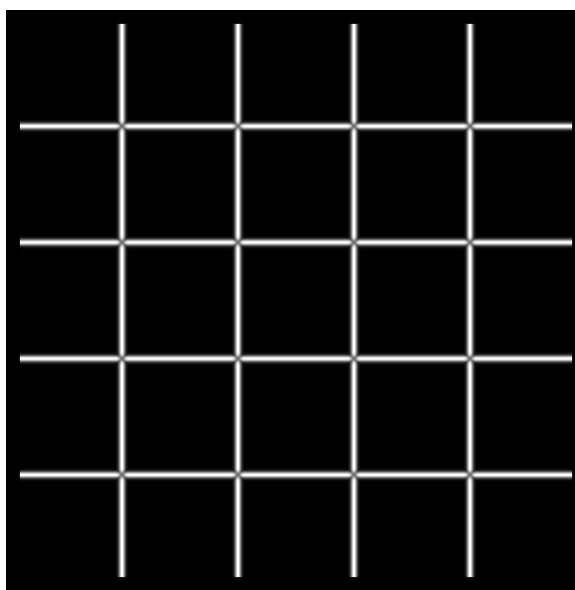
---

Here's what you do

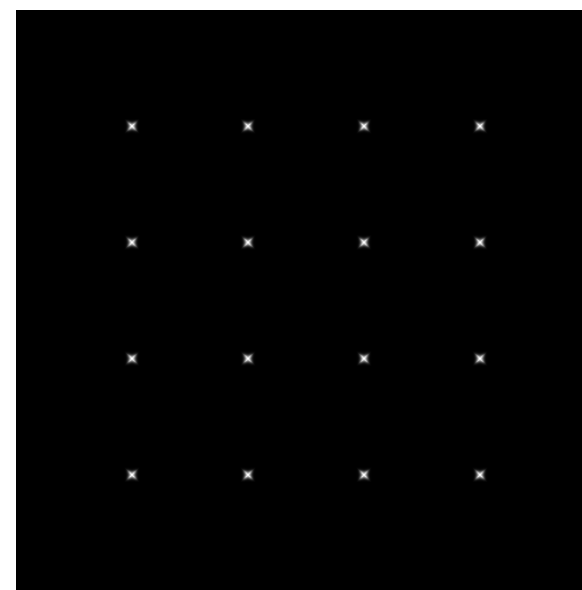
- Compute the gradient at each point in the image
- Create the  **$H$**  matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response ( $\lambda_- > \text{threshold}$ )
- Choose those points where  $\lambda_-$  is a local maximum as features



$I$



$\lambda_+$



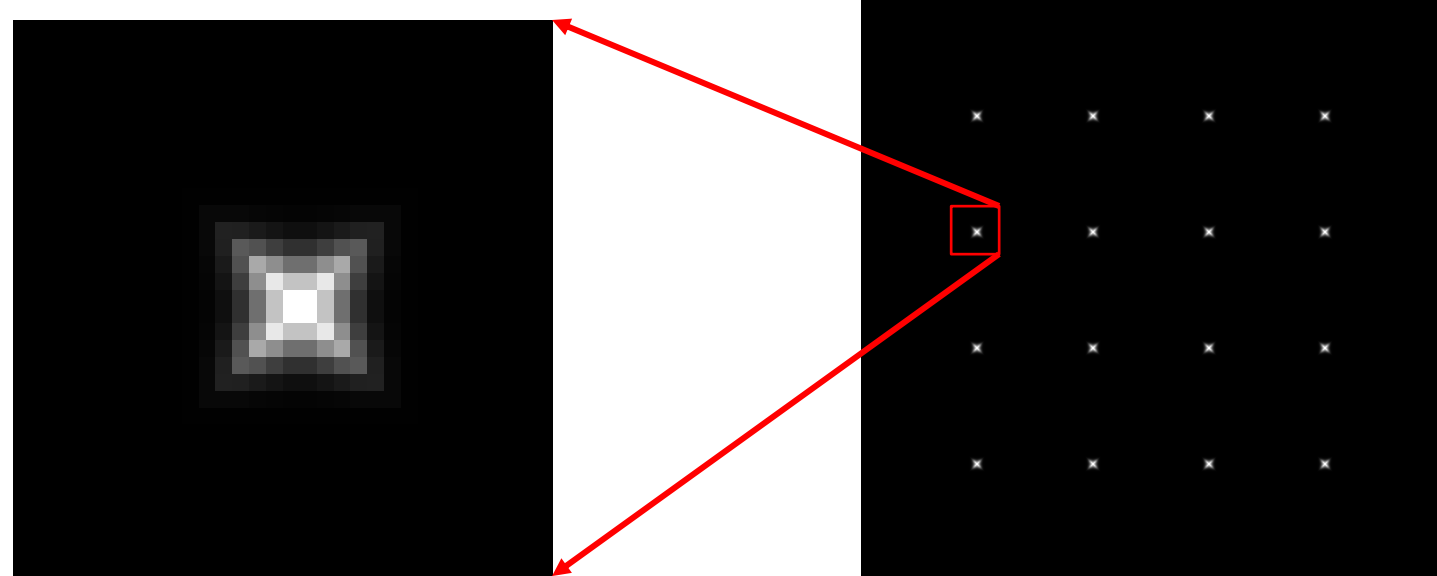
$\lambda_-$

# 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 ( $\lambda_- > \text{threshold}$ )
- Choose those points where  $\lambda_-$  is a local maximum as features



$\lambda_-$

# The Harris operator

---

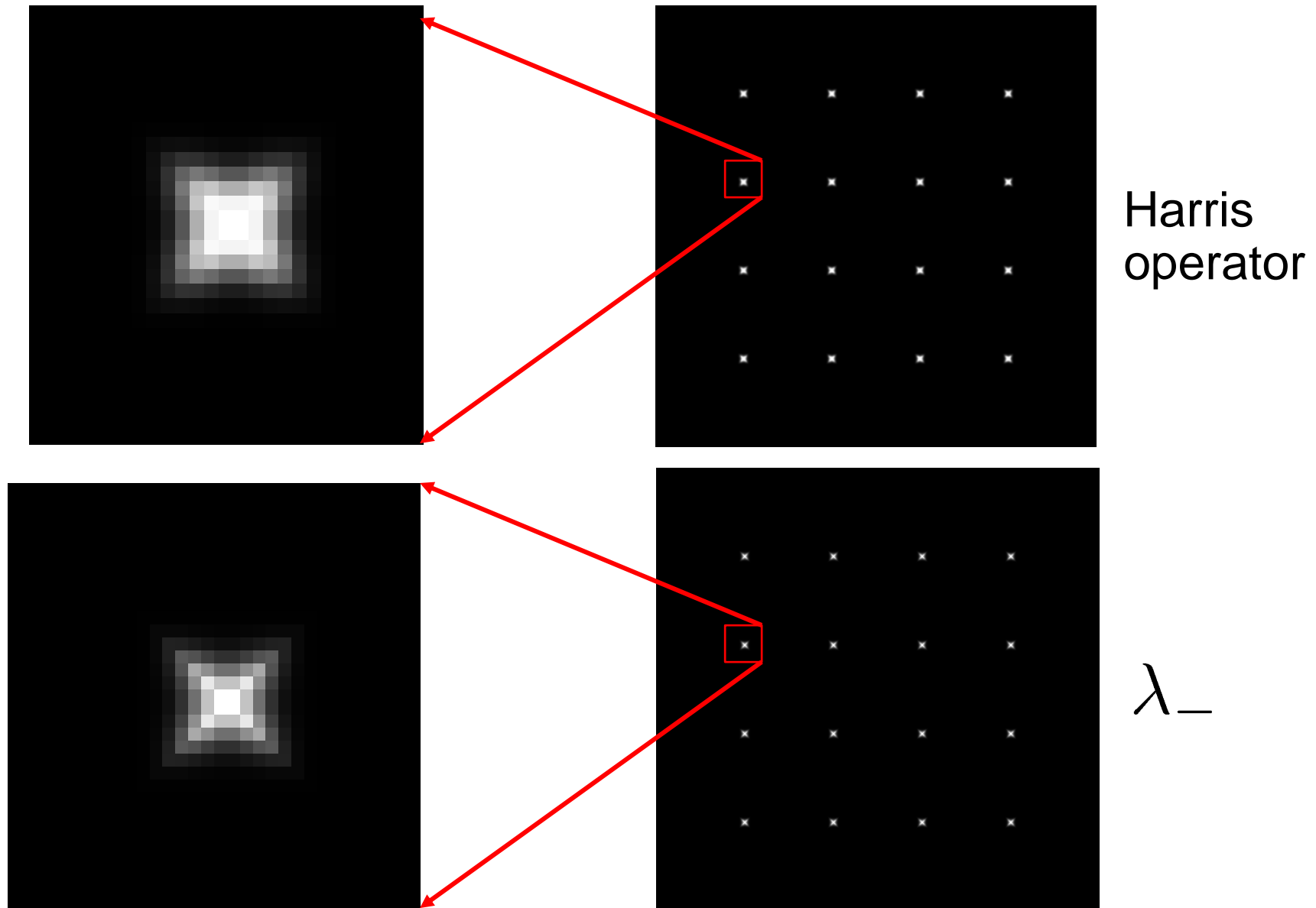
$\lambda_-$  is a variant of the “Harris operator” for feature detection

