

# Local Feature Detectors

---

Selim Aksoy

Department of Computer Engineering

Bilkent University

saksoy@cs.bilkent.edu.tr

Slides adapted from Cordelia Schmid and David Lowe, CVPR 2003 Tutorial,  
Matthew Brown, Microsoft Research, and Steve Seitz, U of Washington.

# Image matching

---

- Image matching is a fundamental aspect of many problems in computer vision.
  - Object or scene recognition
  - Solving for 3D structure from multiple images
  - Stereo correspondence
  - Image alignment & stitching
  - Image indexing and search
  - Motion tracking
- Find “interesting” pieces of the image.
  - Focus attention of algorithms
  - Speed up computation

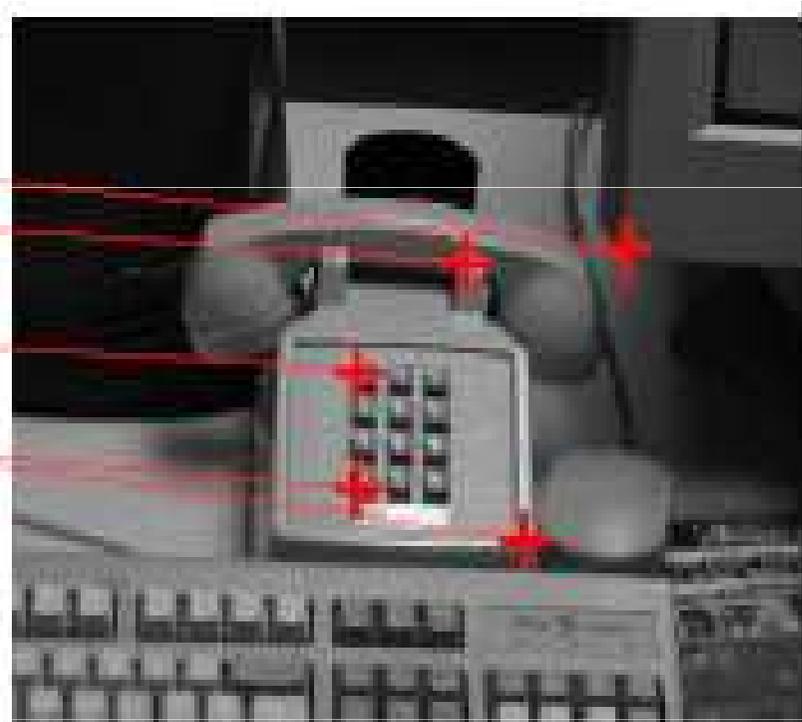
# Image matching applications



Object recognition: Find correspondences between feature points in training and test images.

# Image matching applications

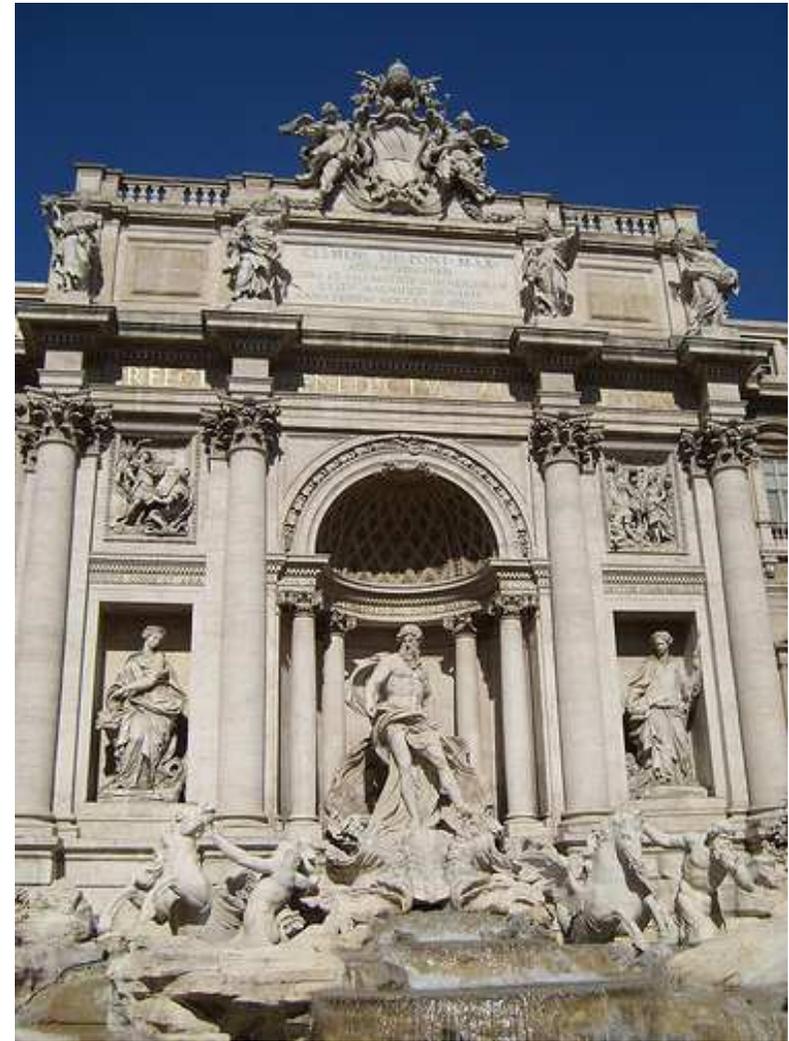
---



3D reconstruction: find correspondences between feature points in two images of the same scene.

# Image matching applications

---

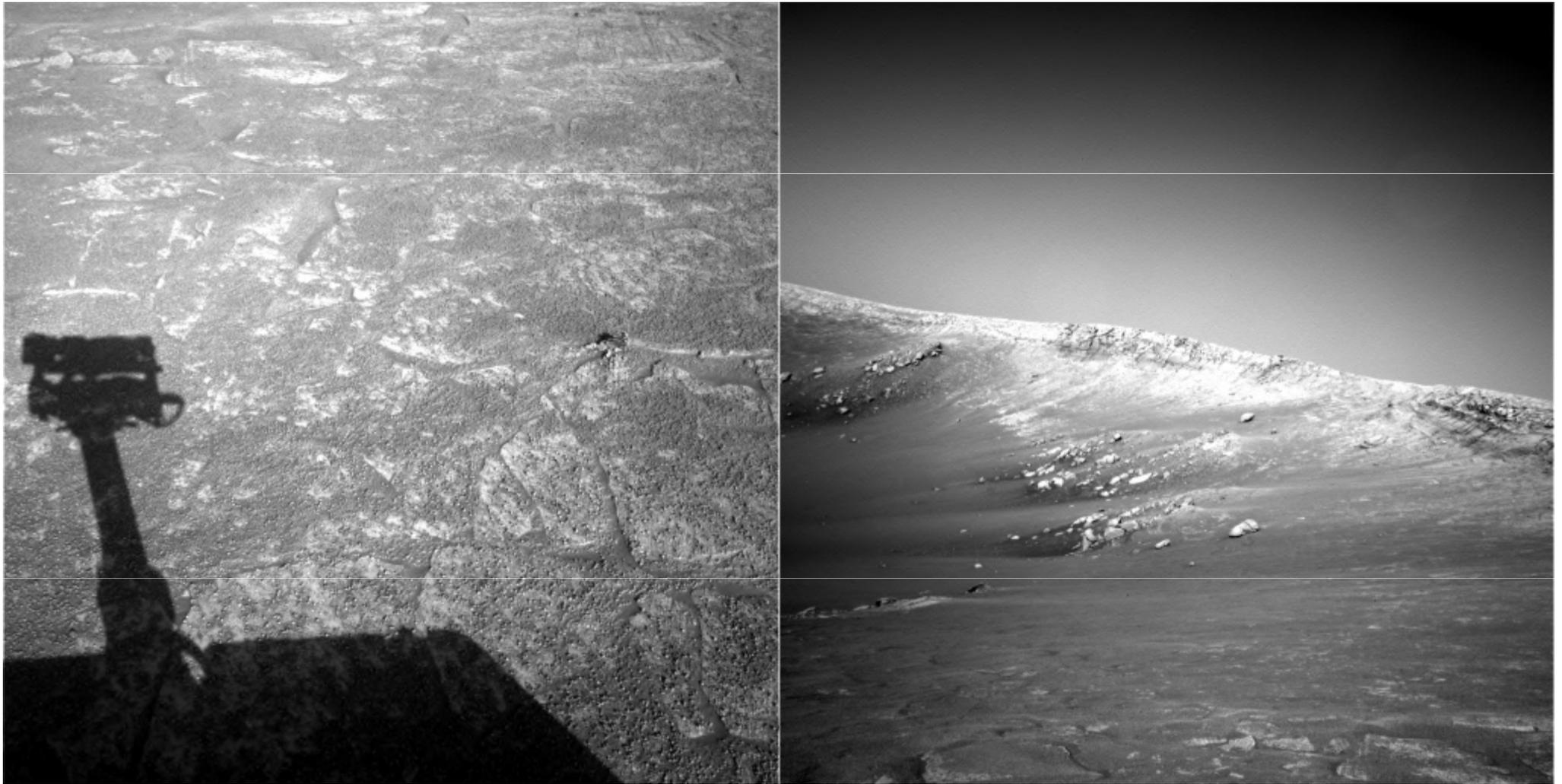


Two images of Rome from Flickr



# Image matching applications

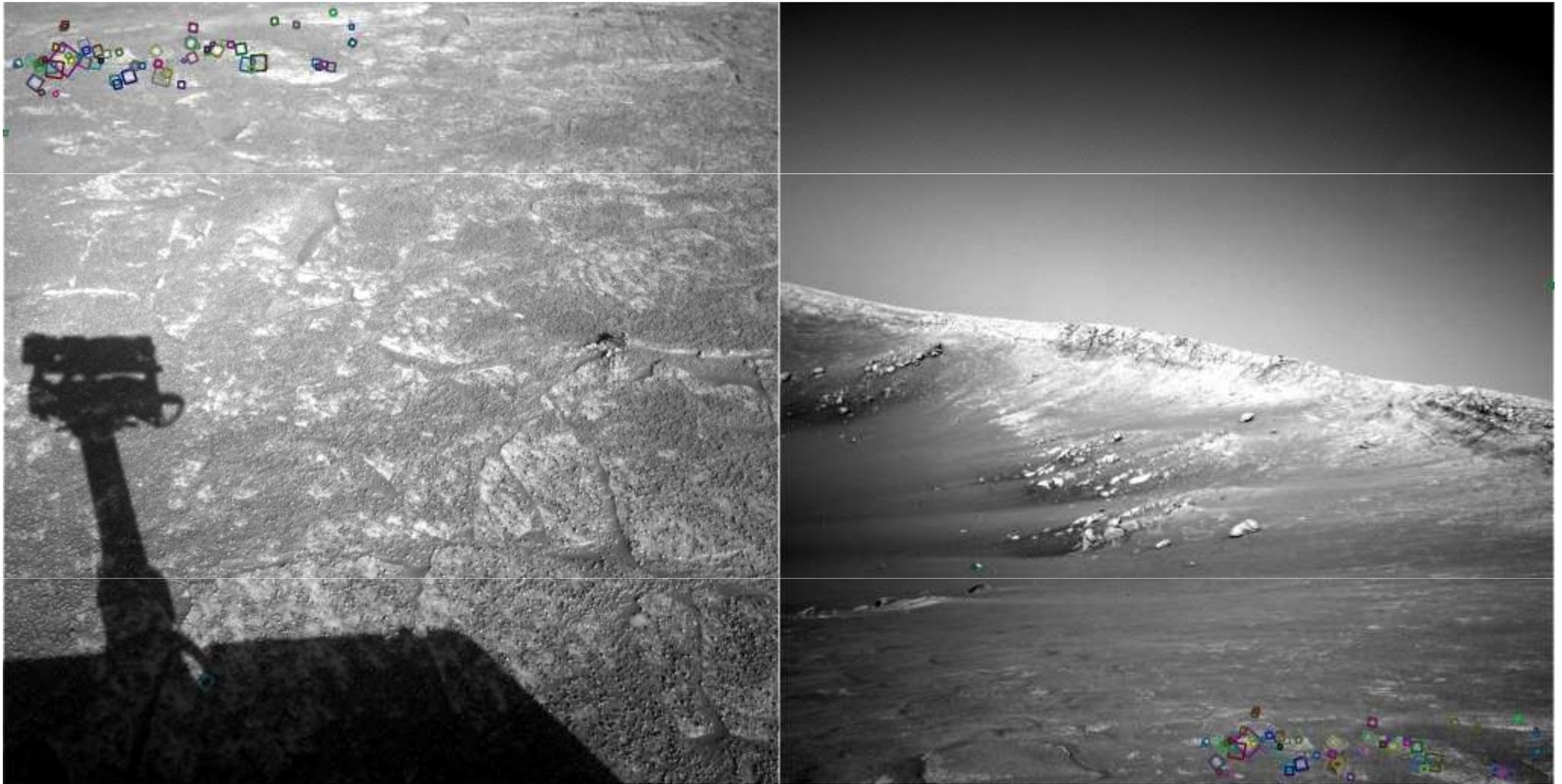
---



Two images from NASA Mars Rover: even harder case

# Image matching applications

---



Two images from NASA Mars Rover: matching using local features

# Image matching applications

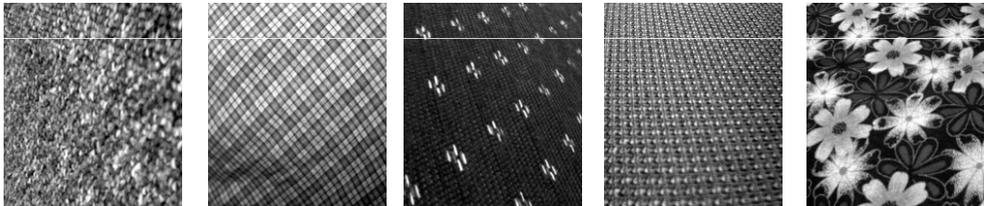
---

## Stereo correspondence



# Image matching applications

## Recognition



## Texture recognition



## Car detection

# Image matching applications

## 3D recognition



# Advantages of local features

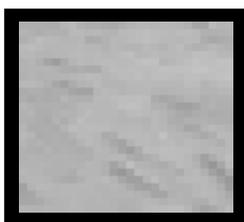
---

- **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

# Local features

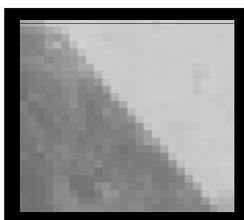
---

- What makes a good feature?
- We want uniqueness.
  - Look for image regions that are unusual.
  - Lead to unambiguous matches in other images.
- How to define “unusual”?



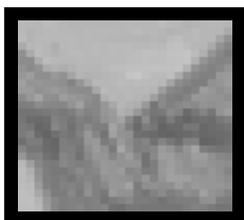
**0D** structure

➔ not useful for matching



**1D** structure

➔ edge, can be localized in 1D,  
subject to the aperture problem



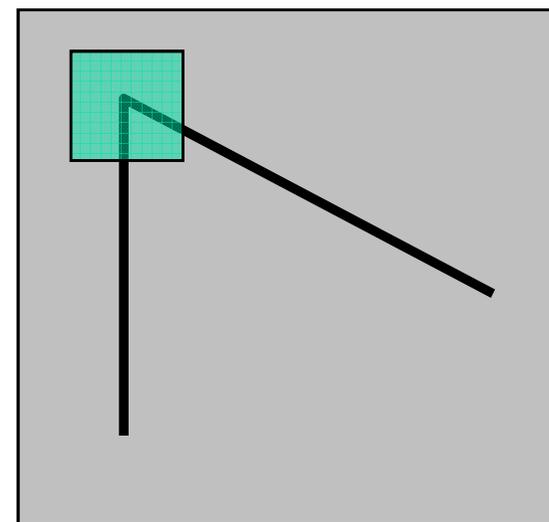
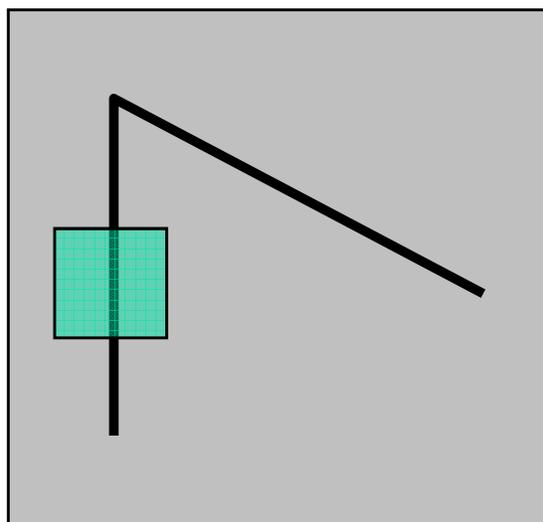
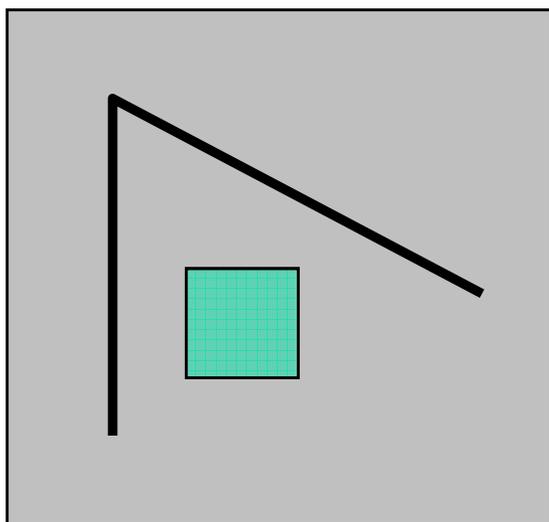
**2D** structure

➔ corner, can be localized in 2D,  
good for matching

# Local measures of uniqueness

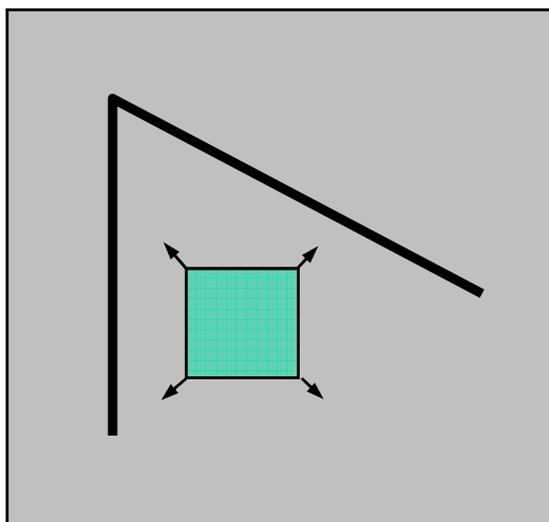
---

- Suppose we only consider a small window of pixels.
  - What defines whether a feature is a good or bad candidate?

