University of Zurich[UZH]

ROBOTICS & PERCEPTION GROUP

Vision Algorithms for Mobile Robotics

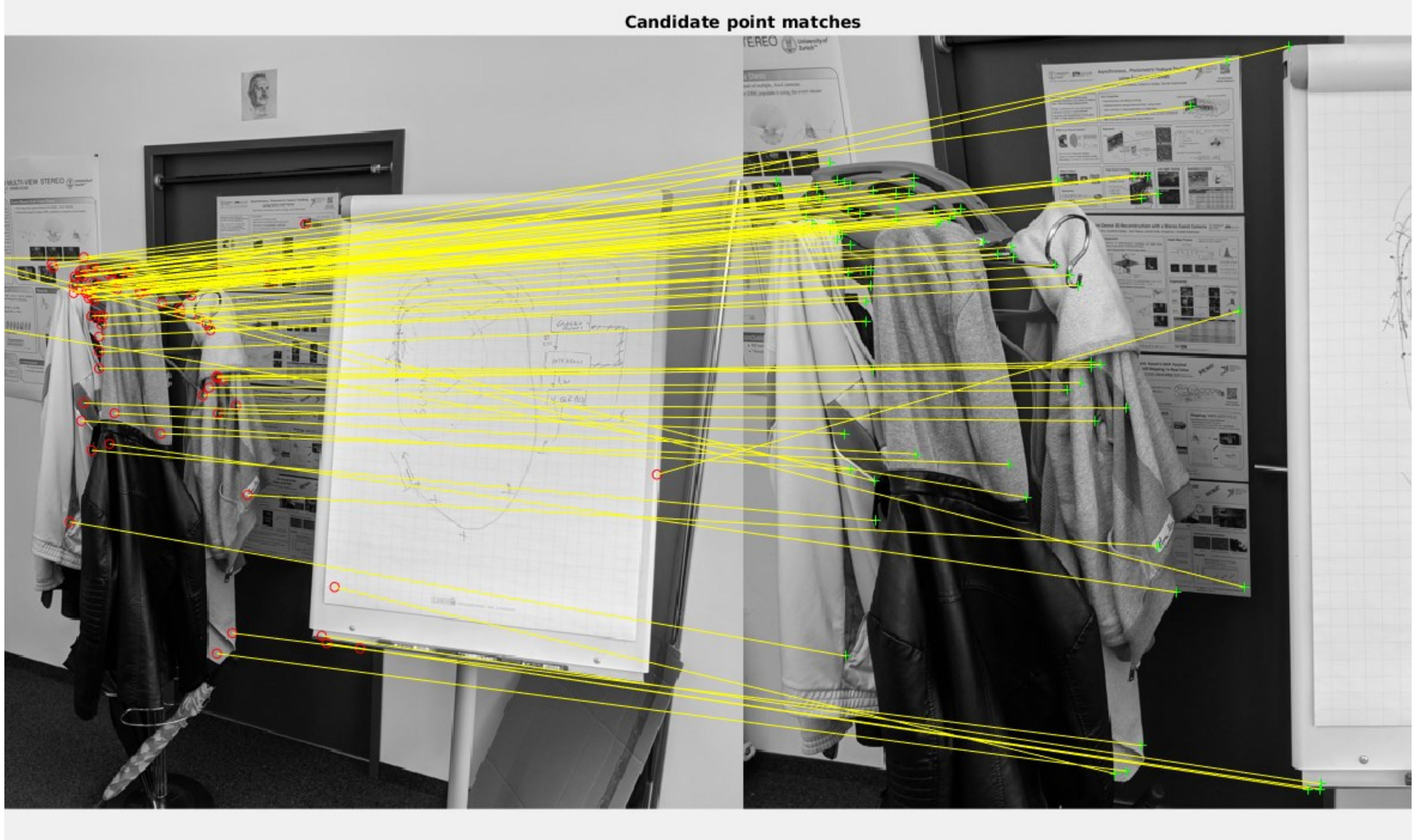
Lecture 06
Point Feature Detection and Matching – Part 2

Davide Scaramuzza

https://rpg.ifi.uzh.ch

Lab Exercise 4 – Today

Implement SIFT blob detection and matching



Main questions

- What features are *repeatable* and *distinctive?*
- How to *describe* a feature?
- How to establish *correspondences*, i.e., compute matches?

Feature Matching

For each point, how to **match** its **corresponding point** in the other image?

- **Brute-force Matching: compare each feature** descriptor **of Image 1 against** the descriptor of **each feature in Image 2** and assign as correspondence the feature with **closest descriptor** (e.g., minimum of SSD). If each image contains N features, we need to perform N^2 **comparisons**.

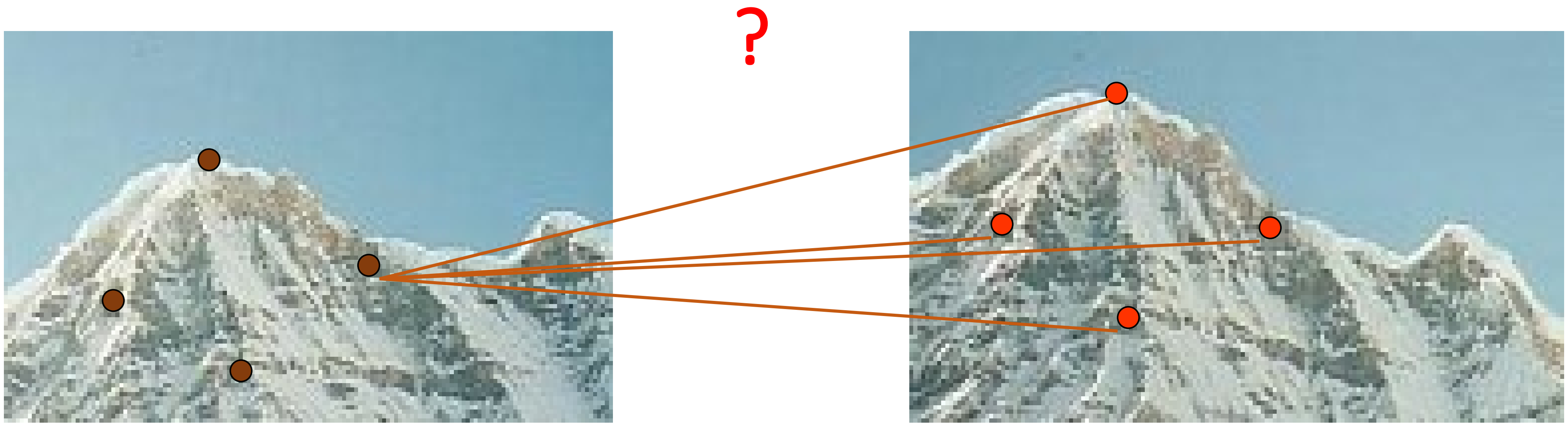
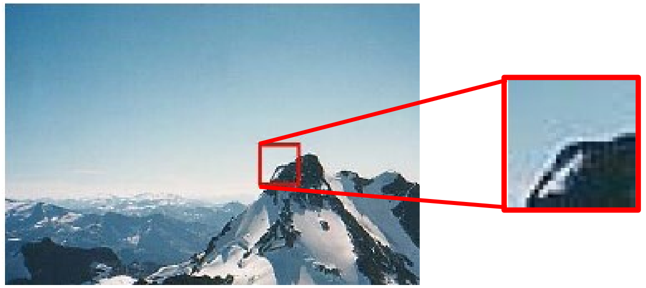


Image 1

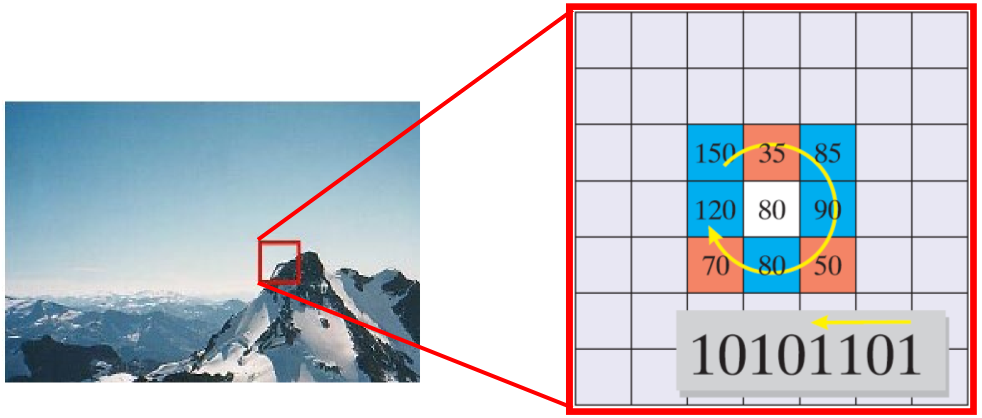
Image 2

Recall: Patch and Census Descriptors

- **Patch descriptor**
  (i.e., patch of intensity values, integer values)

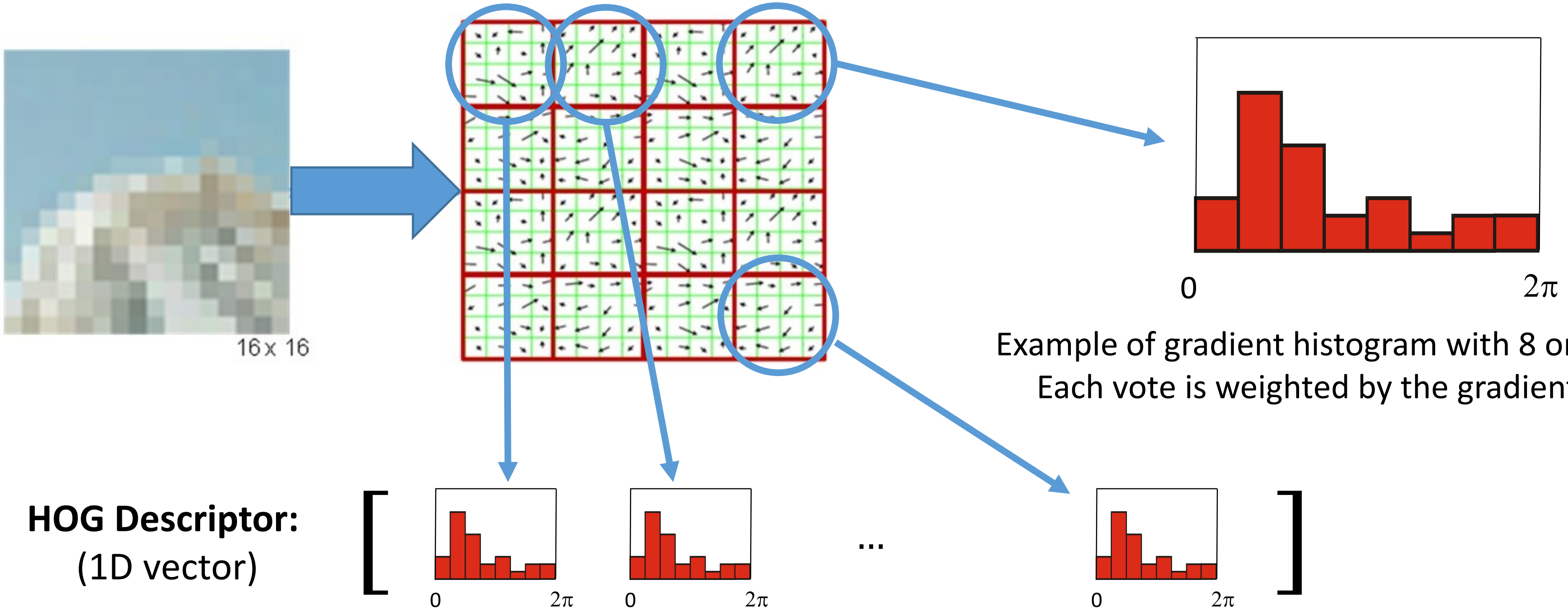


- **Census descriptor** (binary values)



HOG Descriptor (Histogram of Oriented Gradients)

- The patch is divided into a **grid of cells** and for each cell a **histogram of gradient directions weighted by the gradient magnitude** 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**.



Feature Descriptor Invariance

Are feature descriptors **invariant** (robust) to **geometric** and **photometric** changes?

Geometric changes: scale, rotation, viewpoint

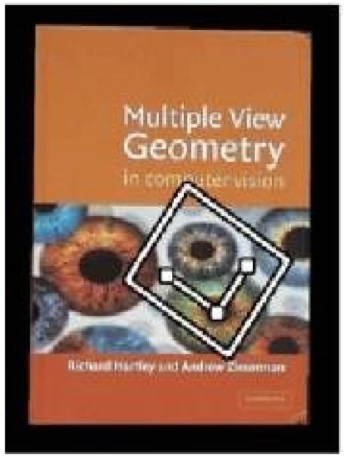


Image 1

Image 2

Photometric changes: illumination

Image 1

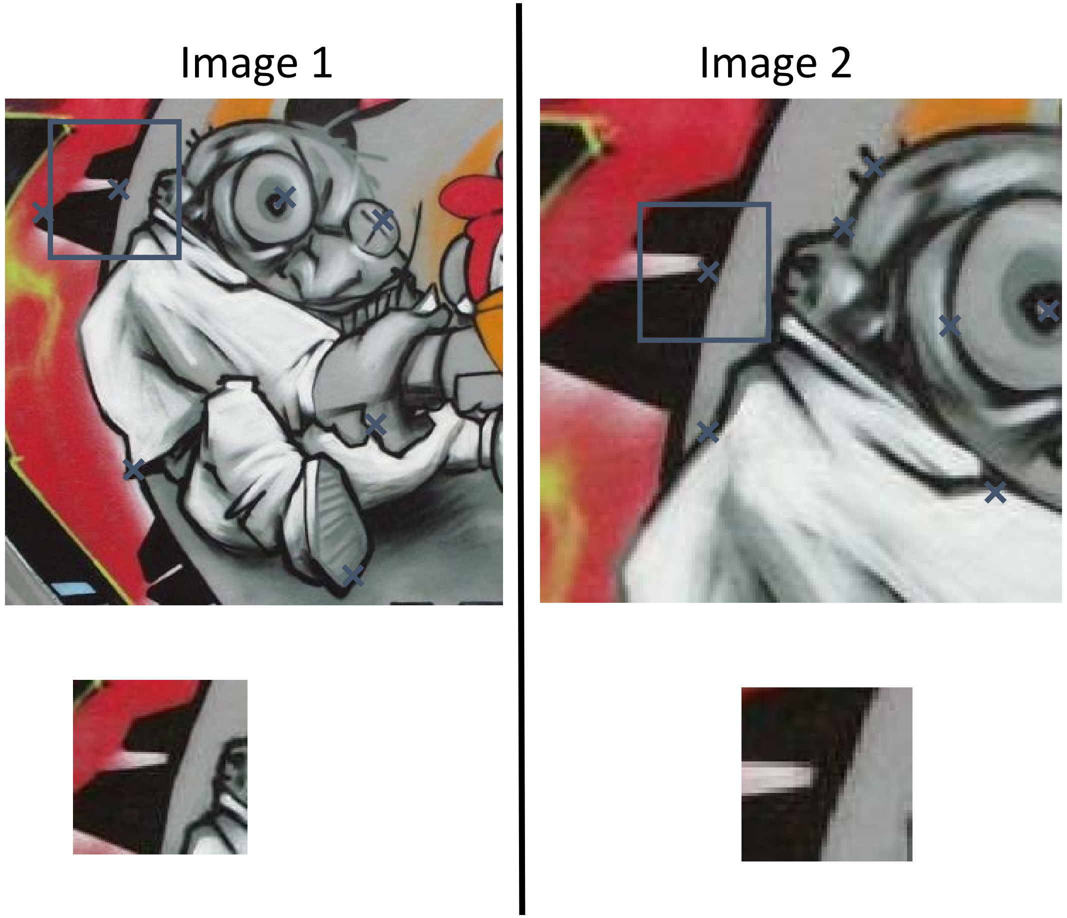
Image 2

Outline

- How to achieve descriptor invariance to:
  - $\boxed{\text{Scale}}$
  - Rotation
  - Viewpoint
- The SIFT blob detector and descriptor
- Other corner and blob detectors and descriptors