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

- The *trace* is the sum of the diagonals, i.e.,  $\text{trace}(H) = h_{11} + h_{22}$
- Very similar to  $\lambda_-$  but less expensive (no square root)
- Called the “Harris Corner Detector” or “Harris Operator”
- Lots of other detectors, this is one of the most popular

# The Harris operator

---



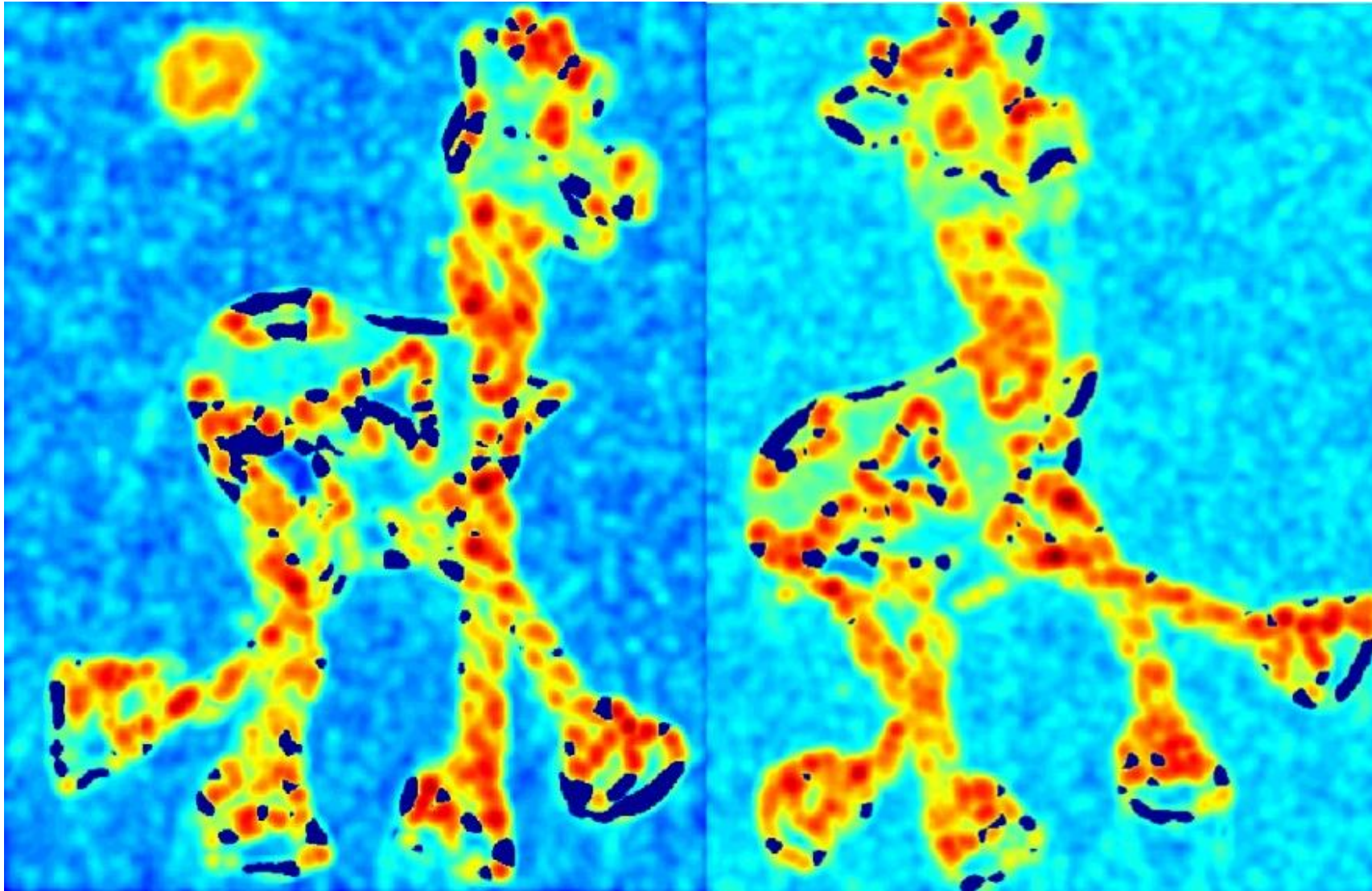
# Harris detector example

---



f value (red high, blue low)

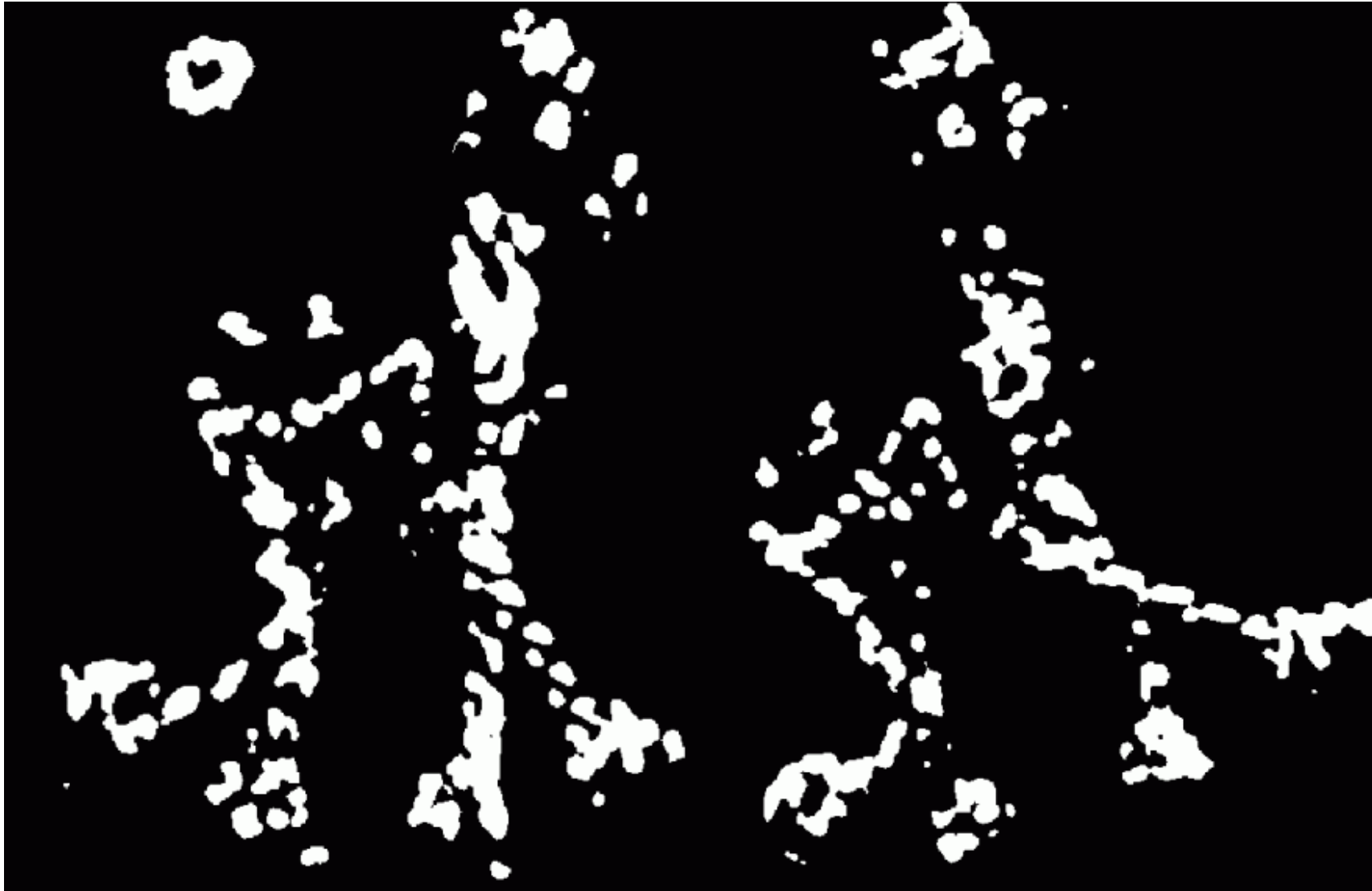
---





Threshold ( $f > \text{value}$ )

---



# Find local maxima of $f$

---



# Harris features (in red)

---



# Invariance

---

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

- Will you still pick up the same features?

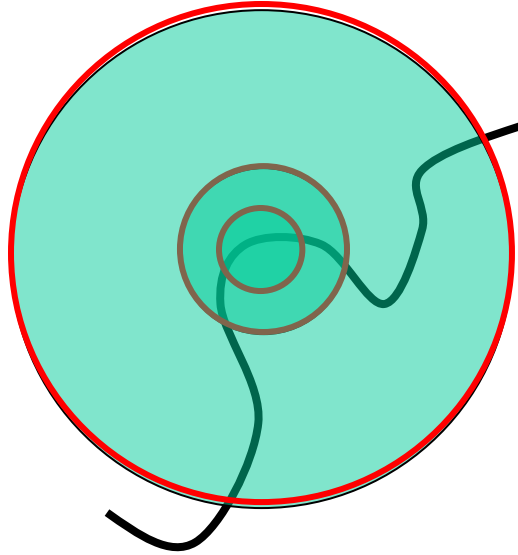
What if you change the brightness?