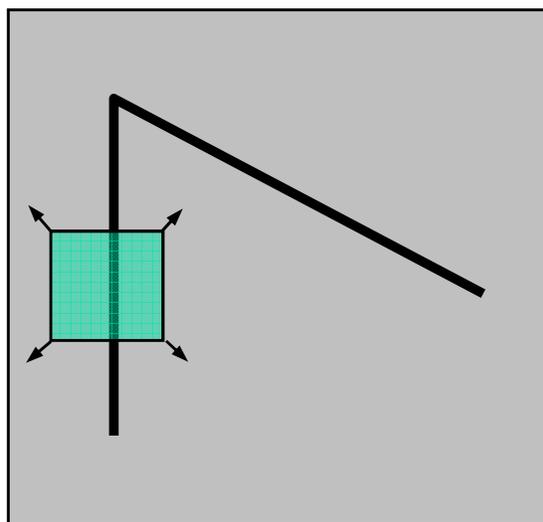


# Local measures of uniqueness

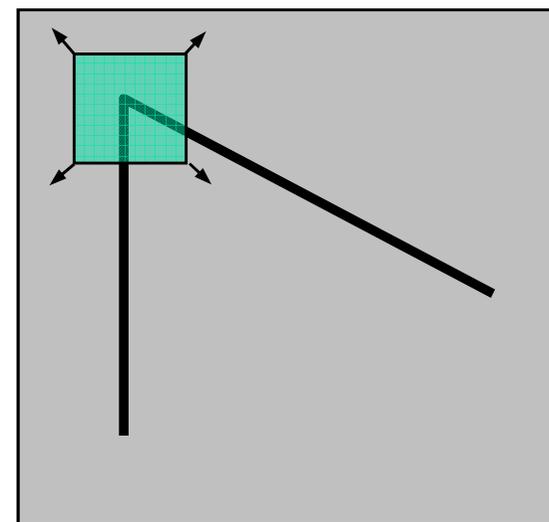
- Local measure of feature uniqueness:
  - How does the window change when you shift it?
  - Shifting the window in *any direction* causes a *big change*.



“flat” region:  
no change in all  
directions



“edge”:  
no change along  
the edge direction



“corner”:  
significant change  
in all directions

# Local features and image matching

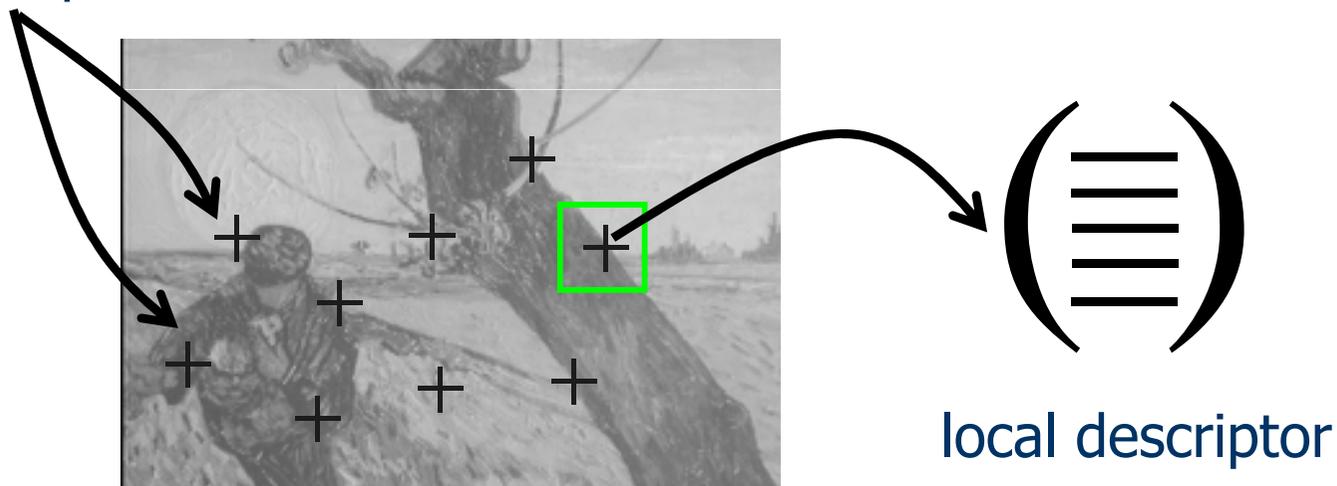
---

- There are three important requirements for feature points to have a better correspondence for matching:
  - Points corresponding to the same scene points should be **detected** consistently over different views.
  - They should be **invariant** to image scaling, rotation and to change in illumination and 3D camera viewpoint.
  - There should be enough information in the neighborhood of the points so that corresponding points can be automatically **matched**.
- These points are also called **interest points**.

# Overview of the approach

---

interest points



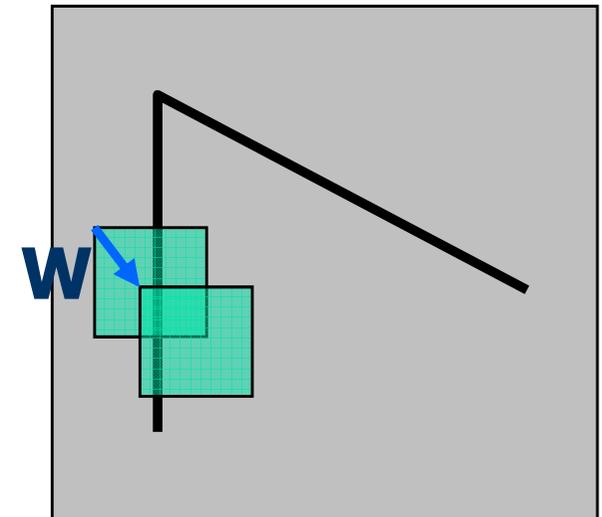
1. Extraction of interest points (characteristic locations).
2. Computation of local descriptors.
3. Determining correspondences.
4. Using these correspondences for matching/recognition/etc.

# Local features: detection

- Consider shifting the window  $W$  by  $(u,v)$ :

- How do the pixels in  $W$  change?
- Auto-correlation function measures the self similarity of a signal and is related to the sum-of-squared difference.
- Compare each pixel before and after the shift by summing up the squared differences (SSD).
- This defines an SSD “error” of  $E(u,v)$ :

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



# Local features: detection

---

- 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 assumed to be small, then first order approximation 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...

# Local features: detection

---

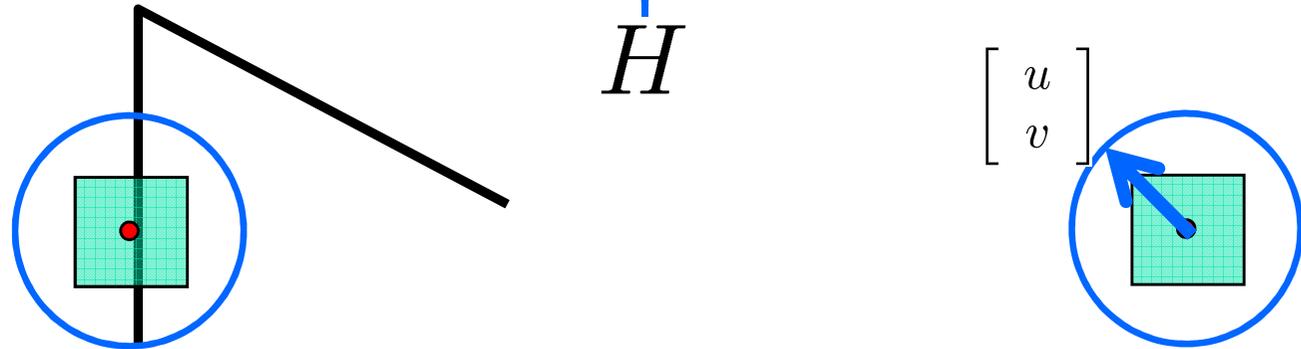
- Sum-of-squared differences error  $E(u,v)$ :

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

# Local features: detection

- 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 eigenvector/eigenvalue 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:

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

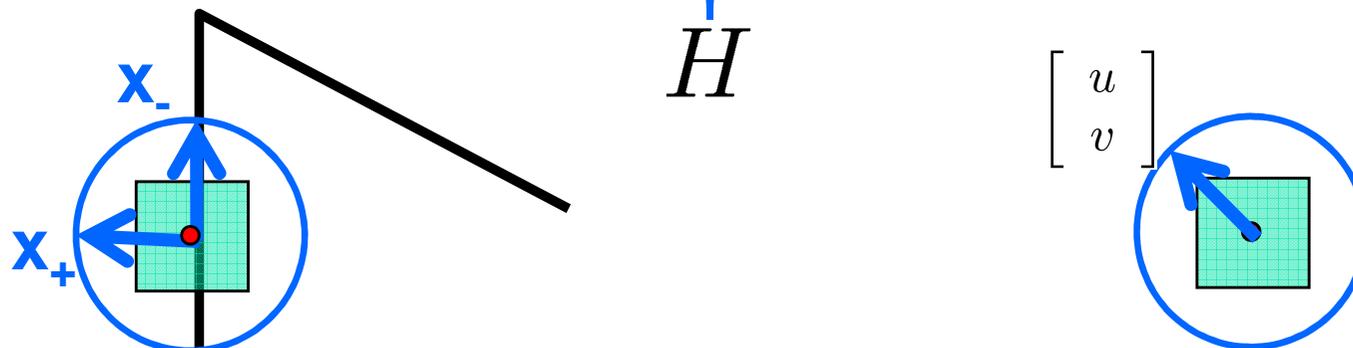
- Once you know  $\lambda$ , you find **x** by solving

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

# Local features: detection

- 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**:

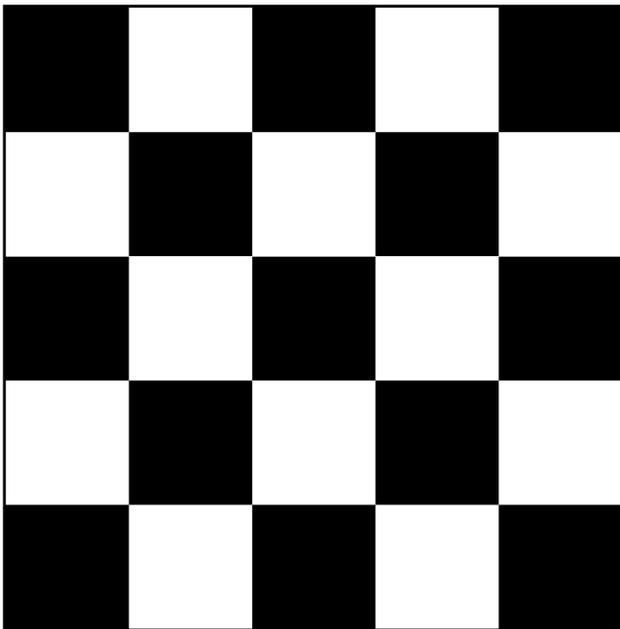
- Define shifts with the smallest and largest change (E value).
- $x_+$  = direction of largest increase in E.
- $\lambda_+$  = amount of increase in direction  $x_+$ .
- $x_-$  = direction of smallest increase in E.
- $\lambda_-$  = amount of increase in direction  $x_-$ .

$$H x_+ = \lambda_+ x_+$$

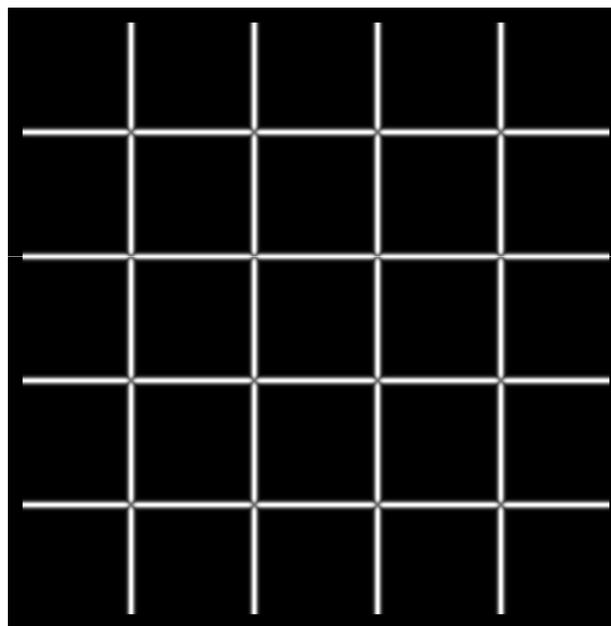
$$H x_- = \lambda_- x_-$$

# Local features: detection

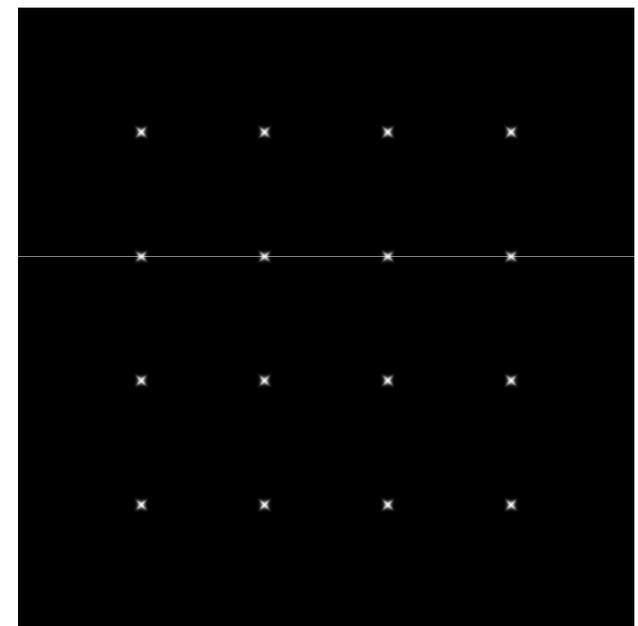
- How are  $\lambda_+$ ,  $x_+$ ,  $\lambda_-$ , and  $x_-$  relevant for feature detection?
  - What's our feature scoring function?
- Want  $E(u,v)$  to be large for small shifts in all directions.
  - The minimum of  $E(u,v)$  should be large, over all unit vectors  $[u \ v]$ .
  - This minimum is given by the smaller eigenvalue ( $\lambda_-$ ) of  $\mathbf{H}$ .



