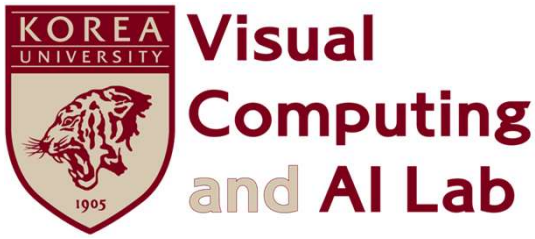
INTRODUCTION TO COMPUTER VISION

Lecture 3 – 2D Features and Matching

Gyeongsik Moon

Visual Computing and AI Lab

Korea University



Slides Credit: Prof. Dr. Davide Scaramuzza

1. Motivation for matching
2. Features for matching
3. Correspondences

1. Motivation for matching

Computer Vision

Goal: Understanding what is happening in images


Vanishing
point

Discrepancy between captured images and real 3D worlds

Early Computer Vision

Goal: Understanding what is happening in images by reconstructing 3D worlds from images

지금은 img to img로 하지만
옛날엔 3D로 재구성하고 이해하려고 했다..

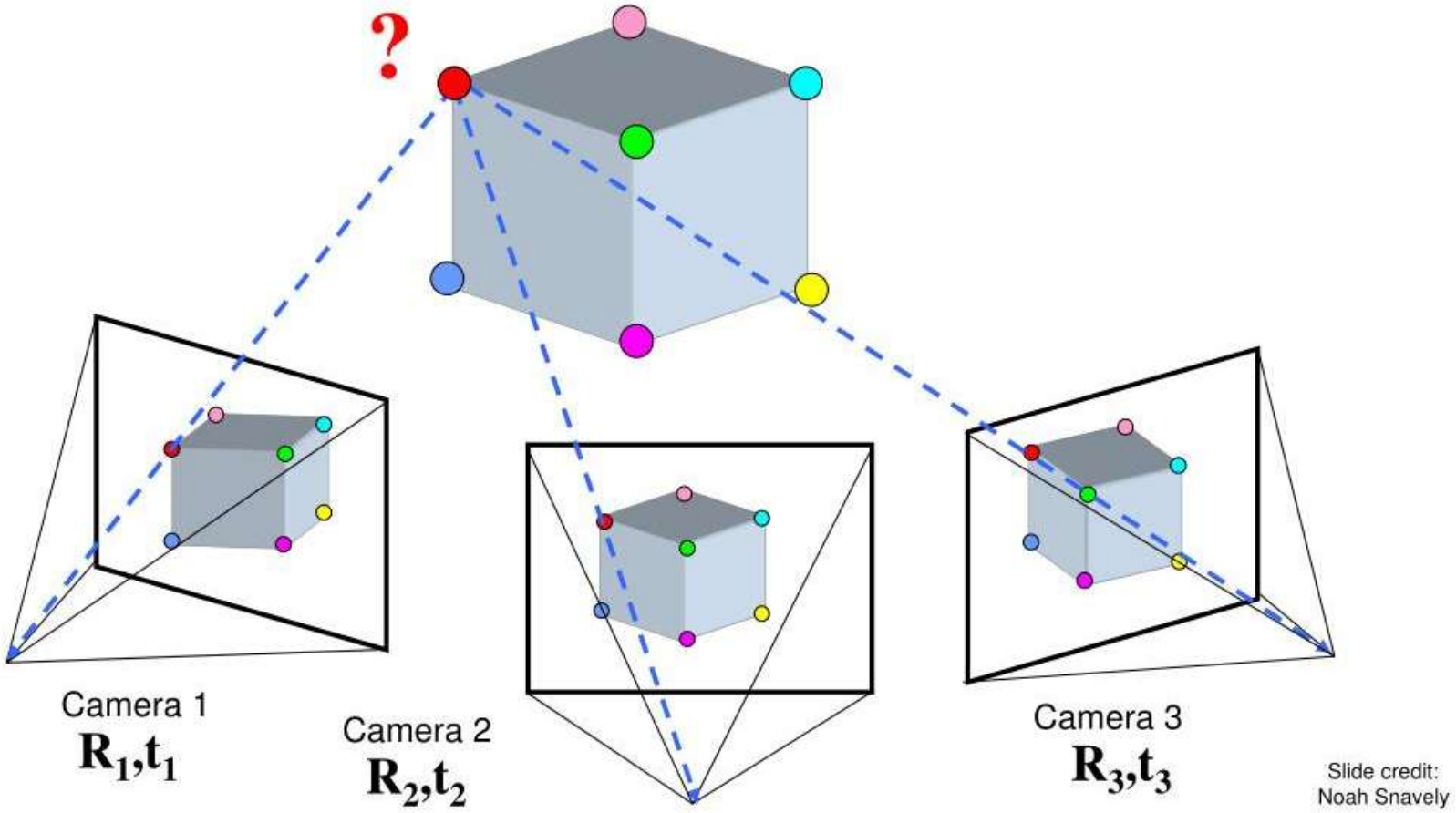
- **Once we recover 3D worlds, we're free from such distortions in images**
  - **Then, we could understand what is happening in images better..?**

Early Computer Vision

Focus: reconstructing 3D worlds from images

Multi-View Geometry (MVG) Theory

Relationship between multi-view 2D observations and 3D space

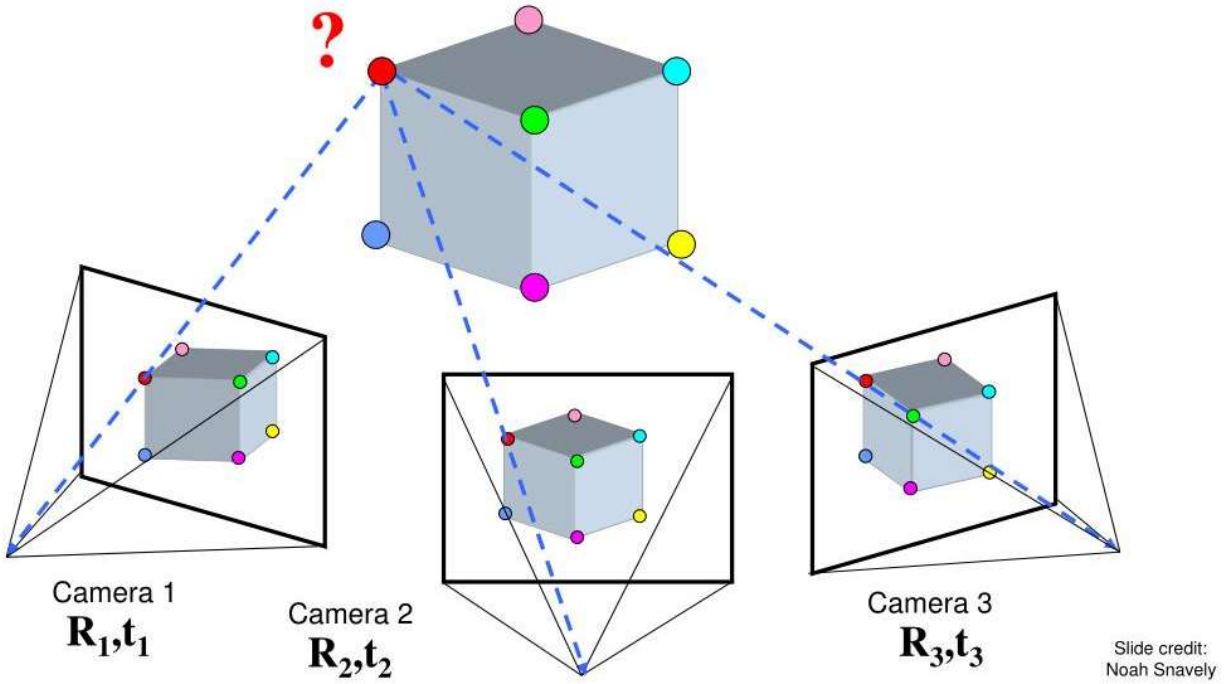


Multi-View Geometry (MVG) Theory

According to MVG (we'll learn this later), if we know

- Camera intrinsic/extrinsic parameters (we learnt what are they in prev. classes)
- *Matched points across multiple viewpoints (same-colored dots in images)*

, then, we can lift the multi-view observations to the 3D space



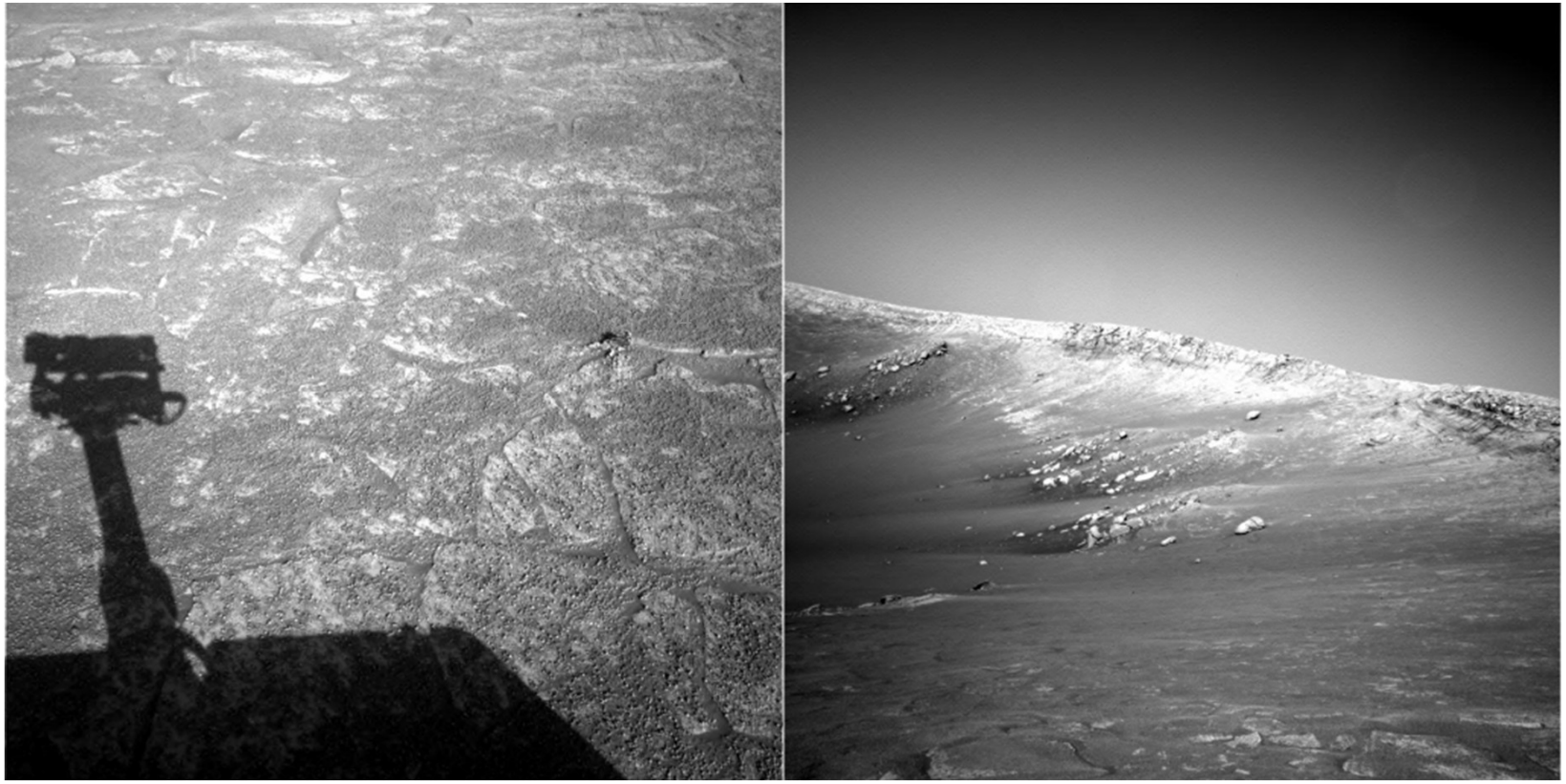
Today's Focus

How can we get correspondences from multi-view images?

This is called matching.