Scale changes

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

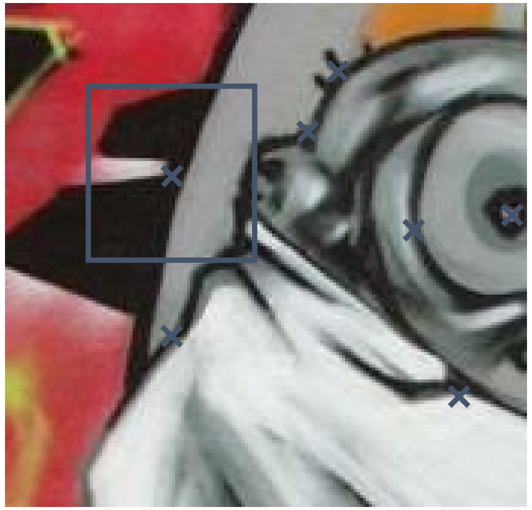


Scale changes

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

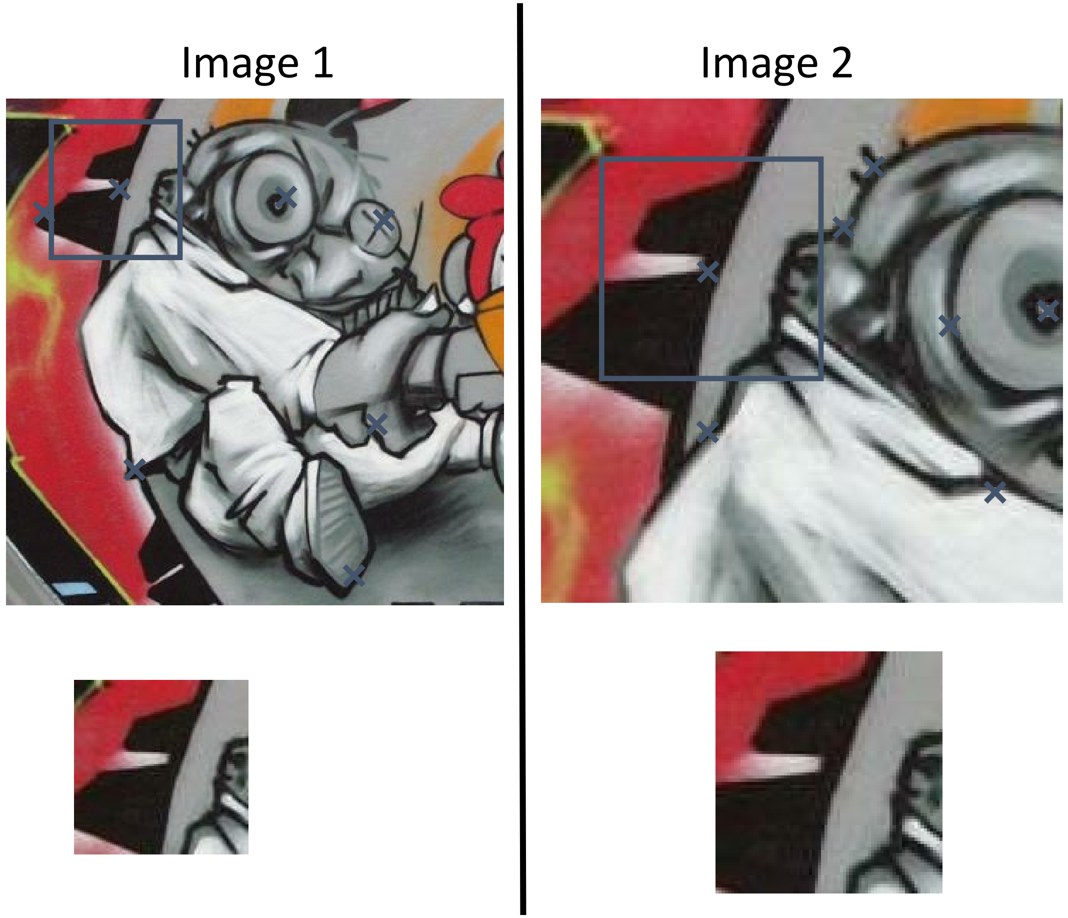
Image 1

Image 2



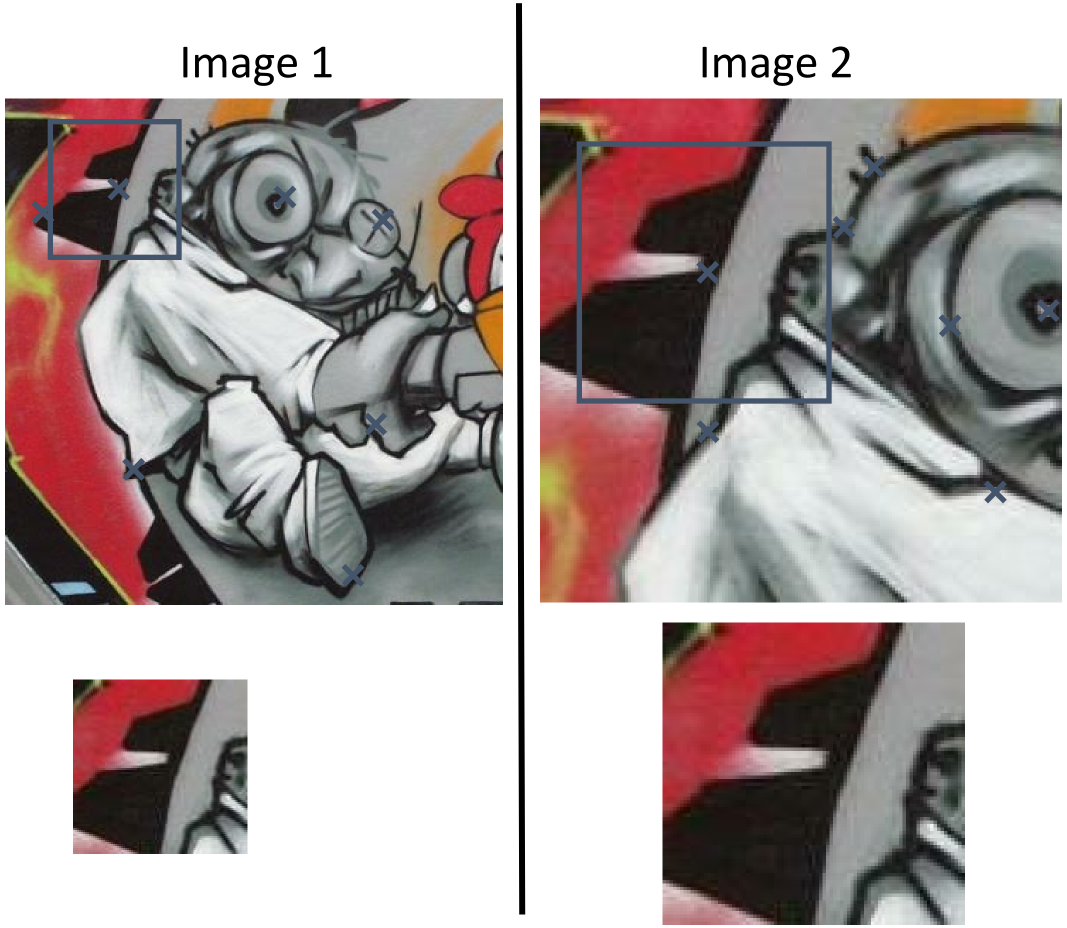
Scale changes

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

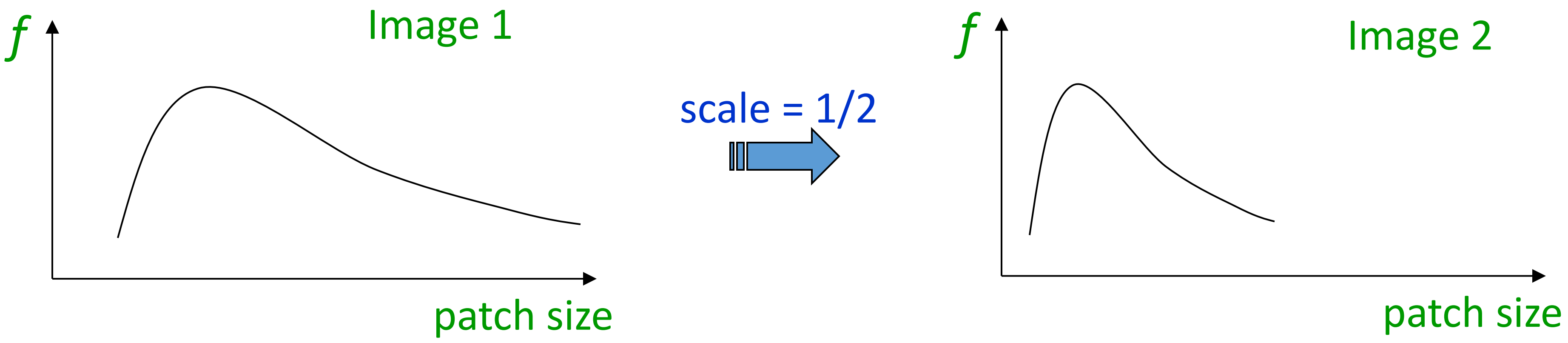


Scale changes

- 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)

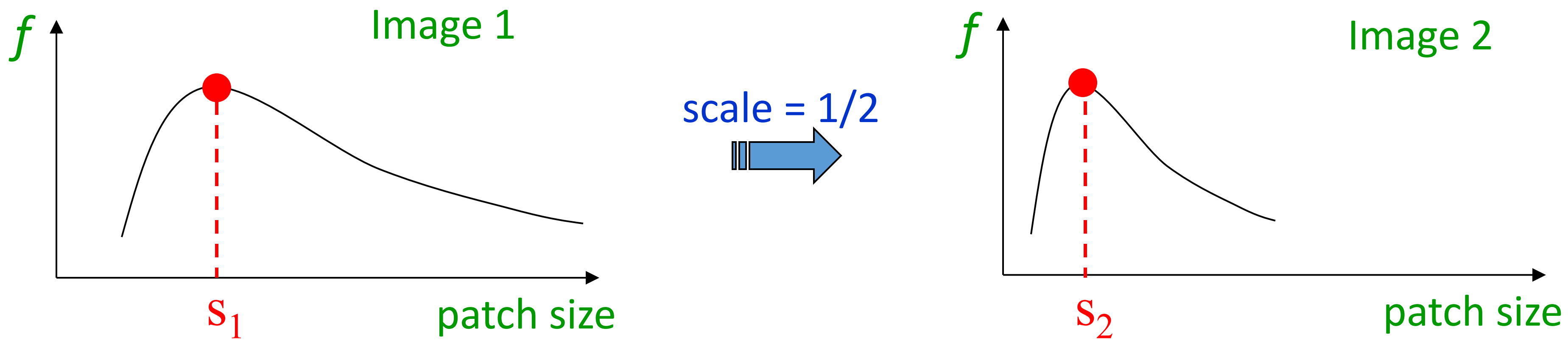
Automatic Scale Selection

- 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)



Automatic Scale Selection

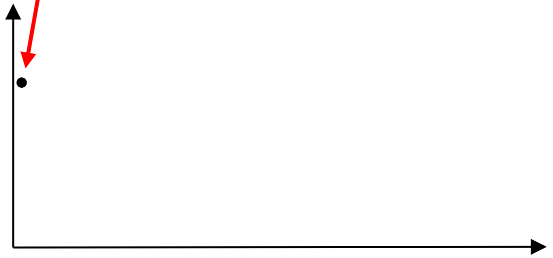
- Idea: Design a function on the image patch, which is **scale invariant** (i.e., it has the same value for corresponding patches, even if they are at different scales)
  - Find **local extrema** of this function
  - The **patch size** at which the local extremum is reached should be **invariant** to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