$I$



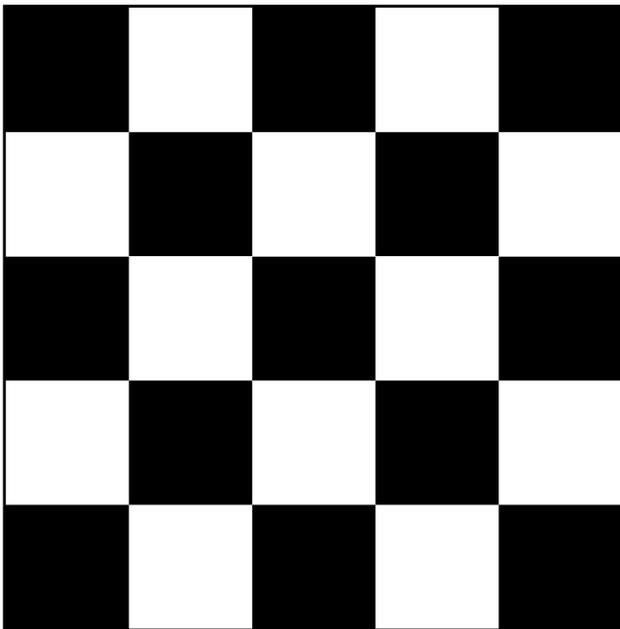
$\lambda_+$



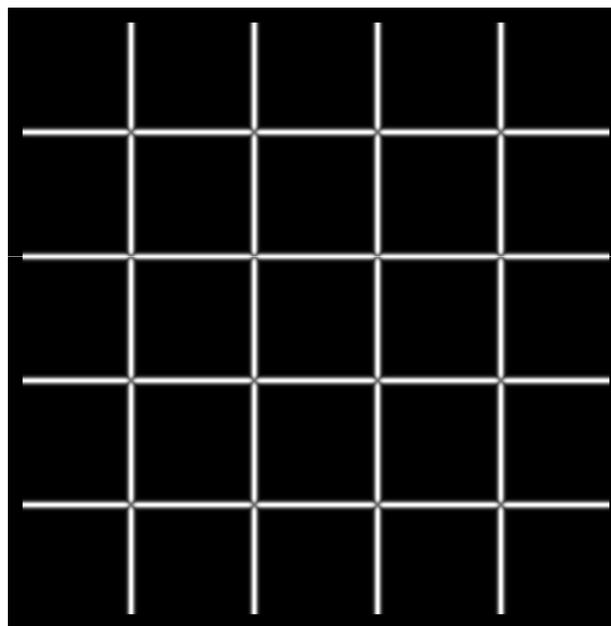
$\lambda_-$

# Local features: detection

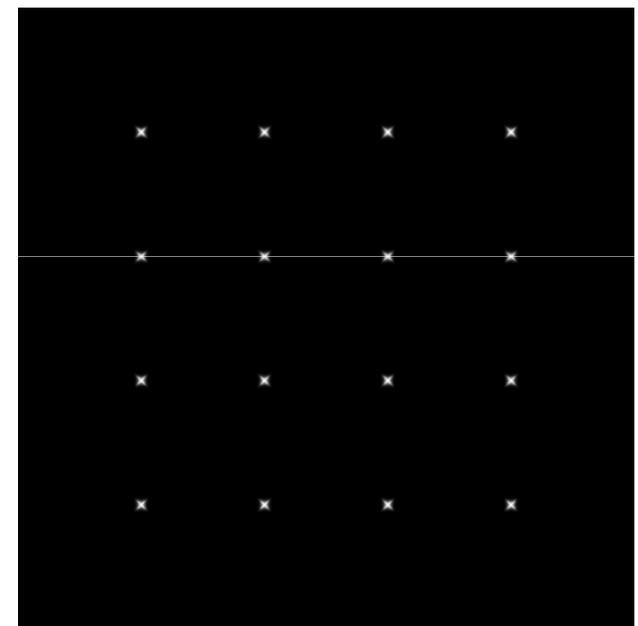
- Here's what you do:
  - Compute the gradient at each point in the image.
  - Create the  $\mathbf{H}$  matrix from the entries in the gradient.
  - Compute the eigenvalues.
  - Find points with large response ( $\lambda_- > \text{threshold}$ ).
  - Choose those points where  $\lambda_-$  is a local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



$I$



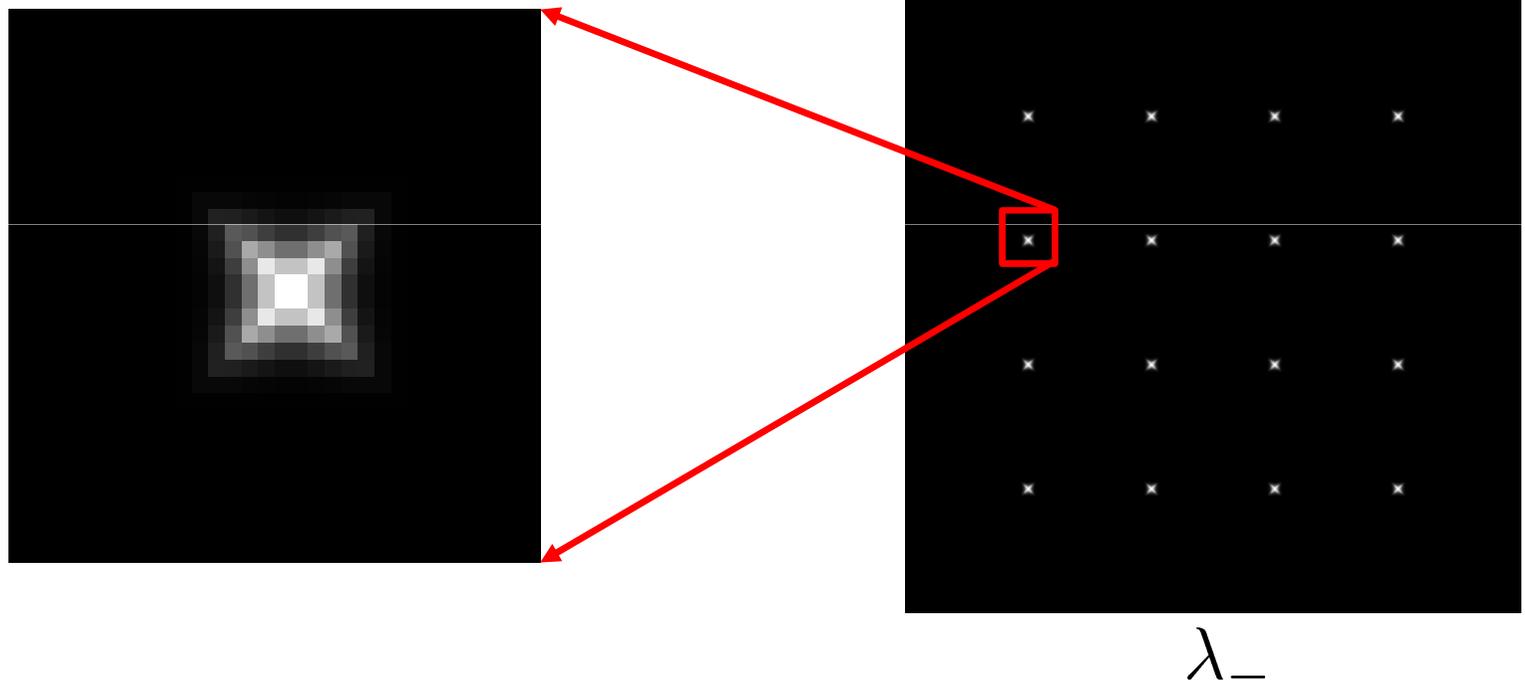
$\lambda_+$



$\lambda_-$

# Local features: detection

- Here's what you do:
  - Compute the gradient at each point in the image.
  - Create the  $\mathbf{H}$  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.



# Harris detector

---

- To measure the corner strength:

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

where

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

$$\det(H) = \lambda_1 \times \lambda_2$$

( $\lambda_1$  and  $\lambda_2$  are the eigenvalues of  $H$ ).

- $R$  is positive for corners, negative in edge regions, and small in flat regions.
- Very similar to  $\lambda_-$  but less expensive (no square root).
- Also called the “Harris Corner Detector” or “Harris Operator”.
- Lots of other detectors, this is one of the most popular.

# Harris detector

1. Compute  $x$  and  $y$  derivatives of image

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

2. Compute products of derivatives at every pixel

$$I_{x2} = I_x \cdot I_x \quad I_{y2} = I_y \cdot I_y \quad I_{xy} = I_x \cdot I_y$$

3. Compute the sums of the products of derivatives at each pixel

$$S_{x2} = G_{\sigma^2} * I_{x2} \quad S_{y2} = G_{\sigma^2} * I_{y2} \quad S_{xy} = G_{\sigma^2} * I_{xy}$$

4. Define at each pixel  $(x, y)$  the matrix

$$H(x, y) = \begin{bmatrix} S_{x2}(x, y) & S_{xy}(x, y) \\ S_{xy}(x, y) & S_{y2}(x, y) \end{bmatrix}$$

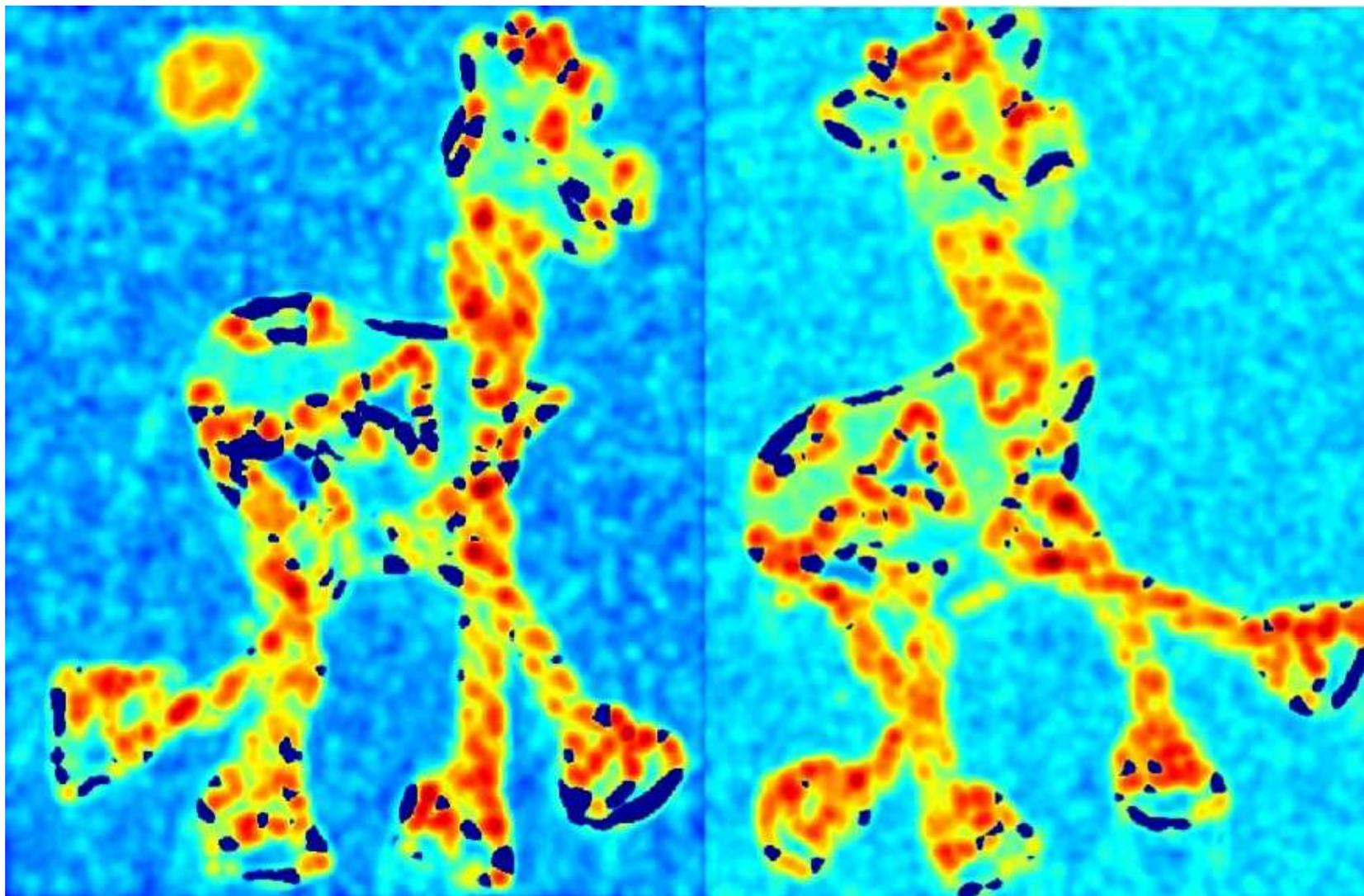
5. Compute the response of the detector at each pixel

$$R = \text{Det}(H) - k(\text{Trace}(H))^2$$

# Harris detector example



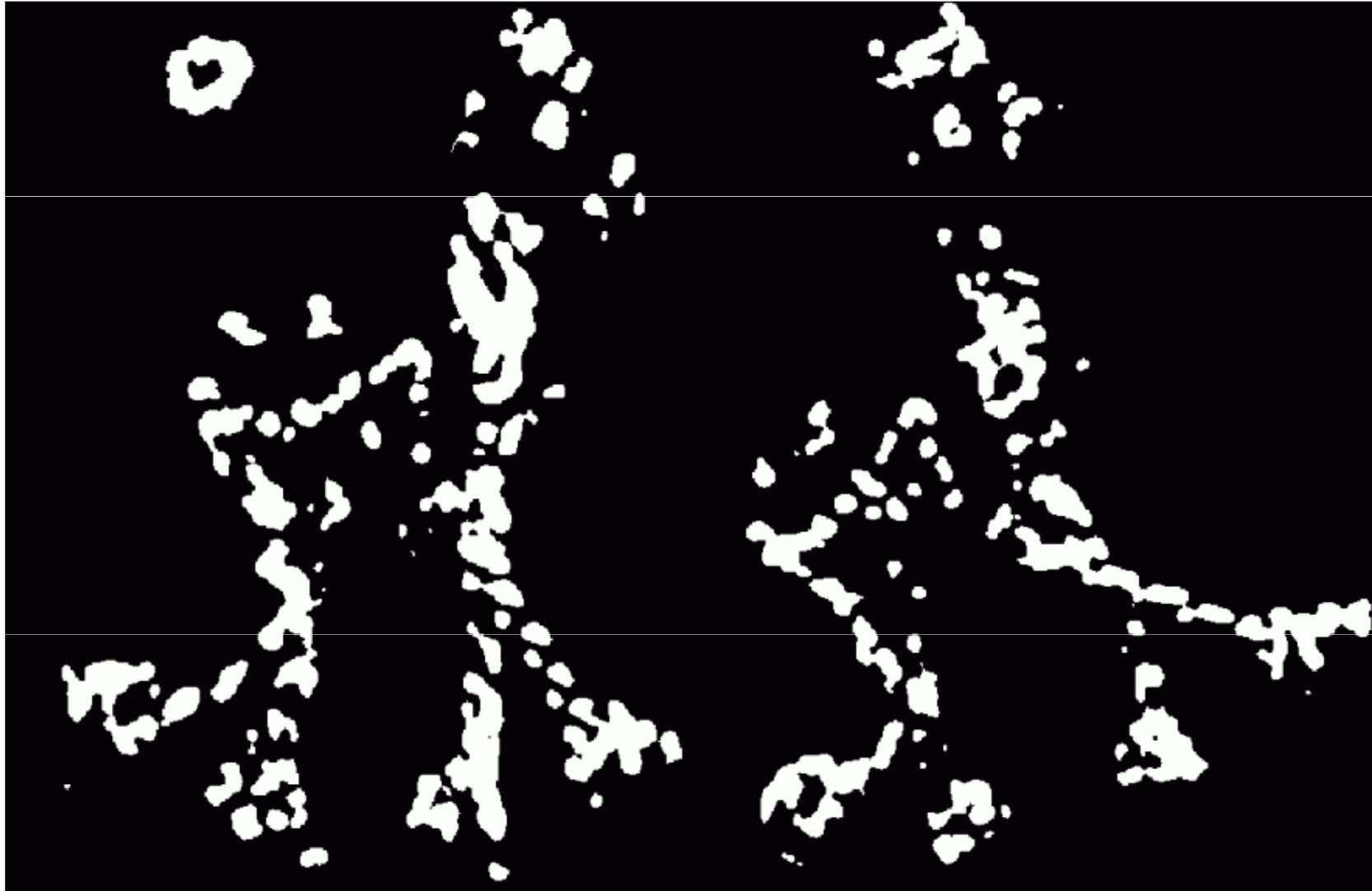
# Harris detector example



R values (red high, blue low)