Scale?

# Scale invariant detection

---

Suppose you're looking for corners

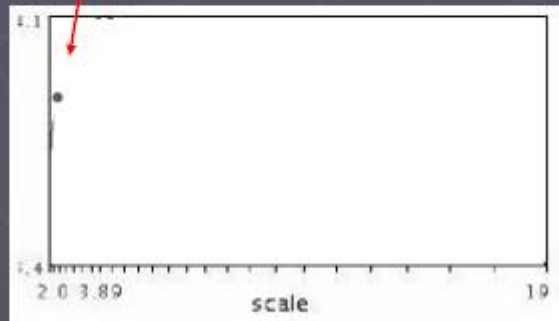


Key idea: find scale that gives local maximum of  $f$

- $f$  is a local maximum in both position and scale
- Common definition of  $f$ : Laplacian  
(or difference between two Gaussian filtered images with different sigmas)

# Automatic scale selection

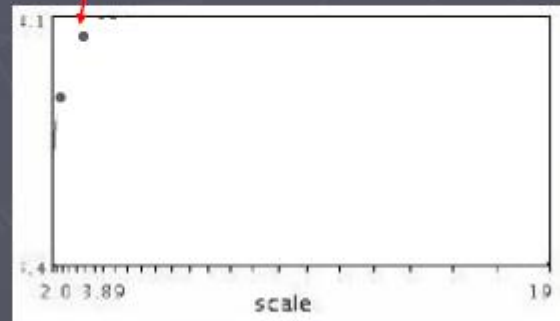
Lindeberg et al., 1996



$$f(I_{l_1...l_m}(x, \sigma))$$

Slide from Tinne Tuytelaars

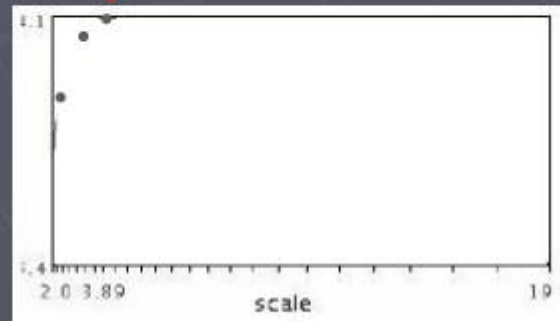
# Automatic scale selection



$$f(I_{l_1...l_m}(x, \sigma))$$

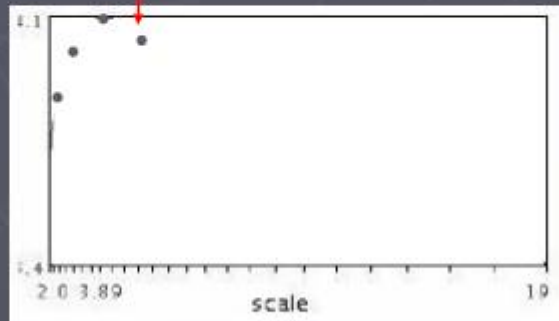


# Automatic scale selection



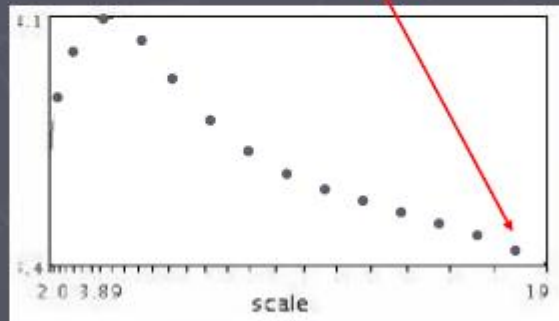
$$f(I_{l_1...l_m}(x, \sigma))$$

# Automatic scale selection



$$f(I_{l \dots l_m}(x, \sigma))$$

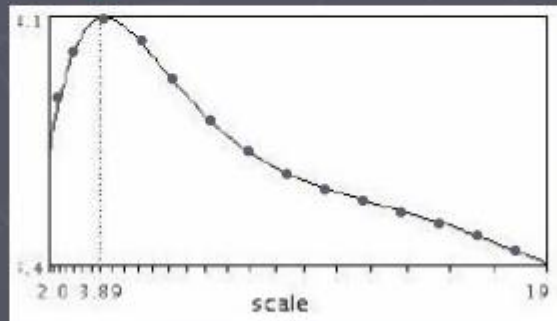
# Automatic scale selection



$$f(I_{l_{\dots}l_m}(x, \sigma))$$

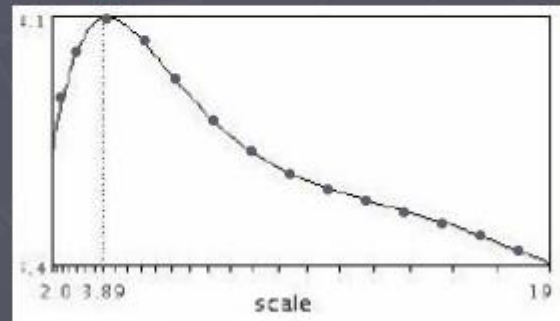


# Automatic scale selection

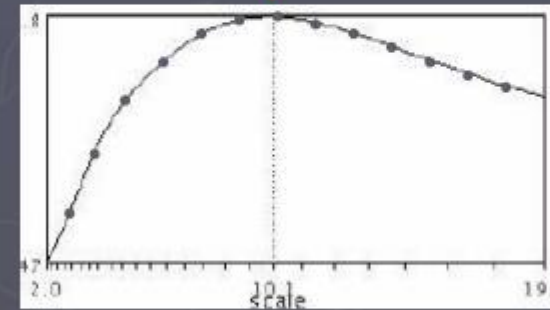


$$f(I_{l_1 \dots l_m}(x, \sigma))$$

# Automatic scale selection



$$f(I_{i_1 \dots i_m}(x, \sigma))$$



$$f(I_{i_1 \dots i_m}(x', \sigma'))$$

# Automatic scale selection

Normalize: rescale to fixed size



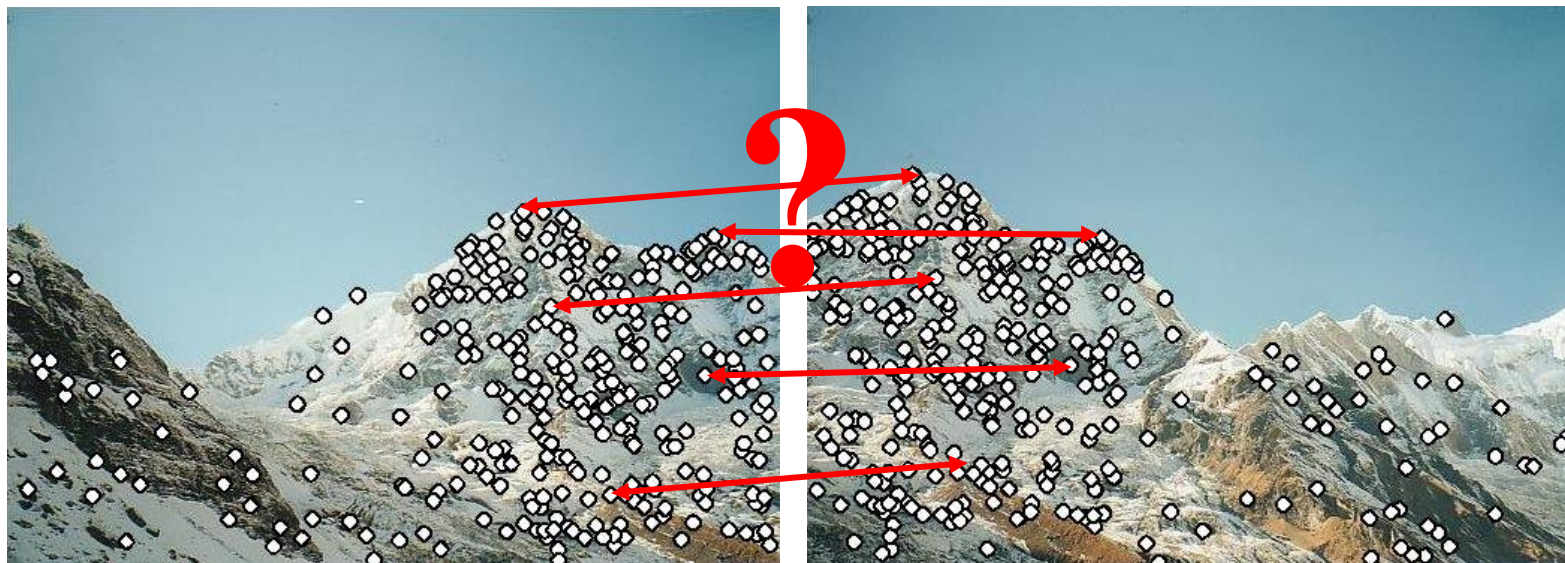


# Feature descriptors

---

We know how to detect good points

Next question: **How to match them?**



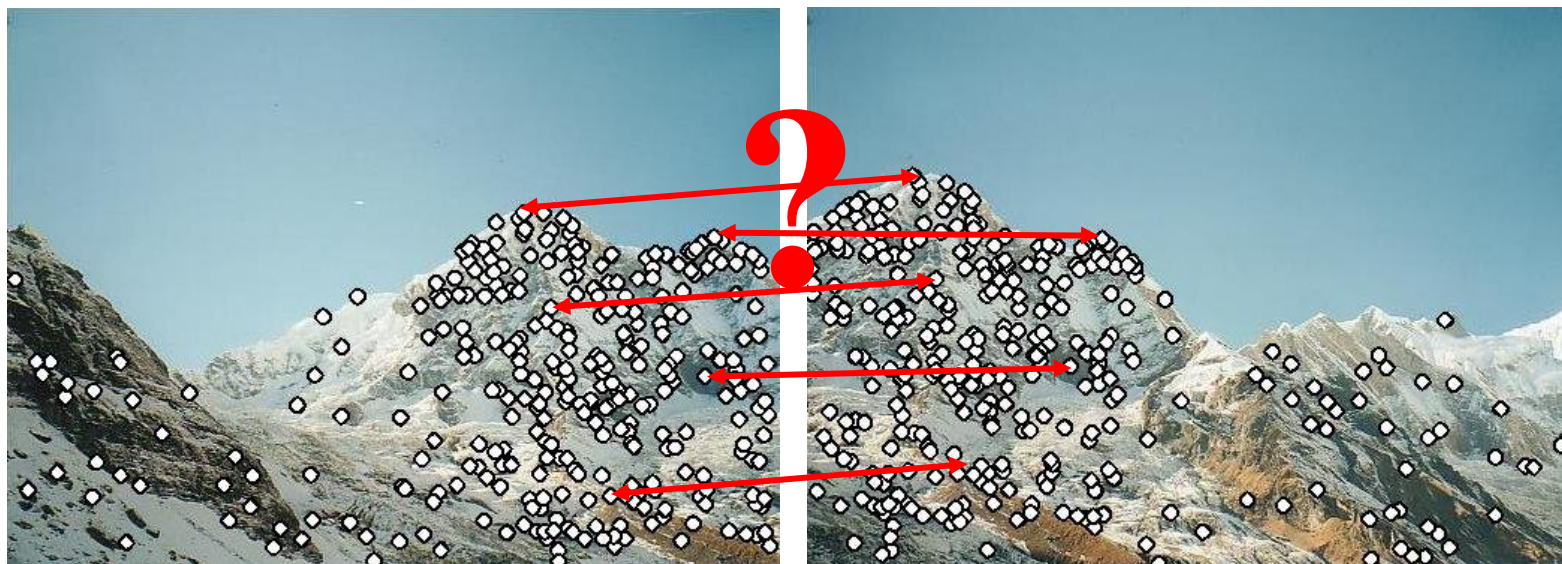


# Feature descriptors

---

We know how to detect good points

Next question: **How to match them?**



Lots of possibilities (this is a popular research area)

- Simple option: match square windows around the point
- State of the art approach: SIFT
  - David Lowe, UBC <http://www.cs.ubc.ca/~lowe/keypoints/>

# Invariance

---

Suppose we are comparing two images  $I_1$  and  $I_2$

- $I_2$  may be a transformed version of  $I_1$