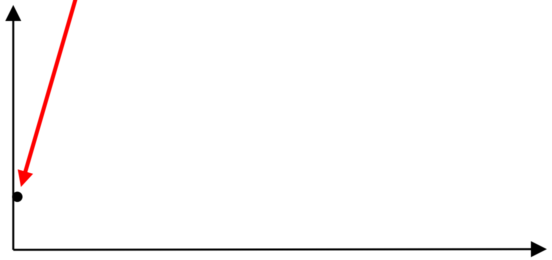


Automatic Scale Selection: Example

Image 1

Image 2

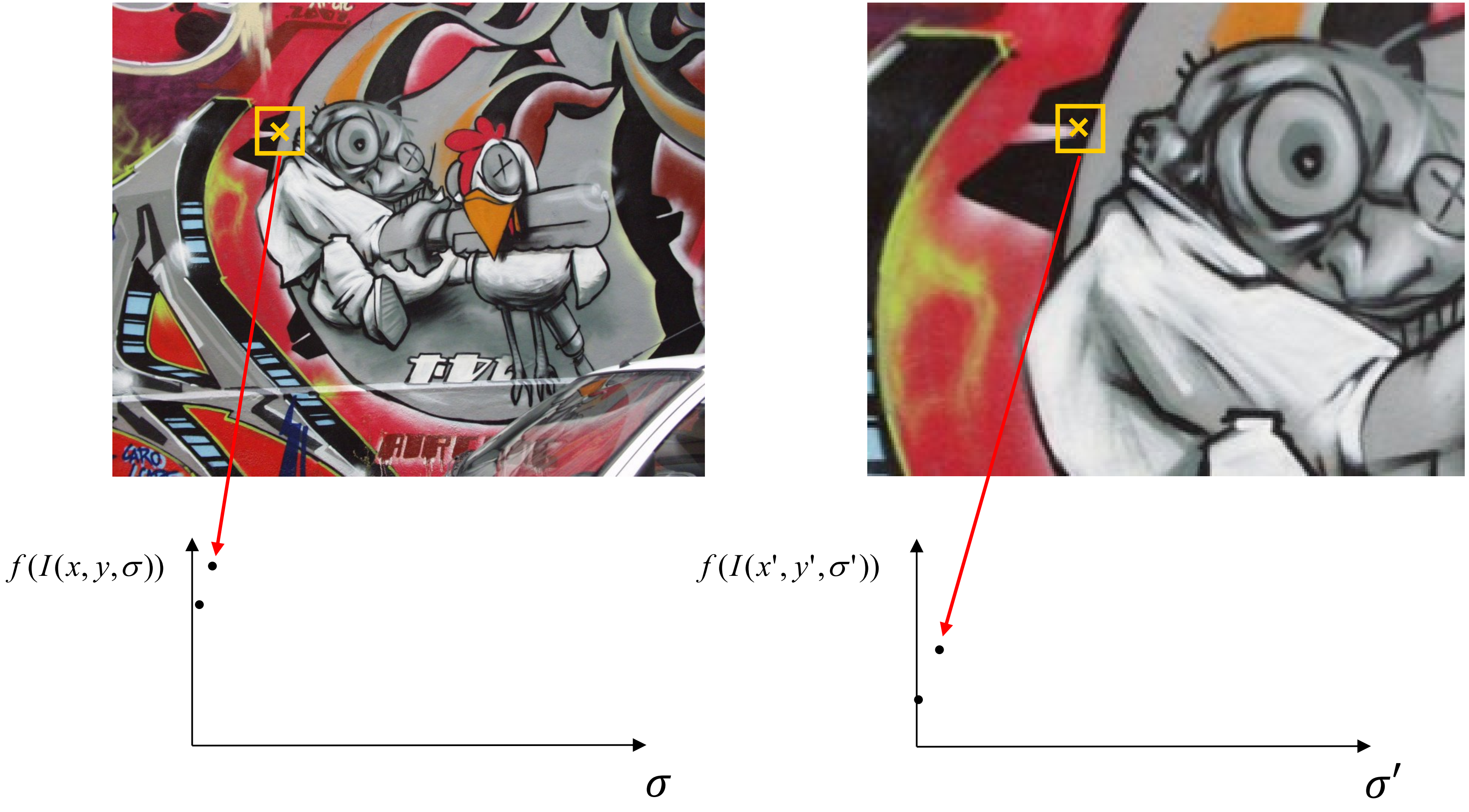




Automatic Scale Selection: Example

Image 1

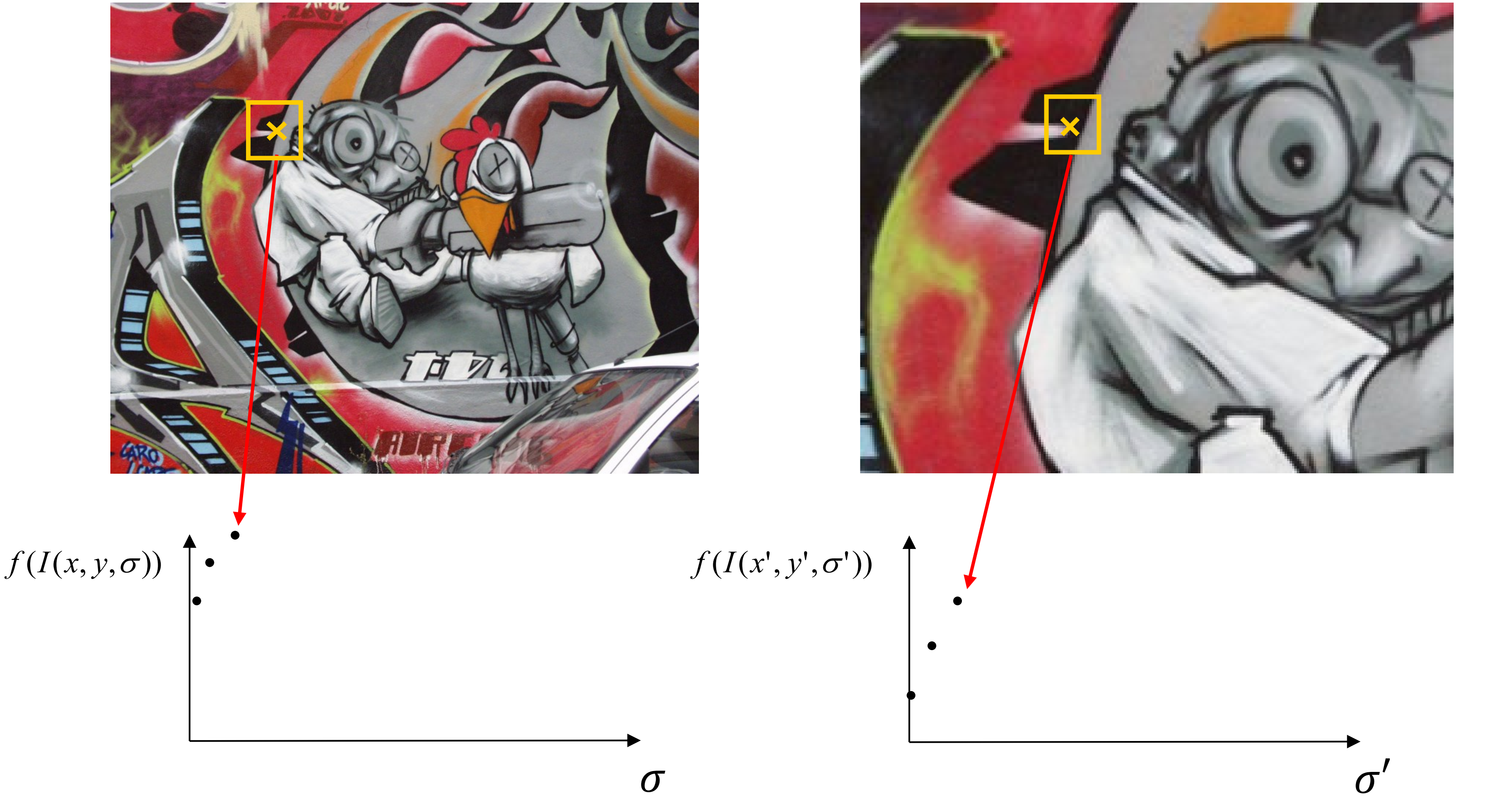
Image 2



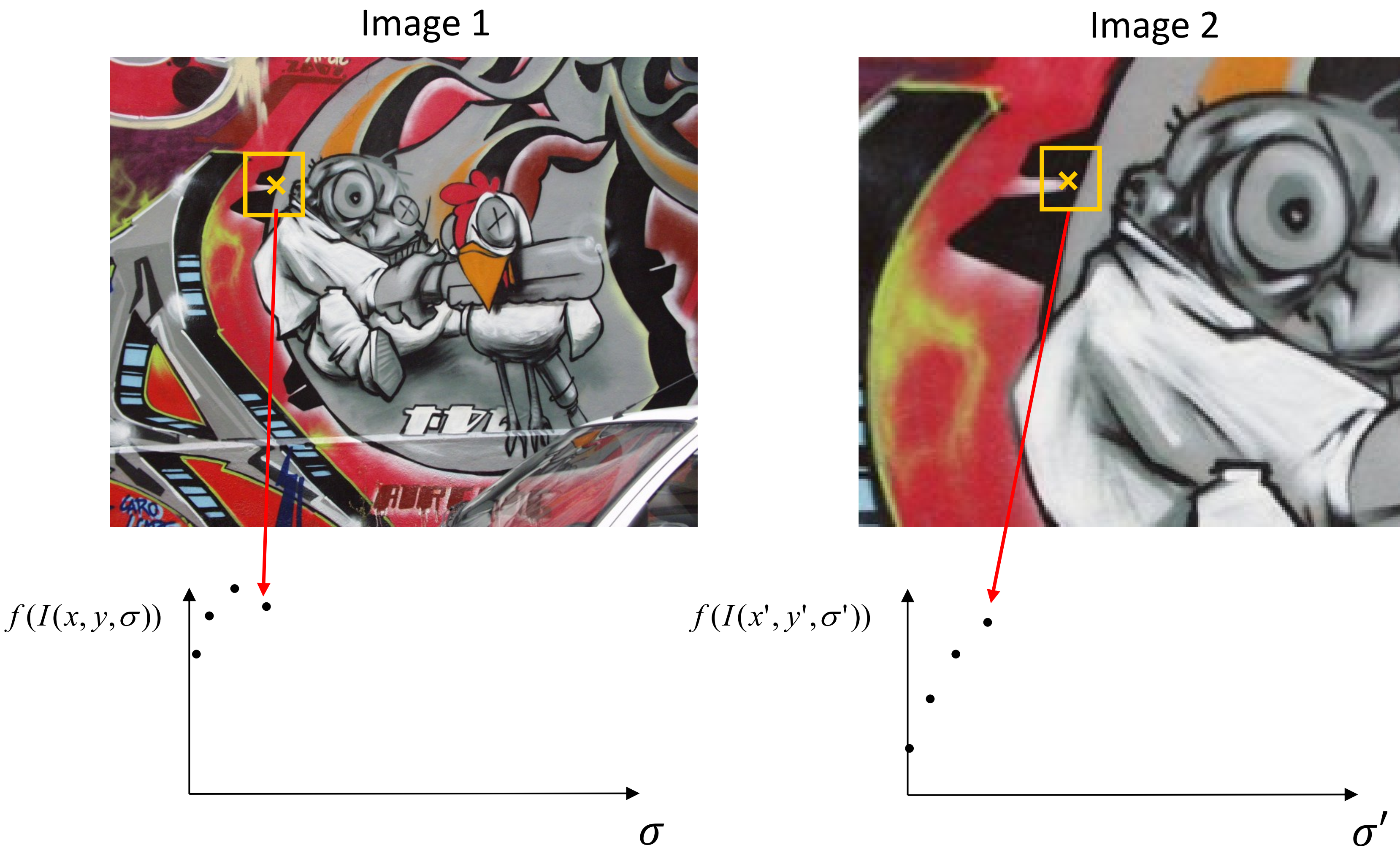
Automatic Scale Selection: Example

Image 1

Image 2



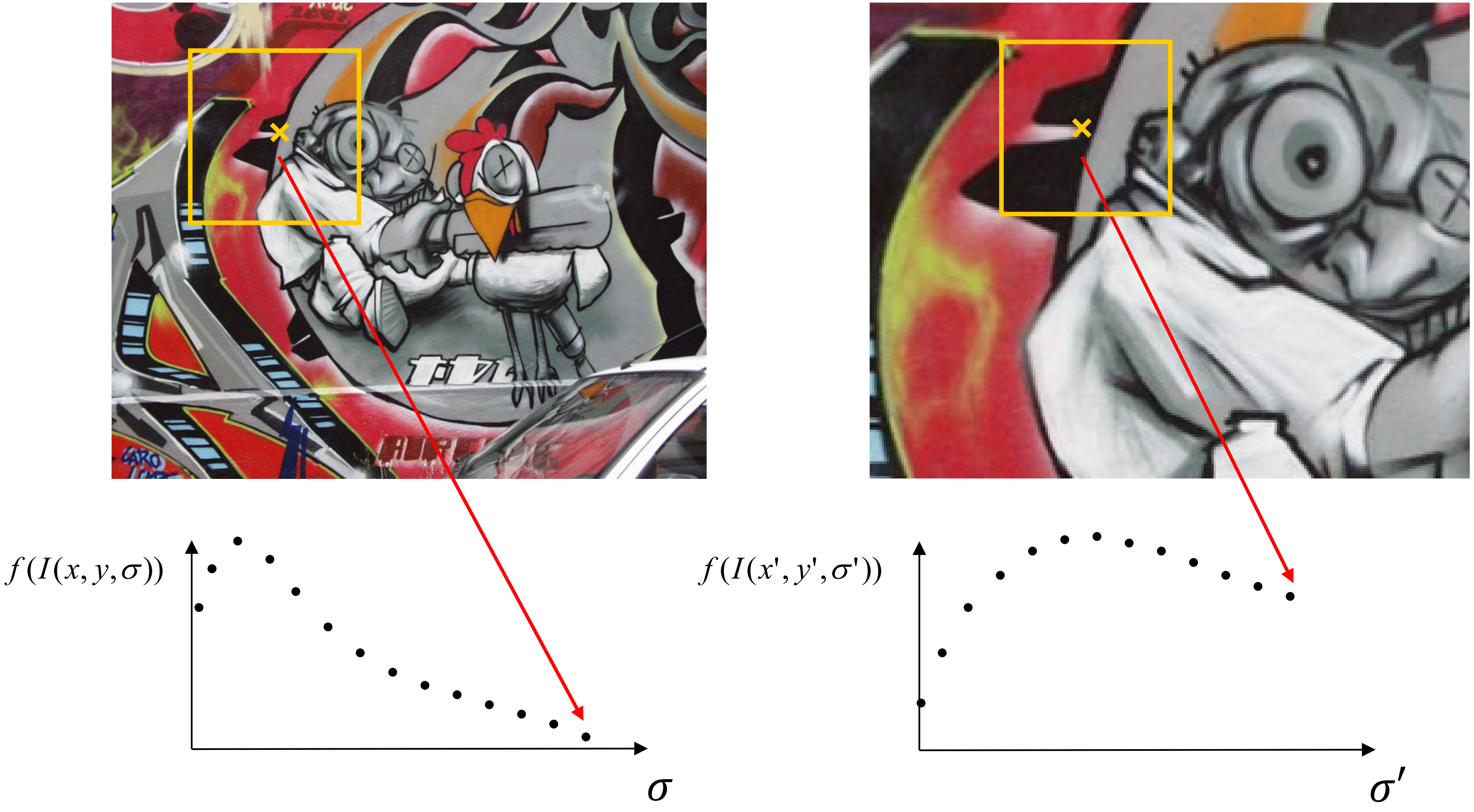
Automatic Scale Selection: Example



Automatic Scale Selection: Example

Image 1

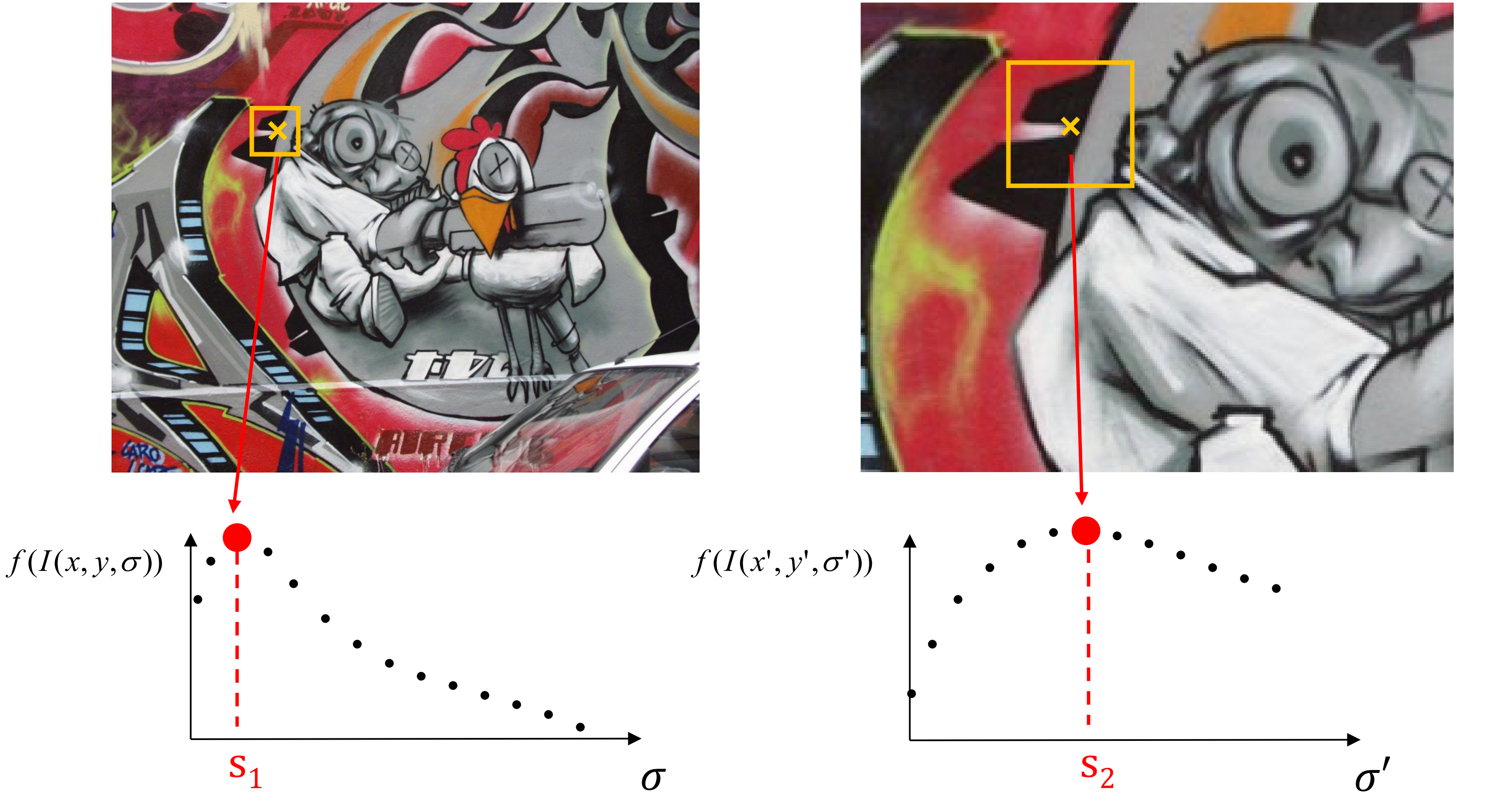
Image 2



Automatic Scale Selection: Example

Image 1

Image 2



Automatic Scale Selection: Example

Image 1

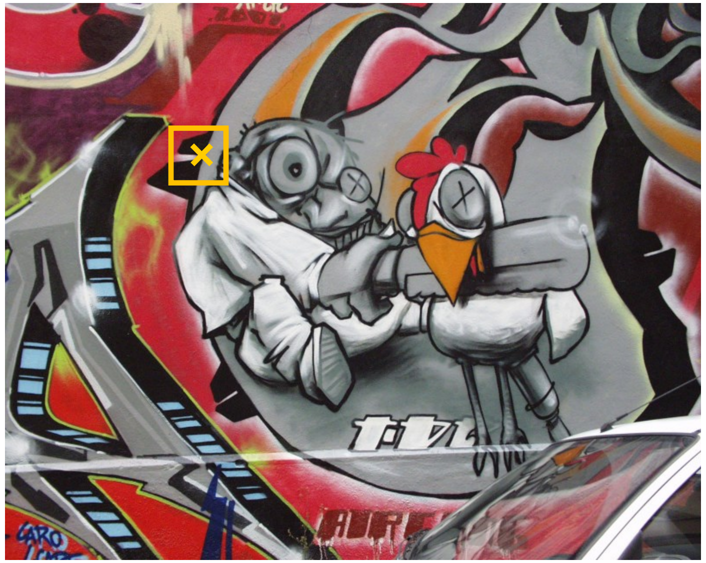


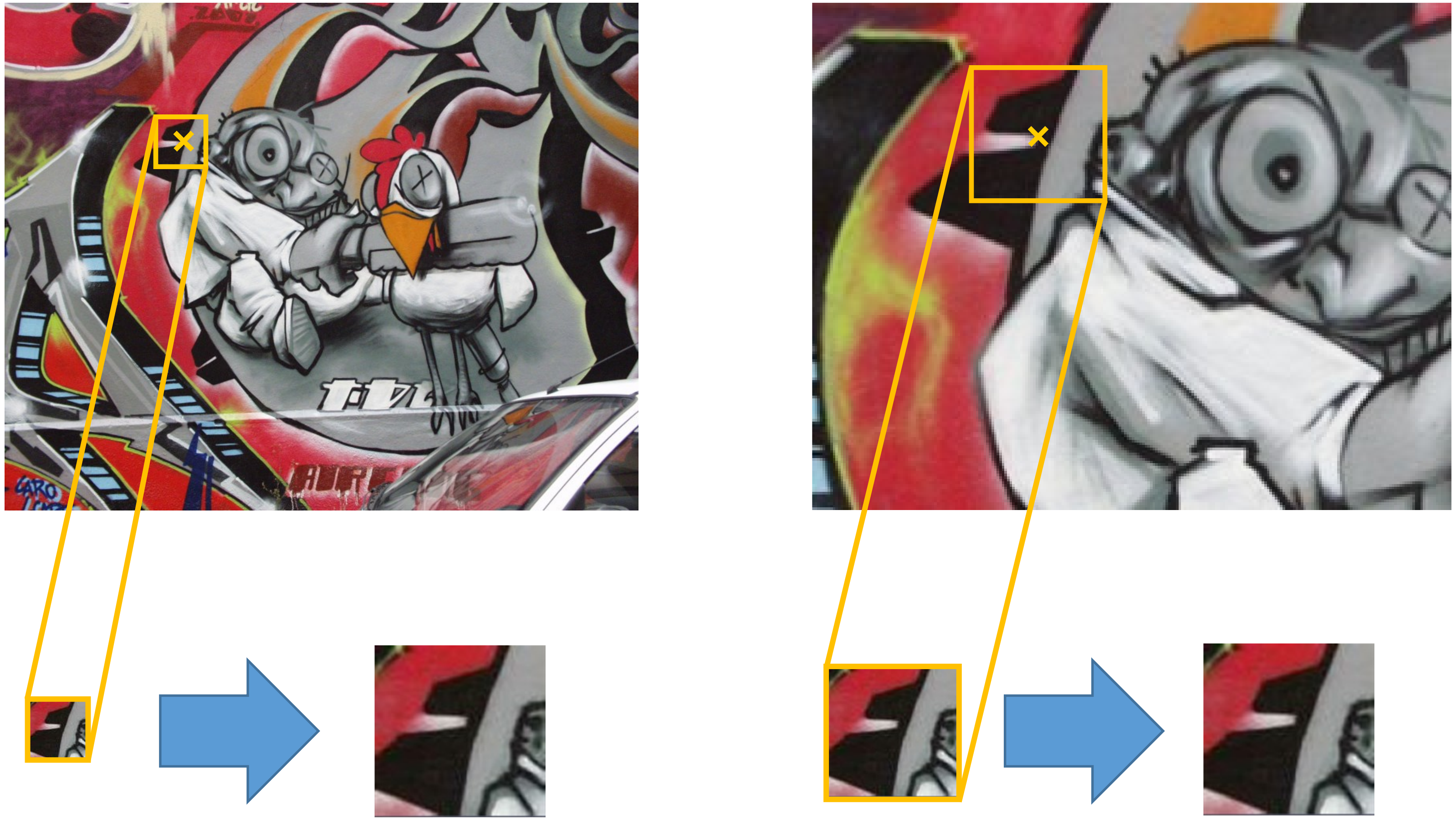
Image 2

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

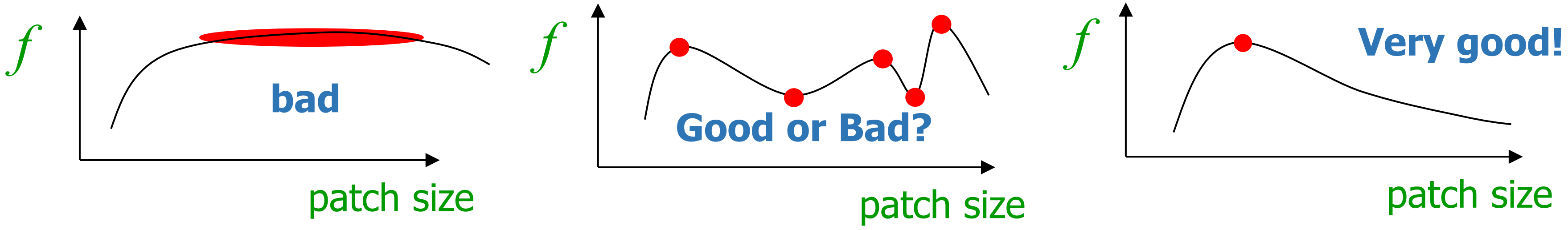
Image 1

Image 2



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?

- 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**
- **Solution**: 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]:

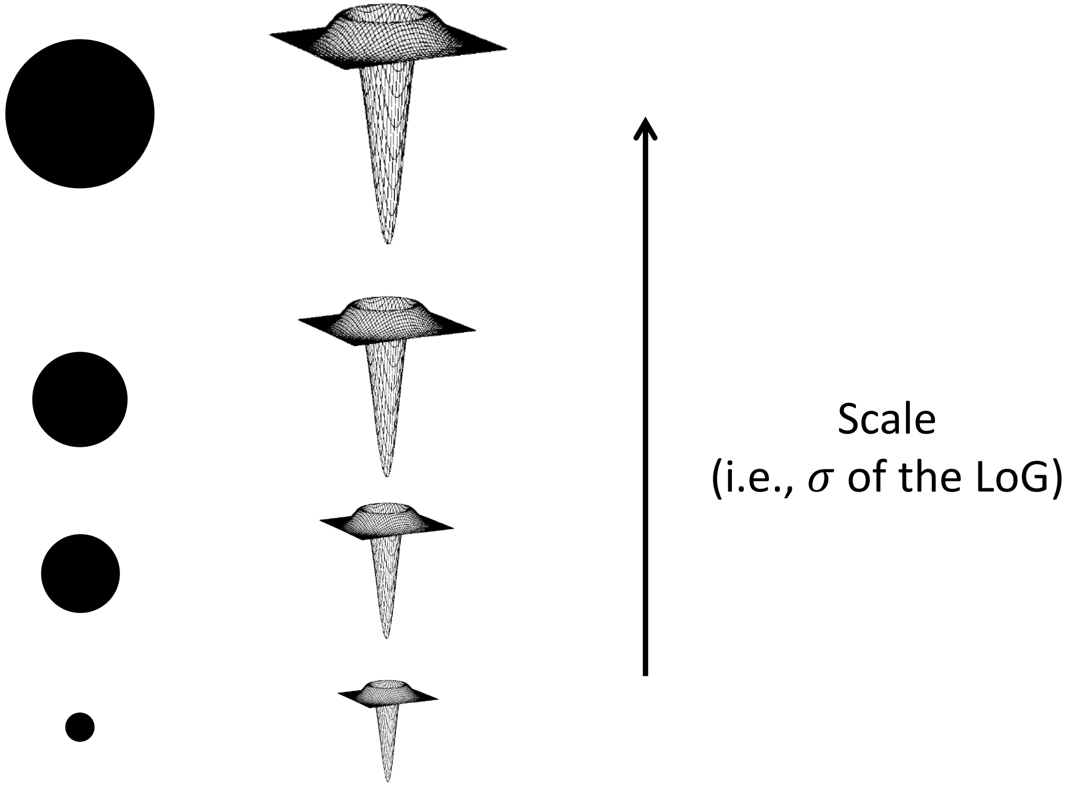
$$LoG(x, y, \sigma) = \nabla^2 G_\sigma(x, y) = \frac{\partial^2 G_\sigma(x, y)}{\partial x^2} + \frac{\partial^2 G_\sigma(x, y)}{\partial y^2}$$

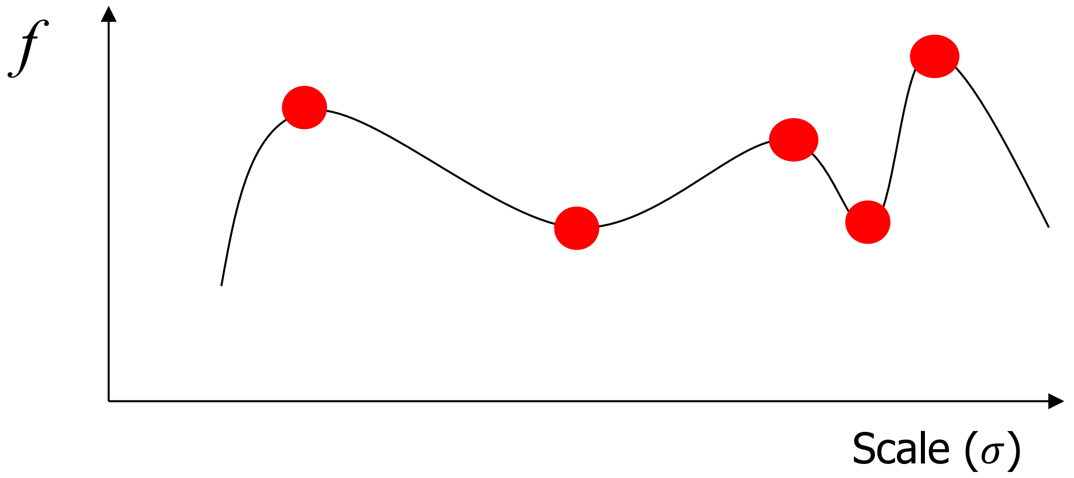
Careful to not confuse it with LOG for edge detection. There we looked for zero crossings across x,y while now we look for local extrema across **scales**

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 scales**





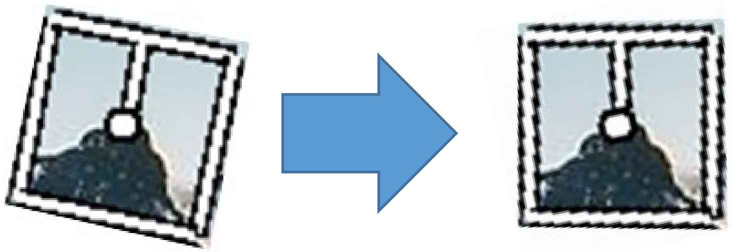
Outline

- How to achieve descriptor invariance to:
  - Scale
  - Rotation
  - Viewpoint
- The SIFT blob detector and descriptor
- Other corner and blob detectors and descriptors

How to achieve invariance to Rotation

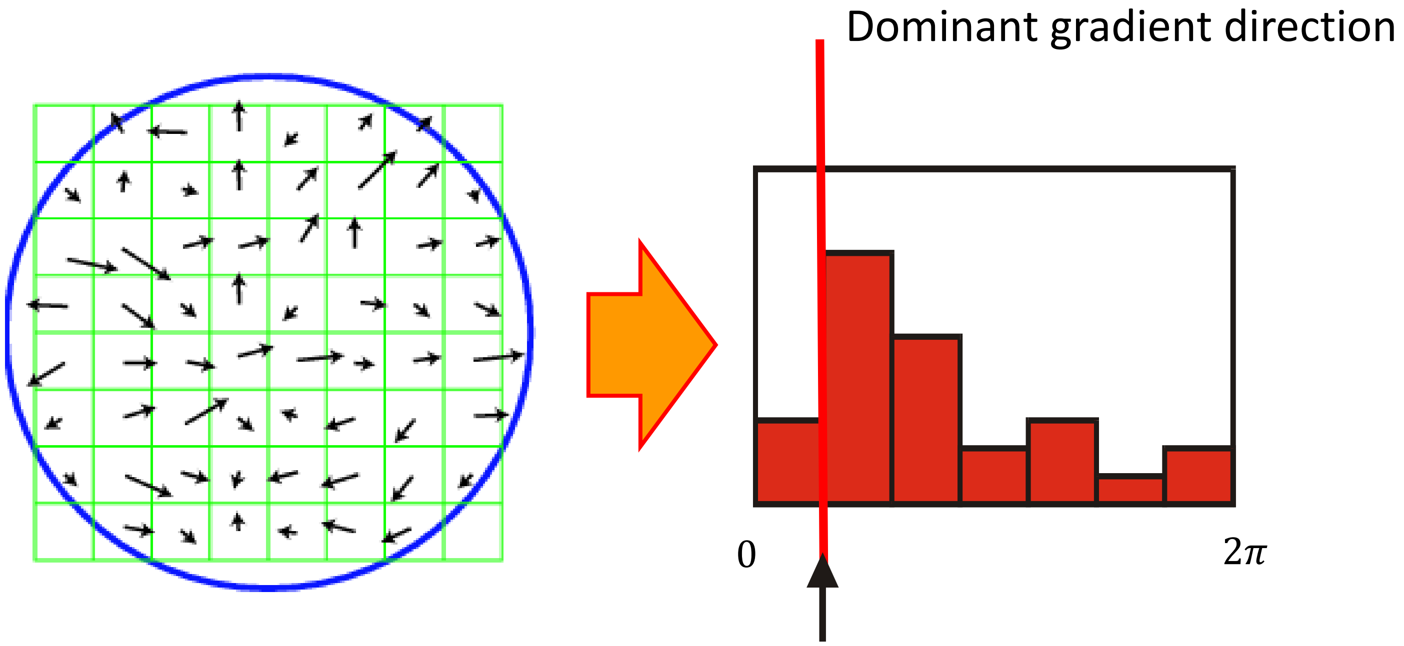
Derotation:

- **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**



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**



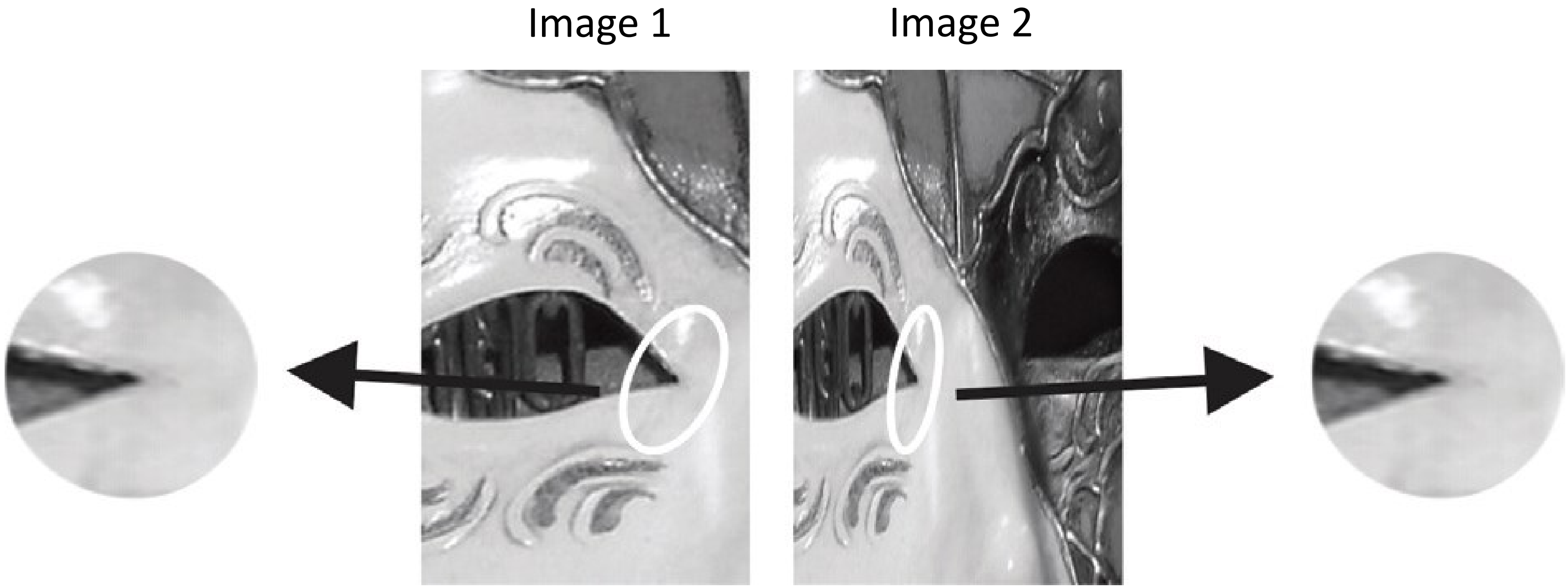
Outline

- How to achieve descriptor invariance to:
  - Scale
  - Rotation
  - Viewpoint
- The SIFT blob detector and descriptor
- Other corner and blob detectors and descriptors

How to achieve invariance to small viewpoint changes?

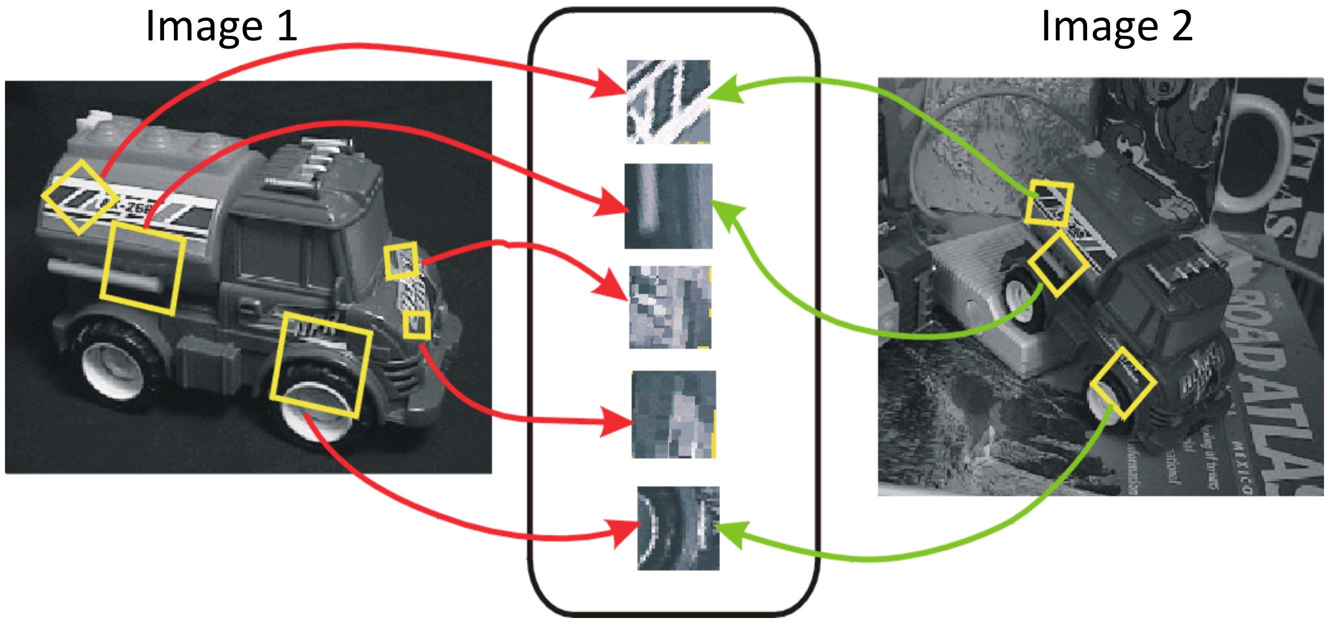
Affine warping provides invariance to small view-point changes

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



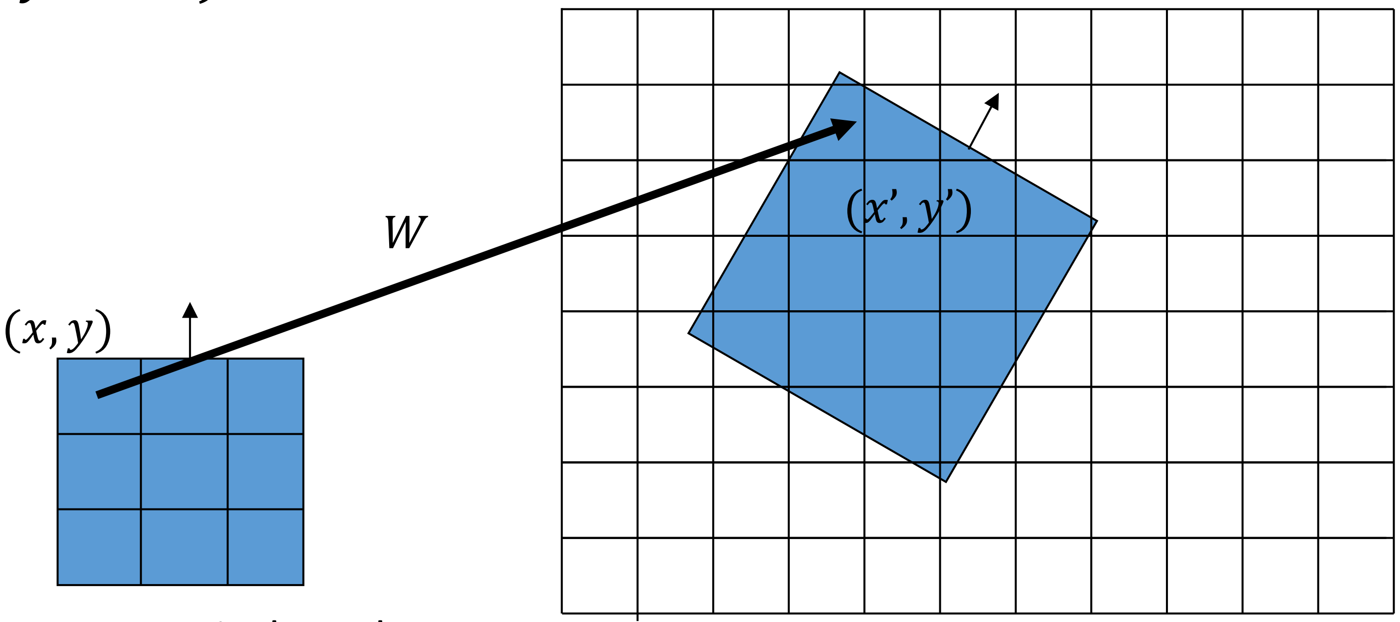
How to warp a patch?

- Start with an “**empty**” canonical patch (all pixels set to 0)
- For each pixel (x, y) in the empty patch, apply the **warping function** $\boldsymbol{W(x, y)}$ to compute the corresponding position in the source image. It will be in floating point and will fall between the image pixels.
- **Interpolate** the intensity values of the 4 closest pixels in the detected image using either of:
  - *Nearest neighbor interpolation*
  - *Bilinear interpolation*
  - *Bicubic interpolation*

Example: Similarity Transformation (rotation, translation, rescaling)

- Warping function W: rotation (θ) plus rescaling (s) and translation (a, b):

$$x' = s(x\cos\theta - y\sin\theta) + a$$
$$y' = s(x\sin\theta + y\cos\theta) + b$$



Empty canonical patch

Patch detected in the image

Example: Rescaling

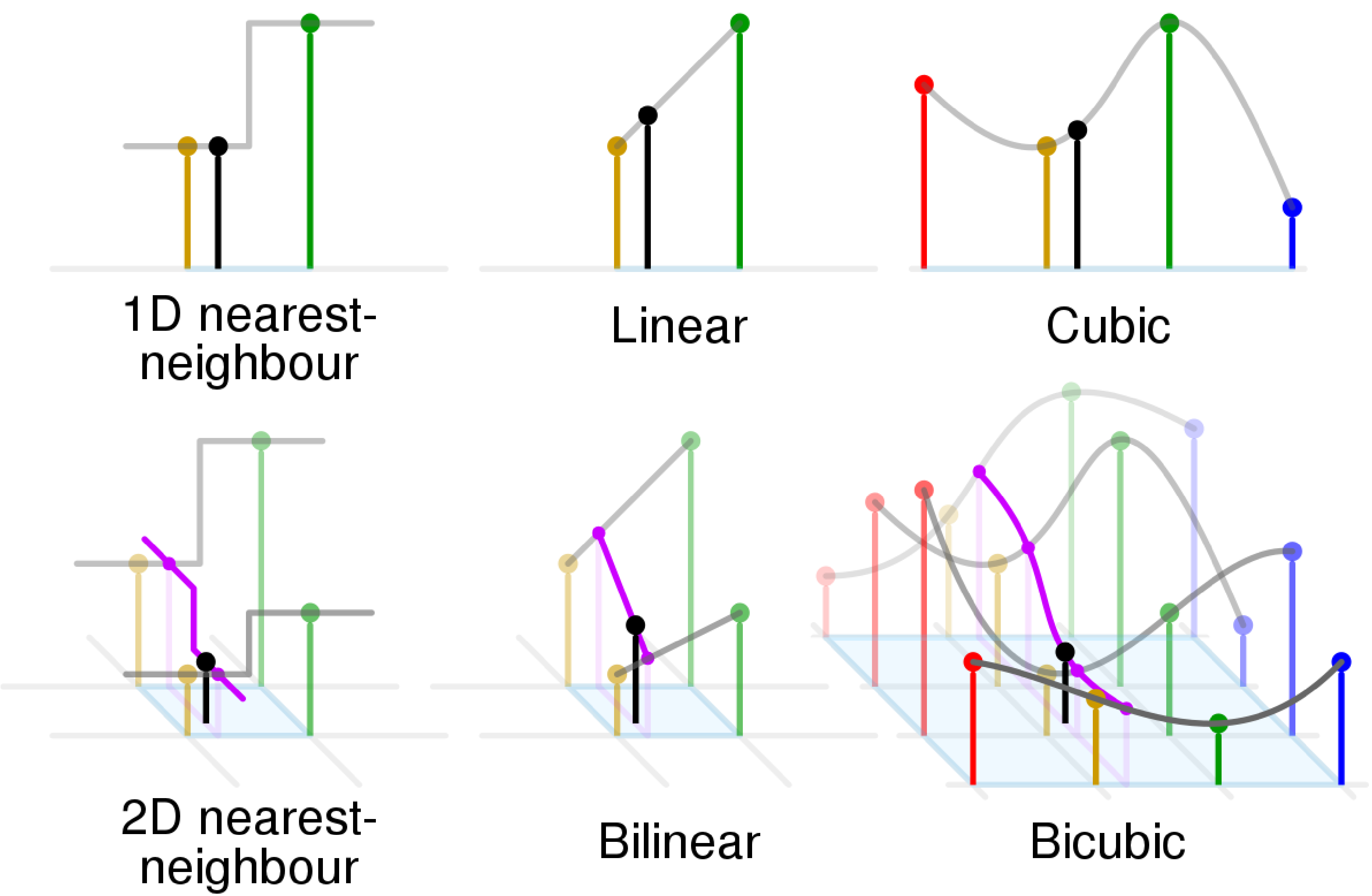
Original image:  x 10

Nearest-neighbor interpolation

Bilinear interpolation

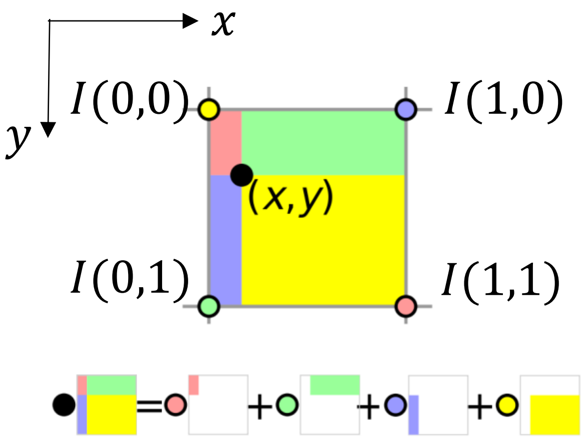
Bicubic interpolation

Nearest Neighbor vs Bilinear vs Bicubic Interpolation



Bilinear Interpolation

- It is an ***extension of linear interpolation*** for interpolating functions of two variables (e.g., x and y) on a *rectilinear 2D grid*.
- The key idea is to **perform linear interpolation first in one direction**, and **then again in the other direction**.
- Although each step is linear in the sampled values and in the position, the interpolation as a whole is **not linear but rather quadratic** in the sample location.



$$I(x,y) = \begin{aligned} & I(0,0)(1-x)(1-y) + \\ & I(0,1)(1-x)(y) + \\ & I(1,0)(x)(1-y) + \\ & I(1,1)(x)(y) \end{aligned}$$

This formula won't be asked at the exam ☺

In this geometric visualization, the value at the black spot is the sum of the value at each colored spot multiplied by the area of the rectangle of the same color.

Disadvantage of Patch Descriptors

- **Disadvantage of patch descriptors**:
  - If the warp is not estimated accurately, very **small errors** in rotation, scale, and view-point will **affect matching score significantly**
  - **Computationally expensive** (need to unwarp every patch)

Outline

- Automatic Scale Selection
- The SIFT blob detector and descriptor
- Other corner and blob detectors and descriptors

SIFT Descriptor

- **S**cale **I**nvariant **F**eature **T**ransform
- Invented by David Lowe in 2004

David Lowe

FOLLOW

Professor Emeritus, Computer Science Dept., University of British Columbia
Verified email at cs.ubc.ca - Homepage

Computer Vision   Object Recognition

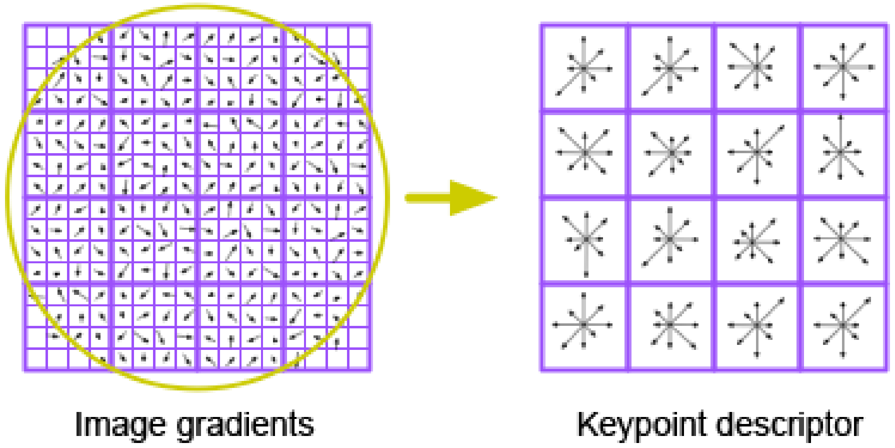
TITLE	CITED BY	YEAR
Distinctive image features from scale-invariant keypoints DG Lowe International journal of computer vision 60 (2), 91-110	76478	2004

Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", Internal Journal of Computer Vision, 2004. PDF

SIFT Descriptor

Descriptor computation:

- Consider a $\mathbf{16 \times 16}$ **pixel patch**
- Multiply the patch by a **Gaussian filter,** compute **dominant orientation,** and **de-rotate patch**
- Compute **HOG descriptor**
  - Divide patch into **4×4 cells**
  - Use **8 bin histograms** (, i.e., 8 directions)
  - **Concatenate all histograms** into a single 1D vector
  - Resulting SIFT descriptor: **4×4×8 = 128 float values**
- Descriptor Matching: SSD (i.e., Euclidean-distance)
- Why 4×4 cells and why 8 bins? See later



Is HOG invariant to additive or affine illumination changes (i.e., $I'(x,y) = \alpha I(x,y) + \beta$)?