# Harris detector example

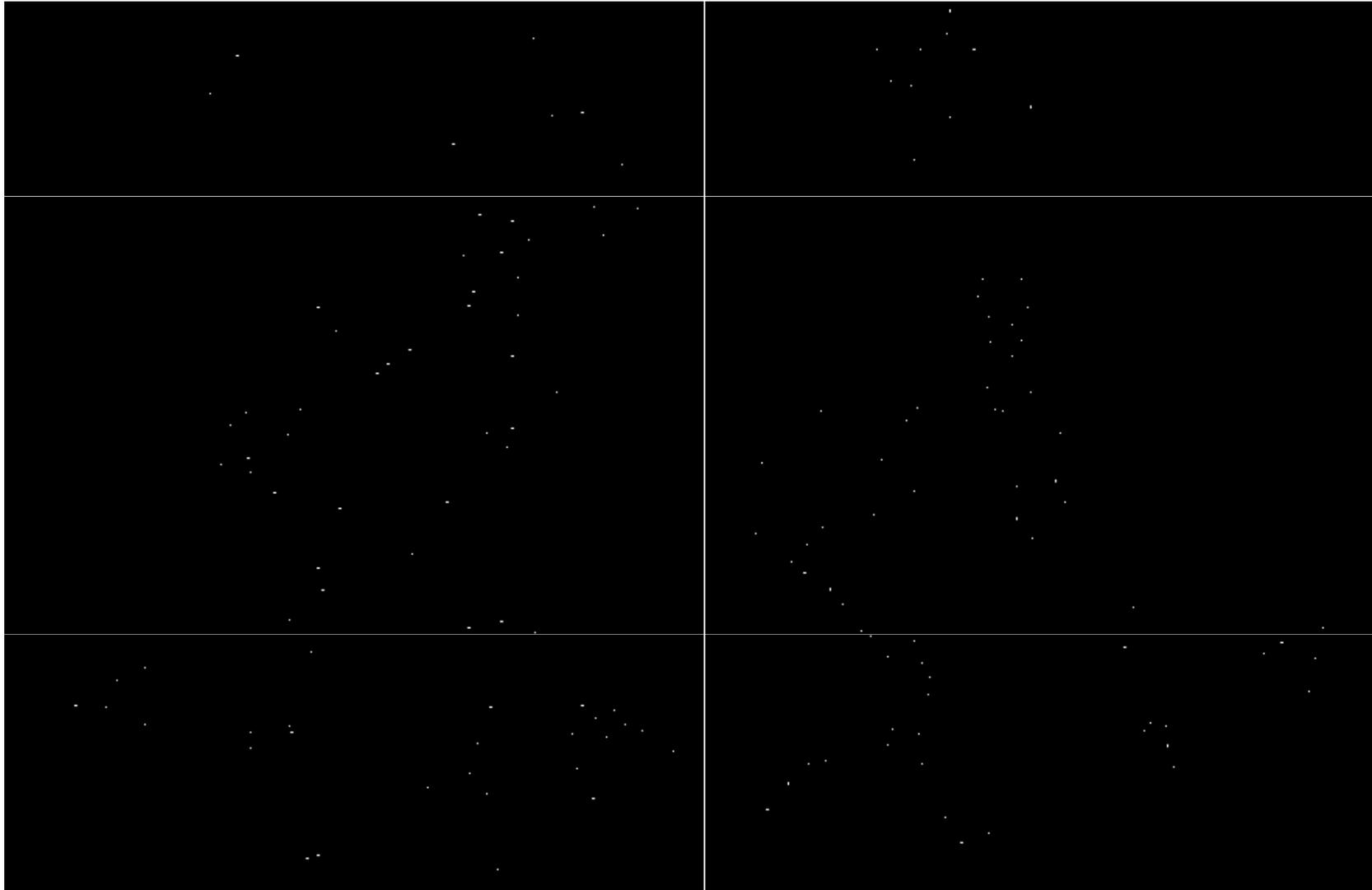
---



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

# Harris detector example

---



Local maxima of R

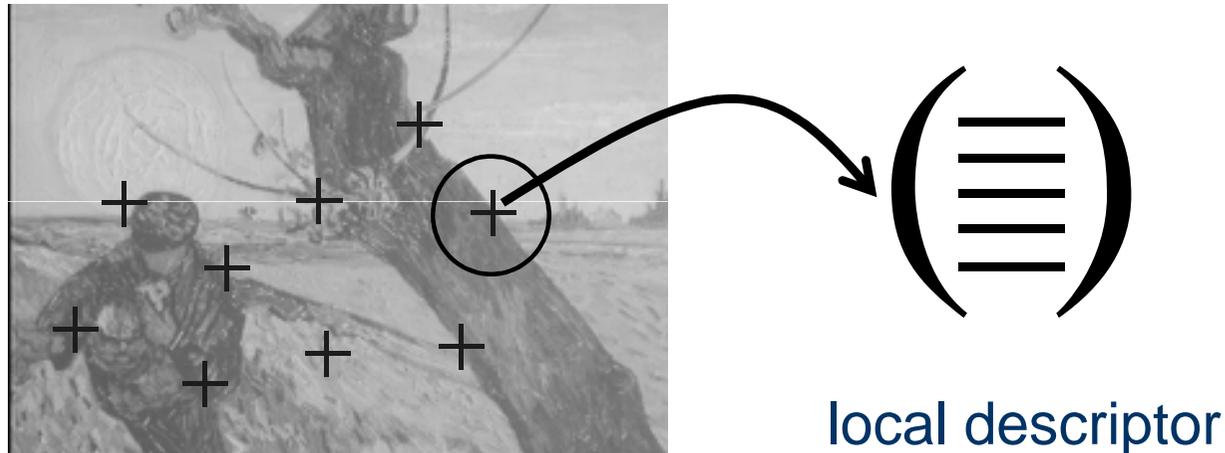
# Harris detector example



Harris features (red)

# Local features: descriptors

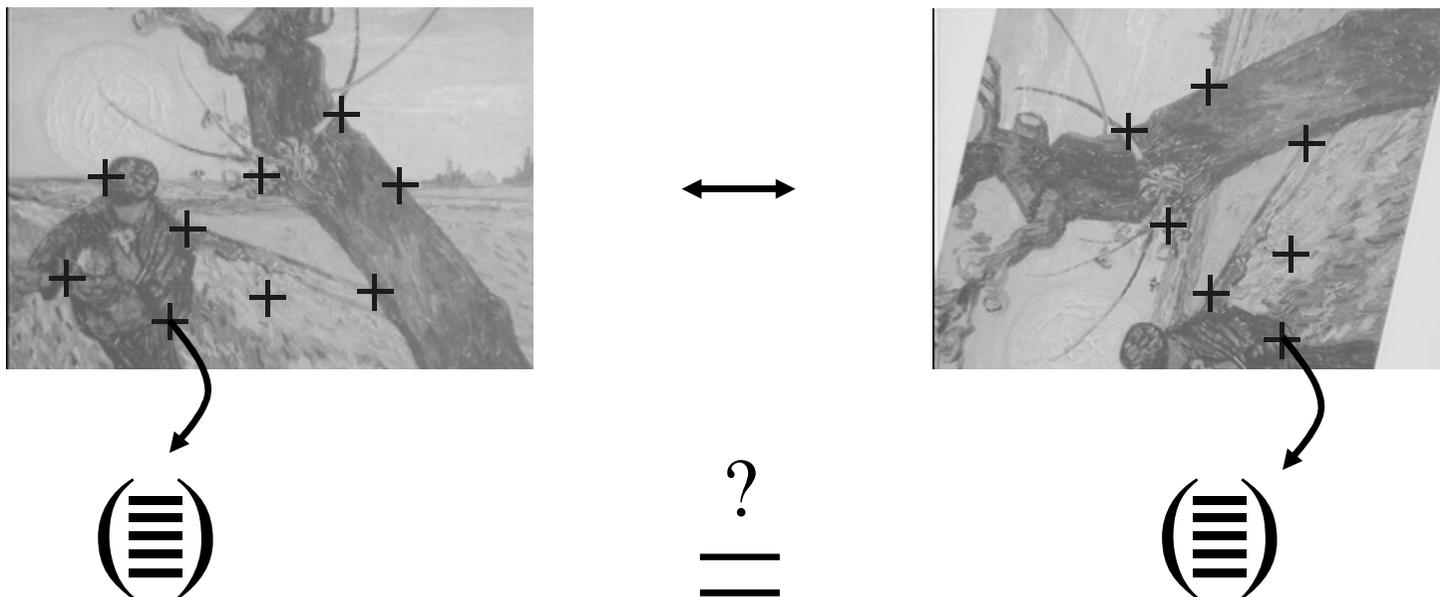
---



- Descriptors characterize the local neighborhood of a point.
  - Gray values within a square window around the point can be used directly.
  - Gray value derivatives or differential invariants can also be used.

# Local features: matching

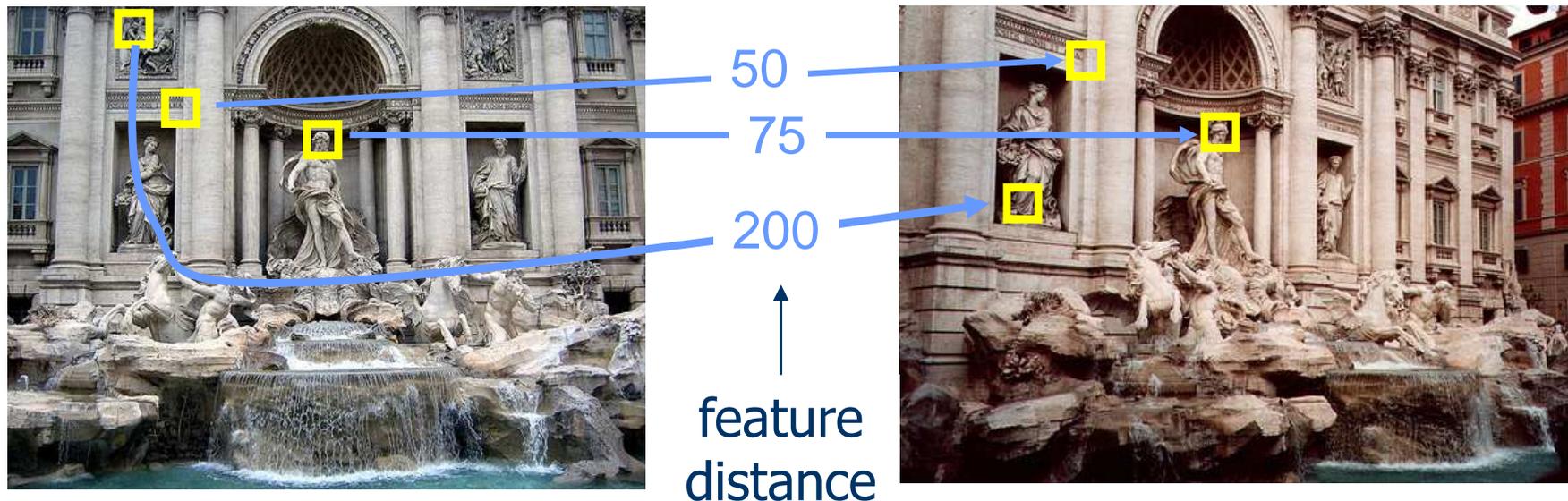
- We know how to detect good features.
- Next question: how to match them?



- Vector comparison using a distance measure can be used.

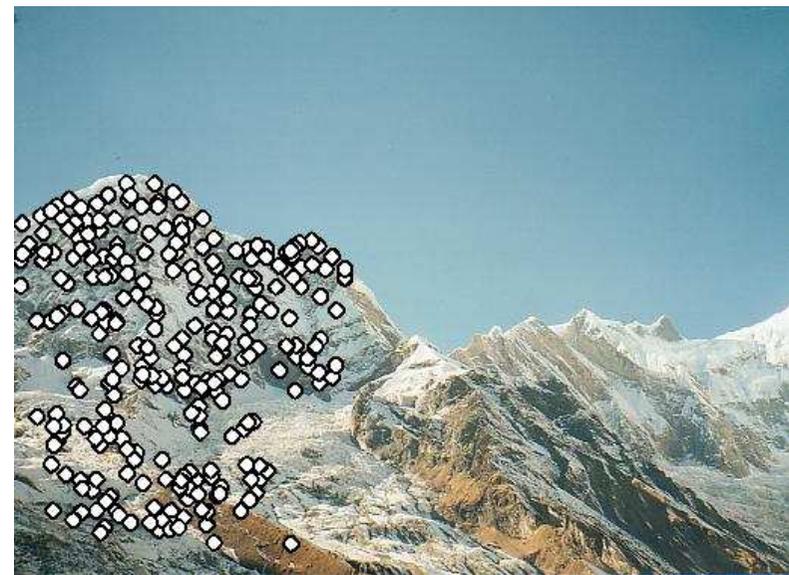
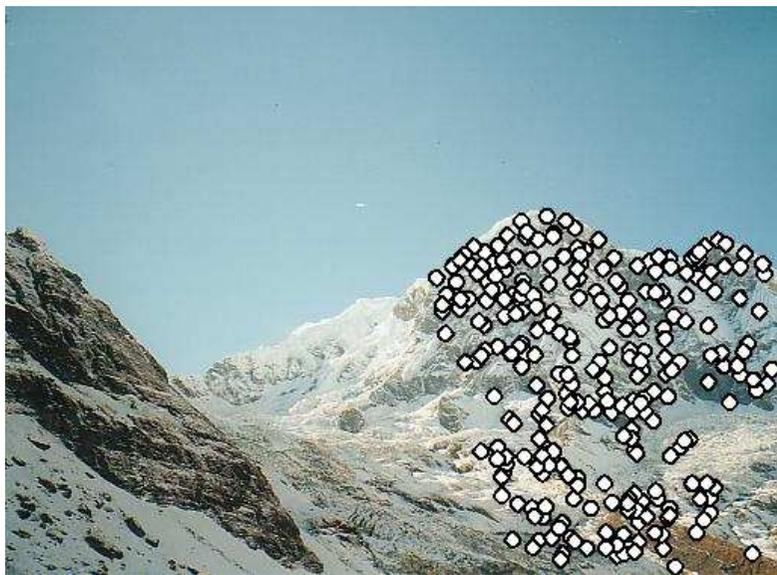
# Local features: matching

- Given a feature in  $I_1$ , how to find the best match in  $I_2$ ?
  1. Define a distance function that compares two descriptors.
  2. Test all the features in  $I_2$ , find the one with minimum distance.