- What kinds of transformations are we likely to encounter in practice?

# Invariance

---

Suppose we are comparing two images  $I_1$  and  $I_2$

- $I_2$  may be a transformed version of  $I_1$
- What kinds of transformations are we likely to encounter in practice?

We'd like to find the same features regardless of the transformation

- This is called transformational ***invariance***
- Most feature methods are designed to be invariant to
  - Translation, 2D rotation, scale
- They can usually also handle
  - Limited 3D rotations (SIFT works up to about 60 degrees)
  - Limited affine transformations (some are fully affine invariant)
  - Limited illumination/contrast changes

# How to achieve invariance

---

Need both of the following:

## 1. Make sure your detector is invariant

- Harris is invariant to translation and rotation
- Scale is trickier
  - common approach is to detect features at many scales using a Gaussian pyramid (e.g., MOPS)
  - More sophisticated methods find “the best scale” to represent each feature (e.g., SIFT)

## 2. Design an invariant feature *descriptor*

- A descriptor captures the information in a region around the detected feature point
- The simplest descriptor: a square window of pixels
  - What’s this invariant to?
- Let’s look at some better approaches...

# Rotation invariance for feature descriptors

---

Find dominant orientation of the image patch

- This is given by  $\mathbf{x}_+$ , the eigenvector of  $\mathbf{H}$  corresponding to  $\lambda_+$ 
  - $\lambda_+$  is the *larger* eigenvalue
- Rotate the patch according to this angle



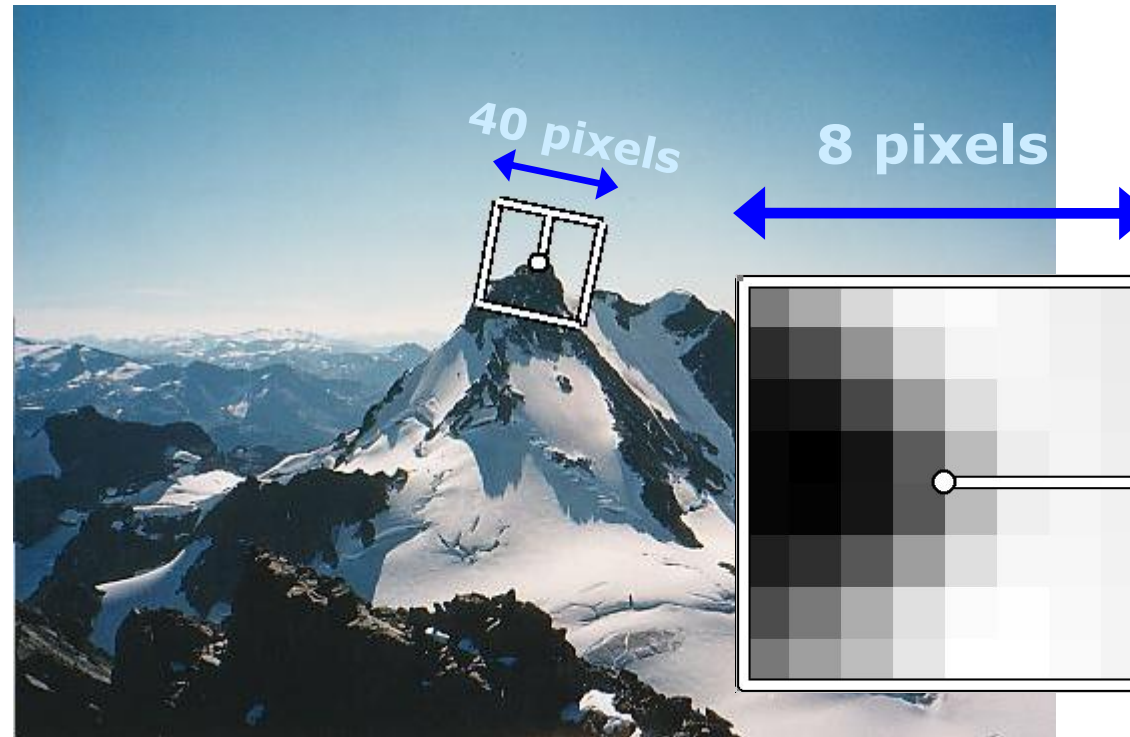
Figure by Matthew Brown

# Multiscale Oriented PatchS descriptor

---

Take 40x40 square window around detected feature

- Scale to 1/5 size (using prefiltering)
- Rotate to horizontal
- Sample 8x8 square window centered at feature
- Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window