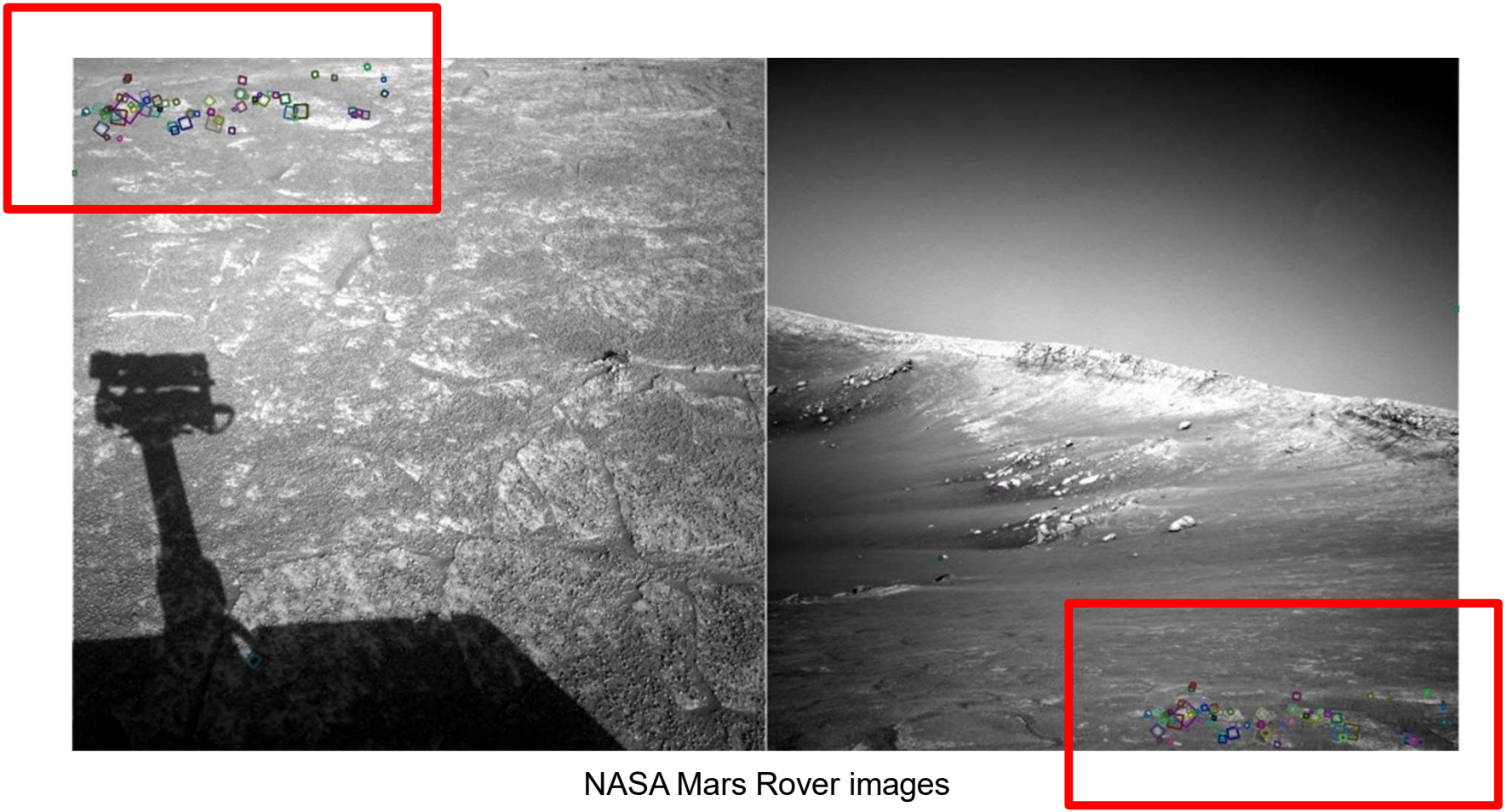
2. Features for matching

Image matching: why is it challenging?



NASA Mars Rover images

Image matching: why is it challenging?



NASA Mars Rover images

view point and camera intrinsic is so different

Example: panorama stitching

How does it work?

AutoStitch: http://matthewalunbrown.com/autostitch/autostitch.html
M. Brown and D. G. Lowe. Recognising Panoramas, International Conference on Computer Vision (ICCV), 2003. PDF.

Local features and alignment

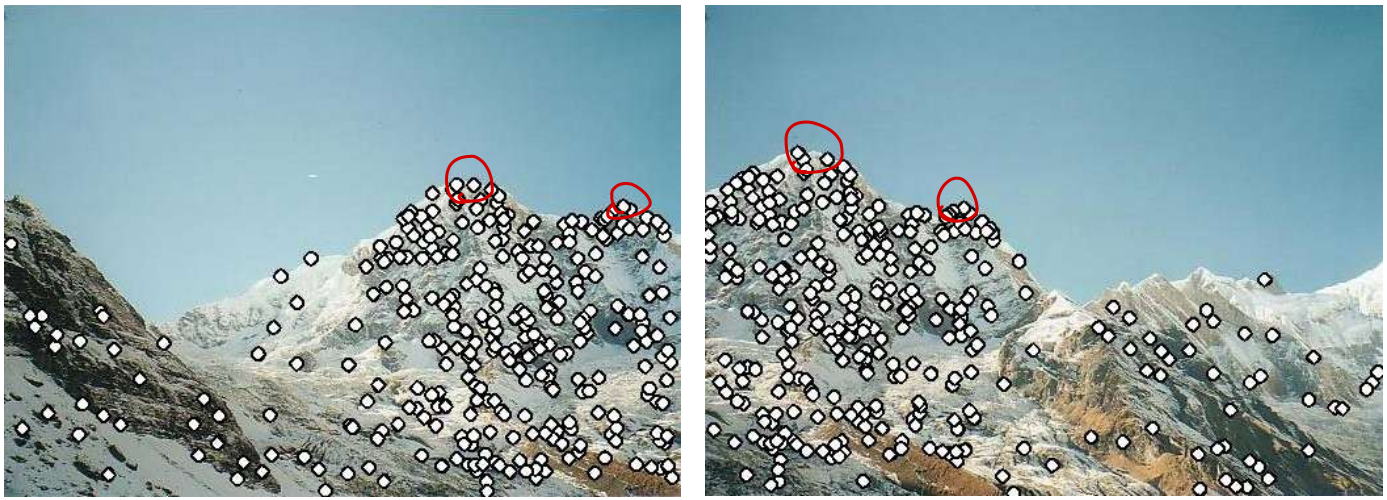
- We need to align two images
- How would you do it?

Local features and alignment

Idea:

= descriptor : points with distinctive property (peak, edges, ...)

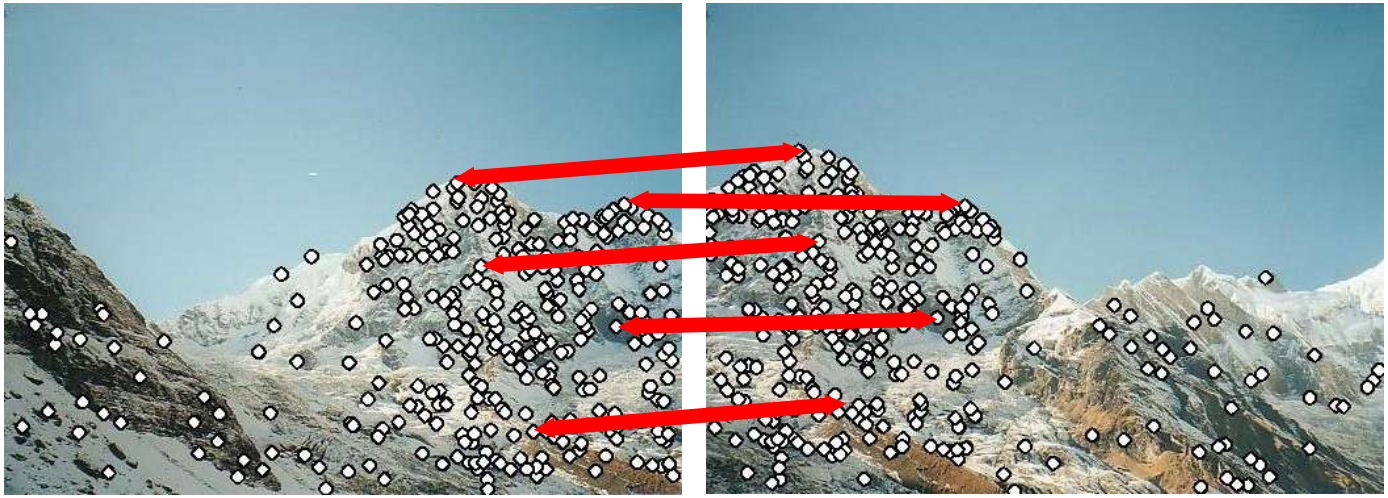
- Detect point features in both images



Local features and alignment

Idea:

- Detect point features in both images
- Find corresponding pairs



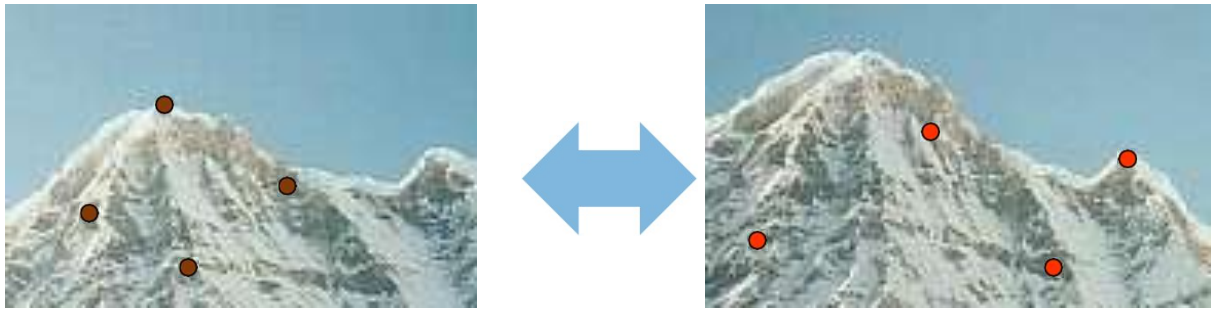
Local features and alignment

Idea:

- Detect point features in both images
- Find corresponding pairs
- Use these pairs to align the images: what image transformation would you use?

Matching with Features

Problem 1: How to **detect** the **same** points **independently** in both images?



no chance to match!

We need a **repeatable** feature **detector.** Repeatable means that the detector should be able to re-detect the same feature in different images of the same scene.

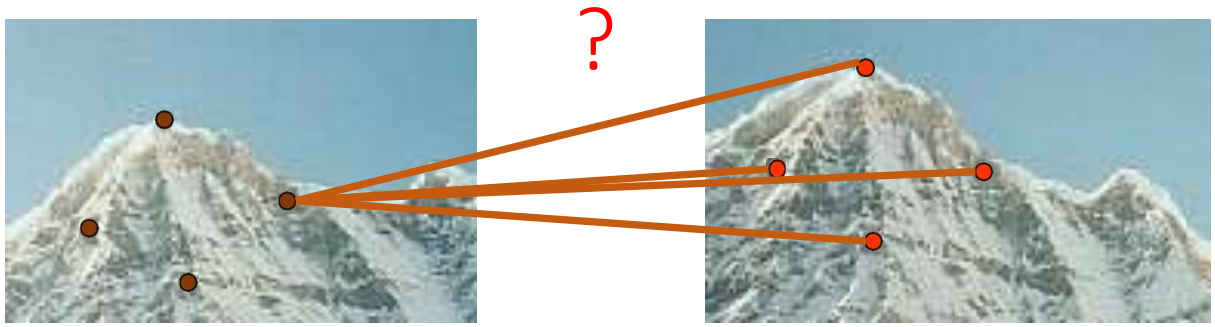
This property is called **Repeatability** of a feature **detector**.

같은 points를 detecting.

pair matching 아님

Matching with Features

Problem 2: For each point, how to **match** its **corresponding point** in the other image

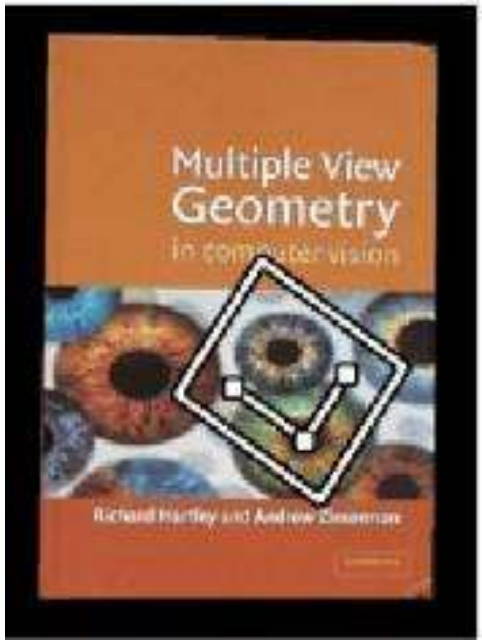


We need a **distinctive** feature descriptor. A descriptor is a "description" of the pixel information around a feature (e.g., patch intensity values, gradient values, etc.). Distinctive means that the descriptor uniquely identifies a feature from other features without ambiguity. This property is called **Distinctiveness** of a feature **descriptor**.

The descriptor must also be **robust to geometric and photometric** changes.