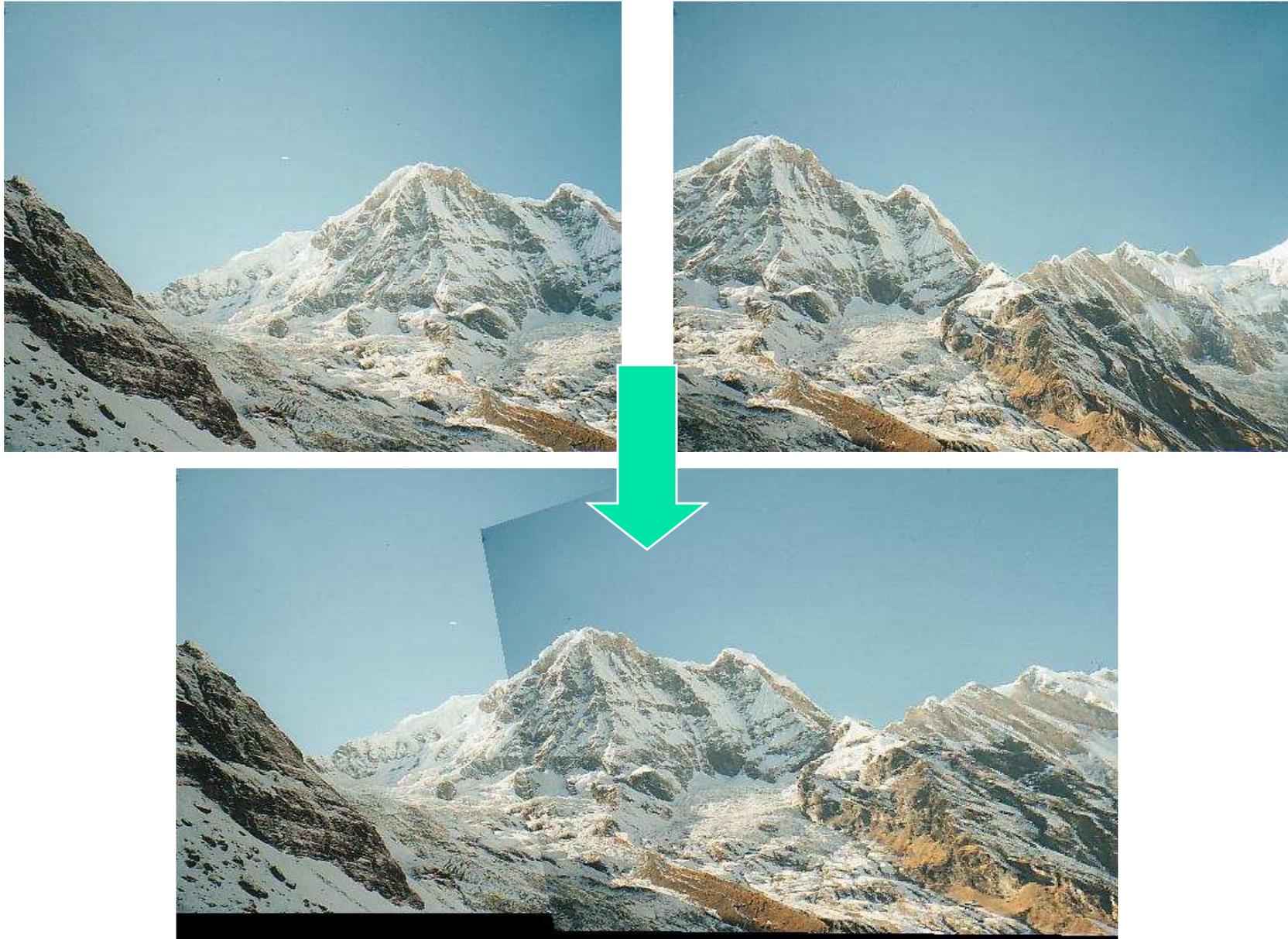


# Matching examples

---



# Matching examples



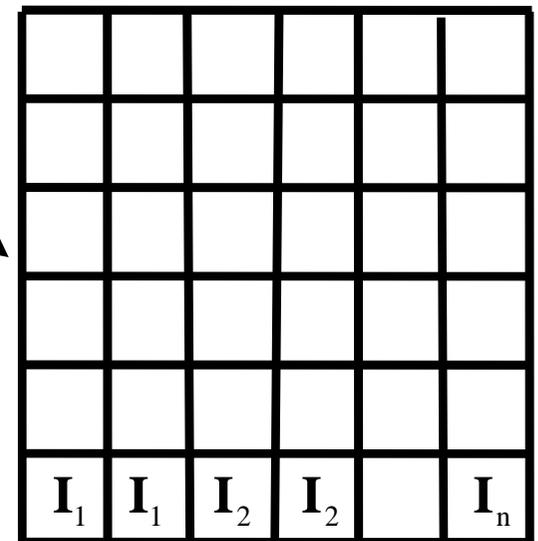
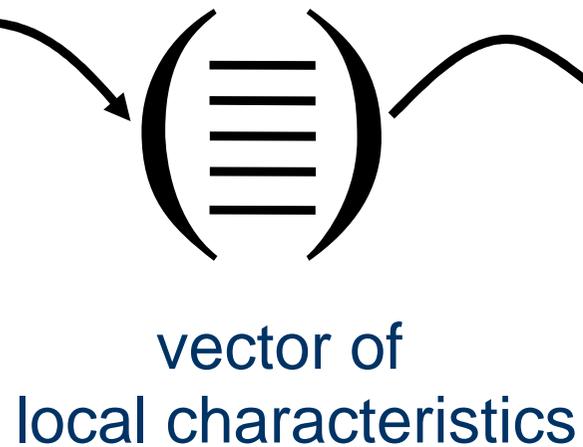
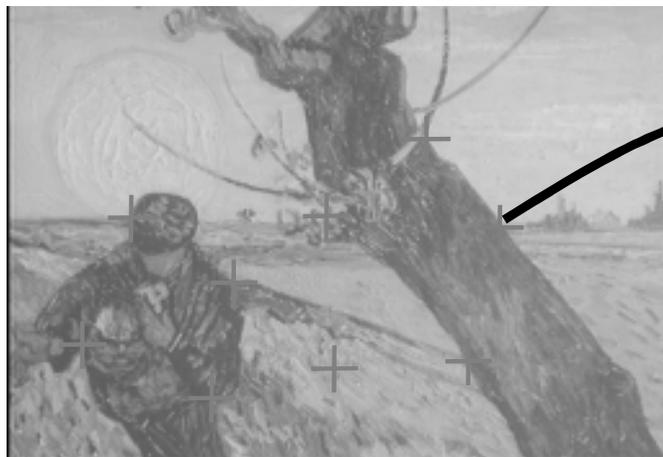
# Retrieval examples

---

- Retrieval in a large database using:
  - voting algorithm
  - additional constraints
- Rapid access with an indexing mechanism.

# Retrieval examples

- Voting algorithm:



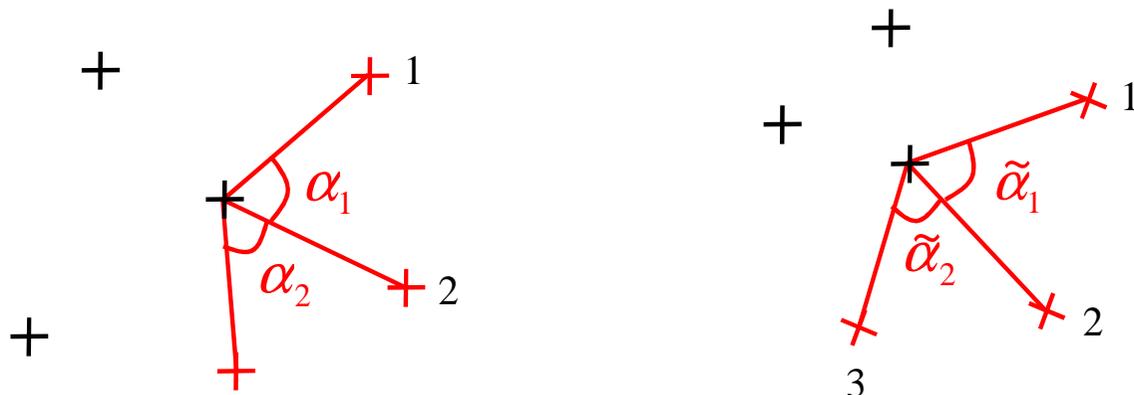
# Retrieval examples

---

- Compute a set of features  $V$  around each interest point for each image in the database.
- For a query image compute the same model.
- Compare the vectors for each of the interest points in the query image with all the models in the database.
- If distance is below some threshold then give a vote to the corresponding model.

# Retrieval examples

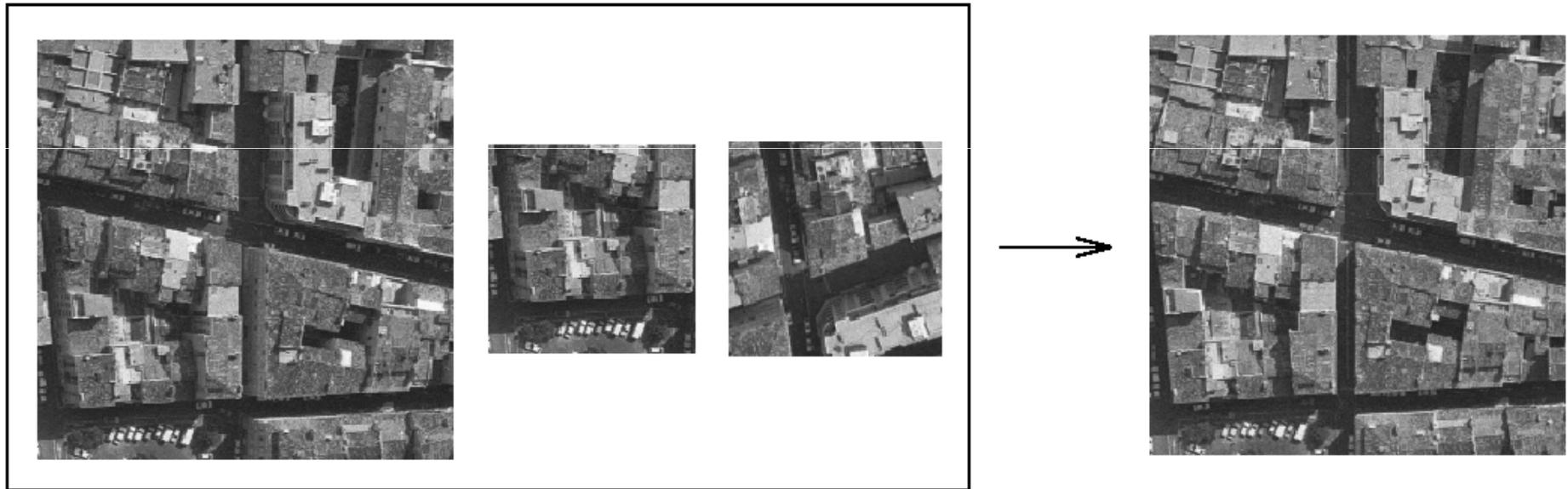
- Semi-local constraints
  - neighboring points should match
  - angles, length ratios should be similar



- Global constraints
- Robust estimation of the image transformation (homography, epipolar geometry)

# Retrieval examples

---

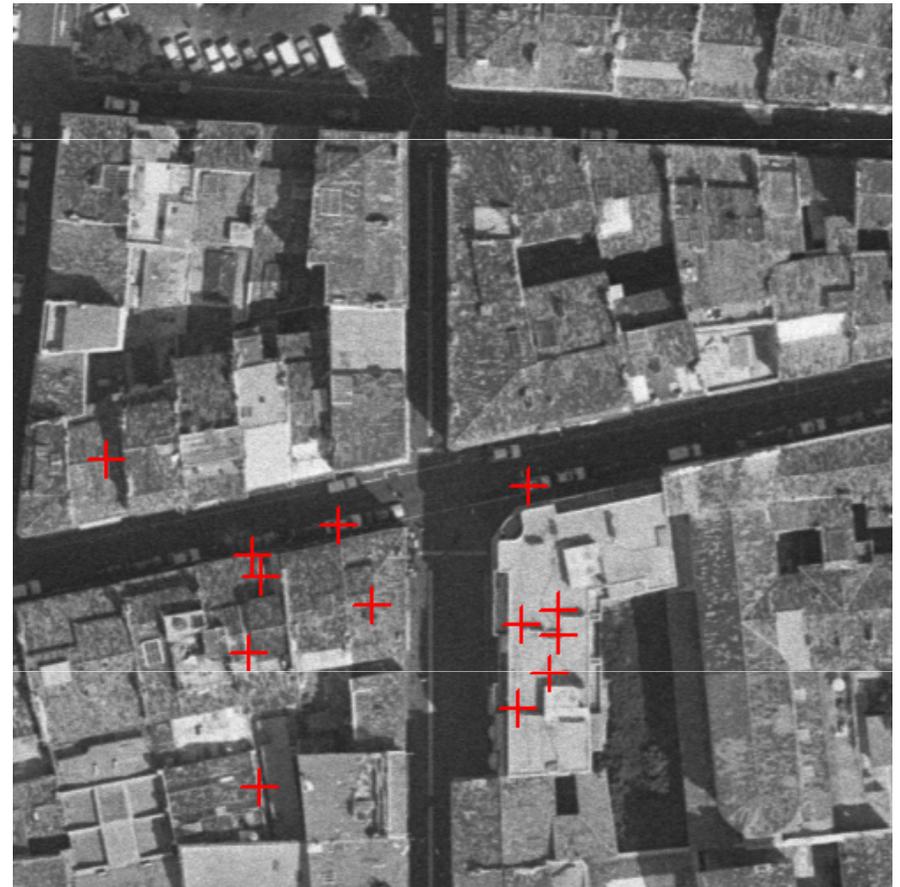
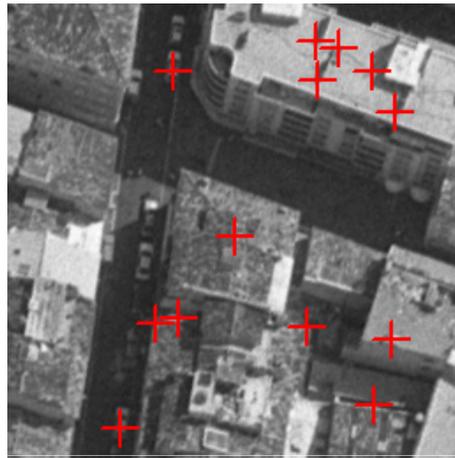


database with  $\sim 1000$  images

The image on the right is correctly retrieved using any of the images on the left.

# Retrieval examples

---



# Summary of the approach

---

- Detection of interest points/regions
  - Harris detector
  - Blob detector based on Laplacian
- Computation of descriptors for each point
  - Gray value patch, differential invariants, steerable filter, SIFT descriptor
- Similarity of descriptors
  - Correlation, Mahalanobis distance, Euclidean distance
- Semi-local constraints
  - Geometrical or statistical relations between neighborhood points
- Global verification
  - Robust estimation of geometry between images

# Local features: invariance

---

- Suppose you rotate the image by some angle.
  - Will you still pick up the same features?
- What if you change the brightness?
- What about scale?
- 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).

# Local features: invariance



original

translated

rotated

scaled

	Translation	Rotation	Scale
Is Harris invariant?	?	?	?
Is correlation invariant?	?	?	?

# Local features: invariance



original

translated

rotated

scaled

	Translation	Rotation	Scale
Is Harris invariant?	YES	YES	NO
Is correlation invariant?	YES	NO	NO

# 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 descriptors

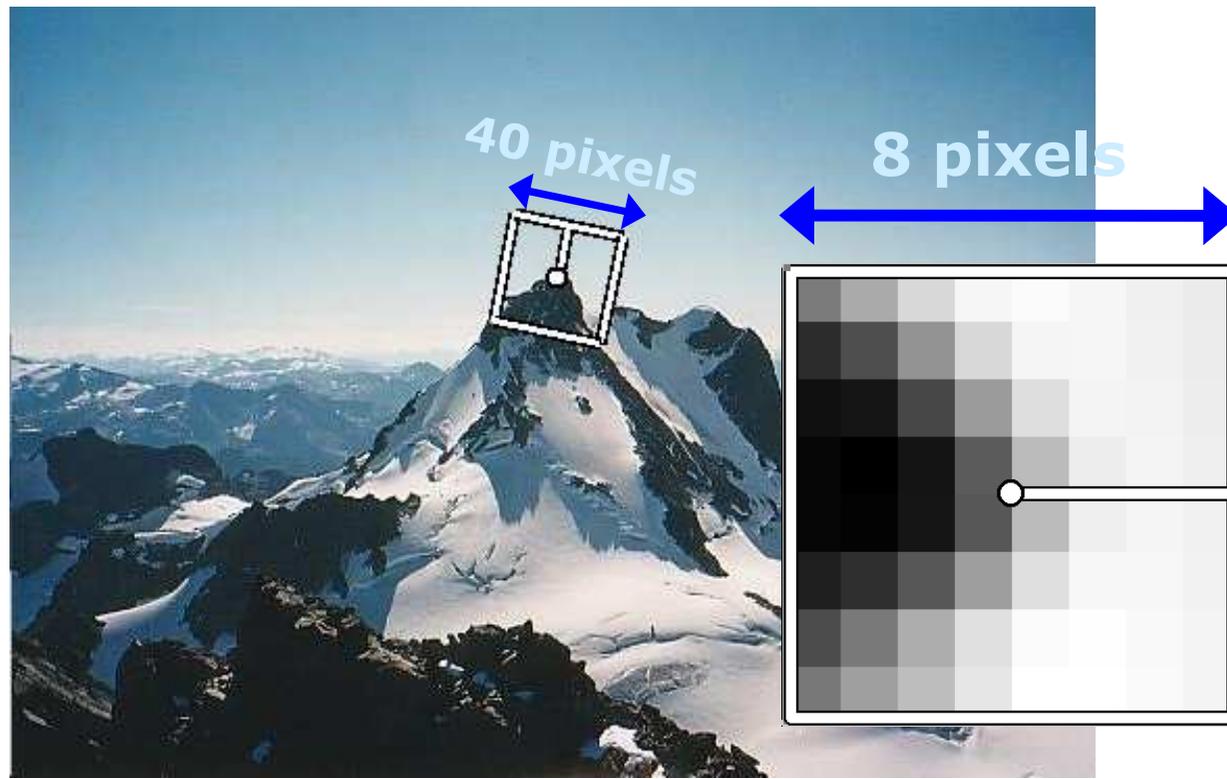
---

- Find dominant orientation of the image patch.
  - This is given by  $\mathbf{x}_+$ , the eigenvector of  $\mathbf{H}$  corresponding to  $\lambda_+$  (larger eigenvalue).
  - Rotate the patch according to this angle.



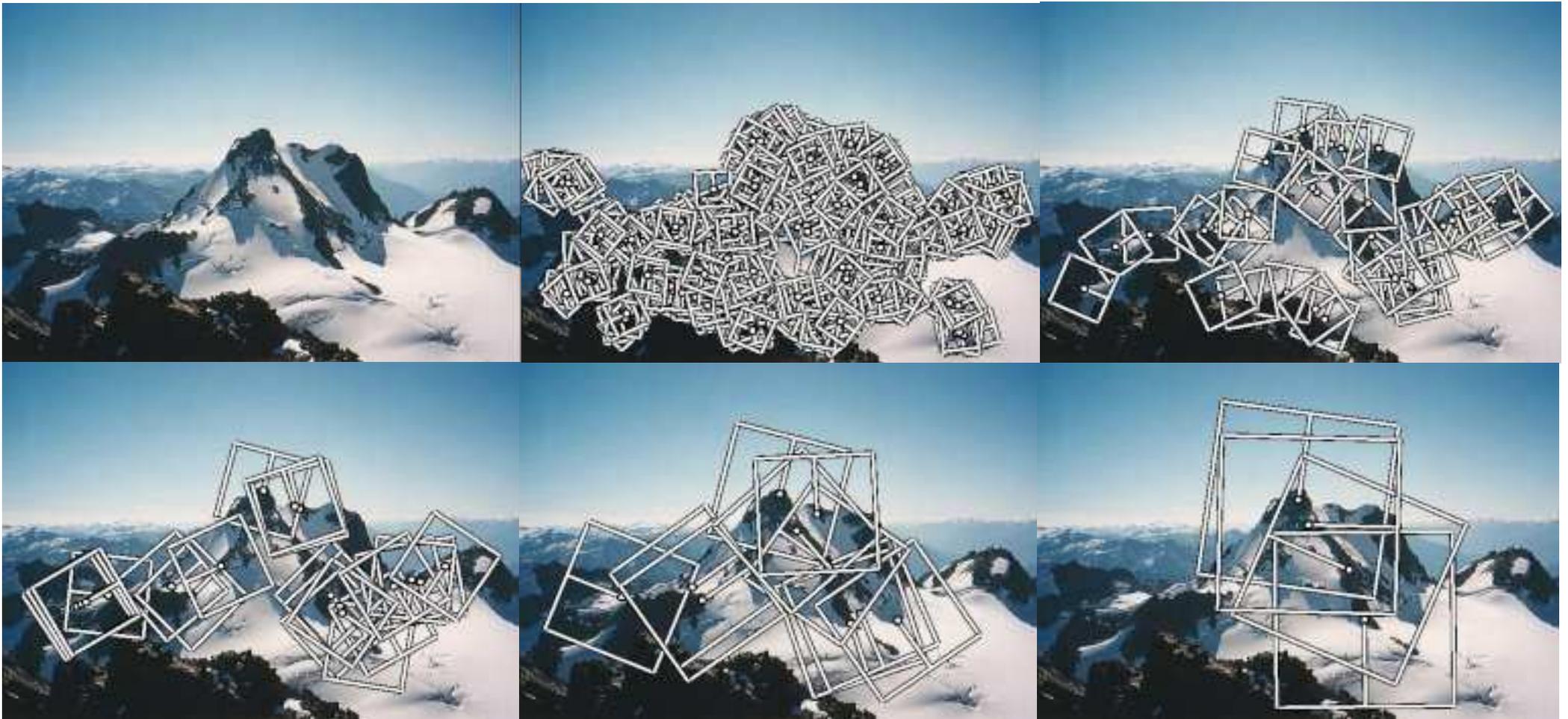
# Multi-scale Oriented Patches (MOPS)

- 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.