Descriptor Normalization

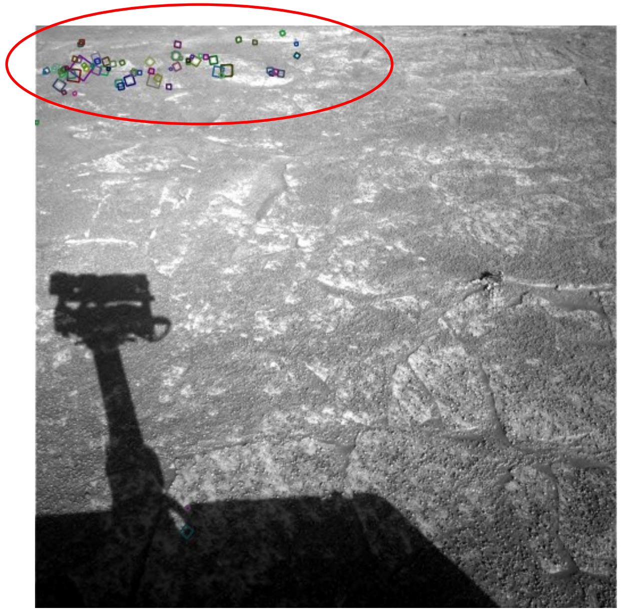
- The HOG descriptor is **invariant to additive illumination** because it is based on gradients
- To make it **invariant affine illumination changes**, the descriptor vector $\boldsymbol{v}$ is then normalized such that its L_2 norm is 1:

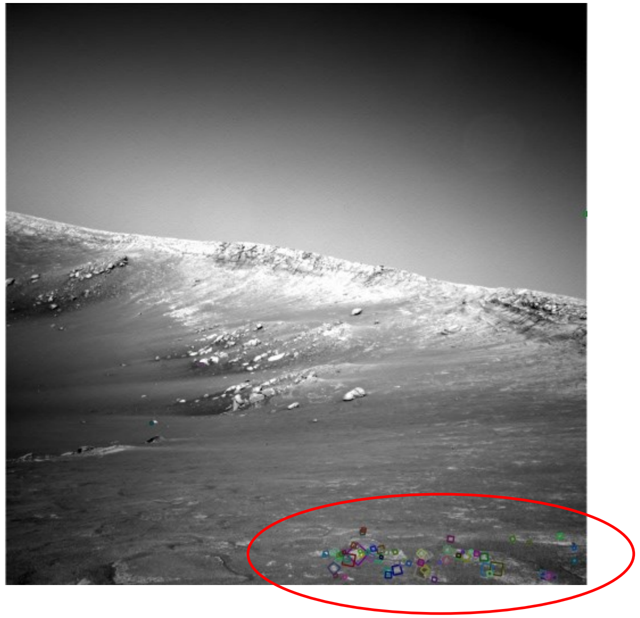
$$\overline{\boldsymbol{v}} = \frac{\boldsymbol{v}}{\sqrt{\sum_i^n v_i^2}}$$

- We can conclude that the **SIFT descriptor is invariant to affine illumination changes**

SIFT Matching Robustness

- Can handle severe **viewpoint changes** (**up to 50 degree** out-of-plane rotation)
- Can handle even **severe non affine changes in illumination** (low to bright scenes)
- More computationally **expensive than the patch descriptor**
- Original SIFT binary files: http://people.cs.ubc.ca/~lowe/keypoints
- OpenCV C/C++ implementation: https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html

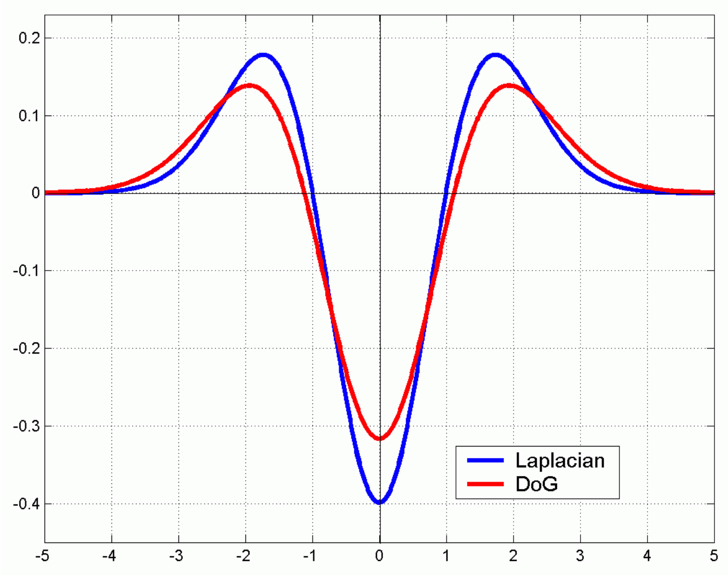




SIFT Detector

- SIFT uses the **Difference of Gaussian (DoG) kernel** instead of Laplacian of Gaussian (LoG) because computationally cheaper

$$LOG(x,y) \approx DoG(x,y) = G_{k\sigma}(x,y) - G_{\sigma}(x,y)$$

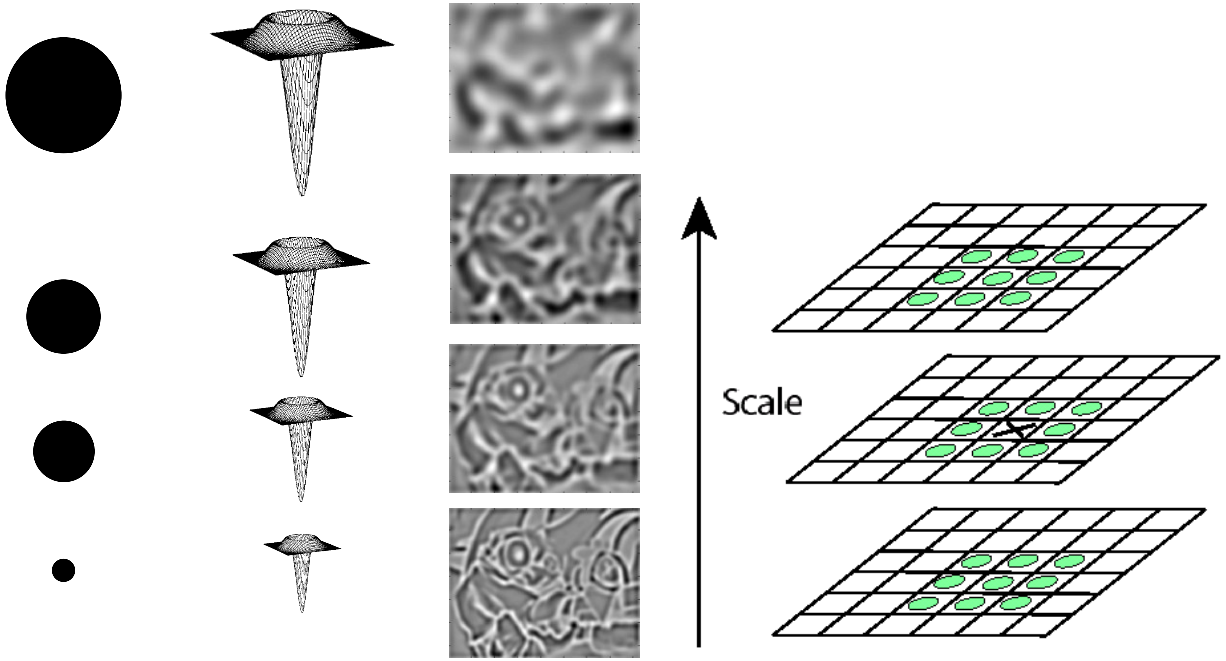


- The proof that LoG can be approximated by a difference of Gaussian comes from the Heat Equation: $\frac{\partial G_\sigma}{\partial \sigma} = \sigma\, \nabla^2 G_\sigma$

SIFT Detector (location + scale)

SIFT keypoints: **local extrema** in **both space and scale** of the **DoG images**

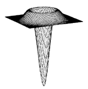
- **Each pixel is compared to 26 neighbors** (below in green): its 8 neighbors in the current image + 9 neighbors in the adjacent upper scale + 9 neighbors in the adjacent lower scale
- If the pixel is a global maximum or minimum (i.e., extrema) with respect to its 26 neighbors then it is selected as SIFT feature

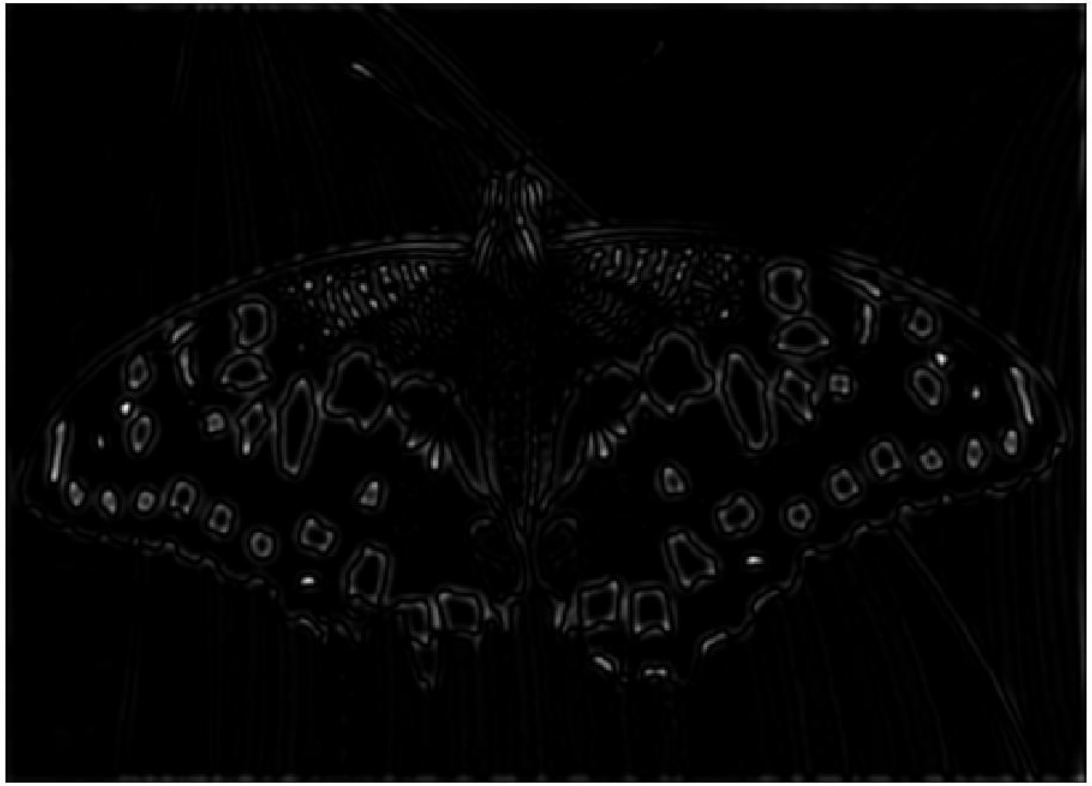


For each extrema, the output of the SIFT detector is the **location** (x, y) and the **scale** s

Example

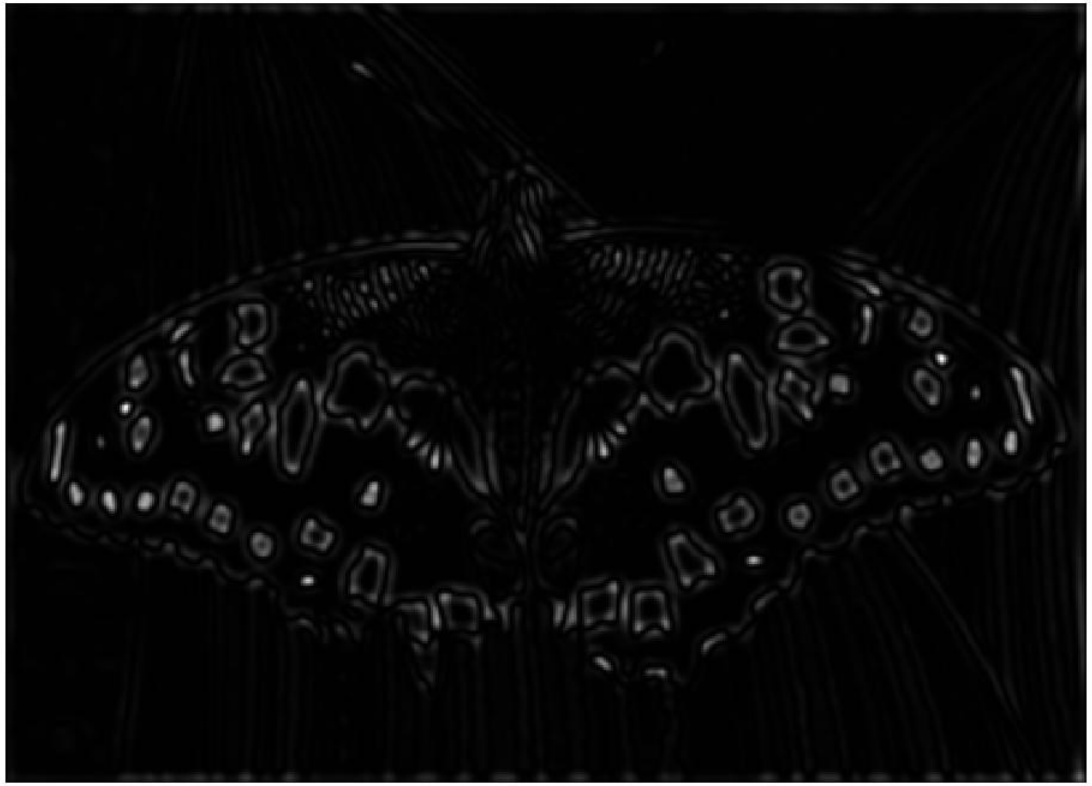
DoG Images example

- 



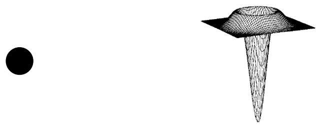
Magnitude of $(G(k\sigma) - G(\sigma))$ | $s = 4; \sigma = 1.6$ |

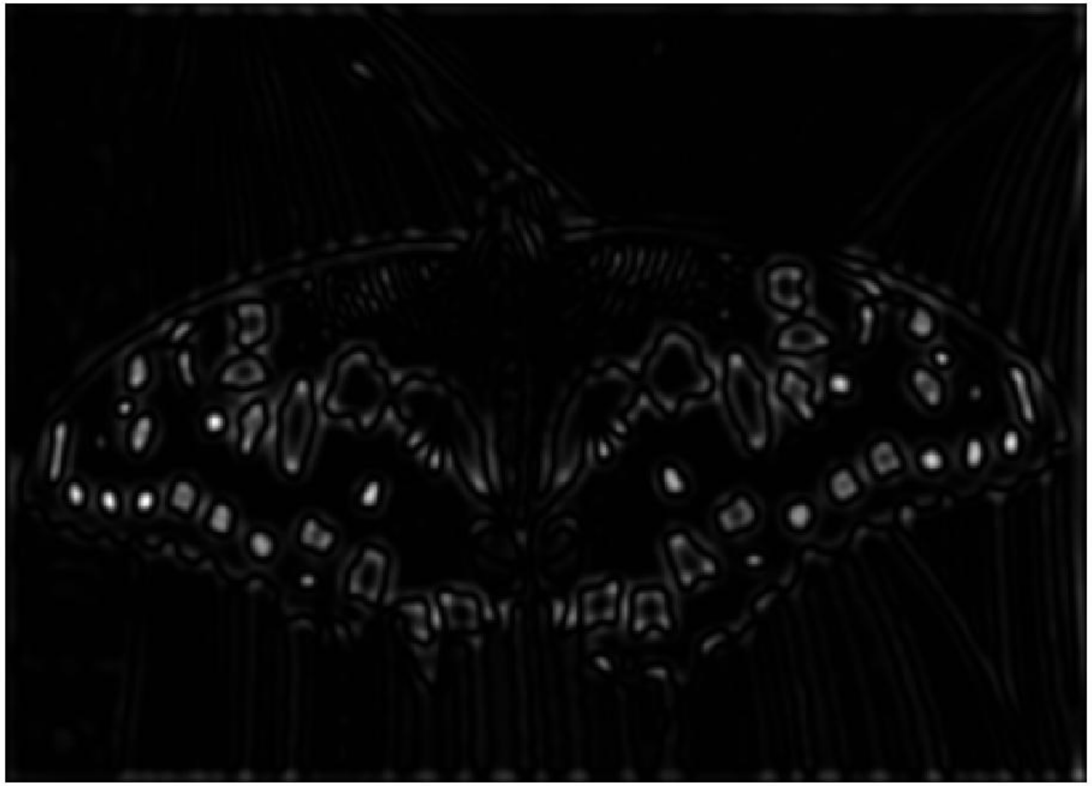
DoG Images example



Magnitude of $(G(k^2\sigma) - G(k\sigma))$| $s = 4; \sigma = 1.6$ |

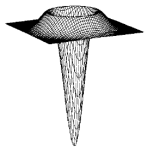
DoG Images example

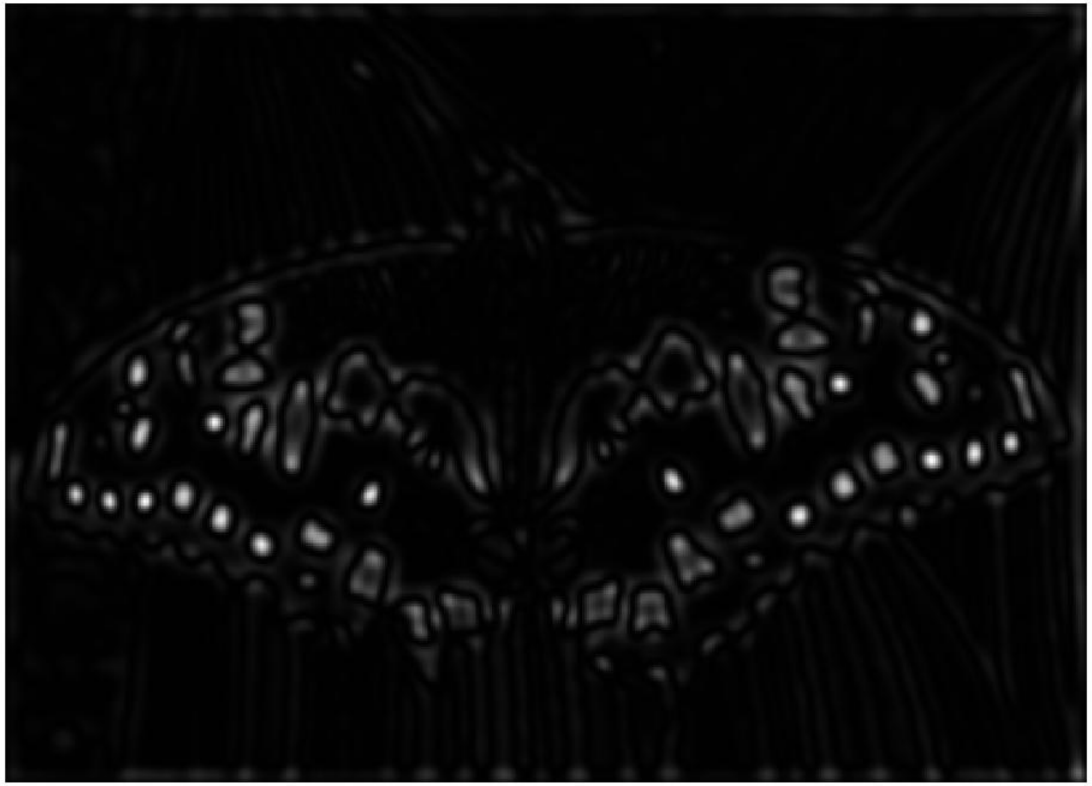




Magnitude of $(G(k^3\sigma) - G(k^2\sigma))|\ s = 4; \sigma = 1.6\ |$

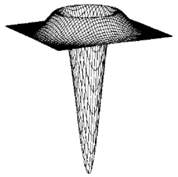
DoG Images example





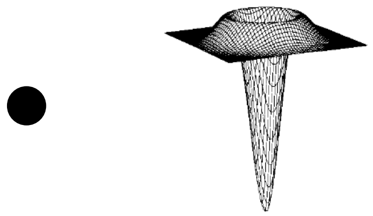
Magnitude of $(G(k^4\sigma) - G(k^3\sigma))$ | $s = 4; \sigma = 1.6$ |

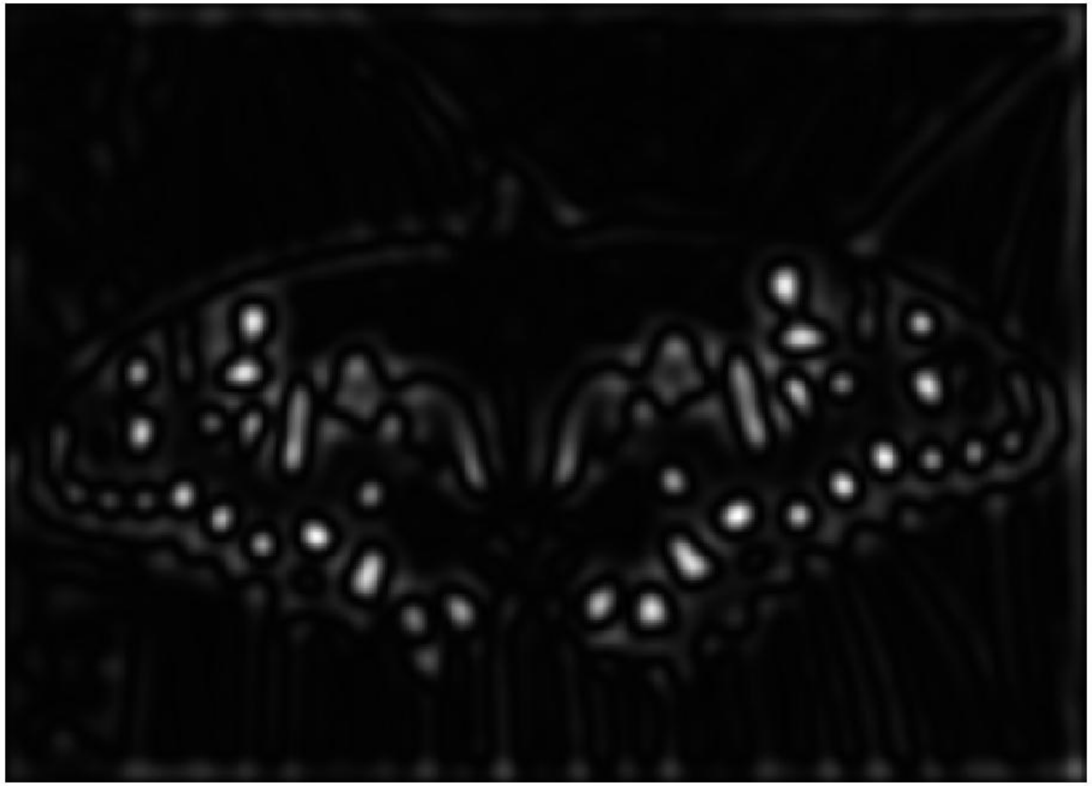
DoG Images example



Magnitude of $(G(k^5\sigma) - G(k^4\sigma))$ | $s = 4; \sigma = 1.6$ |
(second octave shown at the input resolution for convenience)