Geometric changes

- **Rotation**
- **Scale** (i.e., zoom)
- **View point** (i.e, perspective changes)



Photometric Changes (i.e., Illumination changes)

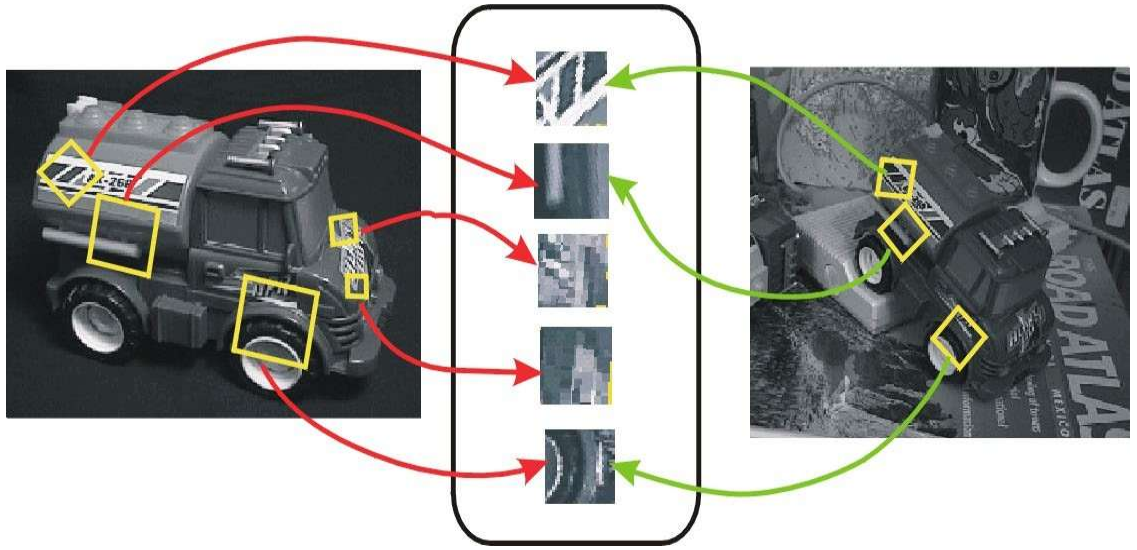
- **Small illumination changes** are modelled with an **affine transformation** (so called *affine illumination changes)*:

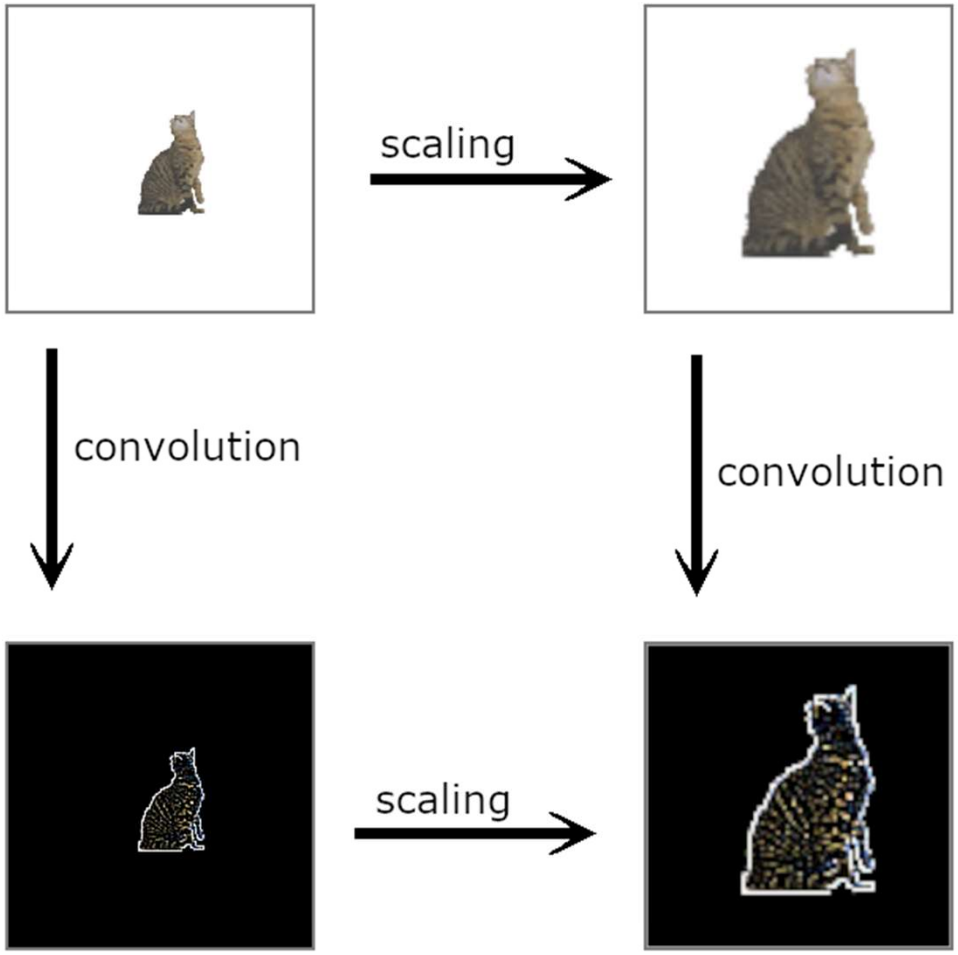
$$I'(x, y) = \alpha I(x, y) + \beta$$

Local Invariant Features

The key to feature detection and matching is to find **repeatable features** and **distinctive descriptors** that are **invariant** to geometric and photometric transformations. Basic steps:

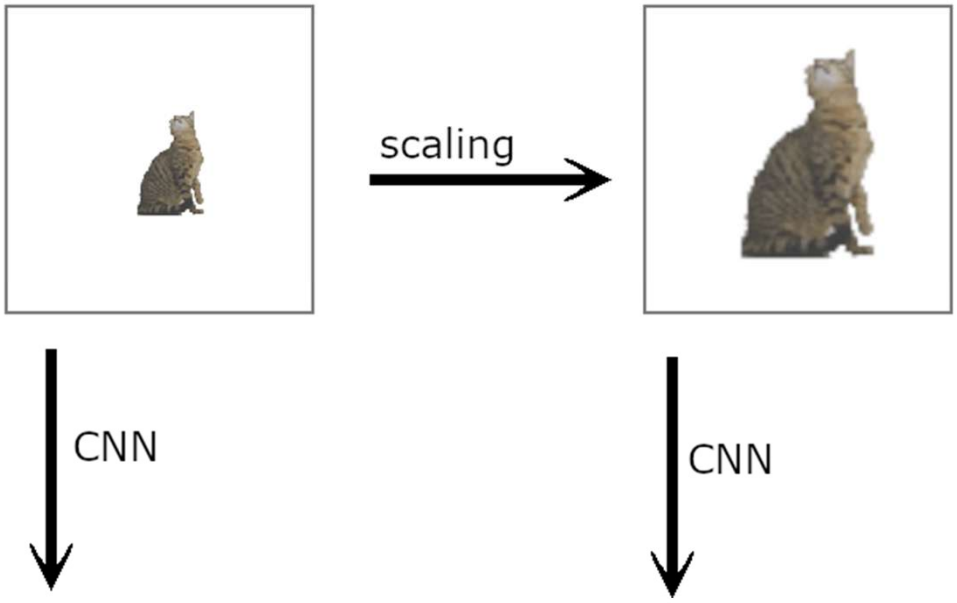
1. Detect repeatable and distinctive interest points
2. Extract invariant descriptors





Equivariance

transformation 에 맞게 output이 바뀜.



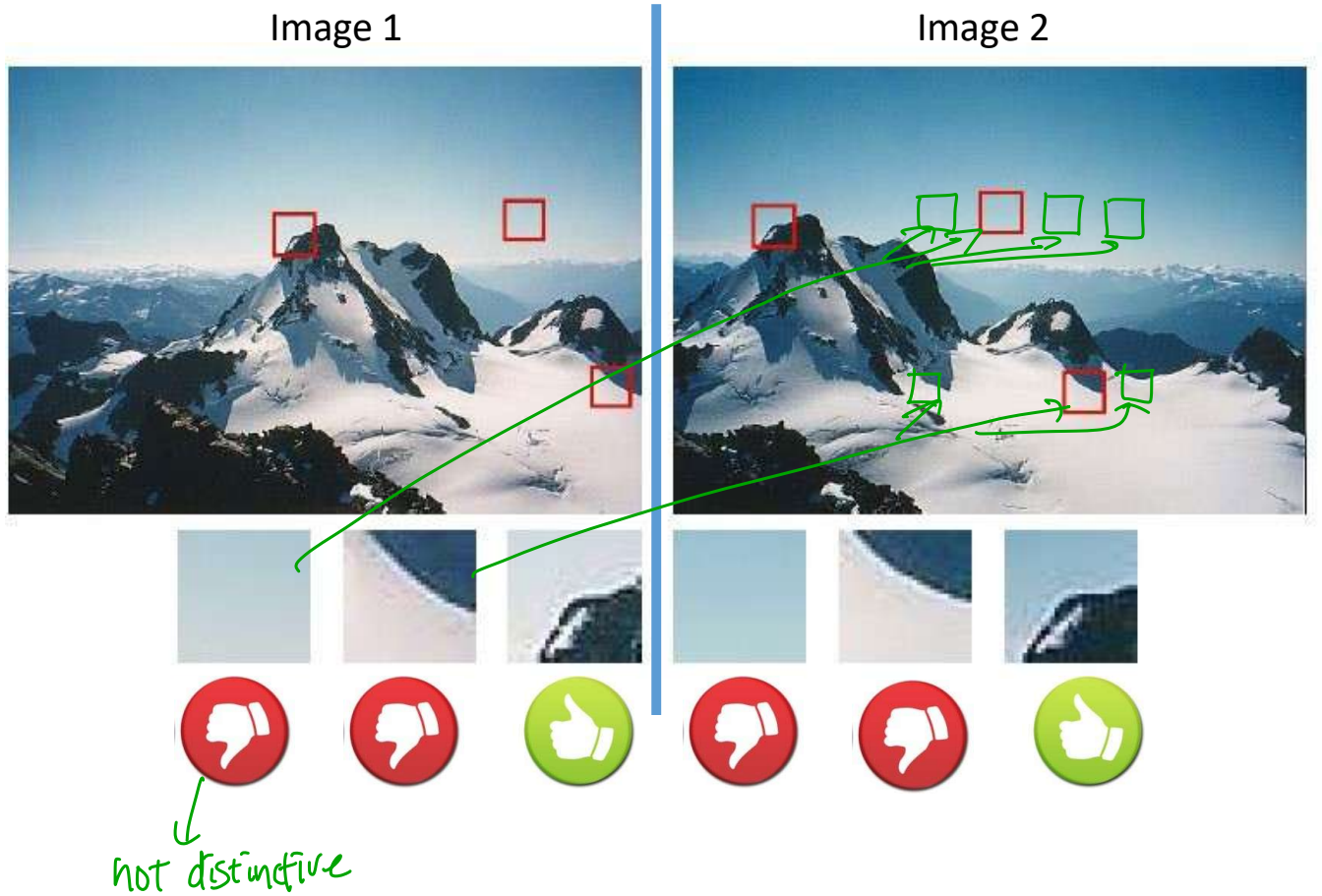
"cat" = "cat"

Invariance

transformation 해도 output 안바뀜.

What is a Repeatable & Distinctive feature?

Consider the images below with some patches. Notice how some patches can be localized or matched with higher accuracy than others

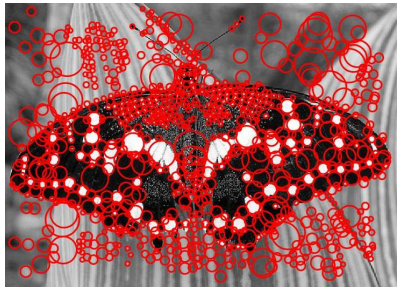


in early CV, we used local patches,

Point Features: Corners vs Blob detectors

- A **corner** is defined as the intersection of two or more edges
  - Corners have **high localization** accuracy → corners are good for visual odometry
  - Corners are **less distinctive than blobs**
  - E.g., *Harris, Shi-Tomasi, SUSAN, FAST*

- A **blob** is any other image pattern **that is not a corner** and differs significantly from its neighbors (e.g., a connected region of pixels with similar color, a circle, etc.)
  - **Blobs have less localization accuracy than corners**
  - Blobs are **more distinctive than corners** → blobs **are better for place recognition**
  - E.g., *MSER, LOG, DOG (SIFT), SURF, CenSurE, etc.*



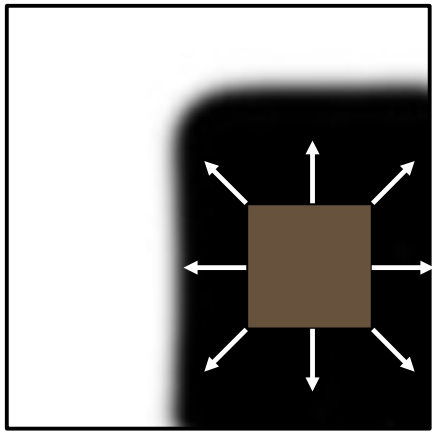
Corner Detection

구체적으로 알 필요 없고 insight를 가져가라. ↓ 예) 이따구는. 왜했는지 어떤 식으로 했는지?

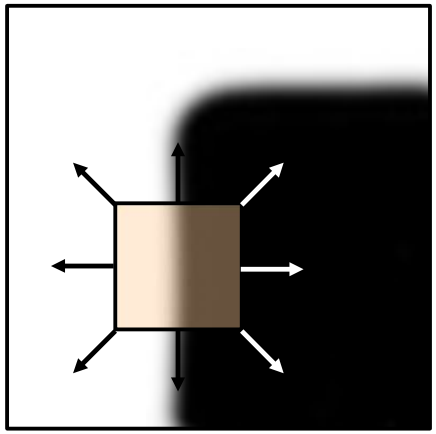
- Key observation: in the region around a corner, image gradient has **two or more** dominant directions
- Corners are **repeatable** and **distinctive**

The Moravec Corner detector (1980)

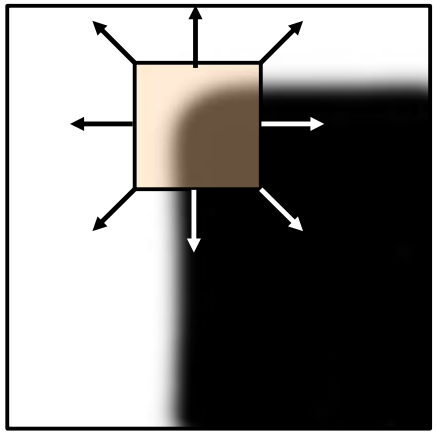
- How do we identify corners? Look at a region of pixels through a small window
- Shifting a window in **any direction** should give raise to **large intensity changes** (e.g., Sum of Squared Differences (SSD))



"flat" region:
no intensity change
(i.e., $\mathrm{SSD} \approx 0$ in all directions)



"edge":
no change along the edge direction
(i.e., $\mathrm{SSD} \approx 0$ along edge but $\gg 0$ in other directions)



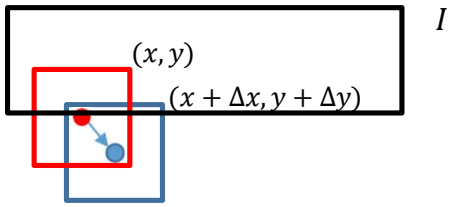
"corner":
significant change in all directions
(i.e., $\mathrm{SSD} \gg 0$ in all directions)

H. Moravec, *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*, PhD thesis, Chapter 5, Stanford University, Computer Science Department, 1980.