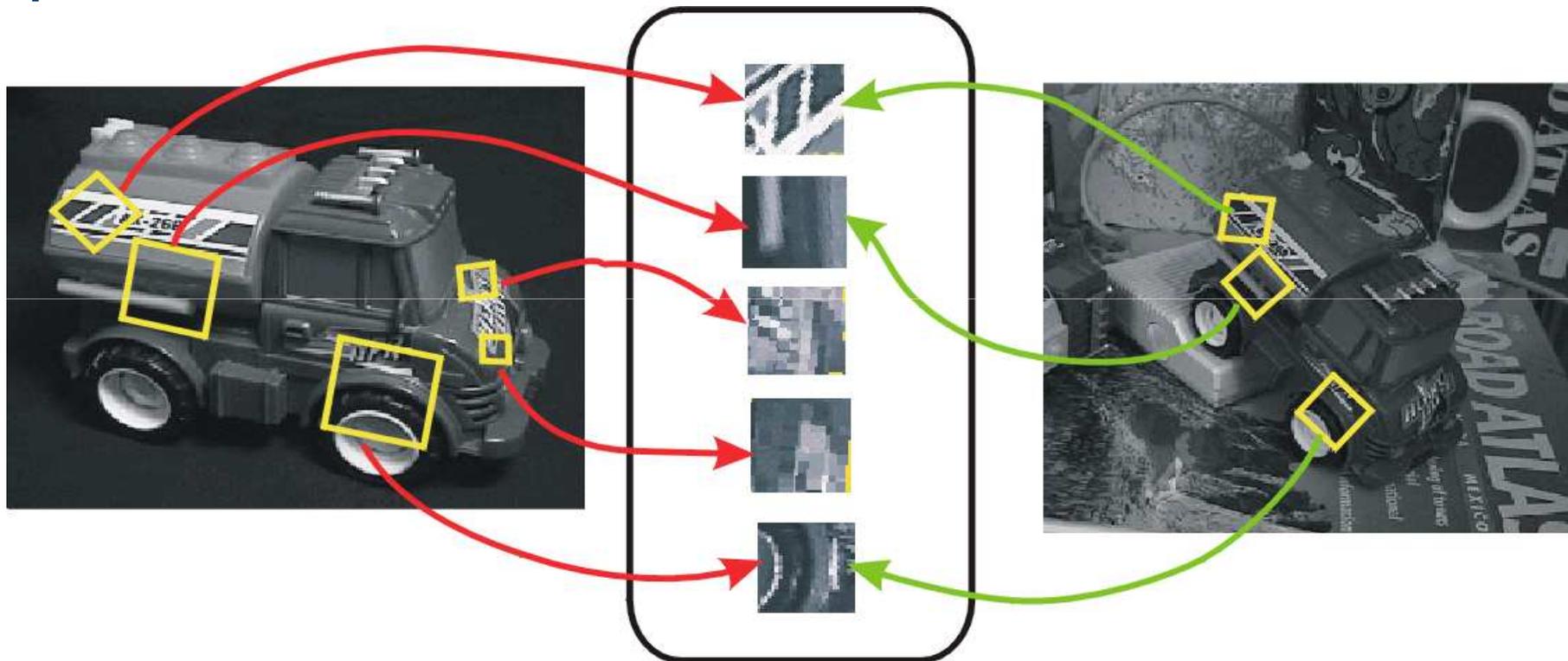
# Multi-scale Oriented Patches (MOPS)

- Extract oriented patches at multiple scales of the Gaussian pyramid.



# Scale Invariant Feature Transform (SIFT)

- The SIFT operator developed by David Lowe is both a detector and a descriptor that are invariant to translation, rotation, scale, and other imaging parameters.



# Overall approach for SIFT

---

1. **Scale space extrema detection**
  - Search over multiple scales and image locations.
2. **Interest point localization**
  - Fit a model to determine location and scale.
  - Select interest points based on a measure of stability.
3. **Orientation assignment**
  - Compute best orientation(s) for each interest point region.
4. **Interest point description**
  - Use local image gradients at selected scale and rotation to describe each interest point region.

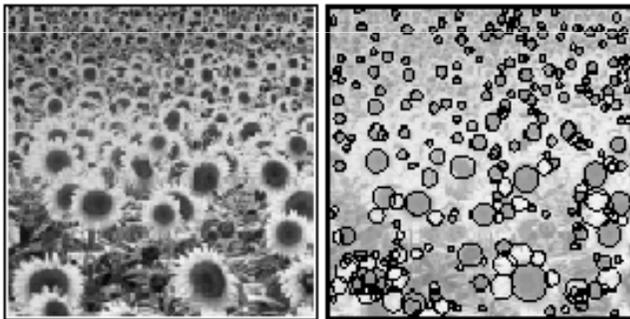
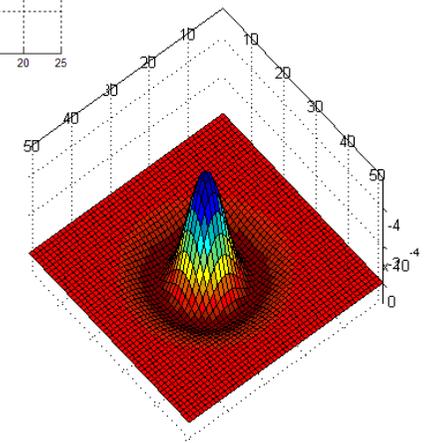
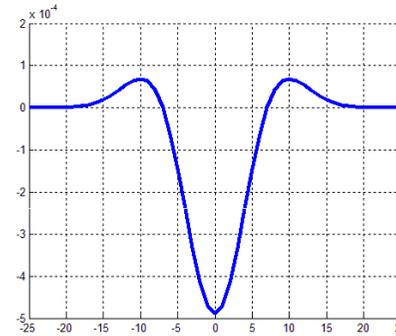
# Scale space extrema detection

---

- **Goal:** Identify locations and scales that can be repeatably assigned under different views of the same scene or object.
- **Method:** search for stable features across multiple scales using a continuous function of scale.
- **Prior work** has shown that under a variety of assumptions, the best function is a **Gaussian function**.
- **The scale space of an image is a function  $L(x,y,\sigma)$**  that is produced from the convolution of a Gaussian kernel (at different scales) with the input image.

# Scale space interest points

- Laplacian of Gaussian kernel
  - Scale normalized
  - Proposed by Lindeberg
- Scale space detection
  - Find local maxima across scale/space
  - A good “blob” detector



$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2} \frac{x^2+y^2}{\sigma^2}}$$

$$\nabla^2 G(x, y, \sigma) = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$$

# Scale space interest points

---

- Scale space function L

- Gaussian convolution

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

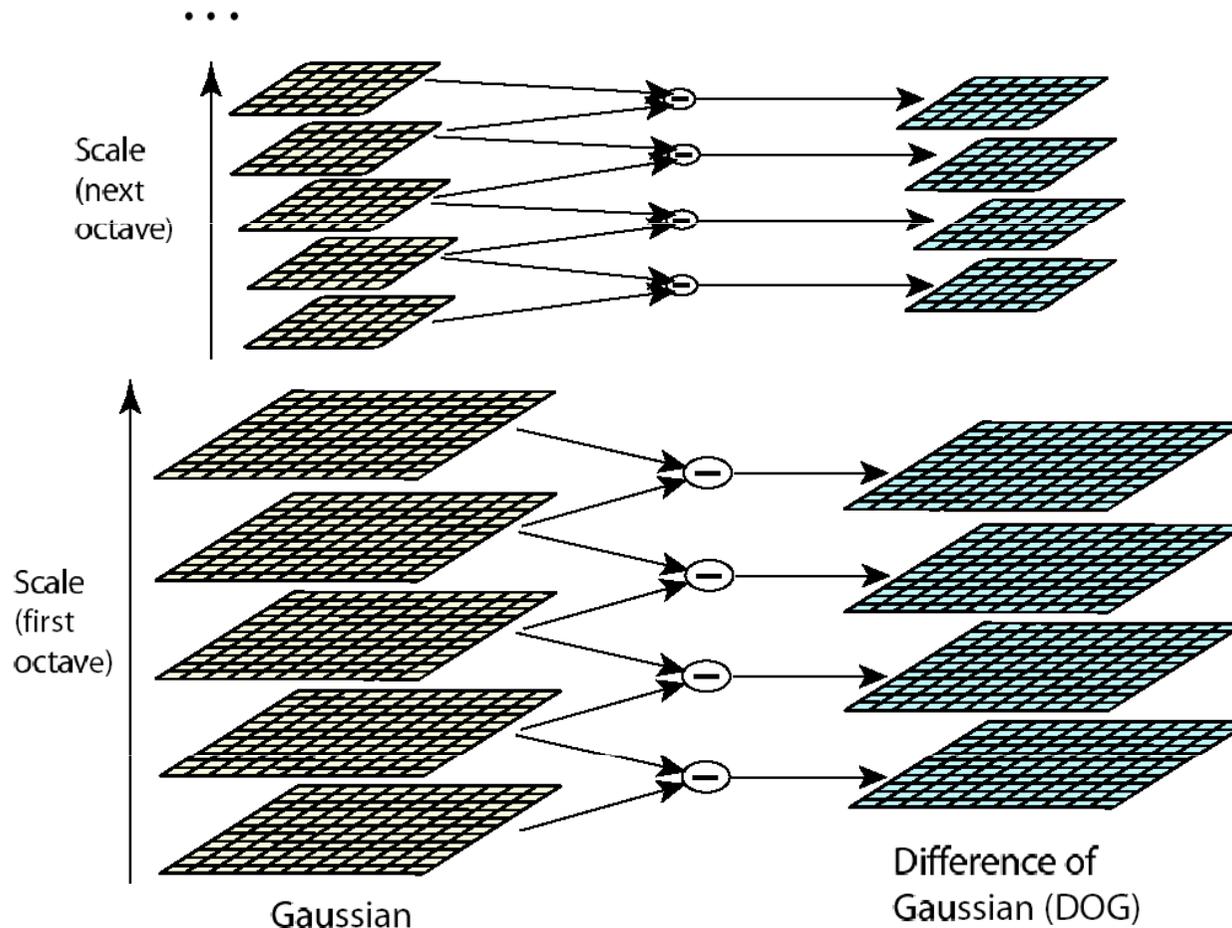
$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$ , where  $\sigma$  is the width of the Gaussian

- **Difference of Gaussian** kernel is a close approximation to scale-normalized Laplacian of Gaussian.

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned} \quad \text{2 scales: } \sigma \text{ and } k\sigma$$

- Can approximate the Laplacian of Gaussian kernel with a difference of separable convolutions.

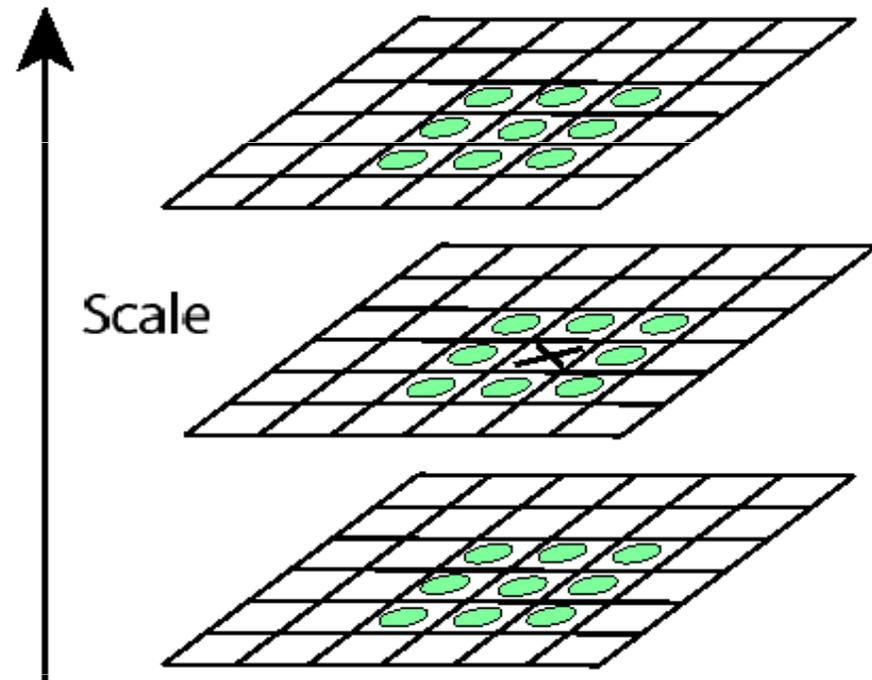
# Lowe's pyramid scheme



For each octave of scale space, the initial image is repeatedly convolved with Gaussian to produce the set of scale space images (left). Adjacent Gaussian images are subtracted to produce difference of Gaussian images (right). After each octave Gaussian image is downsampled by a factor of 2.

# Interest point localization

- Detect maxima and minima of difference of Gaussian in scale space.
- Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below.
- Select only if it is greater or smaller than all the others.



- For each max or min found, output is the location and the scale.

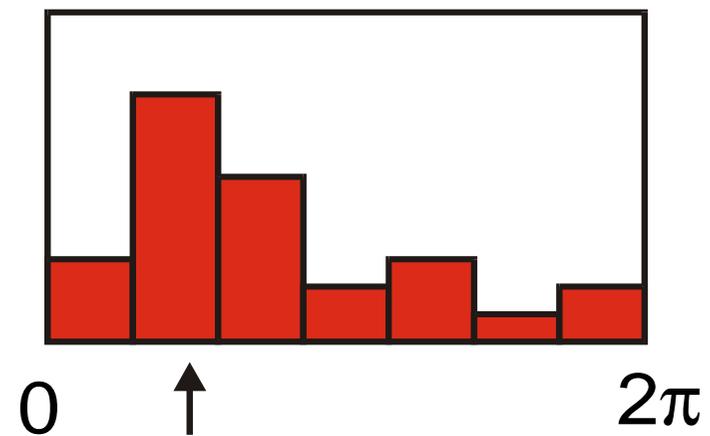
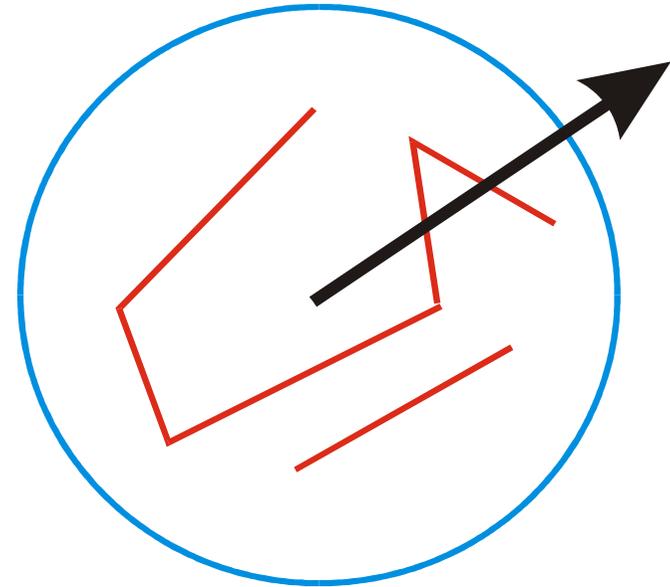
# Interest point localization

---

- Detailed interest point determination
  - Sub-pixel and sub-scale location scale determination.
  - Ratio of principal curvature to reject edges and flats (like detecting corners).
- Once an interest point candidate is found, perform a detailed fit to nearby data to determine
  - location, scale, and ratio of principal curvatures.
- In initial work interest points were found at location and scale of a central sample point.
- In newer work, a 3D quadratic function is fit to improve interpolation accuracy.
- The Hessian matrix is used to eliminate edge responses.