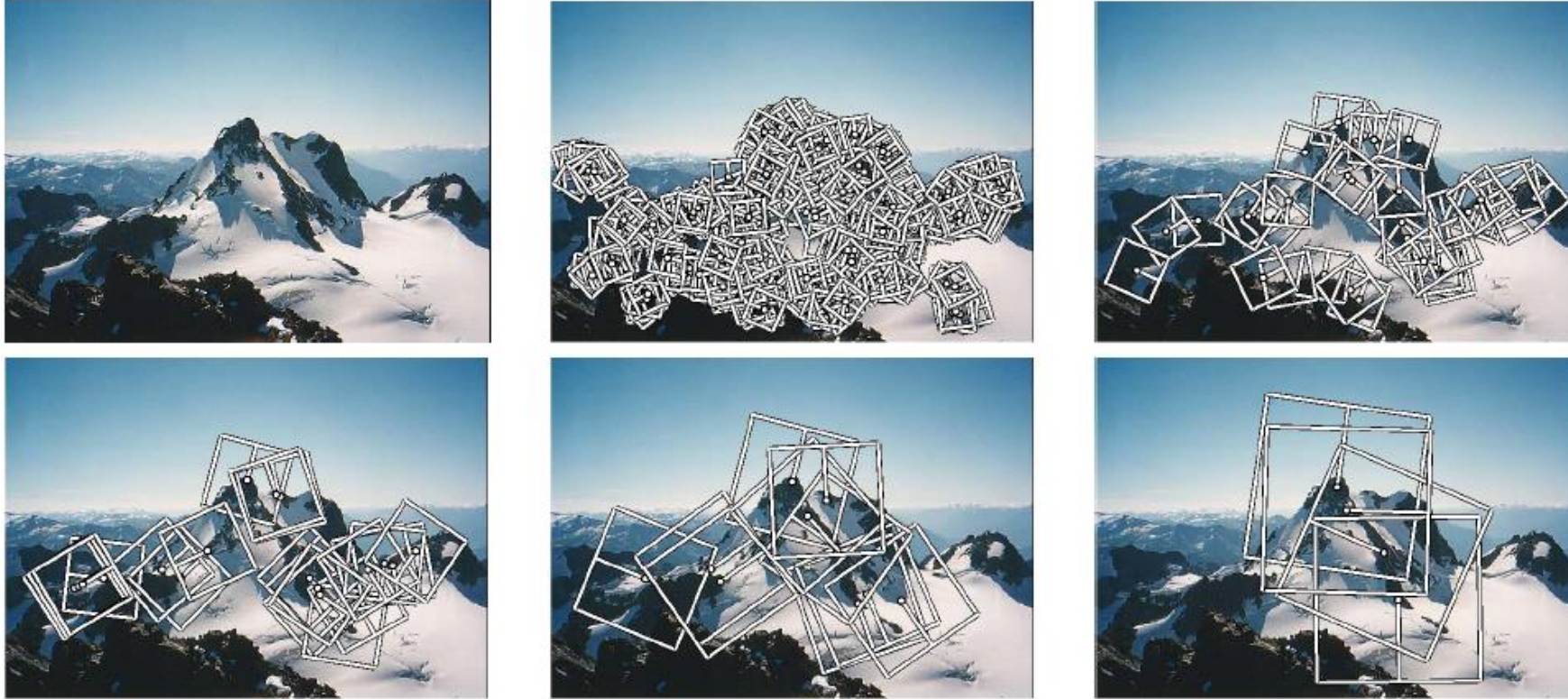


Adapted from slide by Matthew Brown



# Detections at multiple scales

---



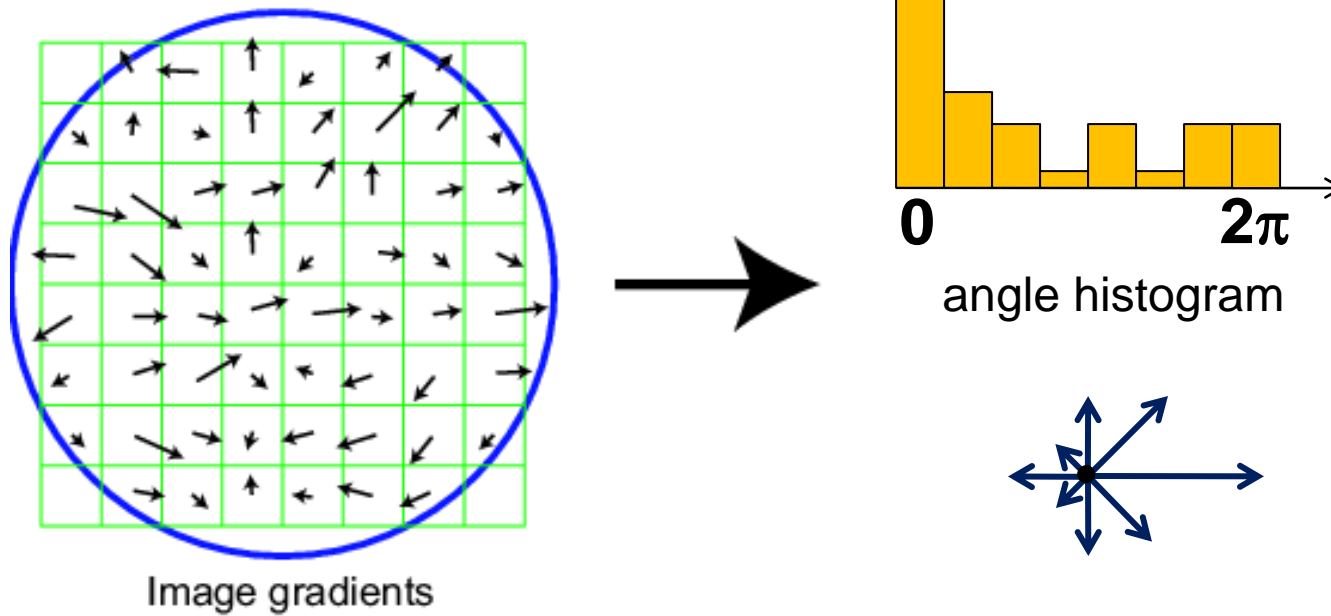
*Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.*



# Scale Invariant Feature Transform

Basic idea:

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



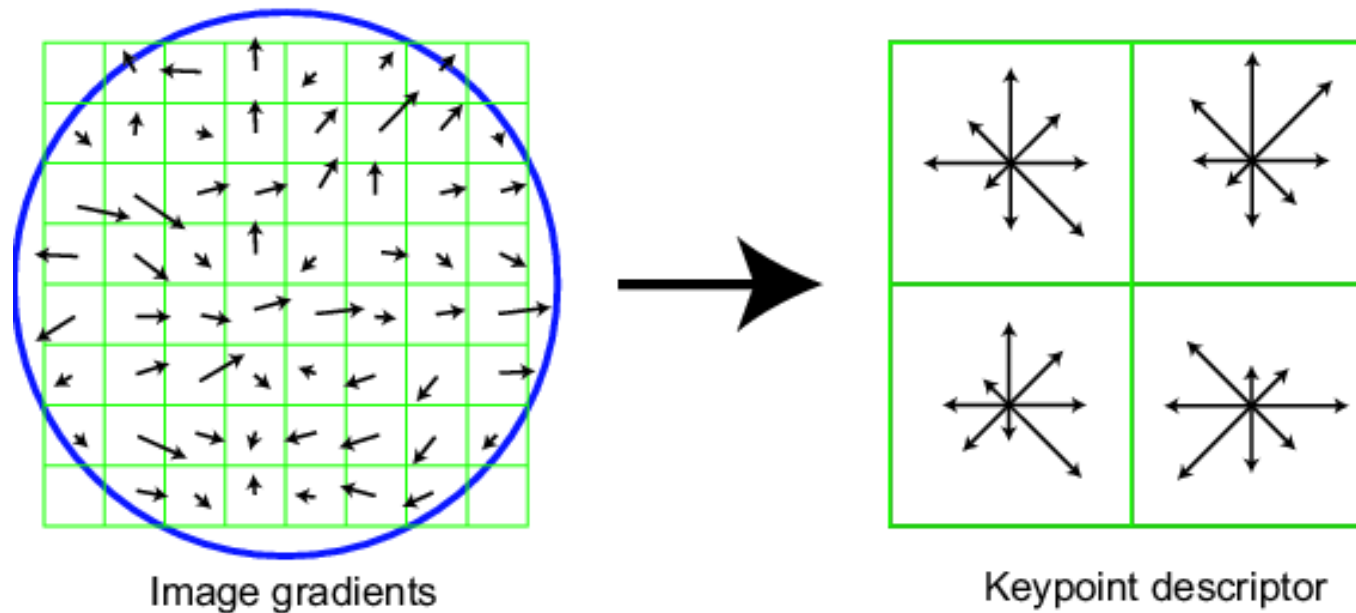
Adapted from slide by David Lowe

# 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

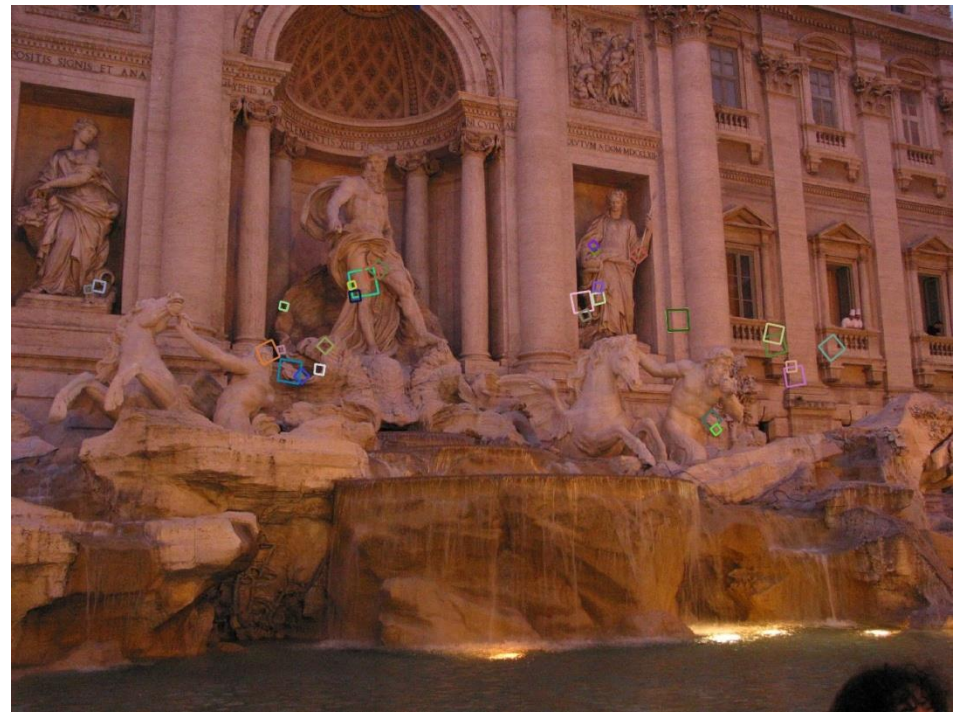


# Properties of SIFT

---

## 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](http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT)



# Maximally Stable Extremal Regions

---

J.Matas et.al. “Distinguished Regions for Wide-baseline Stereo”. BMVC 2002.