The Moravec Corner detector (1980)

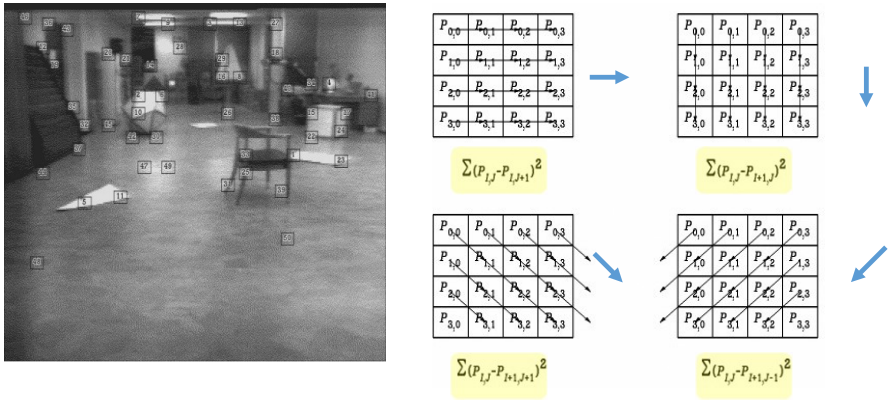
Consider the reference patch centered at (x, y) and the shifted window centered at $(x + \Delta x, y + \Delta y)$. The patch has size Ω. The Sum of Squared Differences (SSD) between them is:

$$SSD(\Delta x, \Delta y) = \sum_{x,y \in \Omega} (I(x, y) - I(x + \Delta x, y + \Delta y))^2$$



"Sums of squares of differences of pixels adjacent in each of ***four directions (horizontal, vertical and two diagonals)*** *over each window are calculated, and the* ***window's interest measure is the minimum of these four sums****. Features are chosen where the* ***interest measure has local maxima****."* [Moravec'80, PhD thesis, Chapter 5, link]

The disadvantage of the Moravec corner detector is that we need to compute four SSDs, one for each shifted version of the patch (1 pixel right, down, down right, and down left).
Can we make it more efficient?
Can we do it without shifting the patch at all?

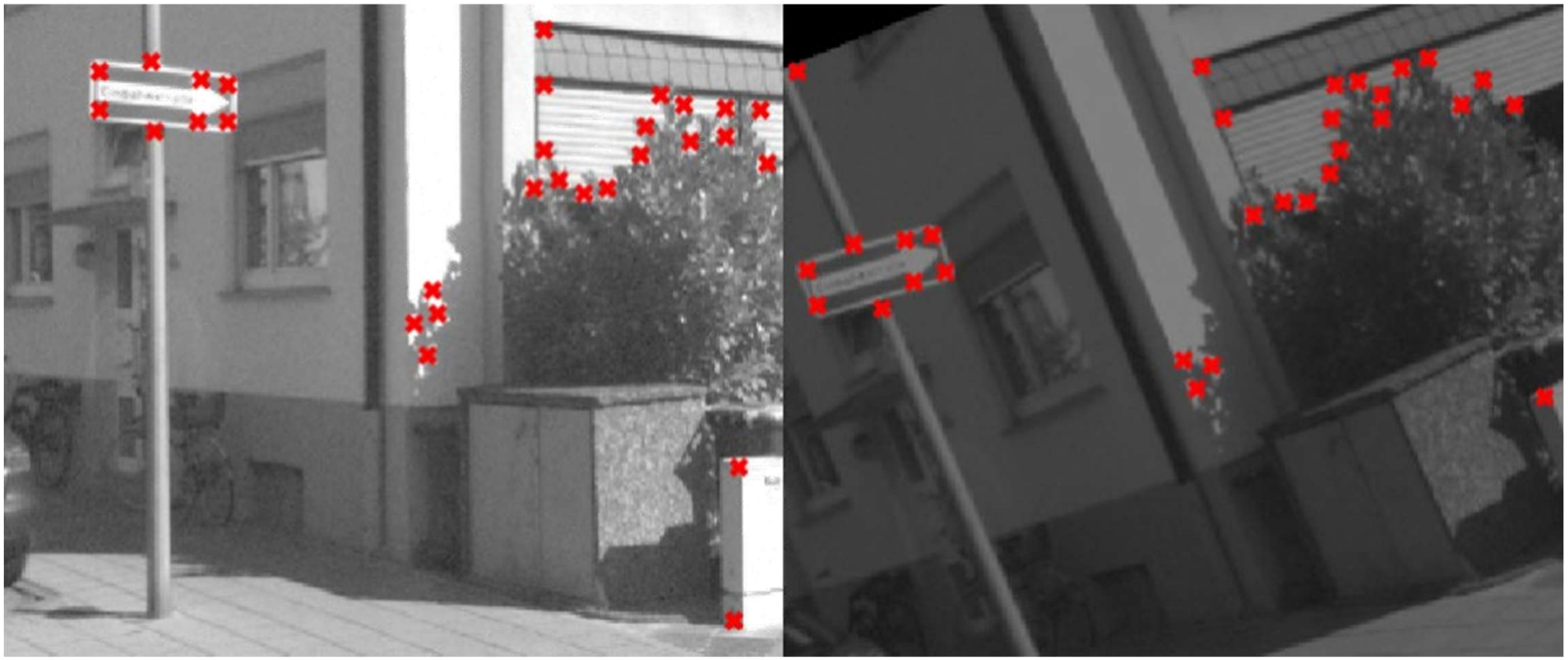


The Harris Corner detector (1988)

It implements the Moravec corner detector without having to physically shift the window but rather by just looking at the patch itself, by using differential calculus.

C.Harris and M.Stephens. “A Combined Corner and Edge Detector”. *Proceedings of the 4th Alvey Vision Conference*. 1988. PDF.

Harris Corner detector



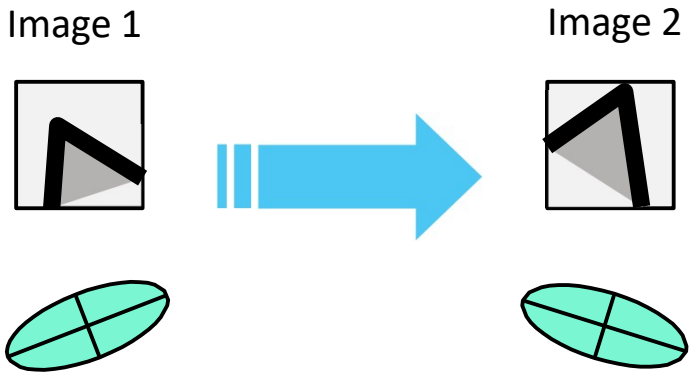
Harris Detector: Some Properties

Repeatability:

- How does the Harris detector behave with **geometric and photometric changes**, i.e. can it re-detect the same corners when the image exhibits changes in
  - **Rotation**,
  - **Scale** (zoom),
  - **View-point**,
  - **Illumination** ?

Harris Detector: Some Properties

- The Harris detector is **rotation invariant**



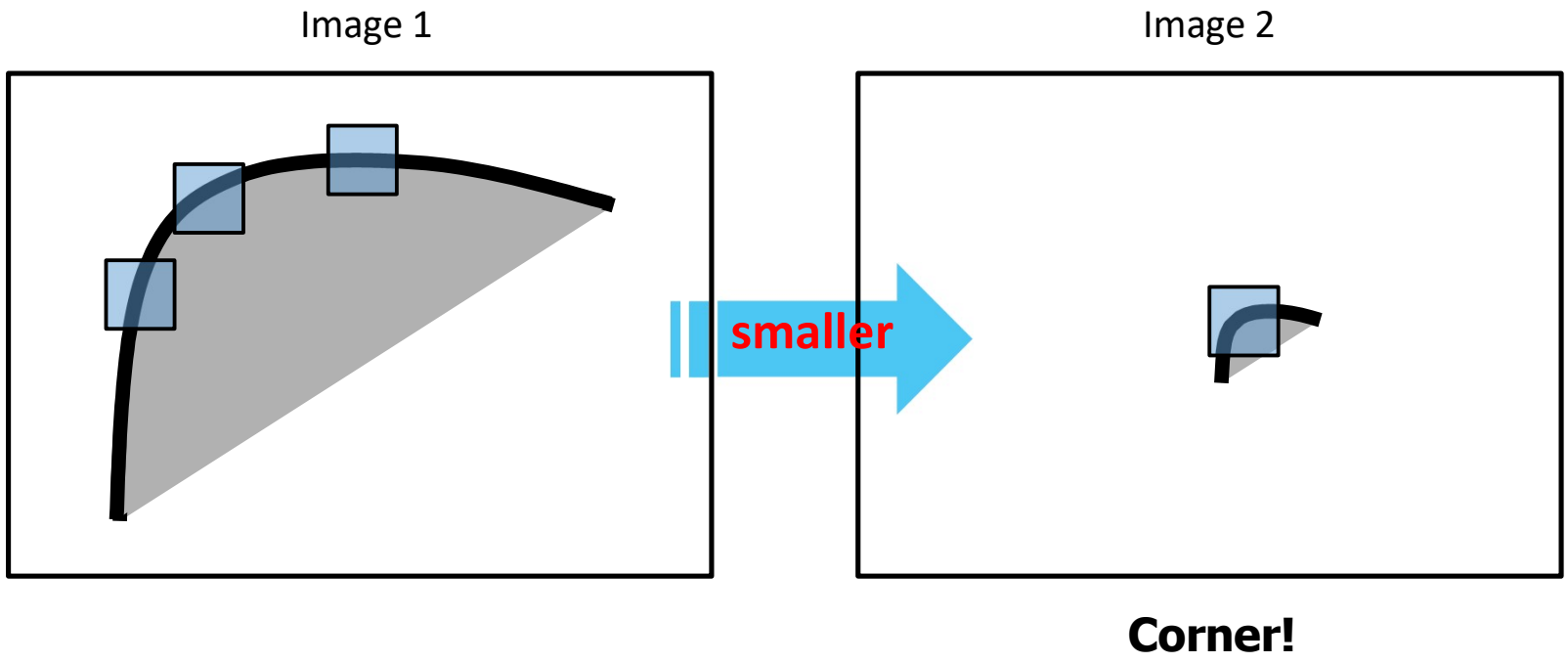
Ellipse rotates but its shape (i.e., eigenvalues of M) remains the same

Corner response R is **invariant to image rotation**

Harris Detector: Some Properties

Scale invariant 하지 않다는게 단점인 이유입니다.

- The Harris detector is **not scale invariant**

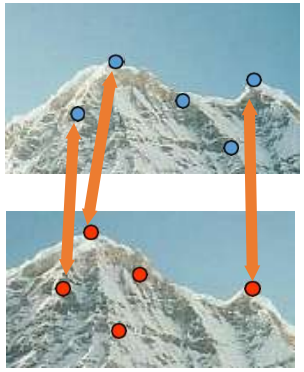


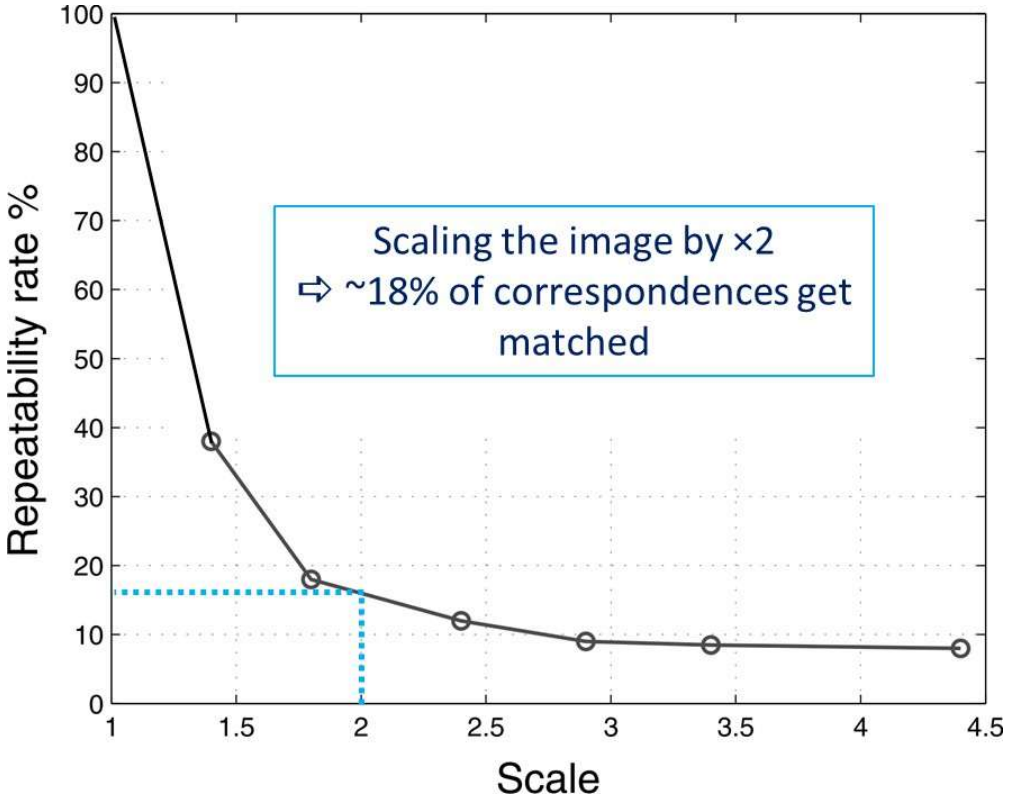
Harris Detector: Some Properties

- **Repeatability** of the Harris detector for different scale changes

Repeatability=

$$\frac{\#\ \text{correspondences detected}}{\#\ \text{correspondences present}}$$