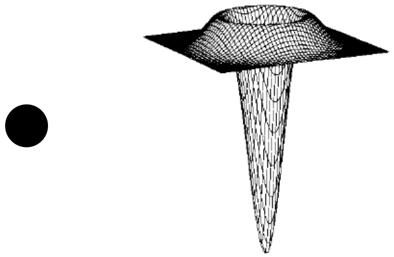
DoG Images example





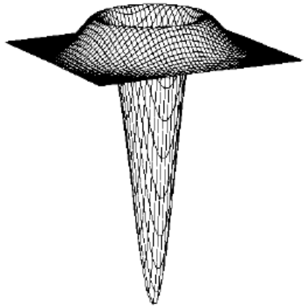
Magnitude of $(G(k^6\sigma) - G(k^5\sigma))$ | $s = 4; \sigma = 1.6$ |
(second octave shown at the input resolution for convenience)

DoG Images example



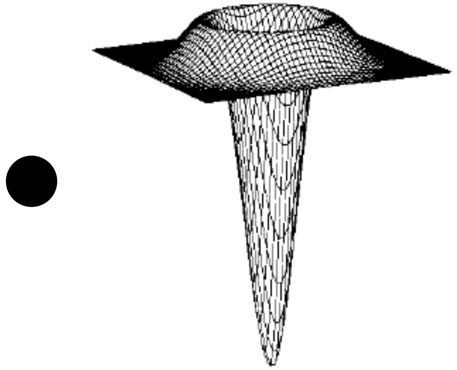
Magnitude of $(G(k^7\sigma) - G(k^6\sigma))$ | $s = 4; \sigma = 1.6$ |
(second octave shown at the input resolution for convenience)

DoG Images example



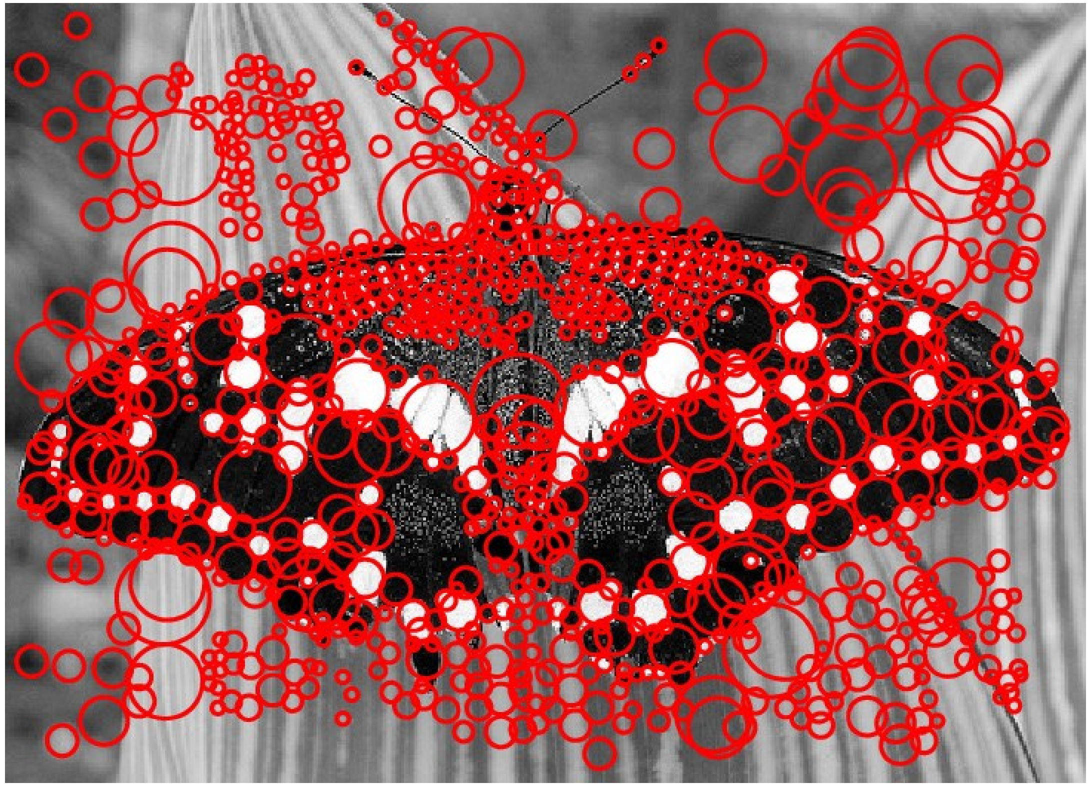
Magnitude of $(G(k^8\sigma) - G(k^7\sigma))$ | $s = 4; \sigma = 1.6$ |
(second octave shown at the input resolution for convenience)

DoG Images example



Magnitude of $(G(k^9\sigma) - G(k^8\sigma))$ | $s = 4; \sigma = 1.6$ |
(third octave shown at the input resolution for convenience)

Local extrema of DoG images across Scale and Space



What are SIFT features like?
Hint: Remember the definition of filtering as template matching

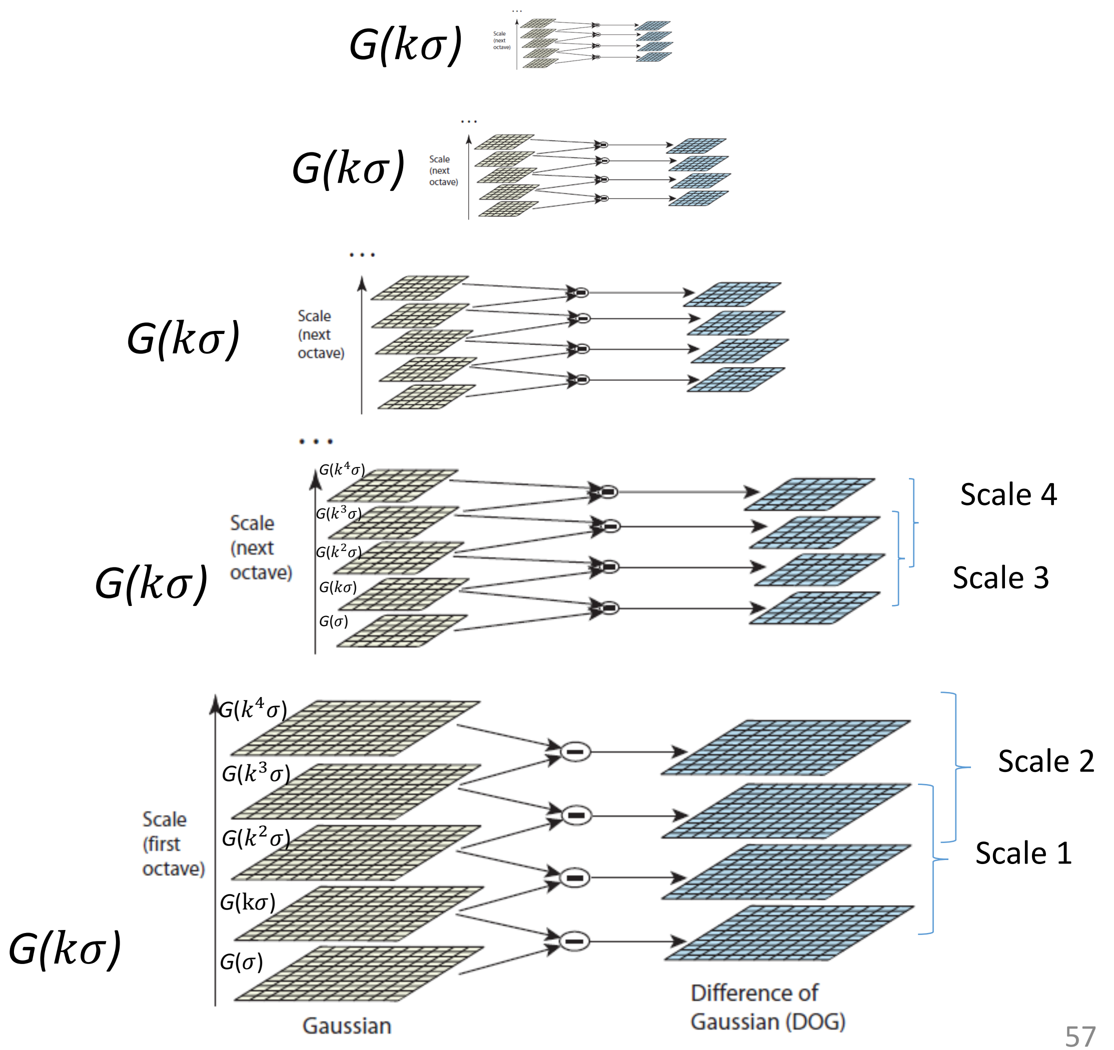
How it is implemented in practice

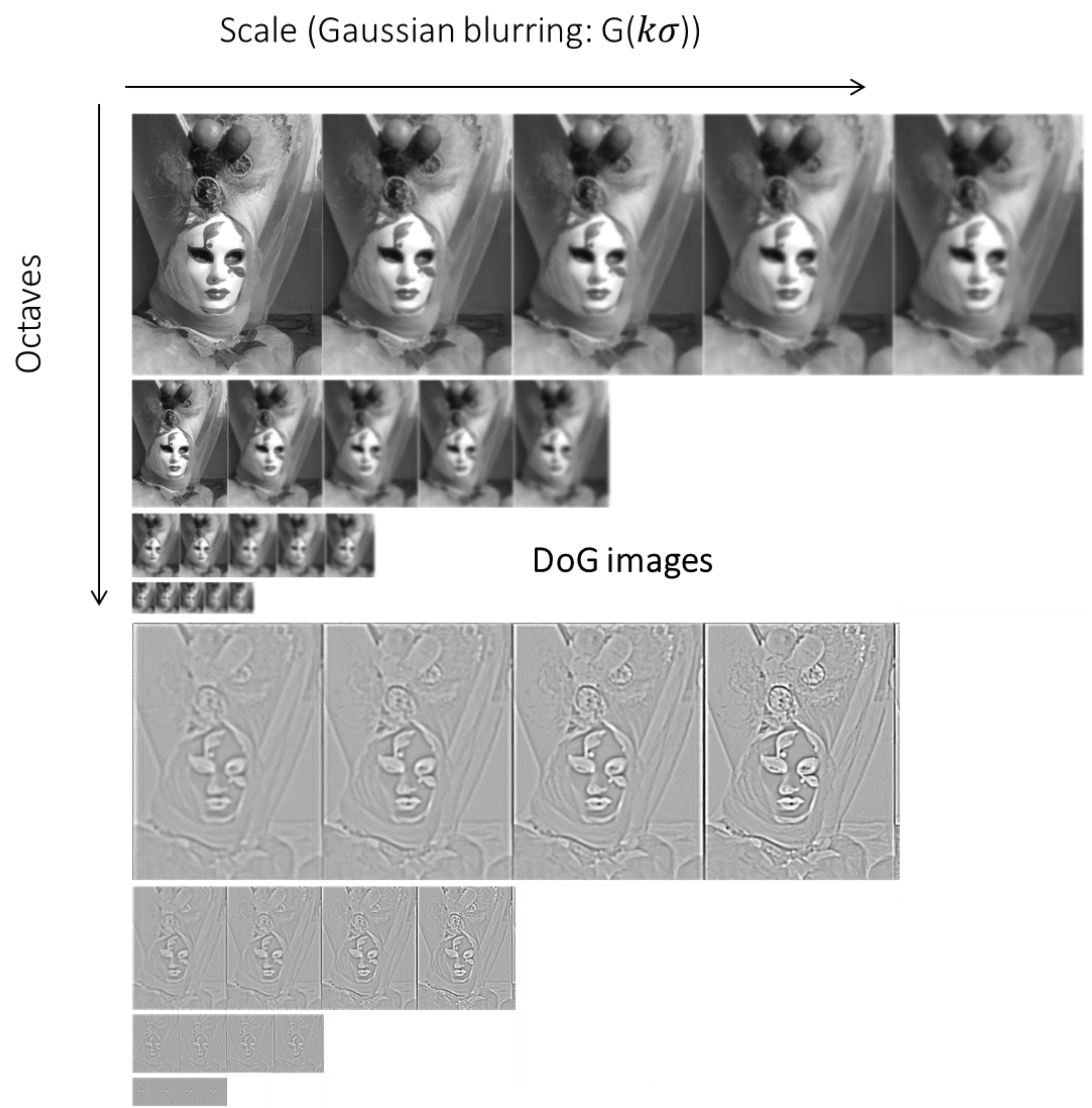
1. Build a Space-Scale Pyramid:

- The initial image is incrementally convolved with Gaussians $G(k^i\sigma)$ to produce blurred images separated by a constant factor k in scale space (shown stacked in the left column).
  - The initial Gaussian G(σ) has σ=1.6
  - k is chosen: $k = 2^{1/s}$, where s is the number of intervals into which each octave of scale space is divided
  - For efficiency reasons, when k^i equals 2, the image is downsampled by a factor of 2 and then the procedure is repeated again up to 5 octaves (pyramid levels)
- Adjacent blurred images are then subtracted to produce the **Difference-of-Gaussian (DoG)** images

2. Scale-Space extrema detection

- Detect local maxima and minima in space-scales (see previous slide)




Scale (Gaussian blurring: G($k\sigma$))
Octaves
DoG images

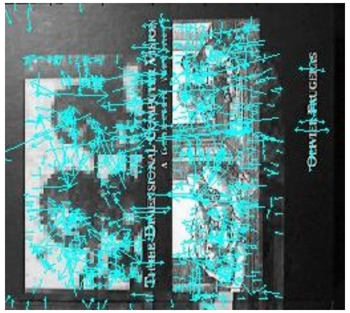
SIFT: Recap

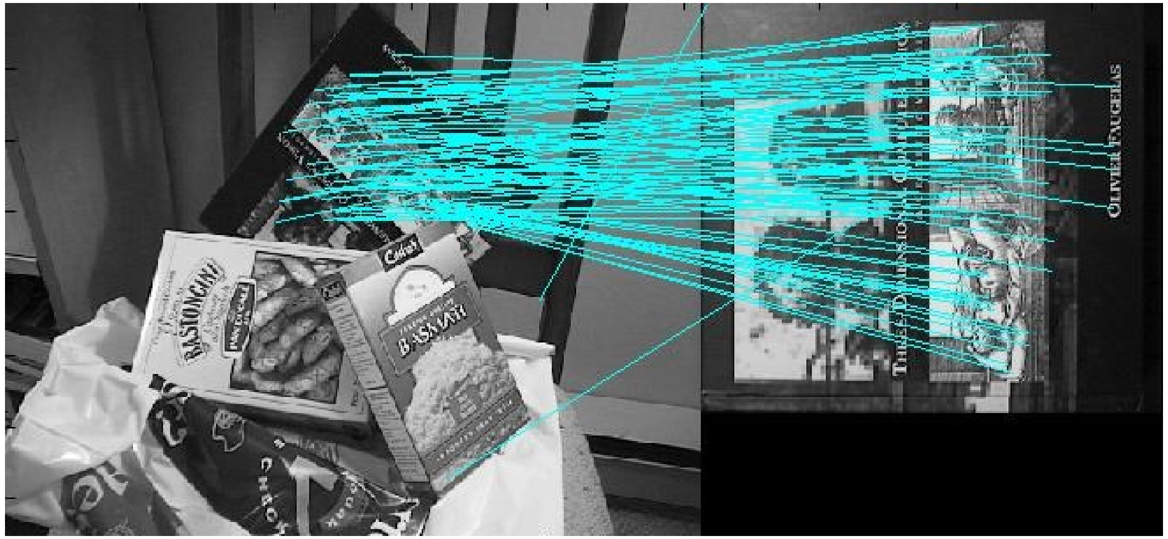
- SIFT: **Scale Invariant** Feature Transform
- An approach to **detect and describe** regions of interest in an image.
  - SIFT detector = **DoG detector**
- SIFT features are **invariant to 2D rotation**, and reasonably invariant to **rescaling**, **viewpoint changes** (up to 50 degrees), and **illumination**
- It runs in real-time but **expensive (**10 Hz on an i7 laptop)
  - The expensive steps are the **scale detection and descriptor extraction**

Original SIFT Demo by David Lowe

Download original SIFT binaries and Matlab function from :
http://people.cs.ubc.ca/~lowe/keypoints

```
>>[image1, descriptor1s, locs1] = sift('scene.pgm');
>>showkeys(image1, locs1);
>>[image2, descriptors2, locs2] = sift('book.pgm');
>>showkeys(image2, locs2);
>>match('scene.pgm','book.pgm');
```





What’s the output of SIFT?

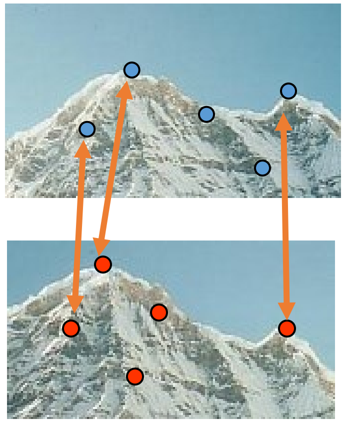
SIFT outputs N features, each one being a data structure containing:

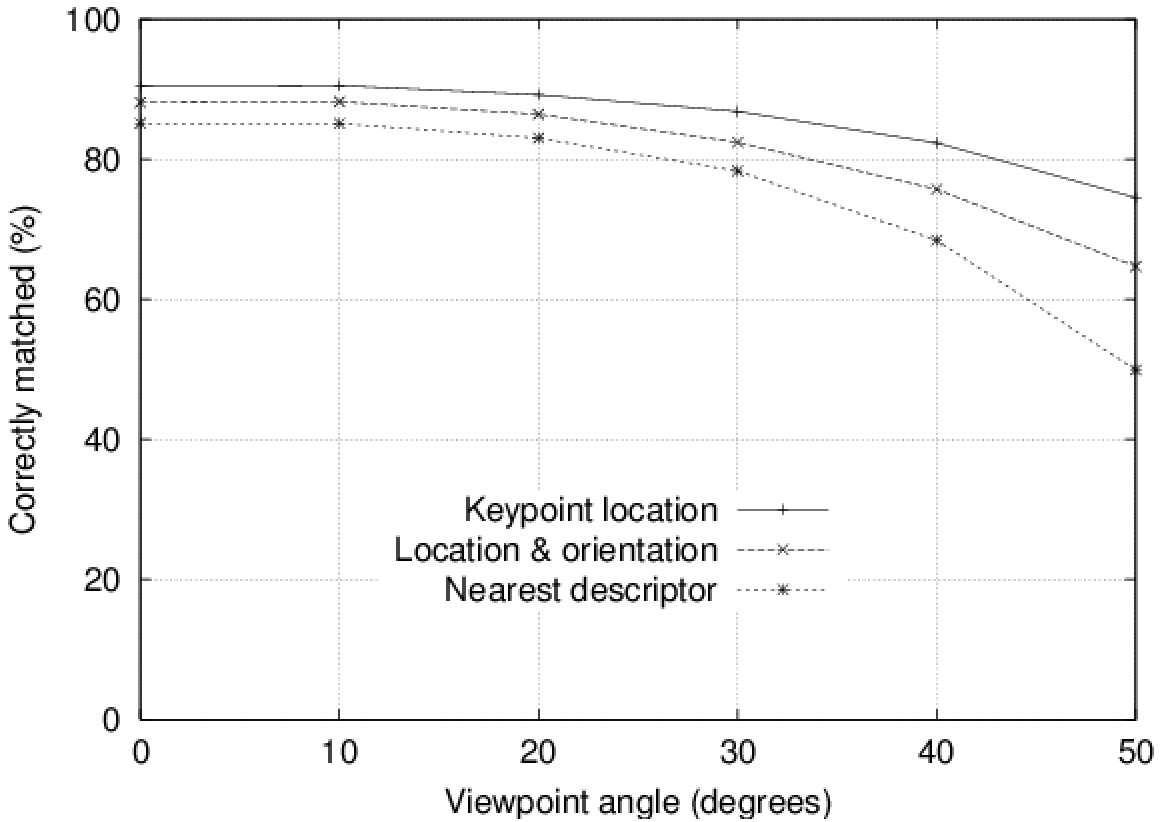
- **Descriptor**: 4x4x8 = 128-element vector
- **Location** (pixel coordinates of the center of the patch): 2-element vector
- **Scale** (i.e., size) of the patch: 1 scalar value
- **Orientation** (i.e., angle of the patch): 1 scalar value