- Maximally Stable Extremal Regions
  - *Threshold* image intensities:  $I > thresh$  for several increasing values of thresh
  - Extract *connected components* (“Extremal Regions”)
  - Find a threshold when region is “Maximally Stable”, i.e. *local minimum* of the relative growth
  - Approximate each region with an *ellipse*



# Feature matching

---

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

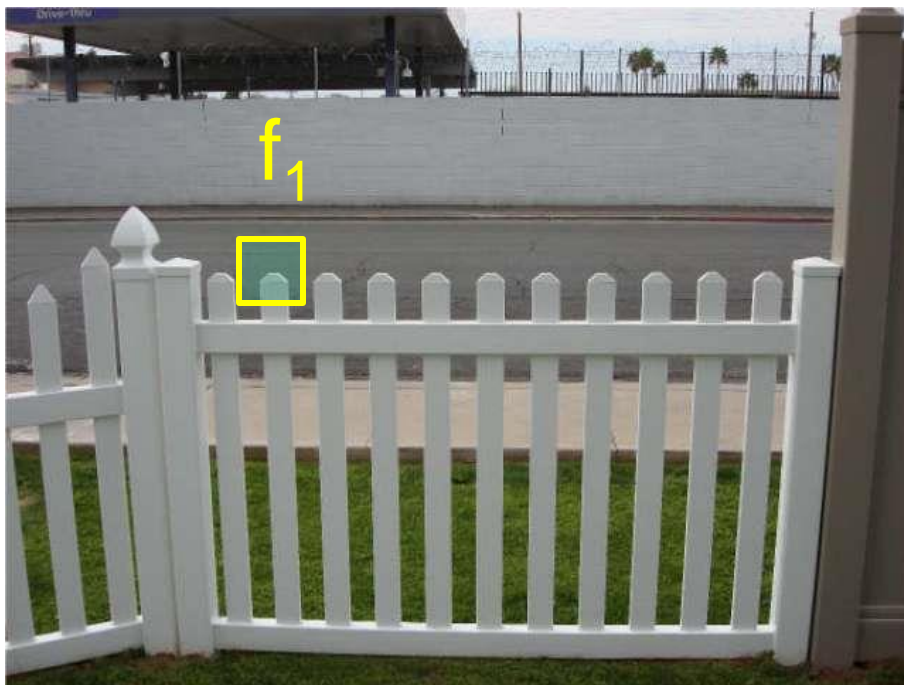
1. Define distance function that compares two descriptors
2. Test all the features in  $I_2$ , find the one with min distance

# Feature distance

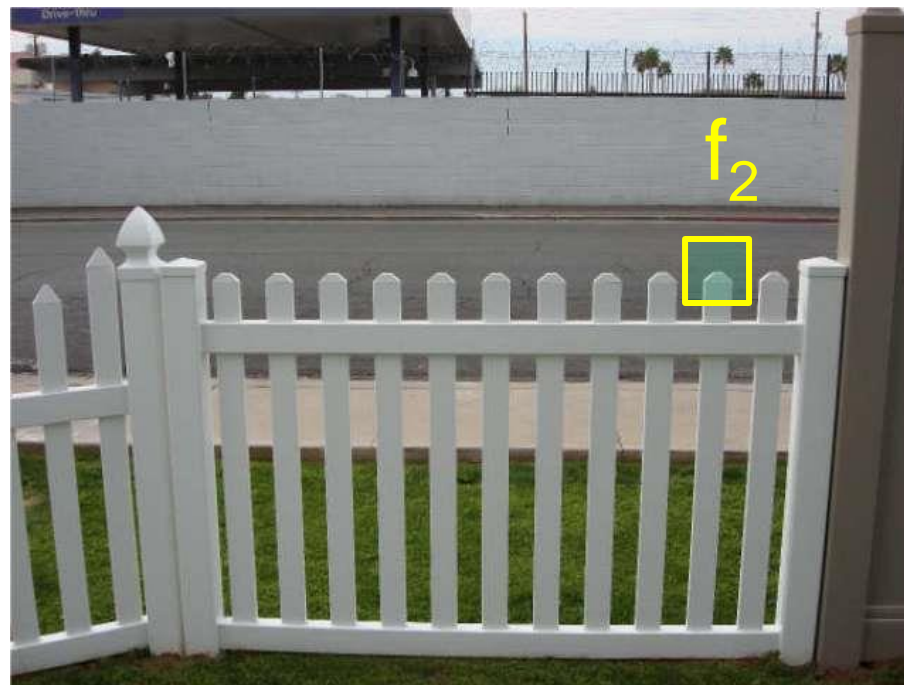
---

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

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



$I_1$



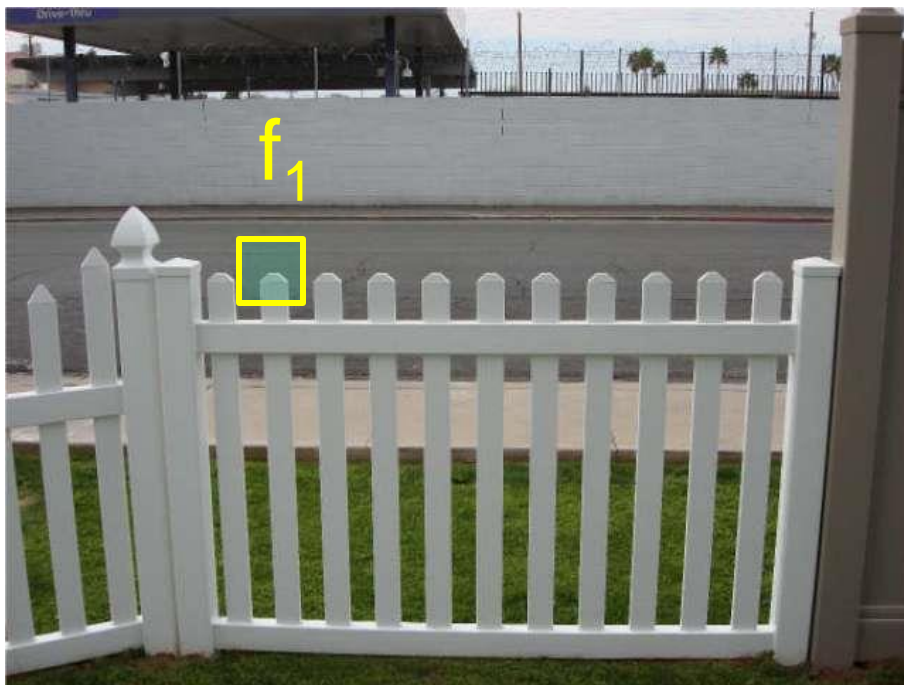
$I_2$



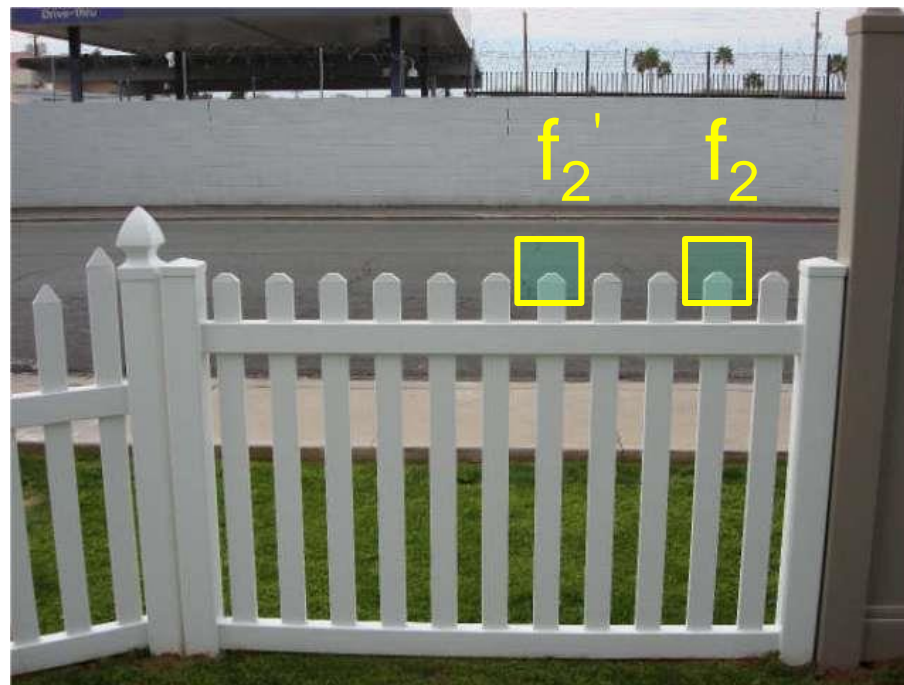
# Feature distance

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

- Better approach: ratio distance =  $\text{SSD}(f_1, f_2) / \text{SSD}(f_1, f_2')$ 
  - $f_2$  is best SSD match to  $f_1$  in  $I_2$
  - $f_2'$  is 2<sup>nd</sup> best SSD match to  $f_1$  in  $I_2$
  - gives small values for ambiguous matches