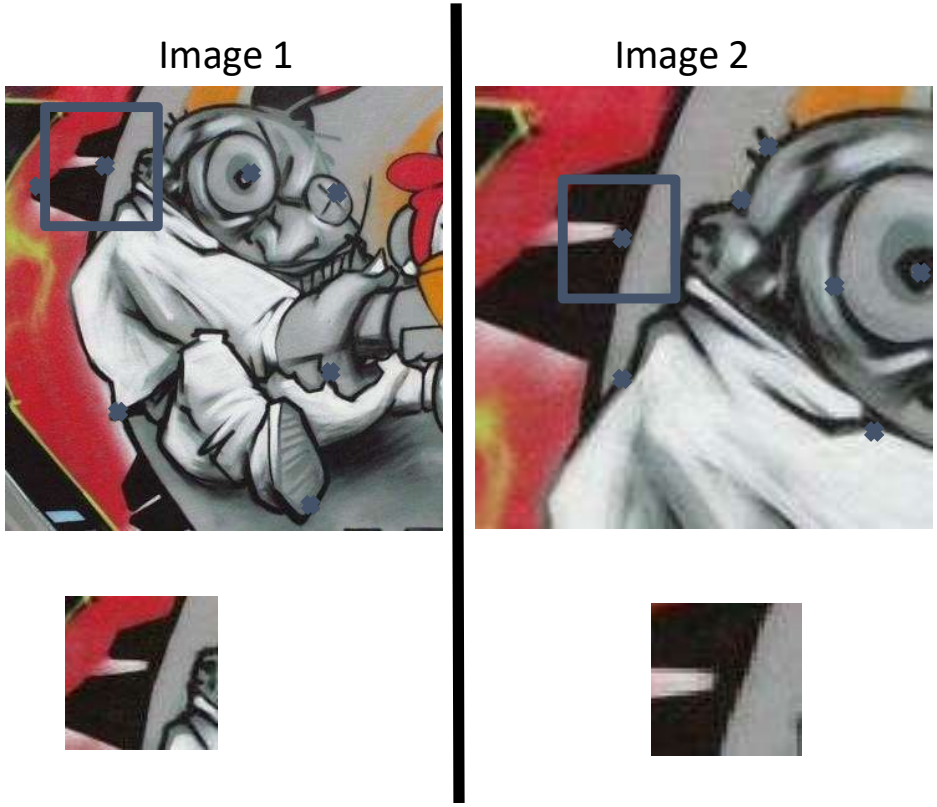


Harris Detector: Some Properties

- Is it invariant to:
  - Affine illumination changes?
    - yes
  - Monotonic illumination changes?
    - yes
  - Viewpoint invariance?
    - Does the same corner look like a corner from a different viewpoint?
      - It depends on the viewpoint change

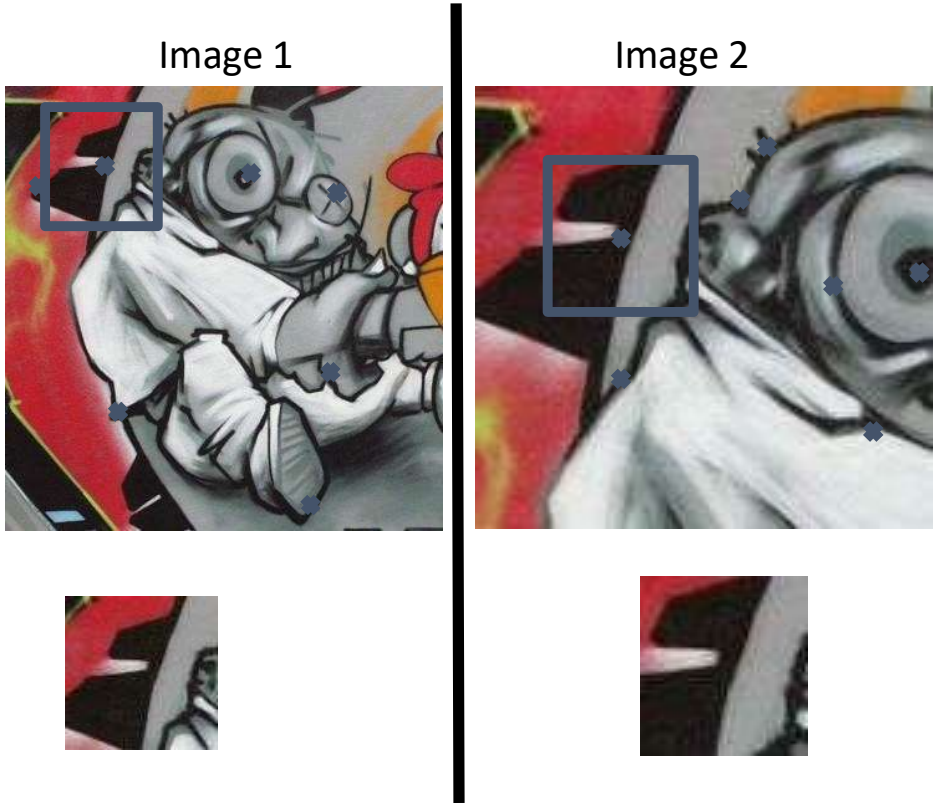
Scale changes

How can we match image patches corresponding to the same feature but belonging to images taken at different scales?



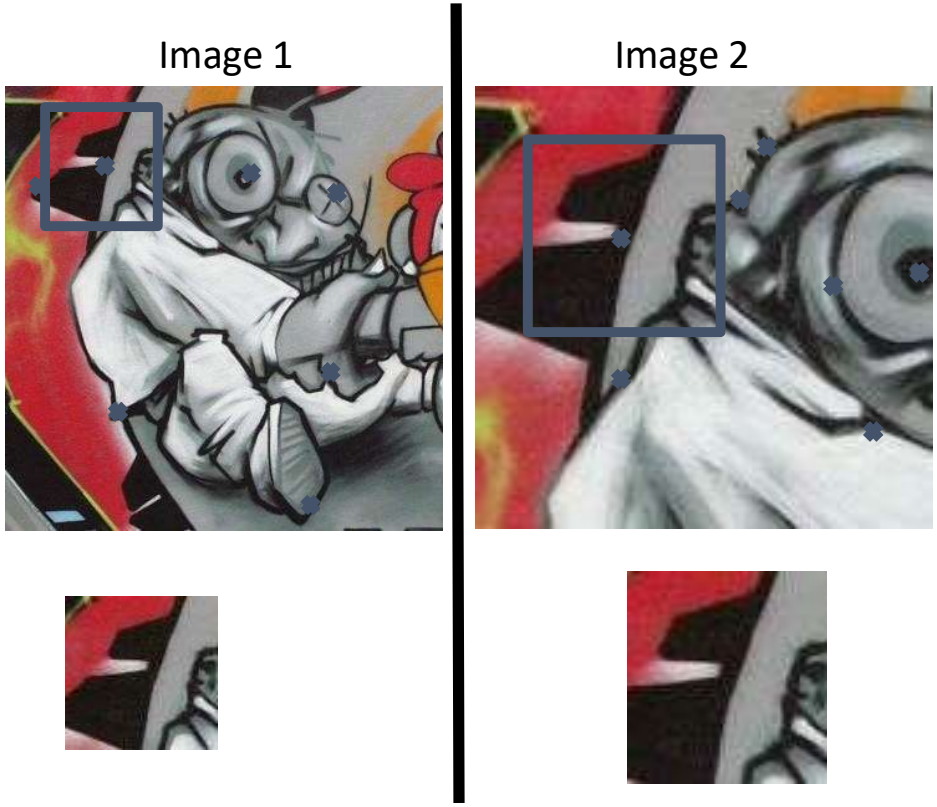
Scale changes

How can we match image patches corresponding to the same feature but belonging to images taken at different scales? Possible solution: rescale the patch



Scale changes

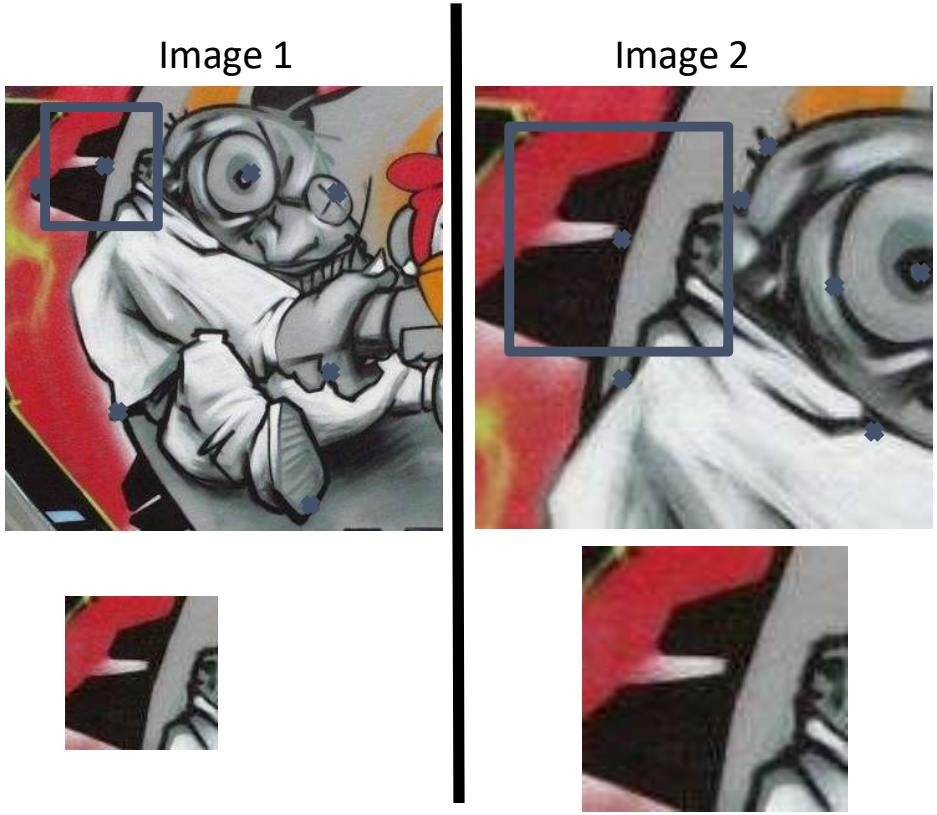
How can we match image patches corresponding to the same feature but belonging to images taken at different scales? Possible solution: rescale the patch



Scale changes

How can we match image patches corresponding to the same feature but belonging to images taken at different scales? Possible solution: rescale the patch

how to find good scales?



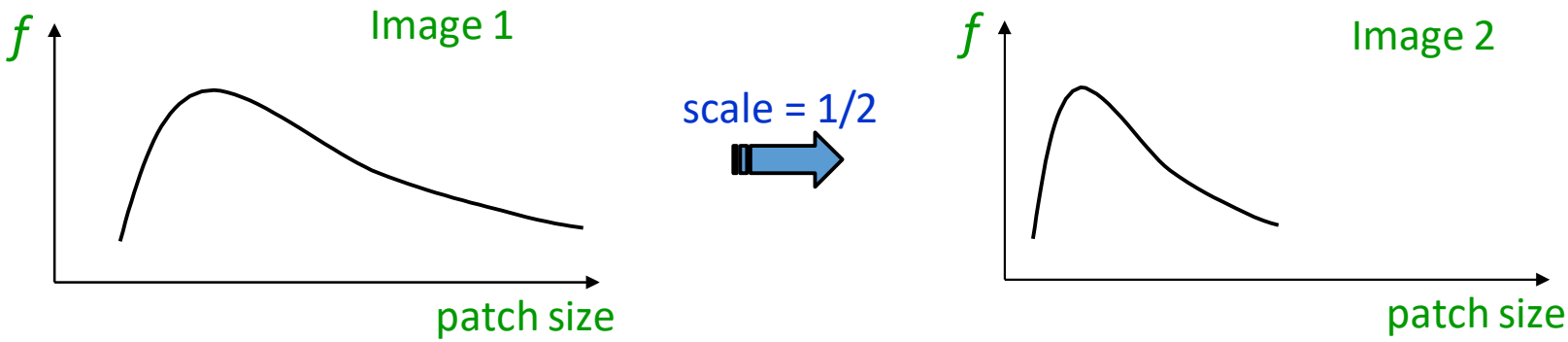
Scale changes

- Scale search is time consuming (needs to be done individually for all patches in one image)
- **Complexity** is N^2S assuming N features per image and S rescalings per feature
- **Solution**: **automatic scale selection:** automatically assign each feature its own "scale" (i.e., size)

by brute force.