# Orientation assignment

- Create histogram of local gradient directions computed at selected scale.
- Assign canonical orientation at peak of smoothed histogram.
- Each key specifies stable 2D coordinates ( $x$ ,  $y$ , scale, orientation).



# Example of interest point detection

Threshold on value at DOG peak and on ratio of principle curvatures (Harris approach).



- (a) 233x189 image
- (b) 832 DOG extrema
- (c) 729 left after peak value threshold
- (d) 536 left after testing ratio of principle curvatures



# Interest point descriptors

---

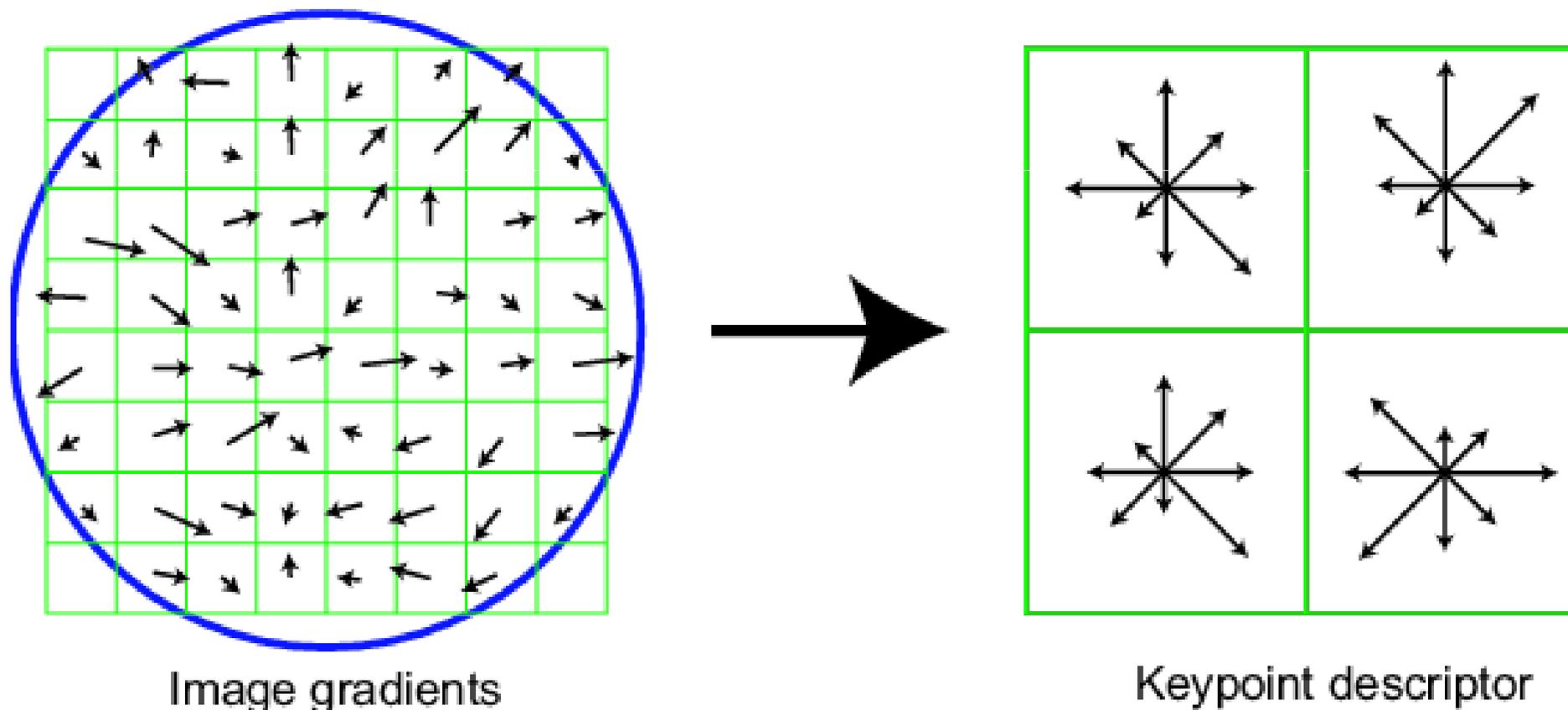
- At this point, each interest point has
  - location,
  - scale,
  - orientation.
- Next step is to compute a descriptor for the local image region about each interest point that is
  - highly distinctive,
  - invariant as possible to variations such as changes in viewpoint and illumination.

# Lowe's interest point descriptor

---

- Use the **normalized** circular region about the interest point.
  - Rotate the window to standard orientation.
  - Scale the window size based on the scale at which the point was found.
- Compute gradient magnitude and orientation at each point in the region.
- **Weight them by a Gaussian** window overlaid on the circle.
- Create an **orientation histogram** over the 4x4 subregions of the window.
- 4x4 descriptors over 16x16 sample array were used in practice. 4x4 times 8 directions gives a vector of **128 values**.

# Lowe's interest point descriptor



In the paper, 4x4 arrays of 8 bin histogram is used, resulting in a total of 128 features for one interest point (shown with 2x2 descriptors over 8x8 array).

# Example applications

---

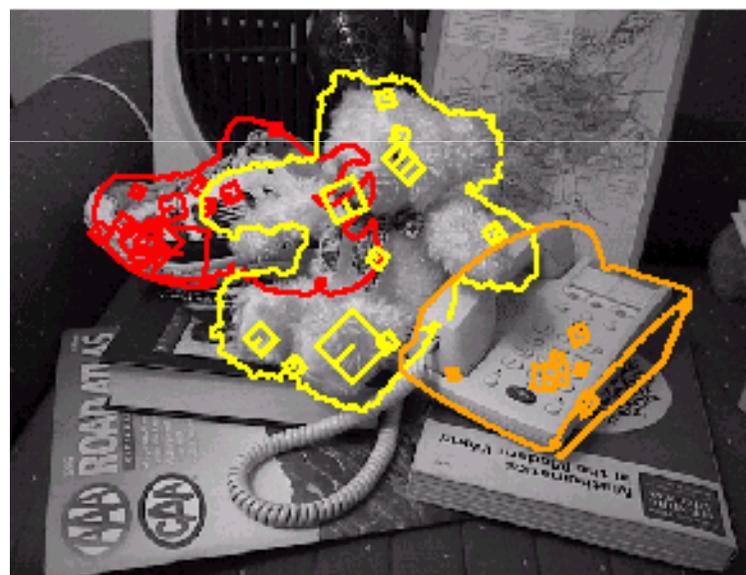
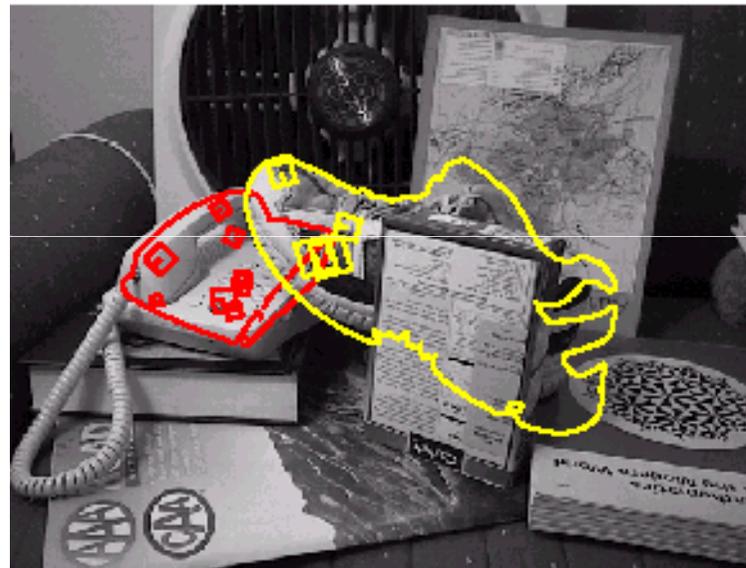
- Object and scene recognition
- Stereo correspondence
- 3D reconstruction
- Image alignment & stitching
- Image indexing and search
- Motion tracking
- Robot navigation

# Examples: 3D recognition

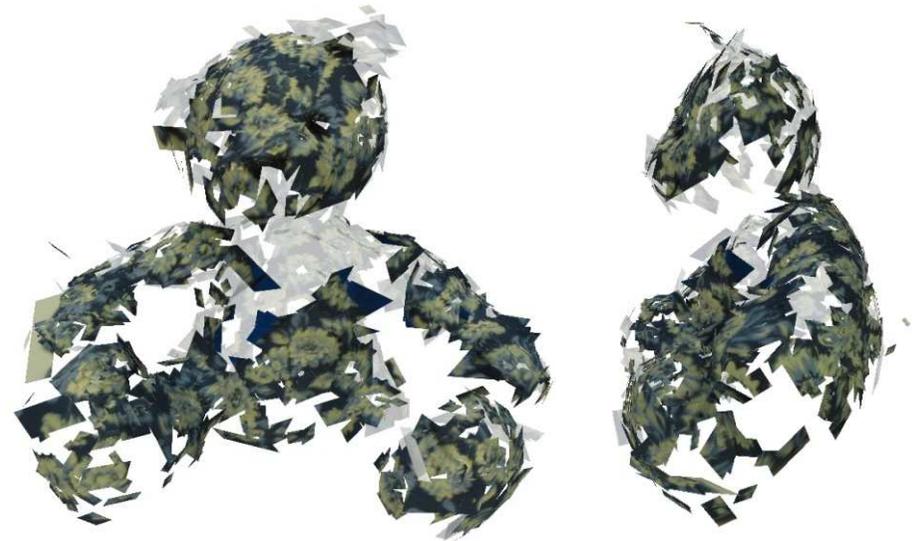
---



# Examples: 3D recognition



# Examples: 3D reconstruction



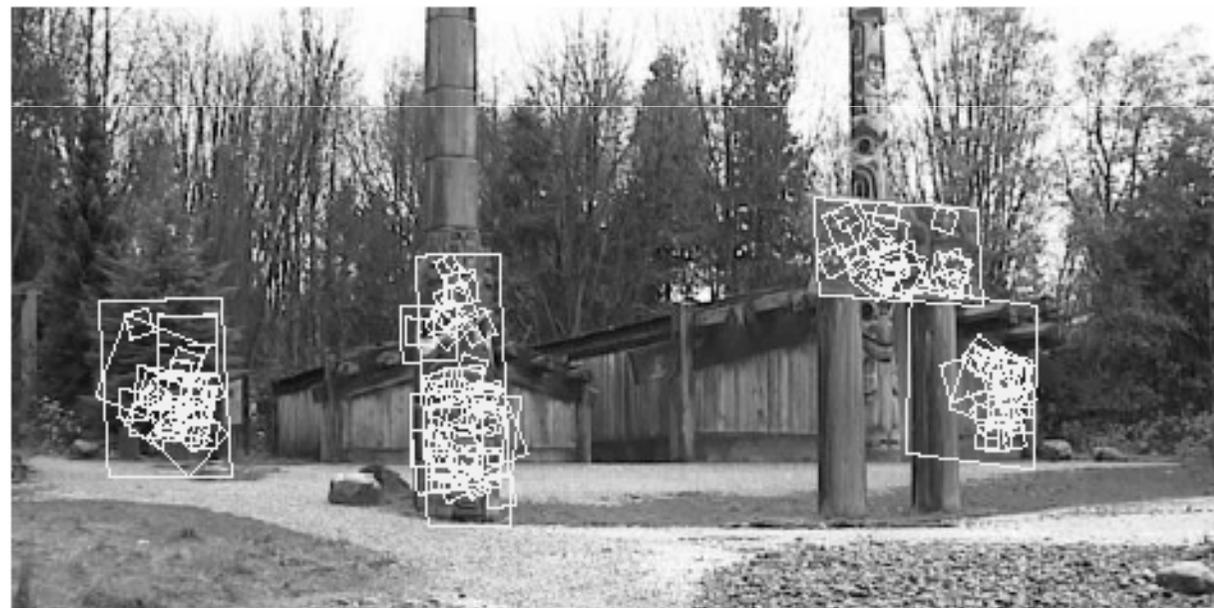
# Examples: 3D recognition



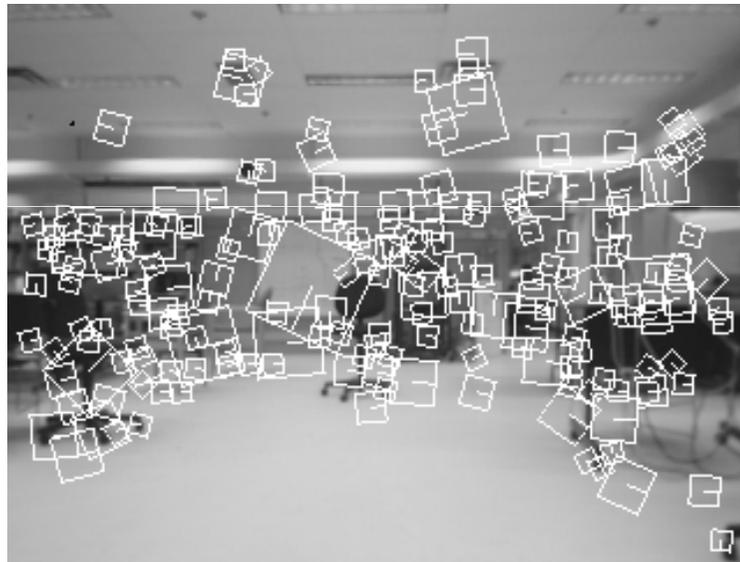
3D object modeling and recognition using affine-invariant patches and multi-view spatial constraints