$I_1$

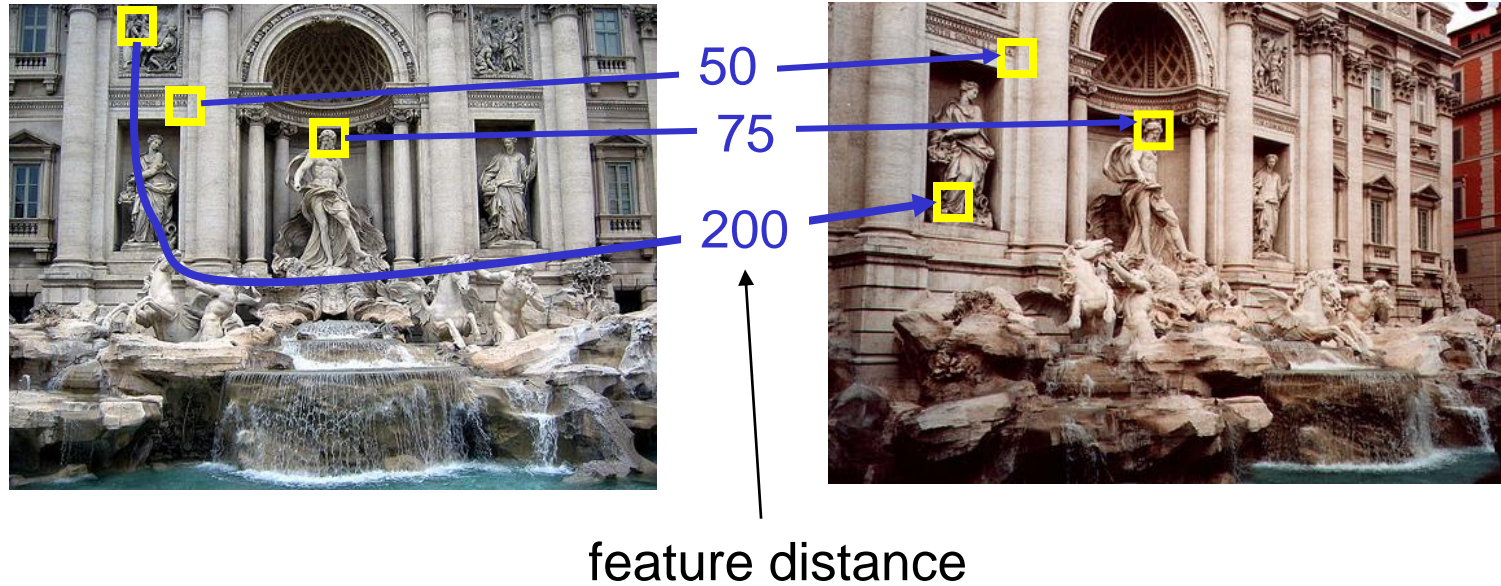


$I_2$

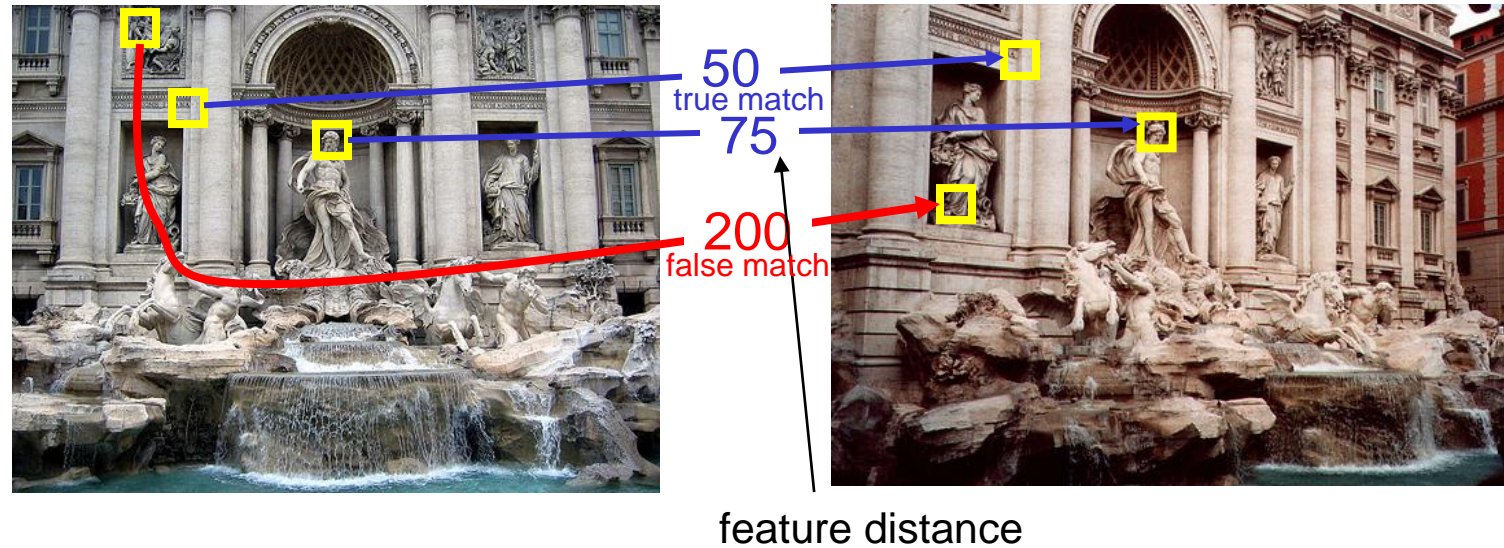


# Evaluating the results

How can we measure the performance of a feature matcher?



# True/false positives



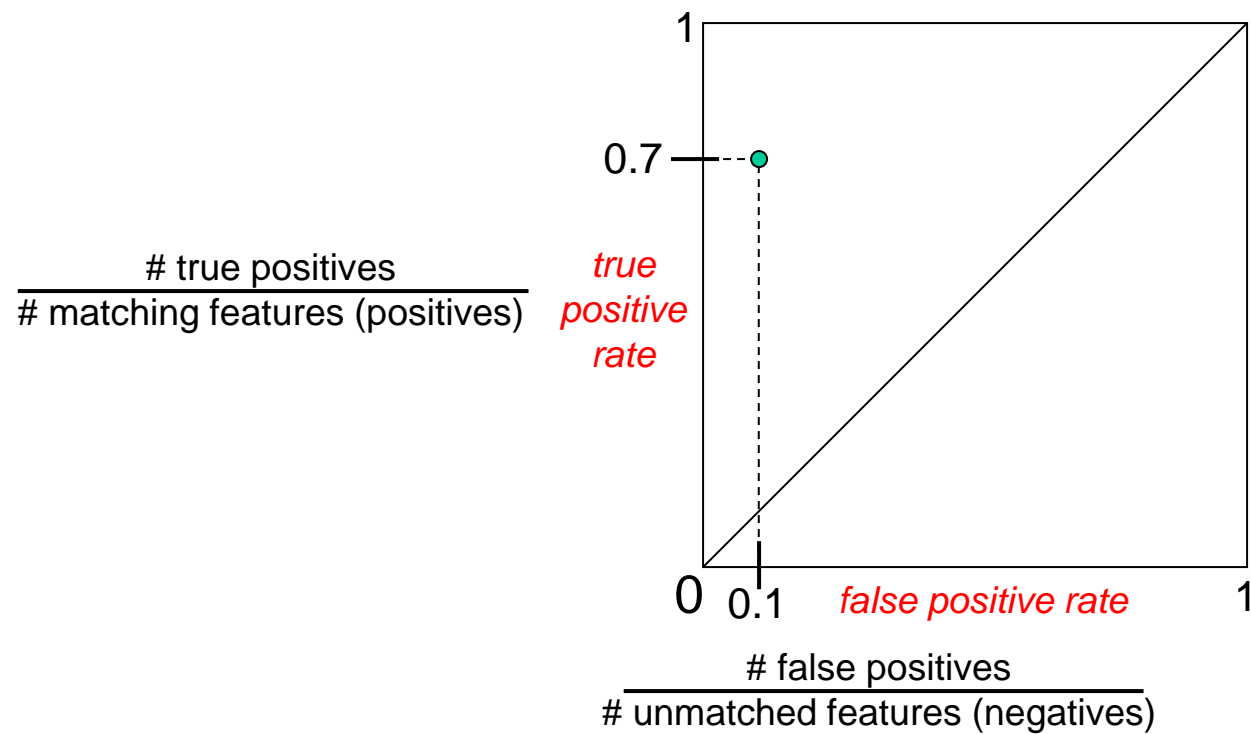
## The distance threshold affects performance

- True positives = # of detected matches that are correct
  - Suppose we want to maximize these—how to choose threshold?
- False positives = # of detected matches that are incorrect
  - Suppose we want to minimize these—how to choose threshold?