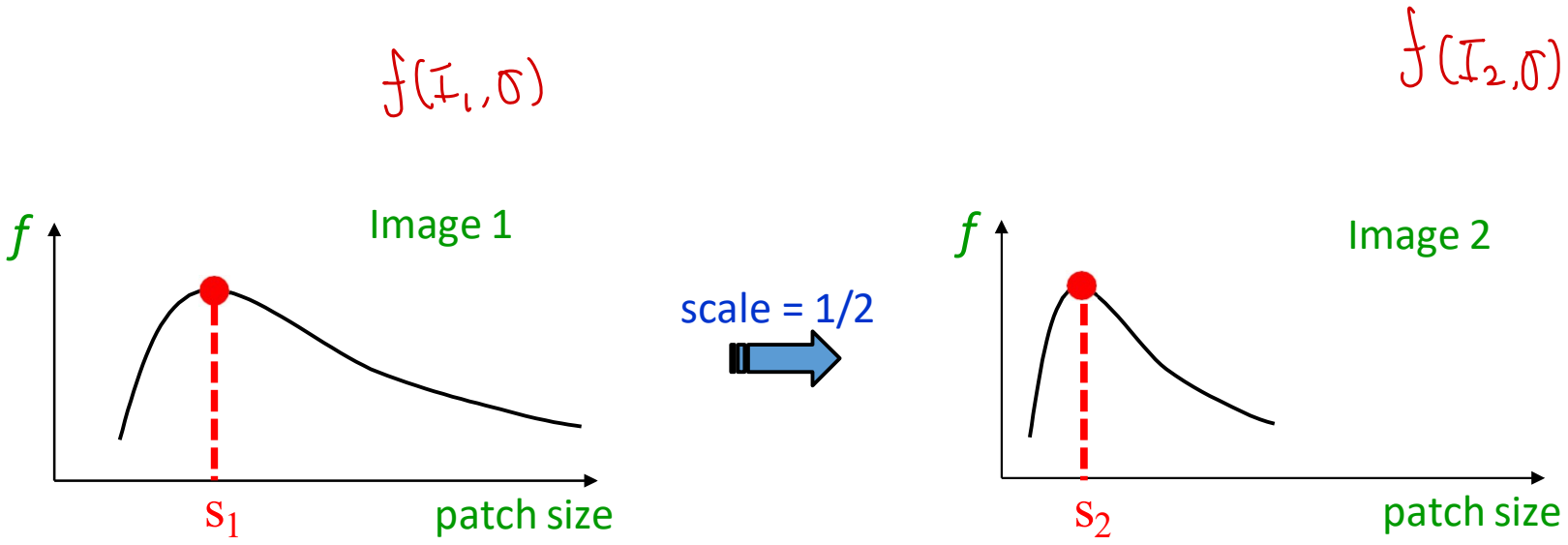
Automatic Scale Selection

- Idea: Design a function on the image patch, which is **scale invariant** (i.e., it has the same value for corresponding patches, even if they are at different scales)
- Function f: the strength of a feature at a given scale



Automatic Scale Selection

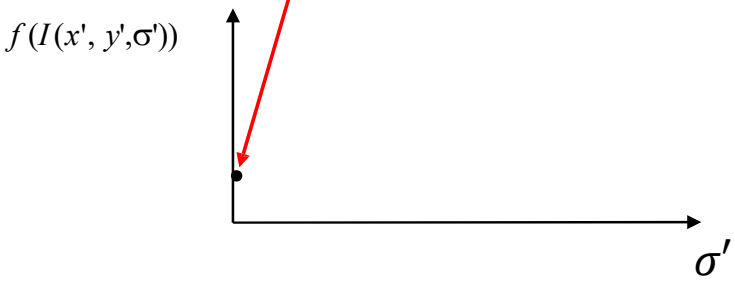
- Idea: Design a function on the image patch, which is **scale invariant** (i.e., it has the same value for corresponding patches, even if they are at different scales)
  - Find **local extrema** of this function
  - The **patch size** at which the local extremum is reached should be **invariant** to image rescaling
  - **Important**: this scale invariant patch size is found in each image **independently**



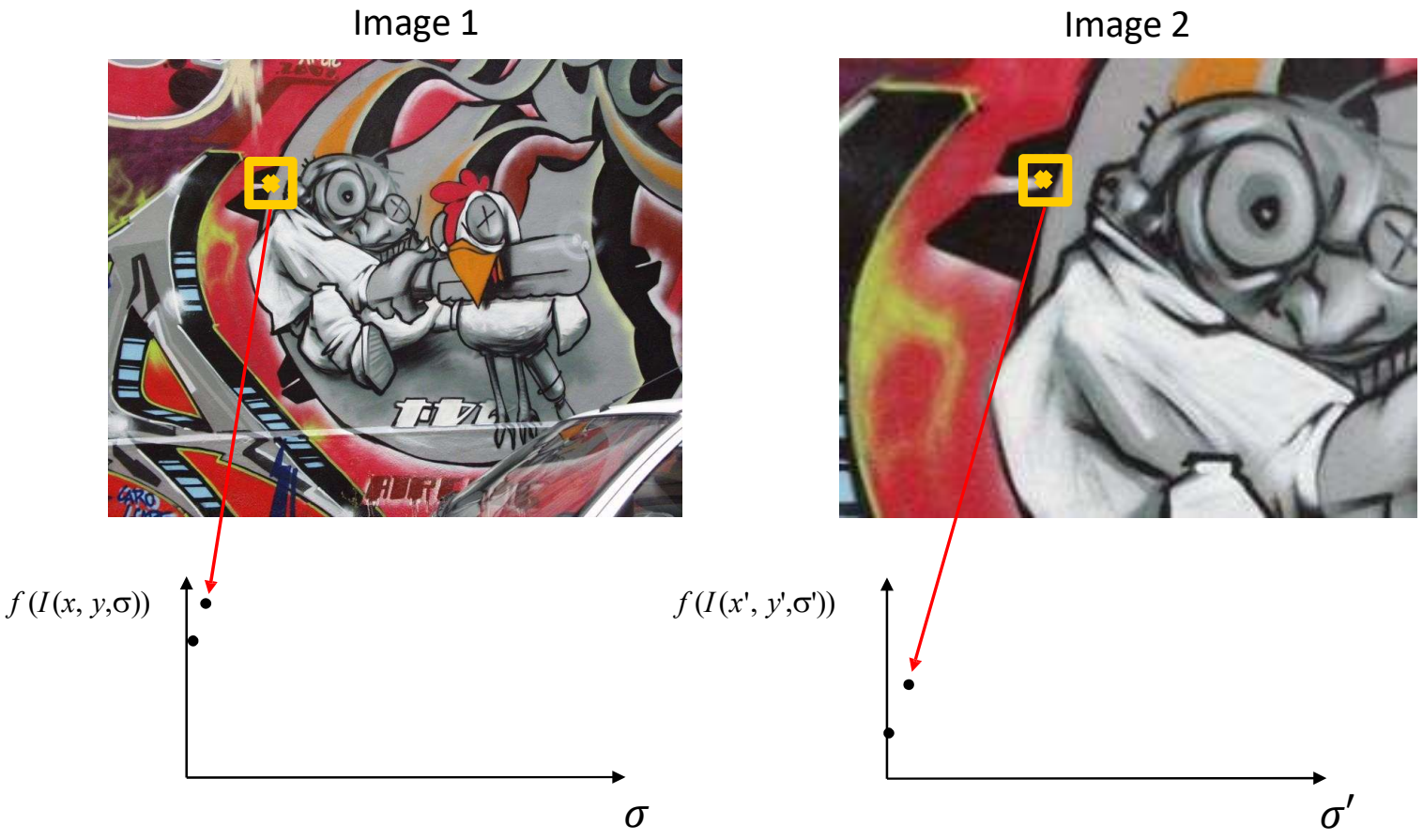
Automatic Scale Selection: Example

Image 1

Image 2

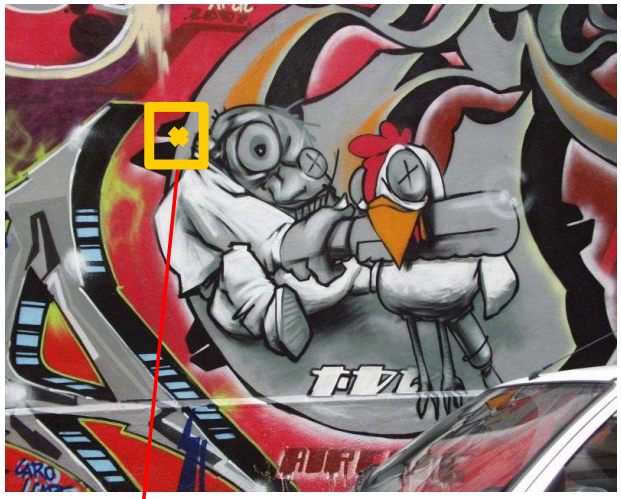


Automatic Scale Selection: Example



Automatic Scale Selection: Example

Image 1



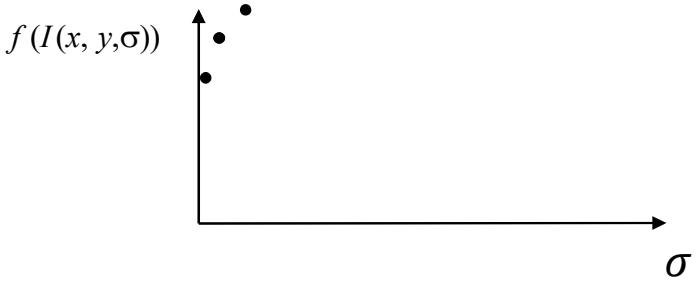
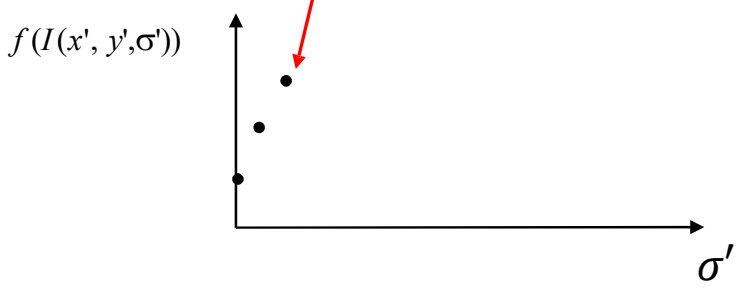
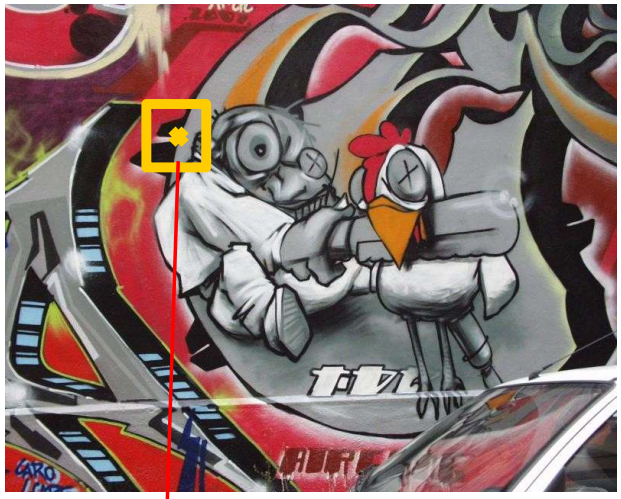


Image 2



Automatic Scale Selection: Example

Image 1



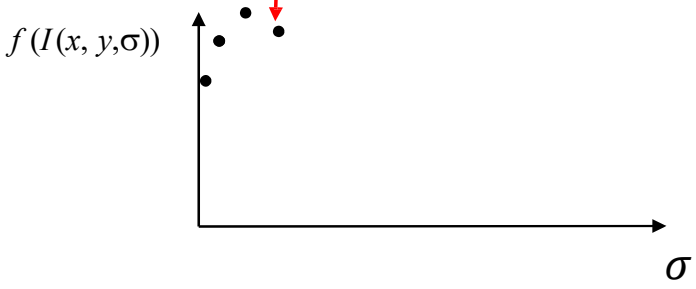
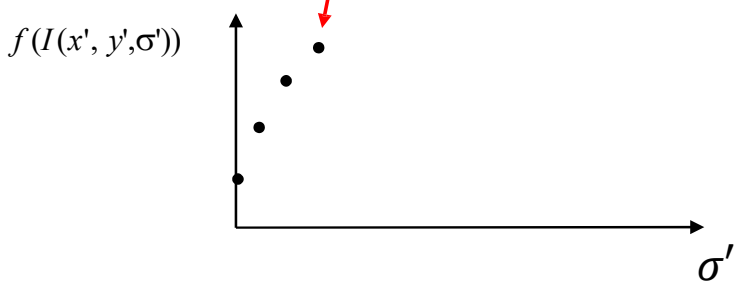
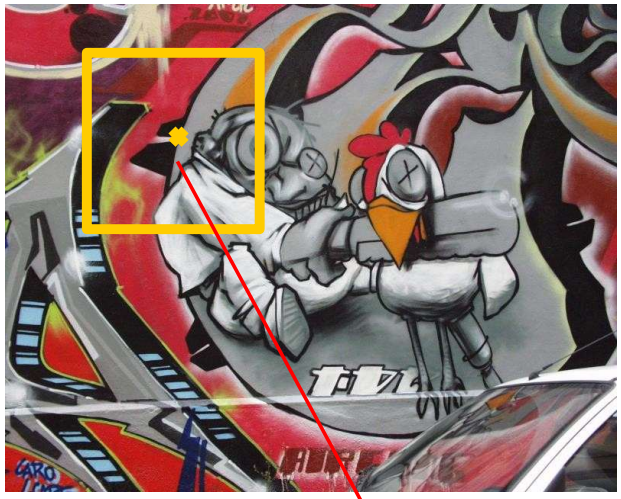