SIFT Repeatability with Viewpoint Changes

Repeatability=

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



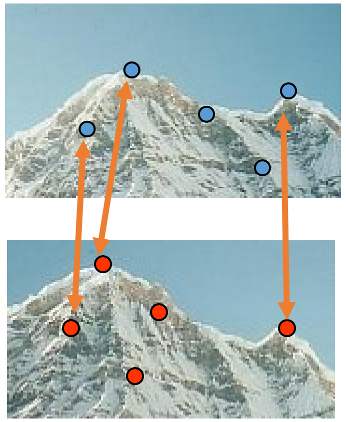


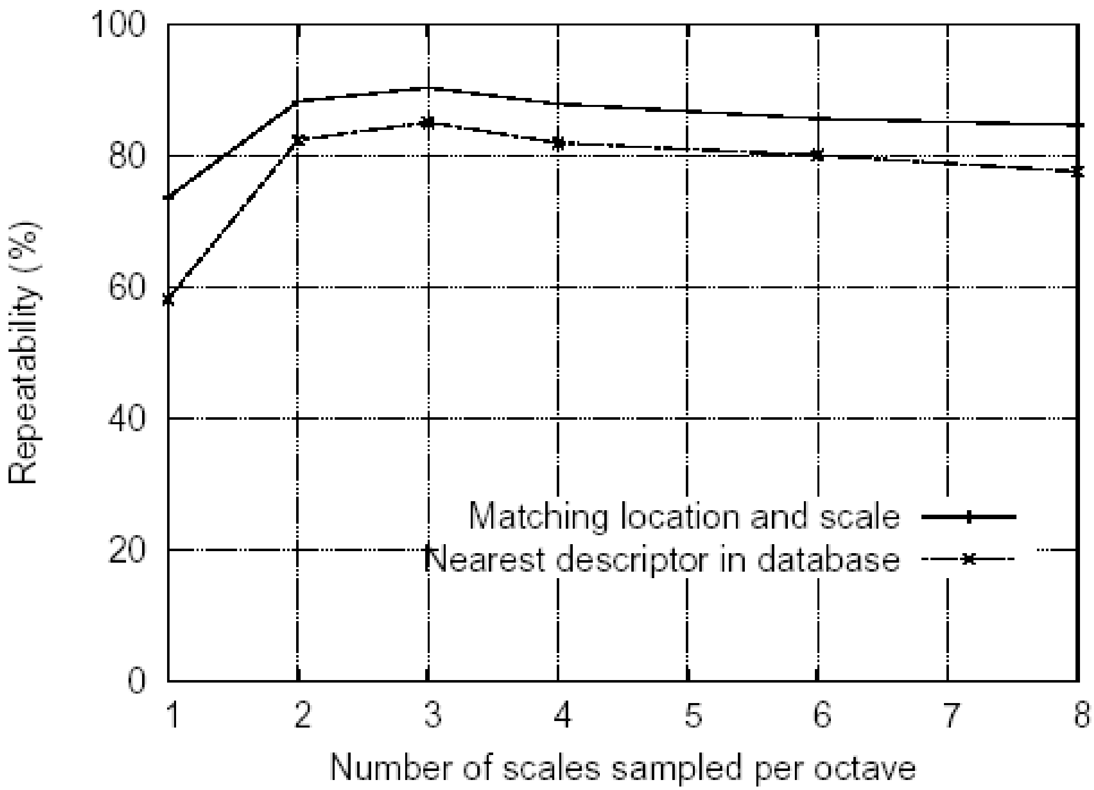
Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", Internal Journal of Computer Vision, 2004. [PDF]

SIFT Repeatability with Number of Scales per Octave

Repeatability=

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

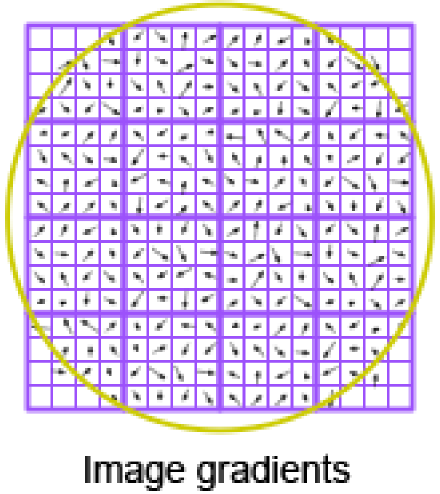




Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", Internal Journal of Computer Vision, 2004. PDF

Influence of Number of Orientations and Number of Sub-patches

The graph shows that a single orientation histogram (n = 1) is very poor at discriminating. The results improve with a 4x4 array of histograms with 8 orientations.



4x4 HOGs

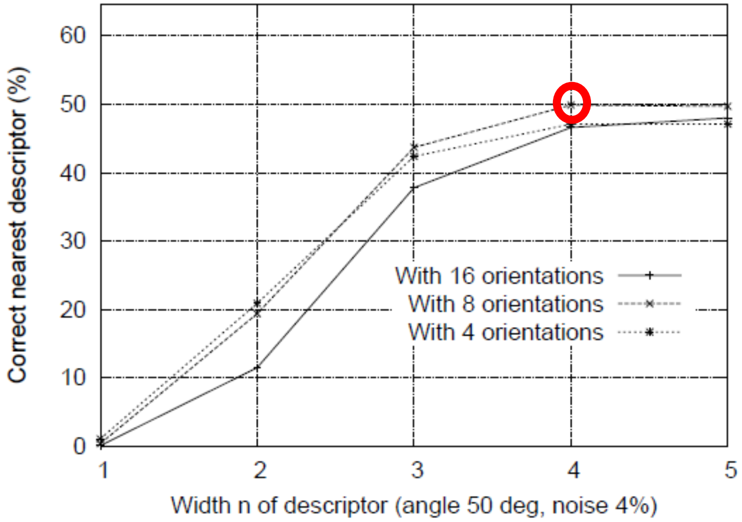
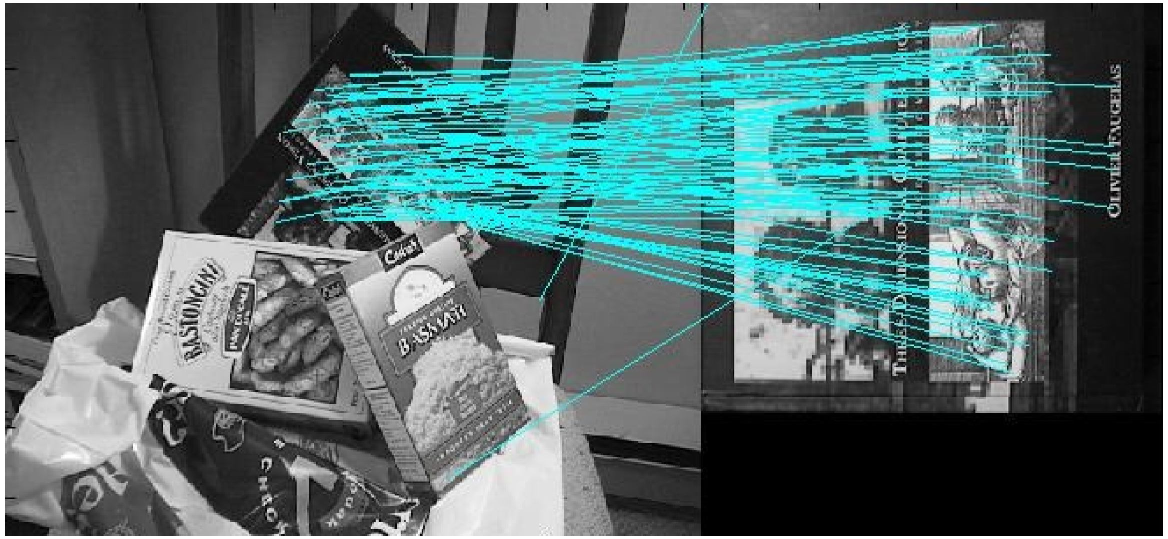


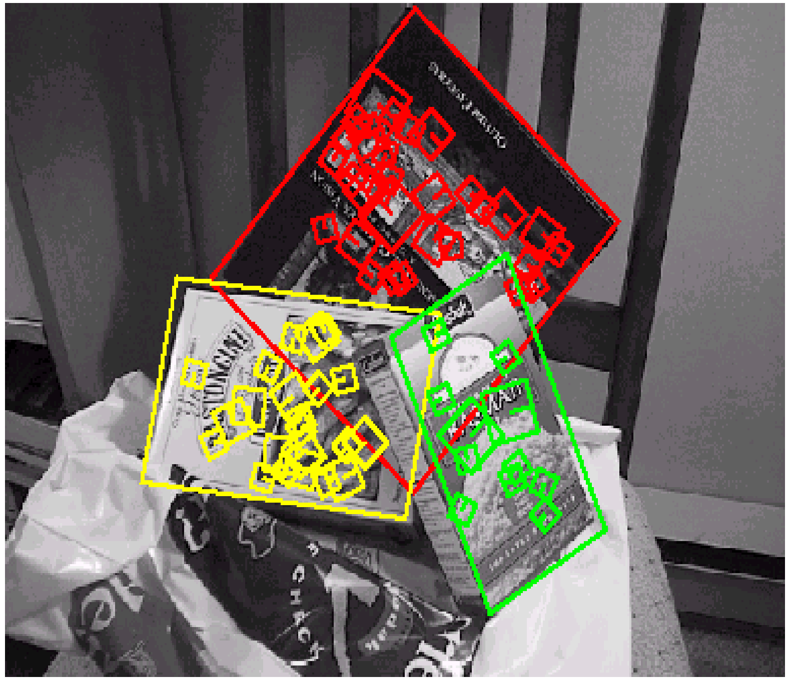
Figure 8: This graph shows the percent of keypoints giving the correct match to a database of 40,000 keypoints as a function of width of the $n \times n$ keypoint descriptor and the number of orientations in each histogram. The graph is computed for images with affine viewpoint change of 50 degrees and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

Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", Internal Journal of Computer Vision, 2004. PDF

Application of SIFT to Object recognition

- Can be implemented easily by returning the object with the largest number of correspondences shared with the template image
- For planar objects, 4-point RANSAC can be used to remove outliers (see Lecture 9).
- For rigid 3D objects, 5-point RANSAC (see Lecture 9).





Application of SIFT to Panorama Stitching

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

How many parameters can we tune for SIFT and Harris?

SIFT

1. Patch size (e.g., 16x16)
2. Sigma of Gaussian to transform the square into a circular patch
3. Number of subpatches (e.g., 4x4)
4. Number of histogram bins (e.g., 8)
5. Threshold to choose dominant orientations
6. Number of octaves (e.g., 5)
7. Number of scales per octave (e.g., 3)
8. Sigma_0 (e.g., 1.6)
9. Distance ratio (e.g., 0.8)

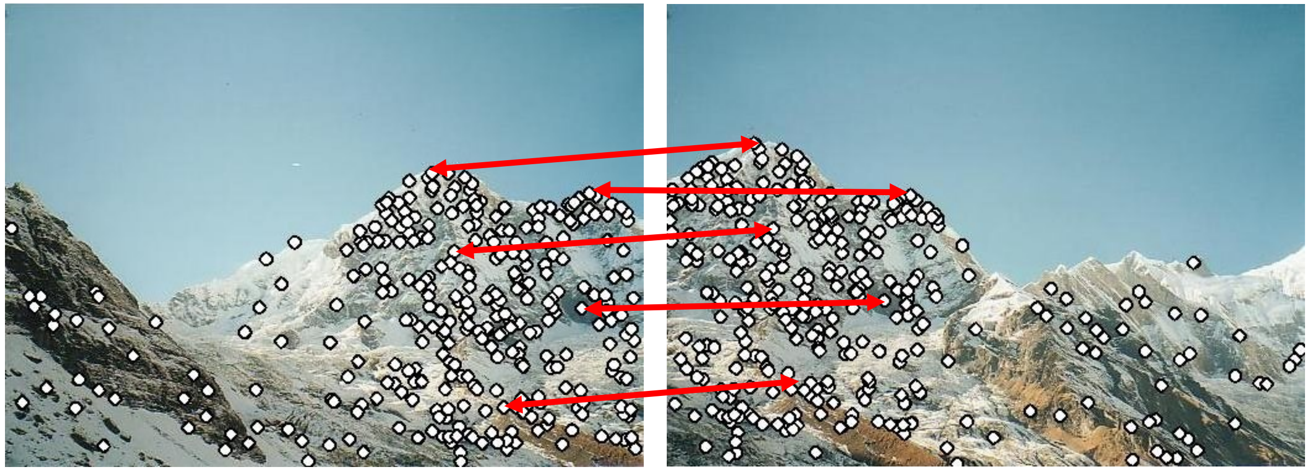
HARRIS

1. Patch size (the larger the size the smaller the sensitivity to fine details)
2. Sigma of Gaussian to transform the square into a circular patch
3. Cornerness response magic number (e.g., 0.04-0.15 for Harris detector)
4. Cornerness response threshold
5. Size of non-maxima suppression: the larger the size the fewer the local maxima (i.e., corners)
6. grid size (image bucketing)

Main questions

- What features are *repeatable* and *distinctive?*
- How to *describe* a feature?
- How to establish *correspondences*, i.e., compute matches?

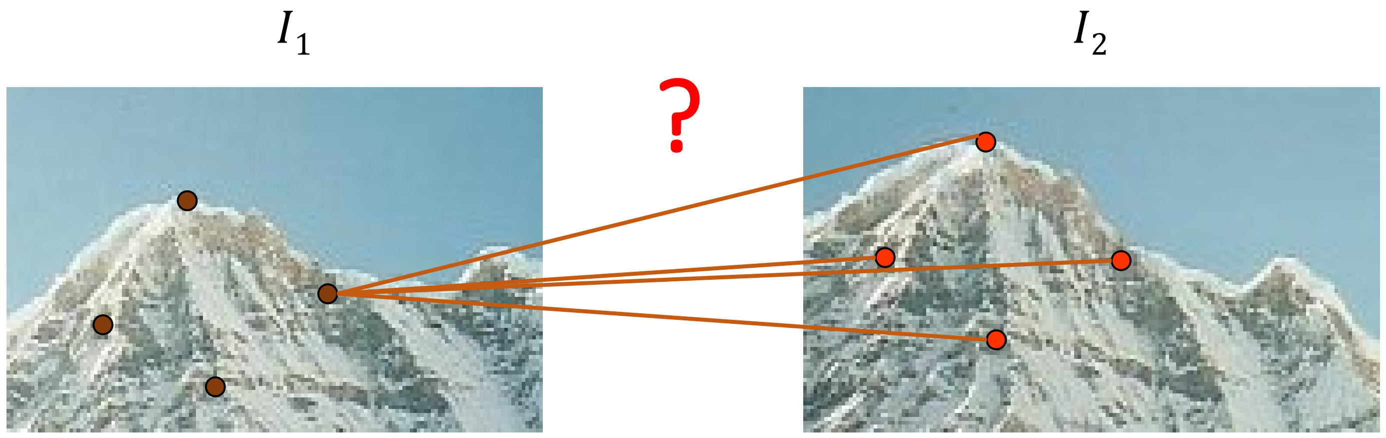
Feature Matching



?

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 ((Z)SSD, (Z)SAD, (Z)NCC or Hamming distance for binary descriptors (e.g., Census, HOG, ORB, BRIEF, BRISK, FREAK)
  2. **Brute-force matching:**
     1. Compare each feature in I_1 against all the features in I_2 (N^2 comparisons, where N is the number of features in each image)
     2. Take the one at minimum distance, i.e., the **closest descriptor**



Feature Matching

- **Issues with closest descriptor**: it returns a match also when the true match is absent
- **Better approach**: add constraint on the ratio between distances from 1st to 2nd closest descriptor:

$$\frac{d_1}{d_2} < Threshold\ (usually\ 0.8)$$

where:

d_1 is the distance from the closest descriptor
d_2 is the distance from the 2nd closest descriptor

Distance Ratio: Explanation

- In SIFT, the nearest neighbor is defined as the keypoint with minimum Euclidean distance. There can be two possible issues if we only take the match with closest descriptor:
    1. Some features in Image 1 may not have a correct match in Image 2 because they arise from background clutter or were not detected in Image 1.
    2. There are multiple matching features in Image 2. How do we solve these ambiguities? Should we just pick the closest descriptor or do we discard them a priori.
- An effective measure is obtained by comparing the distance from the **closest neighbor** to the distance from the **second-closest neighbor**. This measure performs well because correct matches need to have the closest neighbor significantly closer than the closest incorrect match to achieve reliable matching.
- For false matches, there will likely be a number of other false matches within similar distances due to the high dimensionality of the feature space (this problem is known as **curse of dimensionality**).
- We can think of the distance from the **second-closest match as** providing **an estimate of the density of false matches** within this portion of the feature space (case 1) and, at the same time, identifying **specific instances of feature ambiguity** (case 2).

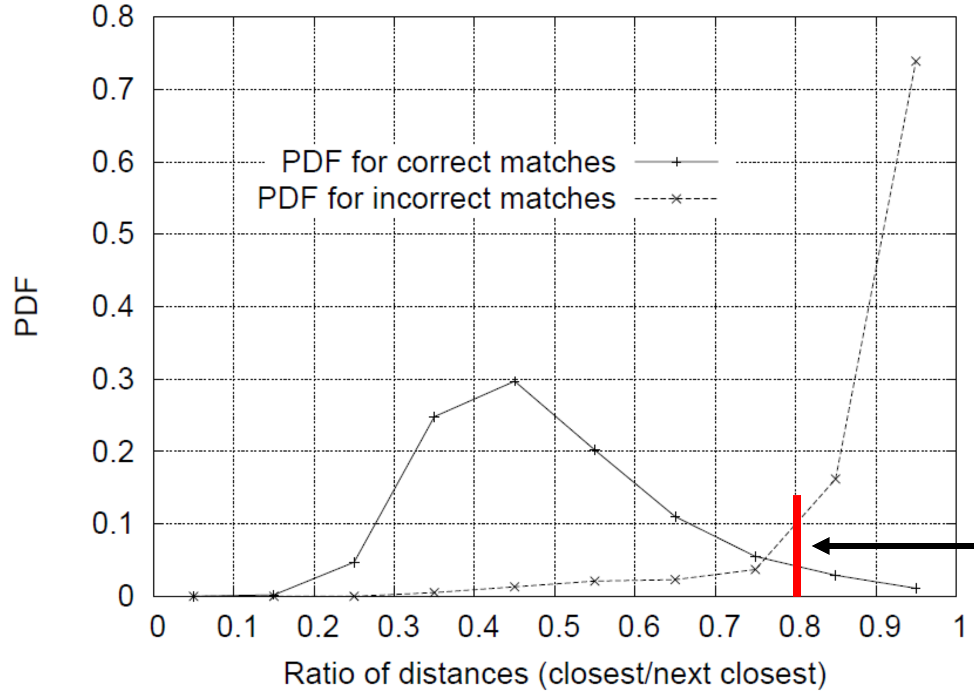
SIFT Feature Matching: Distance Ratio

The SIFT paper recommends to use a threshold on 0.8: $\frac{d_1}{d_2} < 0.8$

Where does this magic number come from?

"A threshold of 0.8 eliminates 90% of the incorrect matches while discarding less than 5% of the correct matches."

"This figure was generated by matching images following random scale and orientation change, with viewpoint change of 30 degrees, and addition of 2% image noise, against a database of 40,000 keypoints."