# Examples: location recognition

---

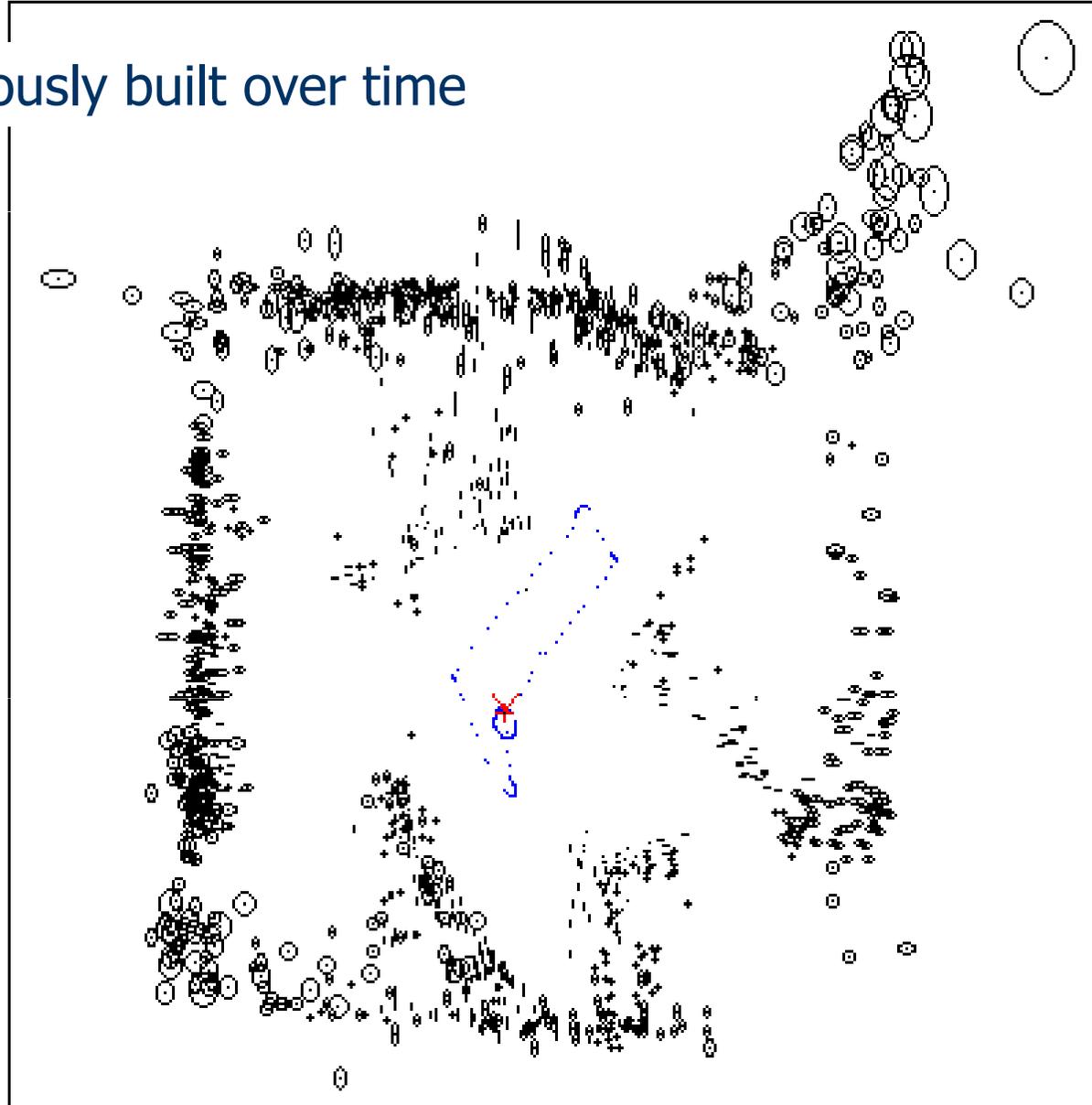


# Examples: robot localization



# Examples: robot localization

Map continuously built over time



# Examples: panoramas

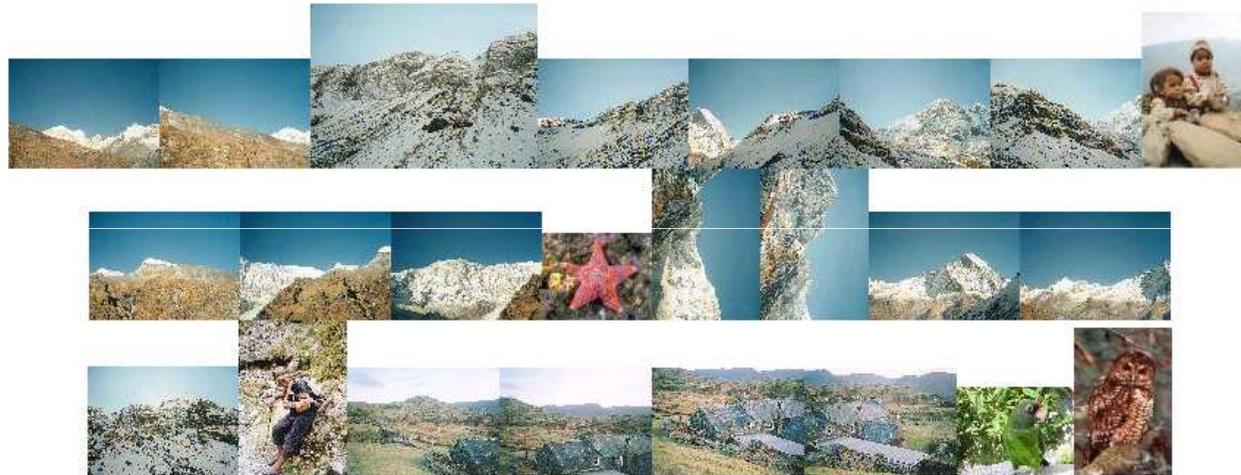
---

- Recognize overlap from an unordered set of images and automatically stitch together.
- SIFT features provide initial feature matching.
- Image blending at multiple scales hides the seams.



Panorama of Lowe's lab automatically assembled from 143 images

# Examples: panoramas



*Input images*

Multiple panoramas from  
an unordered image set



*Output panorama 1*



# Examples: panoramas

## Image registration and blending



(a) 40 of 80 images registered



(b) All 80 images registered



(c) Rendered with multi-band blending

# Sony Aibo

- SIFT usage:
  - Recognize charging station
  - Communicate with visual cards
  - Teach object recognition

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



ERS-7  
Entertainment Robot AIBO

ERS-7 with:  
Wireless LAN  
AIBO MIND software  
Energy Station  
AIBOne  
Pink Ball  
AIBO Cards (15)  
WLAN Manager CD  
Battery & AC Adapter

3rd Generation  
Pre-order Now!

The image shows the AIBO ERS-7 robot, a white and black dog-like robot, surrounded by its accessories. The accessories include a charging station (top left), a clock (top right), a pink ball (bottom center), and a visual card (bottom right). The robot is shown in a standing position, facing right. The accessories are arranged around it, with the charging station and clock at the top, the pink ball in the center, and the visual card at the bottom right. The text 'ERS-7 Entertainment Robot AIBO' is at the top center, and '3rd Generation Pre-order Now!' is at the bottom center. The list of accessories is on the right side.

# Photo tourism: exploring photo collections

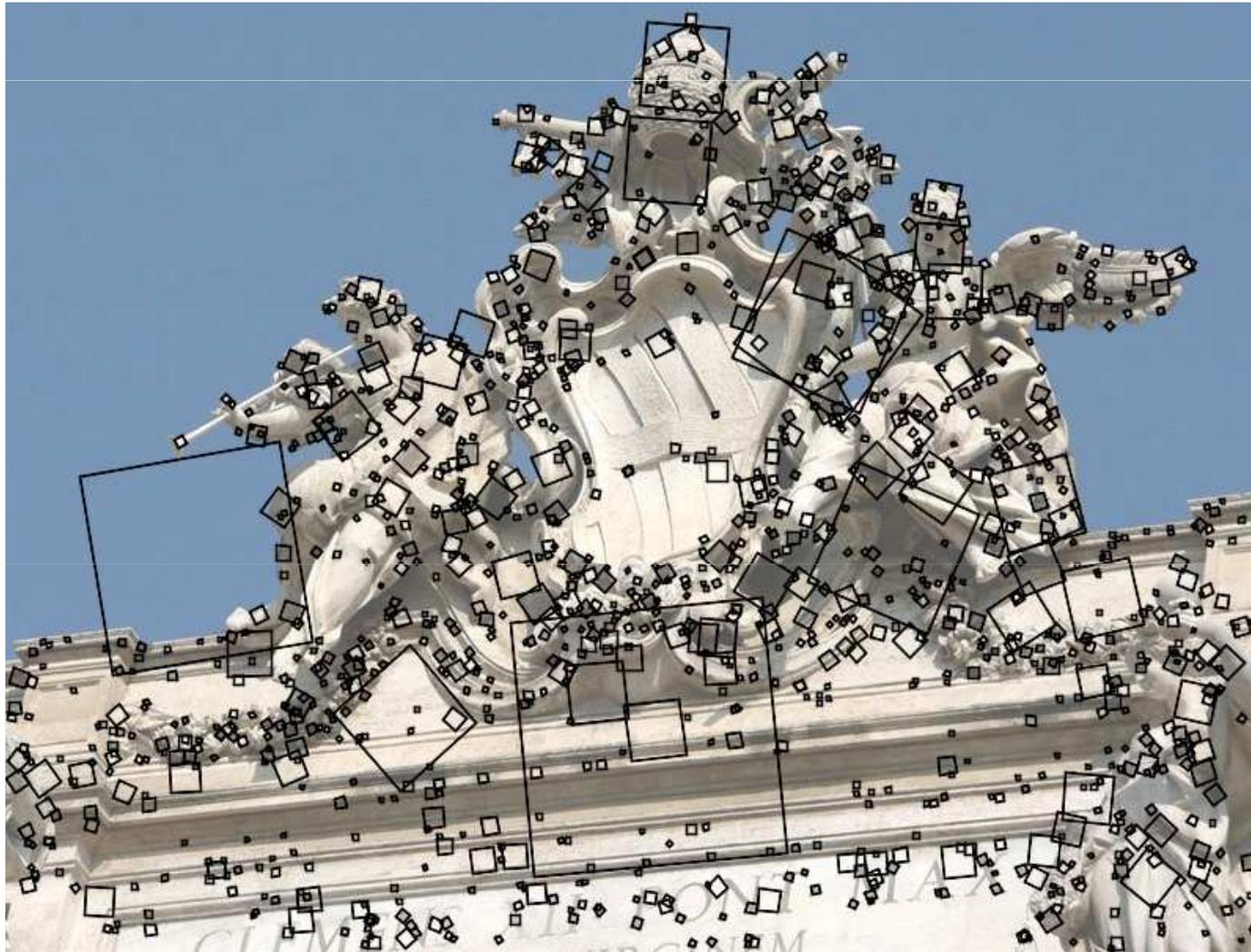
---

- Joint work by University of Washington and Microsoft Research
  - <http://phototour.cs.washington.edu/>
  - <http://research.microsoft.com/IVM/PhotoTours/>
- Photosynth Technology Preview at Microsoft Live Labs
  - <http://labs.live.com/photosynth/>
- Don't forget to check the cool video and demo at <http://phototour.cs.washington.edu/>.

# Photo tourism: exploring photo collections

---

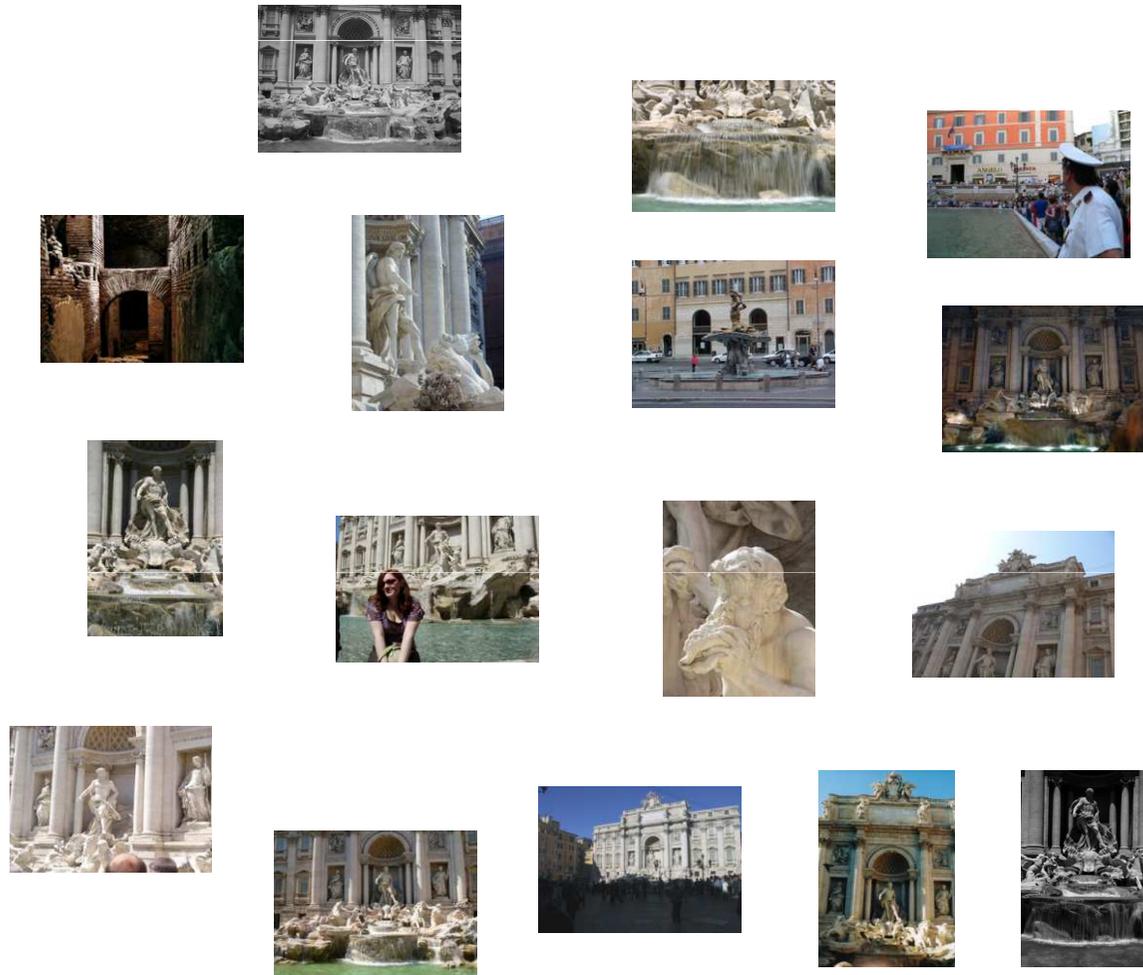
- Detect features using SIFT.



# Photo tourism: exploring photo collections

---

- Detect features using SIFT.



# Photo tourism: exploring photo collections

- Detect features using SIFT.





# Photo tourism: exploring photo collections

---

- Link up pairwise matches to form connected components of matches across several images.

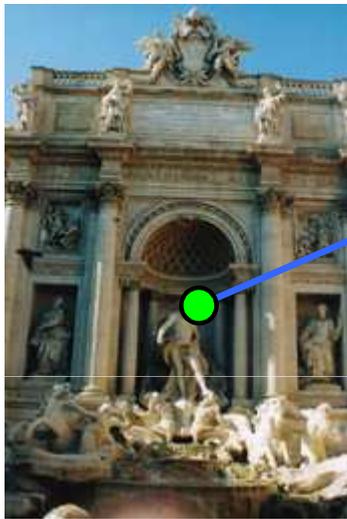


Image 1



Image 2

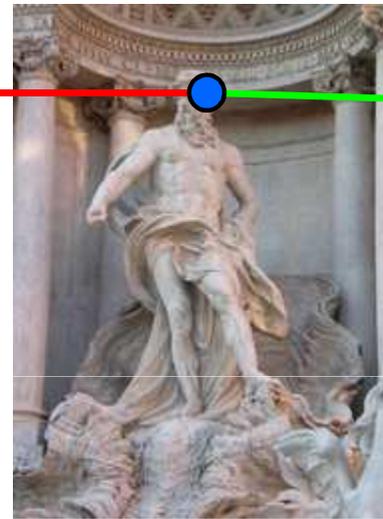


Image 3

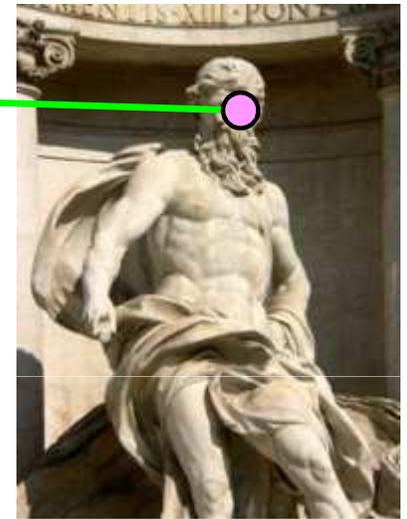
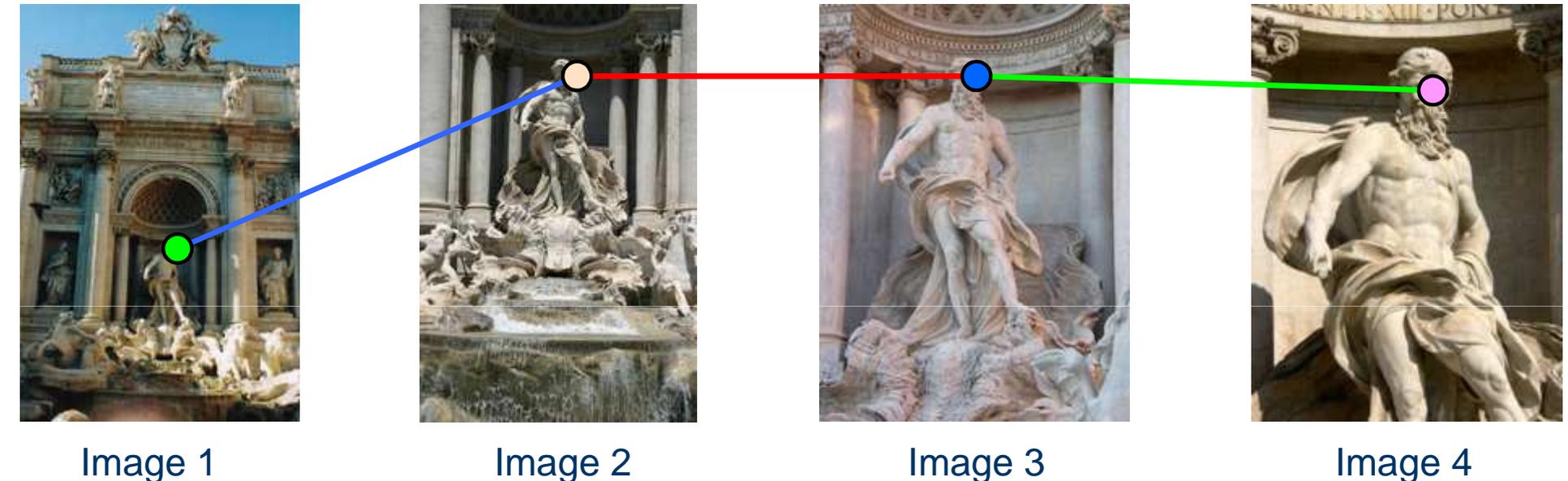


Image 4



# Photo tourism: exploring photo collections

---



# Photo tourism: exploring photo collections

---



Photos are automatically placed inside a sketchy 3D model of the scene; an optional overhead map also shows each photo's location.

# Photo tourism: exploring photo collections



An info pane on the left shows information about the current image and navigation buttons for moving around the collection; the filmstrip view on the bottom shows related images; mousing over these images brings them up as a registered overlay.

# Photo tourism: exploring photo collections



Photographs can also be taken in outdoor natural environments. The photos are correctly placed in 3-D, and more free-form geometric models can be used for inter-image transitions.

# Photo tourism: exploring photo collections



Annotations entered in one image (upper left) are automatically transferred to all other related images.

# Scene summarization for online collections

- <http://grail.cs.washington.edu/projects/canonview>



Scene summary browsing



Enhanced 3D browsing