Image 2



Automatic Scale Selection: Example

Image 1



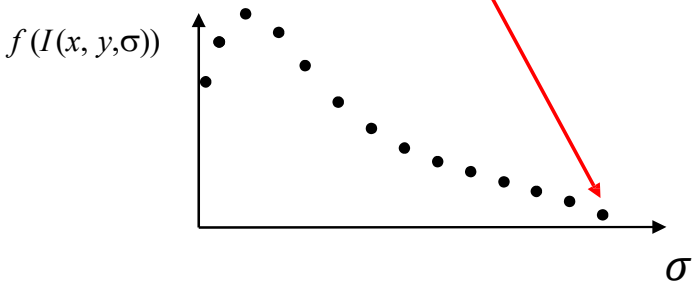
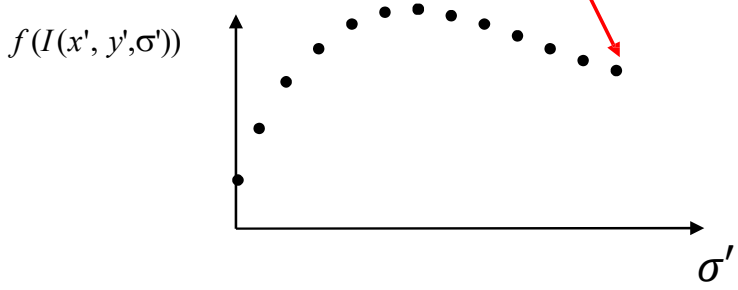
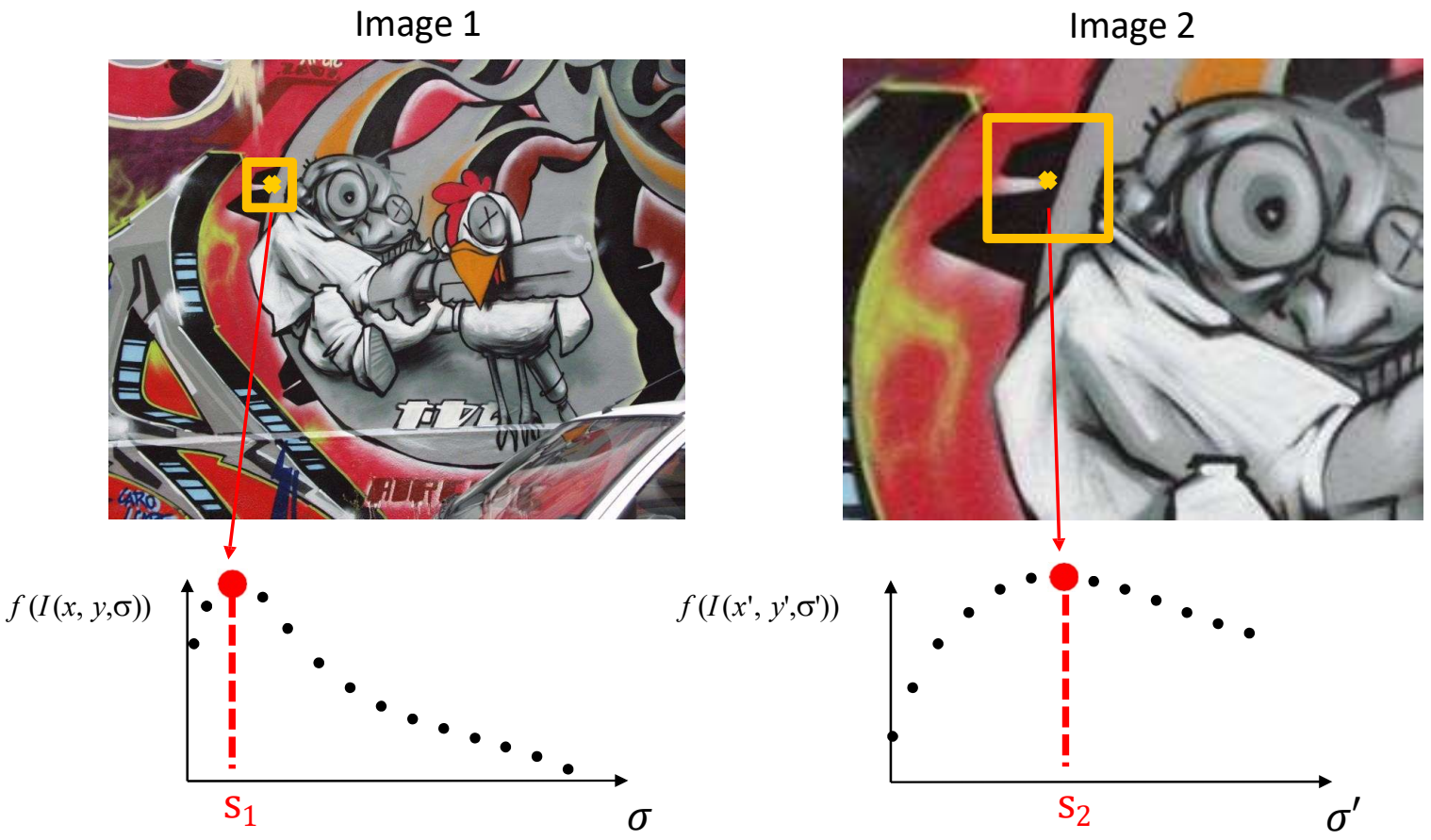


Image 2



Automatic Scale Selection: Example



Automatic Scale Selection: Example

Image 1

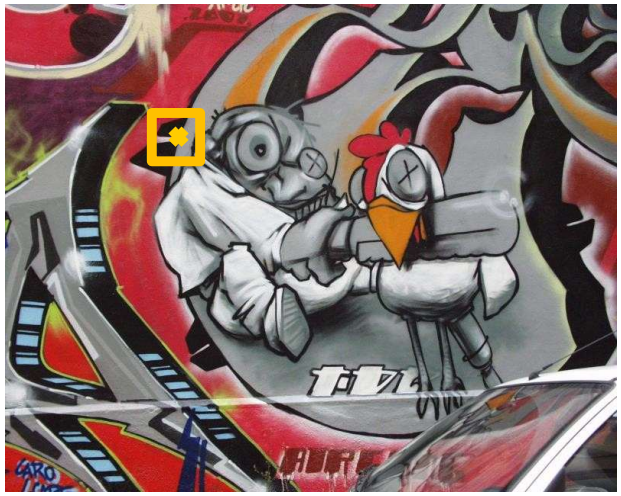
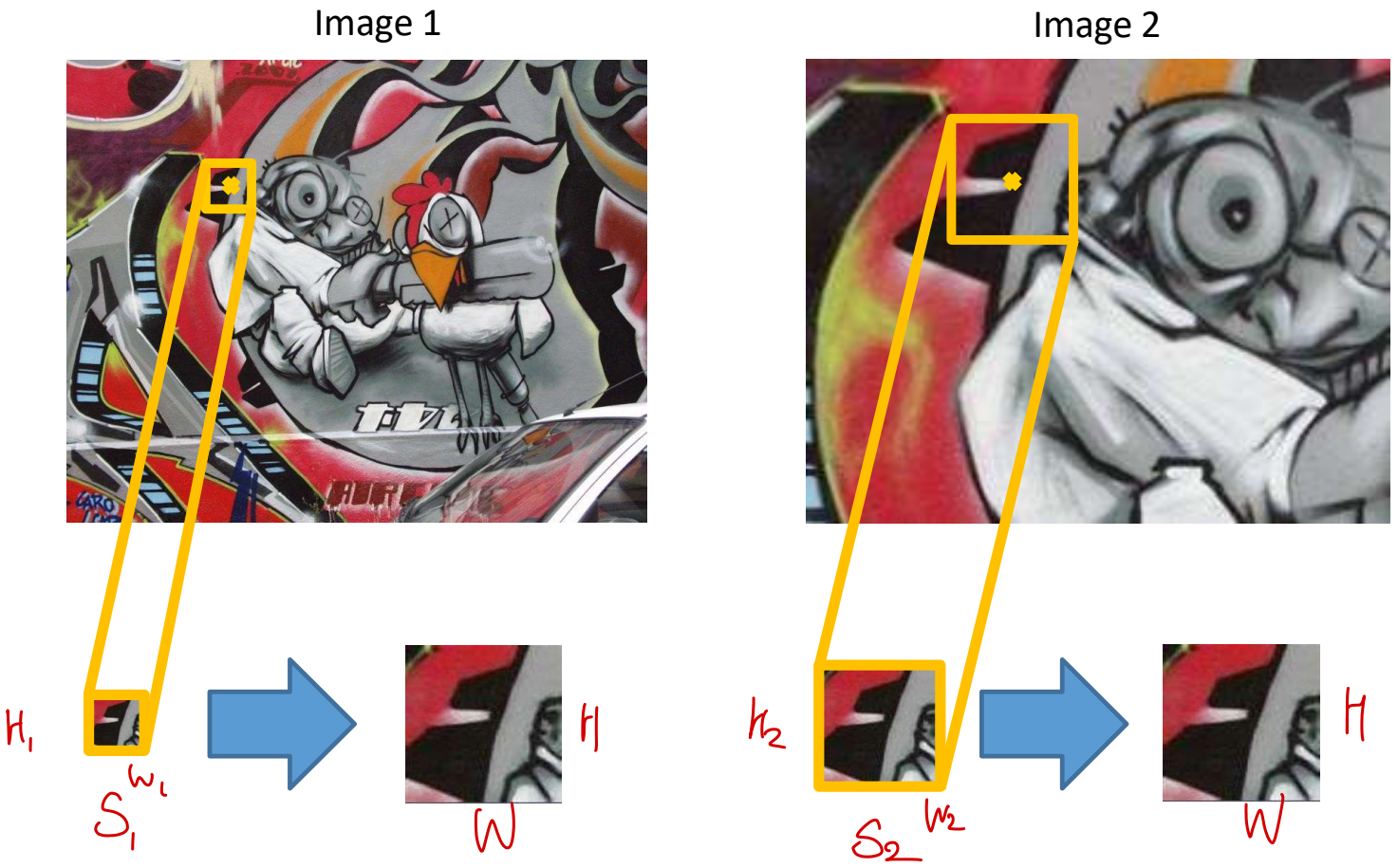


Image 2

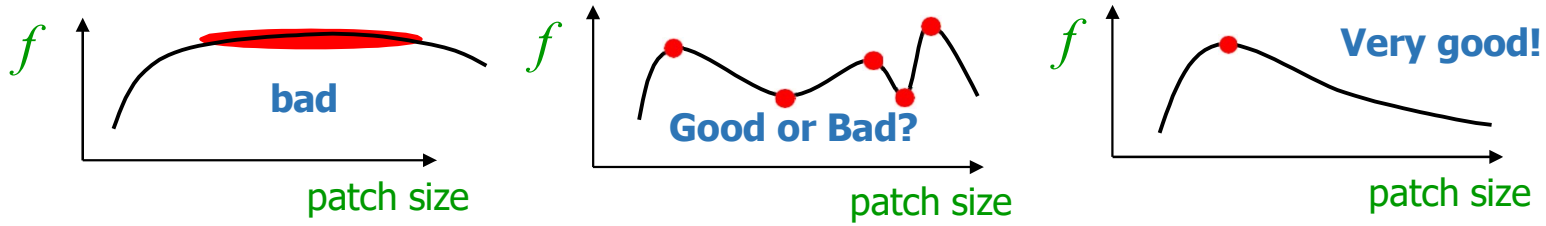
When the right scale is found, the patches must be **normalized to a canonical size** so that they can be compared by SSD.
Patch normalization is done via warping.

Automatic Scale Selection: Example



Automatic Scale Selection

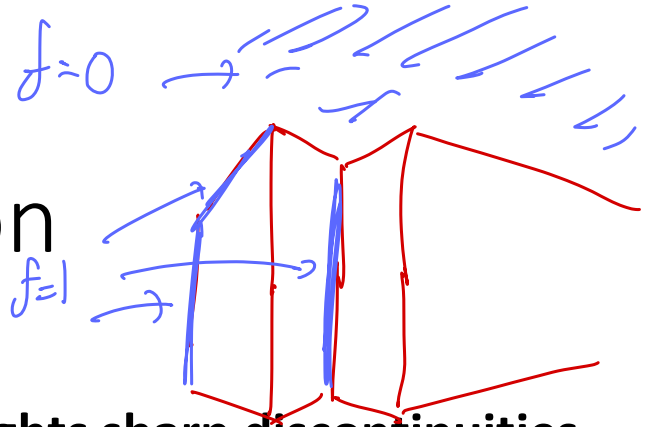
- A “good” function for scale detection should have a single & sharp peak



- What if there are multiple peaks? Is it really a problem?
  - Just select the strongest response
  - Or use multiple peaks!

- What is a good function?
  - **Sharp intensity changes** are good regions to monitor in order to identify the scale

Automatic Scale Selection



- The **ideal function** for determining the scale **is one that highlights sharp discontinuities**
- **Commonly used function f**: convolve image with a **kernel that highlights edges**

$$f = \text{Kernel} * \text{Image}$$

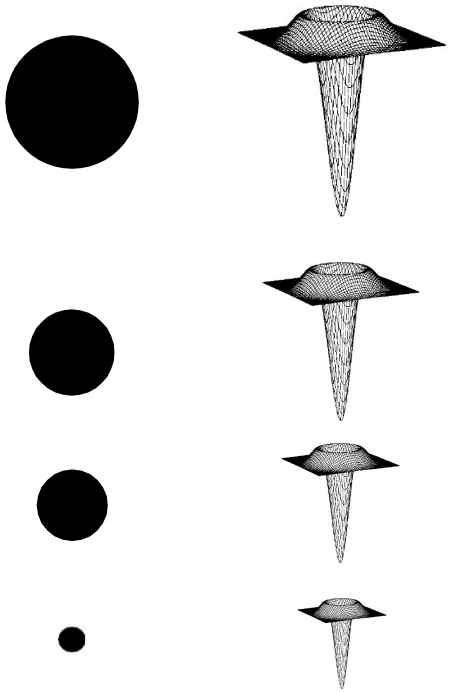
- It has been shown that the **Laplacian of Gaussian kernel** is optimal under certain assumptions [Lindeberg'94]:

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

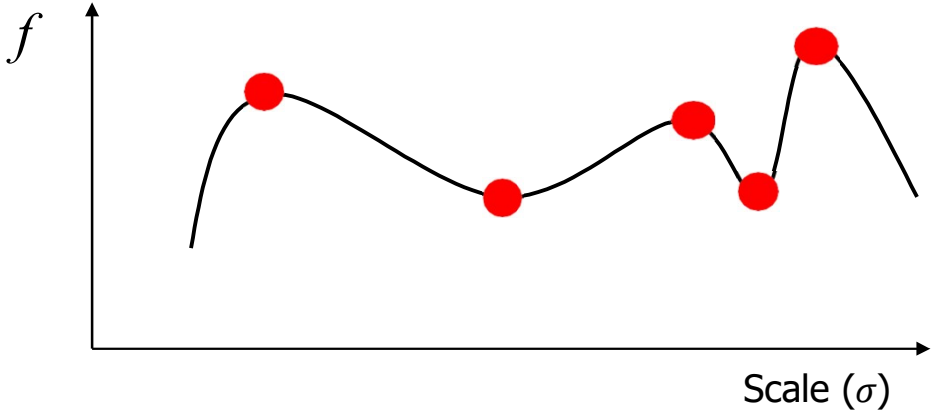
Lindeberg, Scale-space theory: A basic tool for analysing structures at different scales, Journal of Applied Statistics, 1994. PDF.