# Evaluating the results

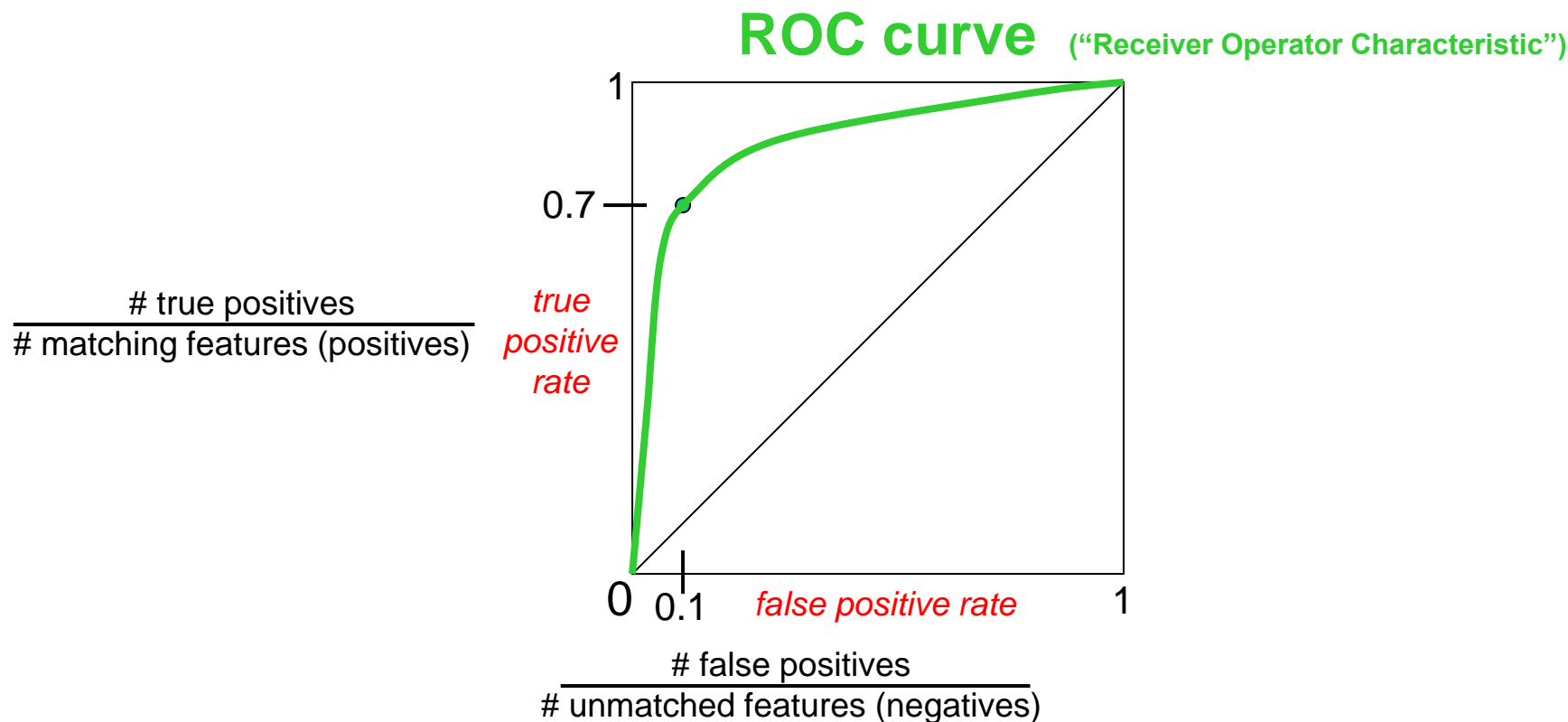
---

How can we measure the performance of a feature matcher?



# Evaluating the results

How can we measure the performance of a feature matcher?

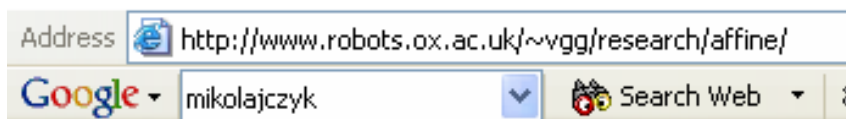


## ROC Curves

- Generated by counting # current/incorrect matches, for different thresholds
- Want to maximize area under the curve (AUC)
- Useful for comparing different feature matching methods
- For more info: [http://en.wikipedia.org/wiki/Receiver\\_operating\\_characteristic](http://en.wikipedia.org/wiki/Receiver_operating_characteristic)

# More on feature detection/description

---



## Affine Covariant Regions

### Publications

#### Region detectors

- *Harris-Affine & Hessian Affine*: [K. Mikolajczyk](#) and [C. Schmid](#), Scale and Affine invariant interest point detectors. In IJCV 1(60):63-86, 2004. [PDF](#)
- *MSER*: [J. Matas](#), [O. Chum](#), [M. Urban](#), and [T. Pajdla](#), Robust wide baseline stereo from maximally stable extremal regions. In BMVC p. 384-393, 2002. [PDF](#)
- *IBR & EBR*: [T. Tuytelaars](#) and [L. Van Gool](#), Matching widely separated views based on affine invariant regions. In IJCV 1(59):61-85, 2004. [PDF](#)
- *Salient regions*: [T. Kadir](#), [A. Zisserman](#), and [M. Brady](#), An affine invariant salient region detector. In ECCV p. 404-416, 2004. [PDF](#)

#### Region descriptors

- *SIFT*: [D. Lowe](#), Distinctive image features from scale invariant keypoints. In IJCV 2(60):91-110, 2004. [PDF](#)

#### Performance evaluation

- [K. Mikolajczyk](#), [T. Tuytelaars](#), [C. Schmid](#), [A. Zisserman](#), [J. Matas](#), [F. Schaffalitzky](#), [T. Kadir](#) and [L. Van Gool](#), A comparison of affine region detectors. Technical Report, accepted to IJCV. [PDF](#)
- [K. Mikolajczyk](#), [C. Schmid](#), A performance evaluation of local descriptors. Technical Report, accepted to PAMI. [PDF](#)

# 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



# Object recognition (David Lowe)

---





# Sony Aibo

## SIFT usage:


- Recognize charging station
- Communicate with visual cards
- Teach object recognition

AIBO® Entertainment Robot  
Official U.S. Resources and Online Destinations



# Laboratorio 5

<https://colab.research.google.com/drive/1KogGm6oE8YoqljYfVR4rV8NjEPKzWBPg?authuser=2#scrollTo=Z361Gh7lcuu->



ReinstallingCV\_Image-Features

## Detección de características

Dado que las esquinas son características interesantes de una imagen. Los algoritmos de detección de características comenzaron con la detección de esquinas. Hay varias técnicas en OpenCV para detectar características.

- Detección de esquinas Harris
- Detección de esquinas Shi-Tomasi
- SIFT (Transformación de características de escala invariable)
- SURF (Funciones robustas aceleradas)
- Algoritmo FAST para la detección de esquinas
- ORB (Breve orientado FAST y girado)

SIFT, SURF están patentados y no están disponibles gratuitamente para uso comercial. Requiere opencv-contrib instalado para poder usarlos

Necesitamos reinstalar el paquete de Contributive de Python, para poder ejecutar los detectores SIFT y SURF pues estos son patentados

```
[ ] !pip uninstall opencv-python -y
# downgrade OpenCV a bit since some non-free features are not available
!pip install opencv-contrib-python==3.4.2.17 --force-reinstall

Uninstalling opencv-python-4.1.2.30:
Successfully uninstalled opencv-python-4.1.2.30
Collecting opencv-contrib-python==3.4.2.17
  Downloading https://files.pythonhosted.org/packages/61/29/fc60b2de1713aa92946992544329f20ccb5e4ba26290f403e04b7da44105/opencv_contrib_python-3.4.2.17-cp36-cp36m-manylinux1_x86_64.whl (30.6MB)
    | 30.6MB 158kB/s
Collecting numpy>=1.11.3
  Downloading https://files.pythonhosted.org/packages/97/af/8a92864a04bfa48f1b5c9b1f8bf2ccb2847f24530026f26dd223de4ca0/numpy-1.19.2-cp36-cp36m-manylinux2010_x86_64.whl (14.5MB)
    | 14.5MB 160kB/s
```

Mostrar todo

# Gracias !!!

---



© Man Bouncing Question Mark Towards Doctor - Artist: [Art Glazer](#)