Automatic Scale Selection

The **correct scale(s)** is (are) found as **local extrema across consecutive smoothed patches**



Scale
(i.e., σ of the LoG)

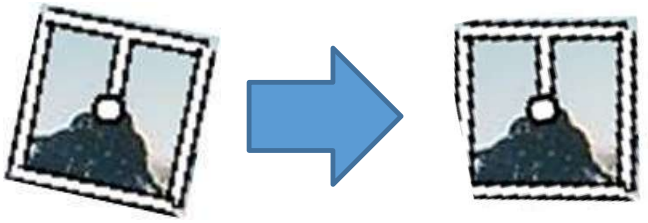


How to achieve invariance to Rotation

Derotation:

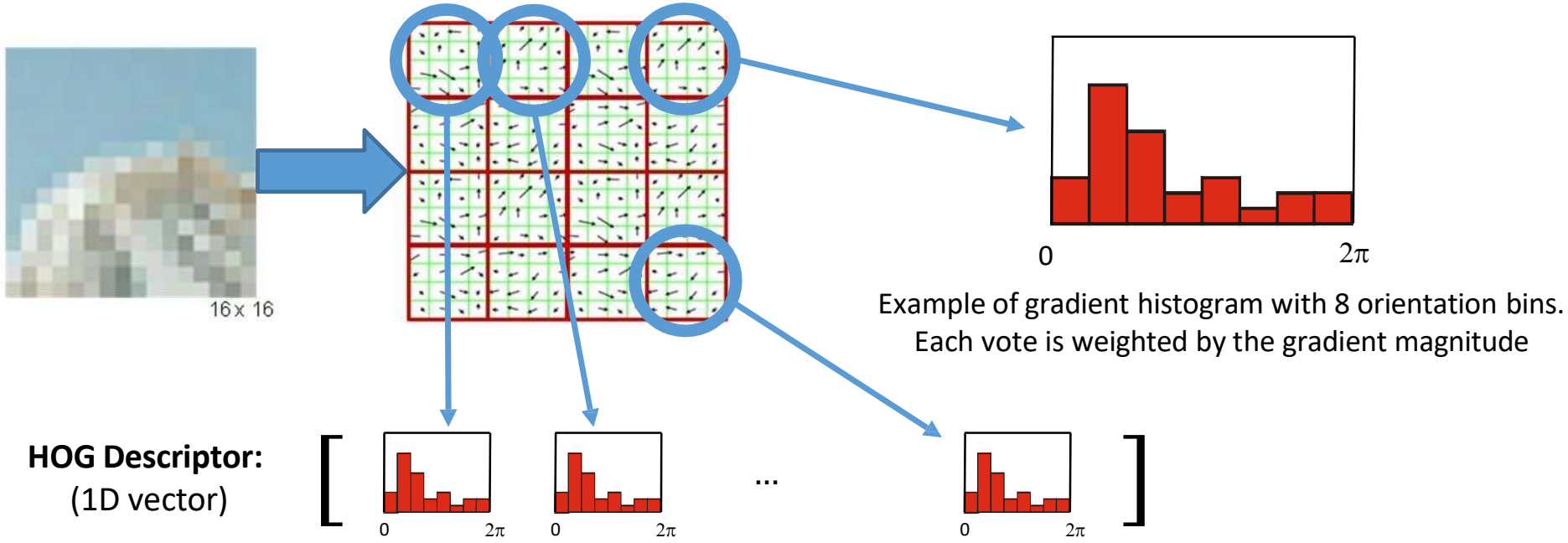
- **Determine patch orientation**
  e.g., eigenvectors of M matrix of Harris or dominant gradient direction (see next slide)
- **Derotate patch through "patch warping"**
  This puts the patches into a **canonical orientation**

→ 다른 viewpoint의 patch를 비교할 수 있게됨.



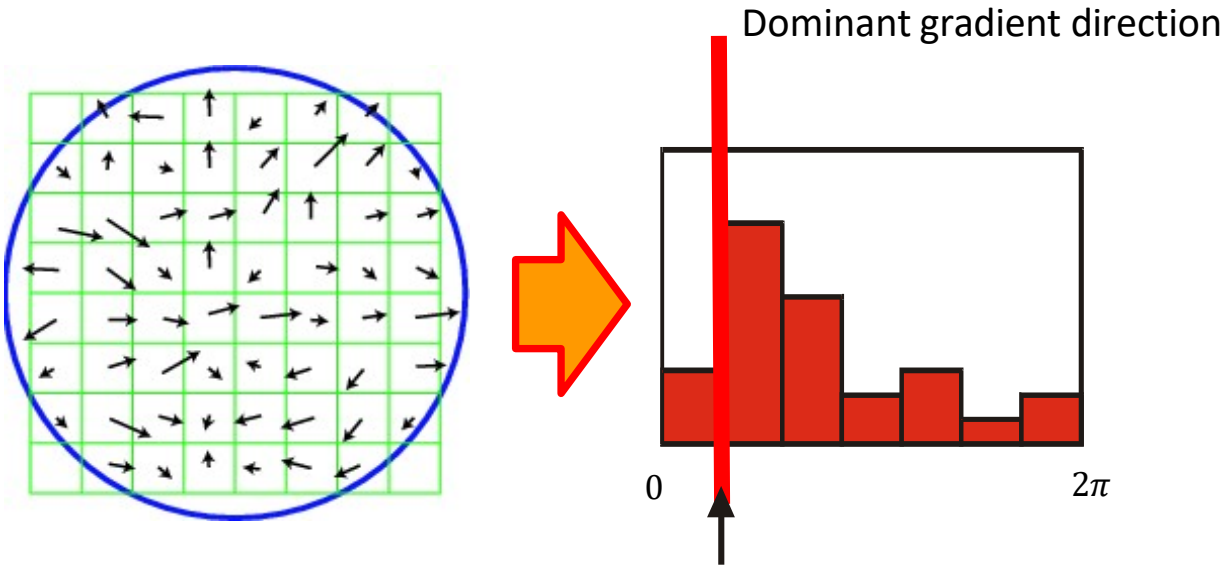
HOG Descriptor (Histogram of Oriented Gradients)

- The patch is divided into a **grid of cells** and for each cell a **histogram of gradient directions** is compiled.
- The HOG descriptor is the **concatenation of these histograms** (used in SIFT)
- Differently from the patch and Census descriptors, HOG has **float values**.



How to determine the patch orientation?

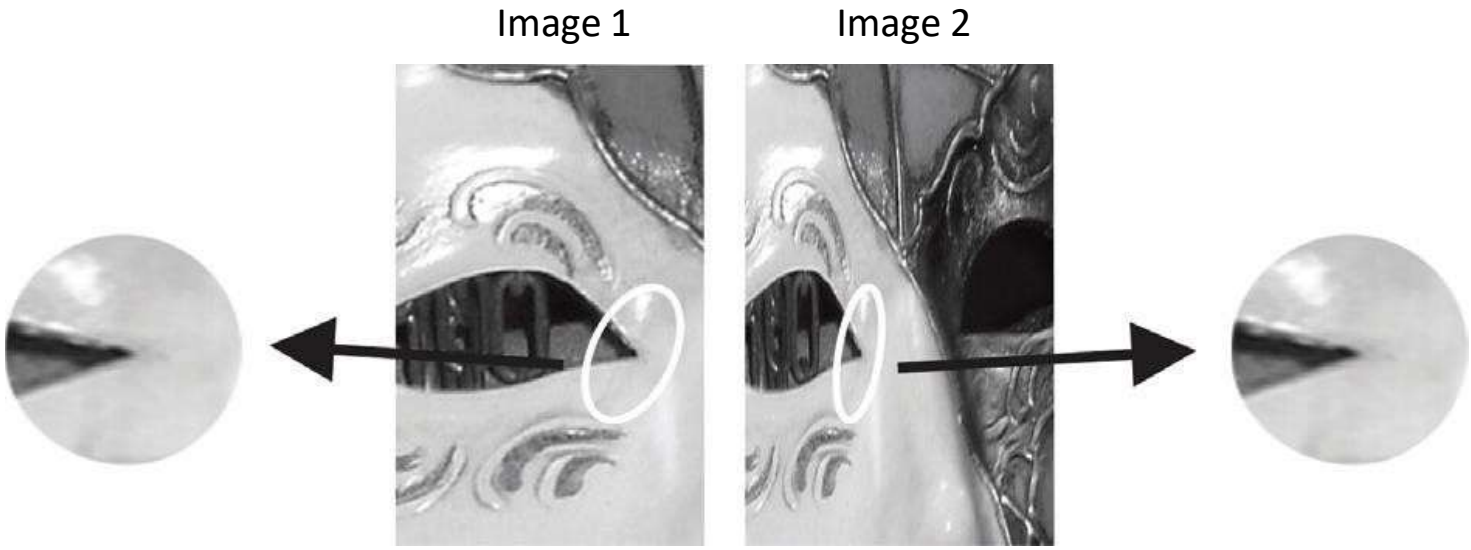
1. First, **multiply the patch by a Gaussian kernel** to make the shape circular rather than square
2. Then, **compute gradients vectors** at each pixel
3. Build a **histogram of gradient orientations**, weighted by the gradient magnitudes. This histogram is a particular case of HOG descriptor (a grid of 1×1 cells)
4. Extract **all local maxima in the histogram**: each local maximum above a threshold is a candidate dominant orientation.
5. Construct a **different keypoint descriptor for each dominant orientation**



How to achieve invariance to small viewpoint changes?

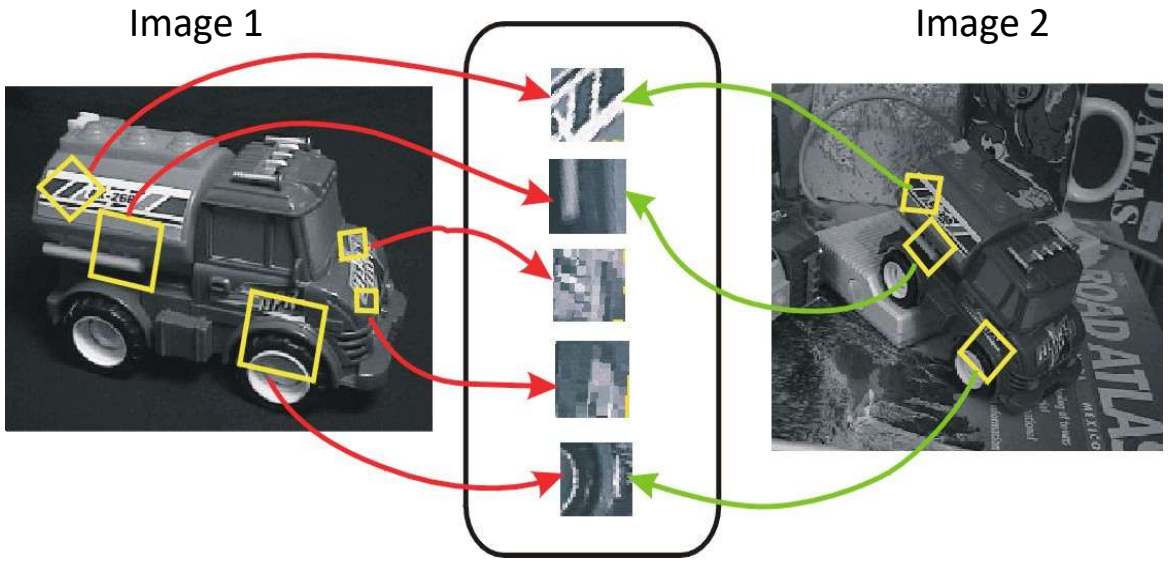
Affine warping provides invariance to small view-point changes

- The **second moment matrix M of the Harris** detector can be used to identify the two **directions of fastest and slowest change of SSD** around the feature (eigen decomposition)
- Out of these two directions, an **elliptic patch is extracted**
- The region inside the ellipse is **normalized to a canonical circular patch**



Recap:
How to achieve Scale, Rotation, and Affine-invariant patch matching

1. **Scale assignment**: compute the scale using the LoG operator. If mutiple local extrema, assign multiple scales. Multiply the patch by a Gaussian kernel to make the shape circular rather than square
2. **Rotation assignment:** use Harris or gradient histogram to find dominant orientation. If multiple local extrema, assign multiple orientations
3. **Affine invariance**: use Harris eigenvectors to extract affine transformation parameters
4. **Warp** the patch into a canonical patch



However, these are not enough..

1. **Scale assignment**: LoG -> slow. Multiple local extrema -> unstable
2. **Rotation assignment:** dominant orientation -> not robust if multiple rotation patterns exist in the patch
3. **Affine invariance**: Harris eigenvectors -> not robust to large viewpoint changes
4. **Warp** the patch into a canonical patch -> huge errors when scale/rotation/affine are used

So, we introduce SIFT!