0.8 separates the areas under the curves (left to right)

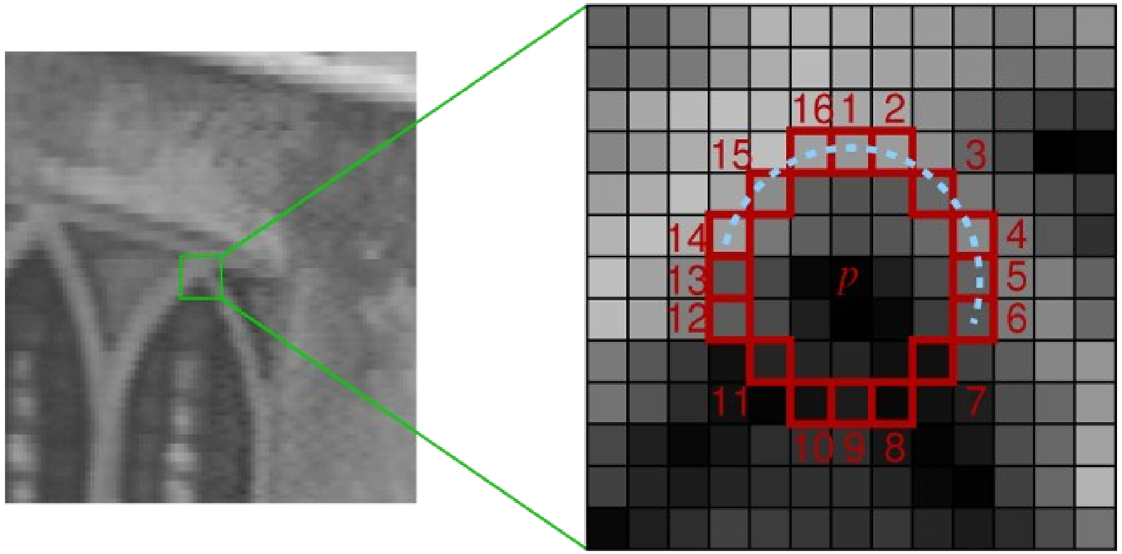
Figure 11: The probability that a match is correct can be determined by taking the ratio of distance from the closest neighbor to the distance of the second closest. Using a database of 40,000 keypoints, the solid line shows the PDF of this ratio for correct matches, while the dotted line is for matches that were incorrect.

Outline

- Automatic Scale Selection
- The SIFT blob detector and descriptor
- Other corner and blob detectors and descriptors

“FAST” Corner Detector

- **FAST**: **F**eatures from **A**ccelerated **S**egment **T**est
- Analyses intensities along a ring of 16 pixels centered on the pixel of interest $\boldsymbol{p}$
- $\boldsymbol{p}$ is a FAST corner **if** a set of ***N*** contiguous pixels on the ring are:
  - all brighter than the pixel intensity $\boldsymbol{I(p) + threshold}$,
  - or all darker than $\boldsymbol{I(p) - threshold}$
- Common value of ***N*: 12**
- A simple **classifier** is used to **check the quality** of corners and reject the weak ones
- **FAST is the fastest corner detector ever made:** can process 100 million pixels per second (**<3ms per VGA image**)
- **Issue**: it is very **sensitive to image noise** (high in low light). This is why Harris is still more common despite a bit slower
- In fact, FAST was initially proposed to **find candidate corner regions to scout with the Harris detector**



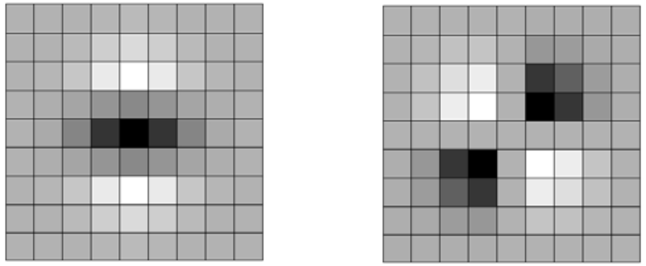
Rosten, Drummond, Fusing points and lines for high performance tracking, International Conference on Computer Vision (ICCV), 2005. PDF.

Rosten, Porter, Drummond, “Faster and better: a machine learning approach to corner detection”, IEEE Trans. Pattern Analysis and Machine Intelligence, 2010. PDF.

“SURF” Blob Detector & Descriptor

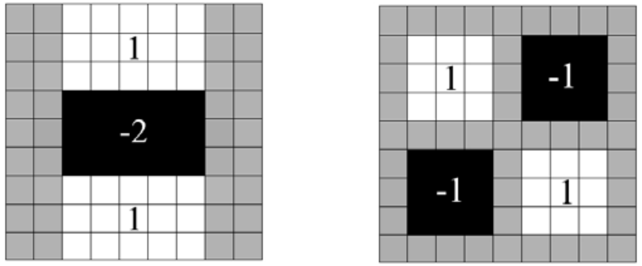
- **SURF: *S*peeded *U*p *R*obust *F*eatures**
- Similar to **SIFT** but much faster
- **Basic idea: approximate Gaussian and LoG filters using box filters**
- Results comparable with SIFT, plus:
  - Faster computation
  - Generally shorter descriptors

Original second order partial derivatives of a Gaussian



$\frac{\partial^2 G(x,y)}{\partial y^2}$ $\frac{\partial^2 G(x,y)}{\partial x \partial y}$

SURF Approximation using **box filters**



Bay, Tuytelaars, Van Gool, " Speeded Up Robust Features ", European Conference on Computer Vision (ECCV) 2006. PDF.

SWISS MADE

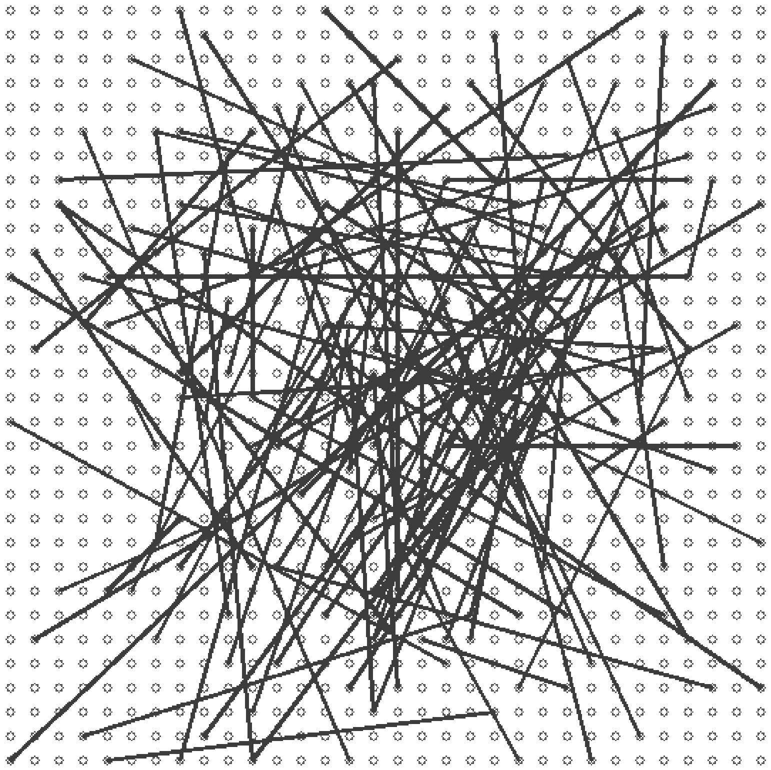
“BRIEF” Descriptor (can be applied to corners or blobs)

- **BRIEF: B**inary **R**obust **I**ndependent **E**lementary **F**eatures

- **Goal**: high speed description computation and matching

- **Binary** descriptor formation:
  - Smooth image
  - **for each** detected keypoint (e.g. FAST),
  - ***sample*** 128 intensity pairs $({p_1}^i, {p_2}^i)$ $(i = 1, \dots, 128)$ within a squared patch around the keypoint
  - Create an empty 128-element descriptor
  - **for each** i^{th} **pair**
    - **if** $I_{{p_1}^i} < I_{{p_2}^i}$ **then *set*** i^{th} bit of descriptor to **1**
    - **else** to **0**

- The **pattern is generated randomly** (or **learned**) only once; then, **the same pattern is used for all patches**
- **Pros**: **Binary descriptor**: allows **very fast** Hamming distance matching (count of the number of bits that are different in the descriptors matched)
- **Cons**: **Not scale/rotation invariant**



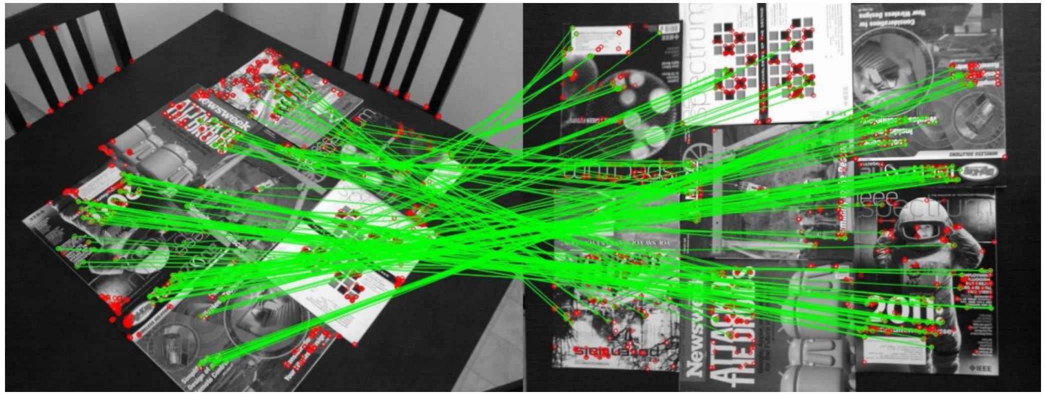
Pattern for intensity pair samples – generated randomly

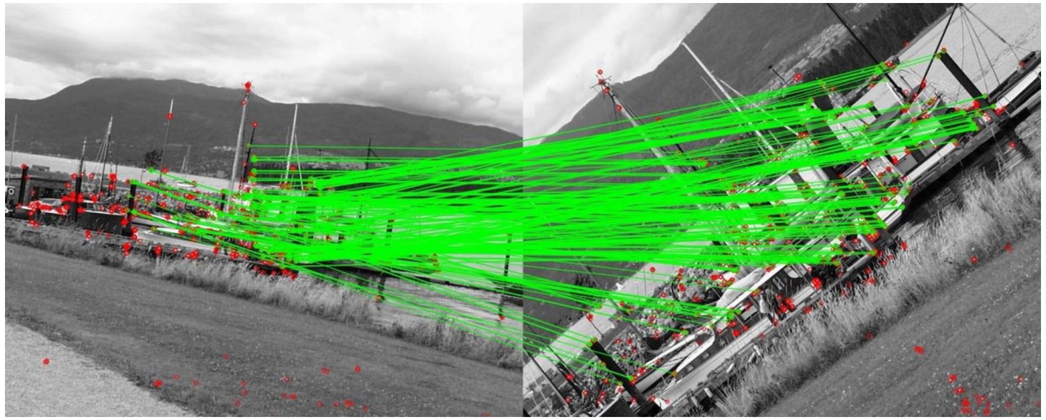
SWISS MADE

Calonder, Lepetit, Strecha, Fua, BRIEF: Binary Robust Independent Elementary Features, European Conference on Computer Vision (ECCV), 2010. PDF.

“ORB” Descriptor (can be applied to corners or blobs)

- **ORB: *O*riented FAST and *R*otated *B*RIEF**
- Keypoint detector originally based on **FAST**
- Binary descriptor based on BRIEF but adds an orientation component to make it **rotation invariant**

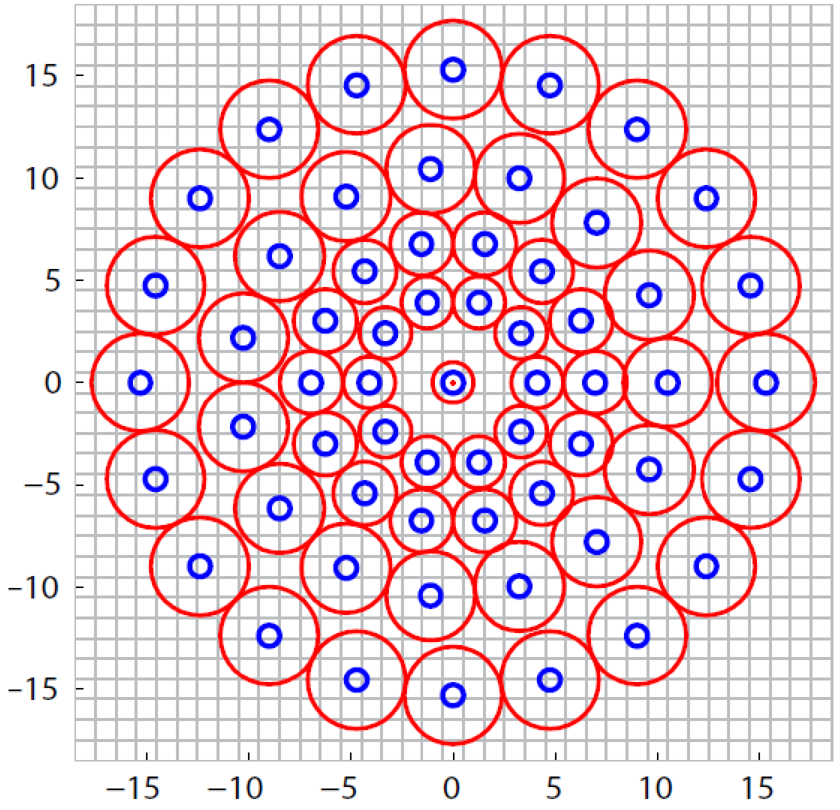




Rublee,Rabaud, Konolige, Bradski,“ORB: an efficient alternative to SIFT or SURF".
IEEE International Conference on Computer Vision (ICCV), 2011. PDF.

"BRISK" Descriptor (can be applied to corners or blobs)

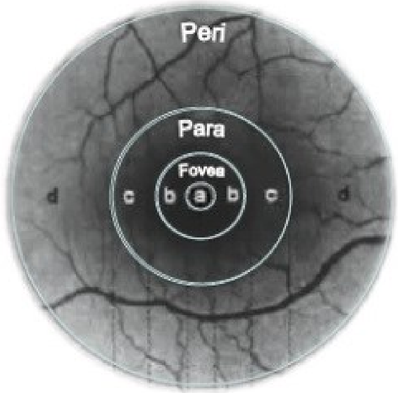
- **BRISK: B**inary **R**obust **I**nvariant **S**calable **K**eypoints
- Keypoint detector based on **FAST**
- Binary descriptor
- Both **rotation and scale invariant**
- **Binary descriptor**, formed by pairwise intensity comparisons **(like BRIEF) but on a radially symmetric sampling pattern**
  - **Red circles**: size of the smoothing kernel applied
  - **Blue circles**: smoothed pixel value used
- Detection and descriptor speed: **10 times faster than SURF**
- **Slower than BRIEF, but scale- and rotation- invariant**



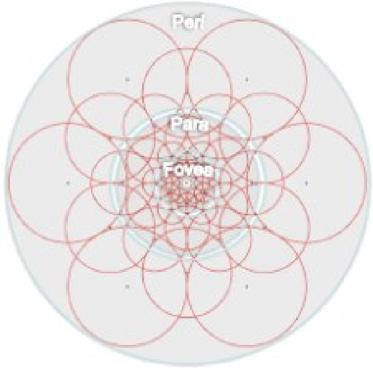
Leutenegger, Chli, Siegwart. BRISK: Binary Robust invariant scalable keypoints, ICCV 2011. PDF

"FREAK" Descriptor (can be applied to corners or blobs)

- **FREAK: F**ast **R**etina **K**eypoint
- **Rotation and scale invariant**
- Binary descriptor
- **Sampling pattern** similar to BRISK but uses a more pronounced "**retinal" (i.e., log-polar)** sampling pattern inspired by the human retina: higher density of points near the center
- **Pairwise** intensity comparisons form **binary** strings similar to BRIEF
- Pairs are **learned** (as in ORB)
- Circles indicate size of smoothing kernel
- **Coarse-to-fine matching** (cascaded approach): first compare the first half of bits; if distance smaller than threshold, proceed to compare the next bits, etc.
- Faster to compute, less memory and **than SIFT, SURF or BRISK**



Human retina



FREAK sampling pattern

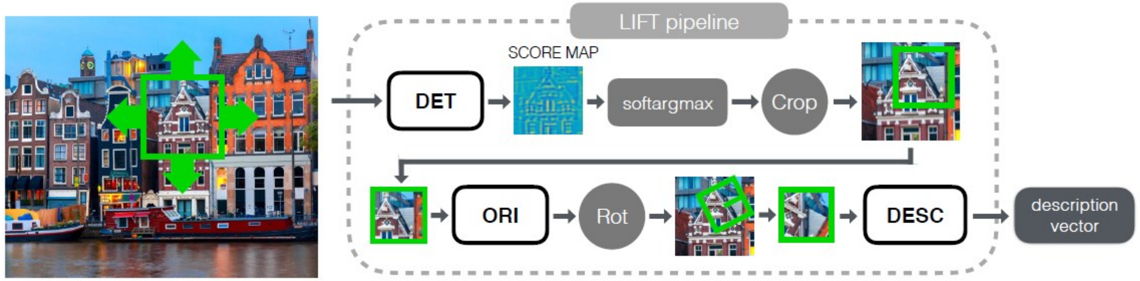
SWISS MADE

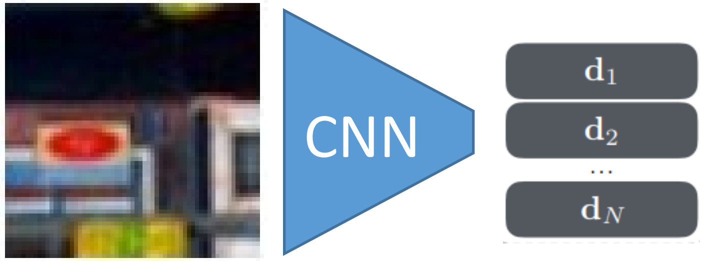
Alahi, Ortiz, Vandergheynst. FREAK: Fast Retina Keypoint, Conference on Computer Vision and Pattern Recognition (CVPR), 2012. PDF.

“LIFT” Detector and Descriptor

- **LIFT: L**earned **I**nvariant **F**eature **T**ransform
- **Rotation, scale, viewpoint and illumination invariant**
- **Learned feature detector, orientation, and descriptor** (self-supervised via SFM)
- **Not learned** jointly but **disjointly**:
  1. First, a network predicts a **feature location**
  2. Second, another network predicts the patch **orientation** which is used to derotate the patch.
  3. Then another neural network is used to generate a **patch descriptor** (128 dimensional) from the derotated patch.
- **Illumination invariance** is achieved by randomizing illuminations during training.
- **LIFT descriptor beats SIFT in repeatability**

Keypoints with scales and orientations





A CNN predicts descriptor

Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, Pascal Fua,
LIFT: Learned Invariant Feature Transform, European Conference on Computer Vision (ECCV) 2016. PDF.

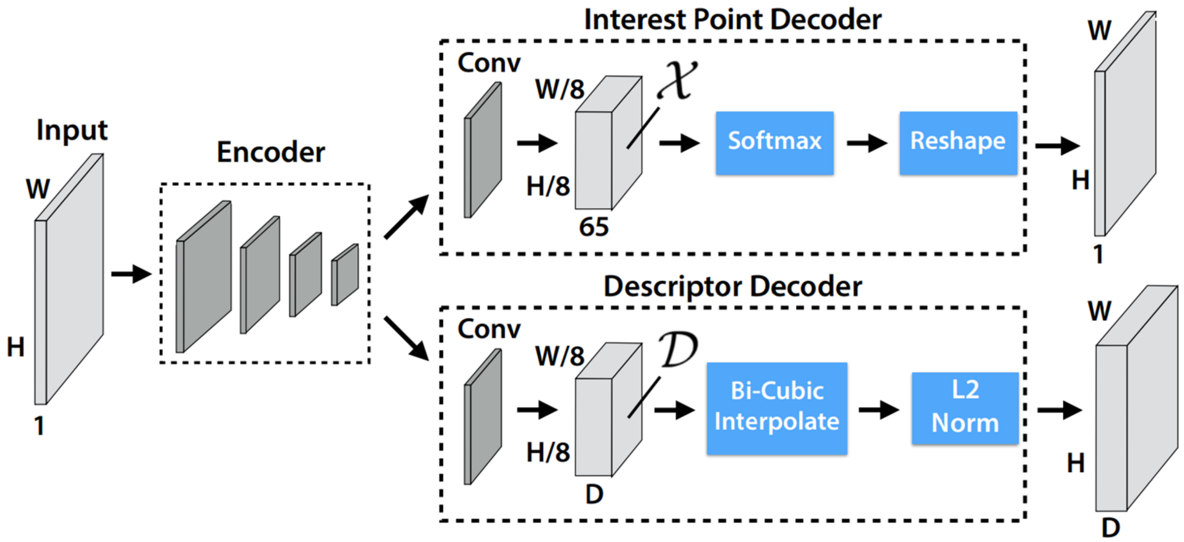
SWISS MADE

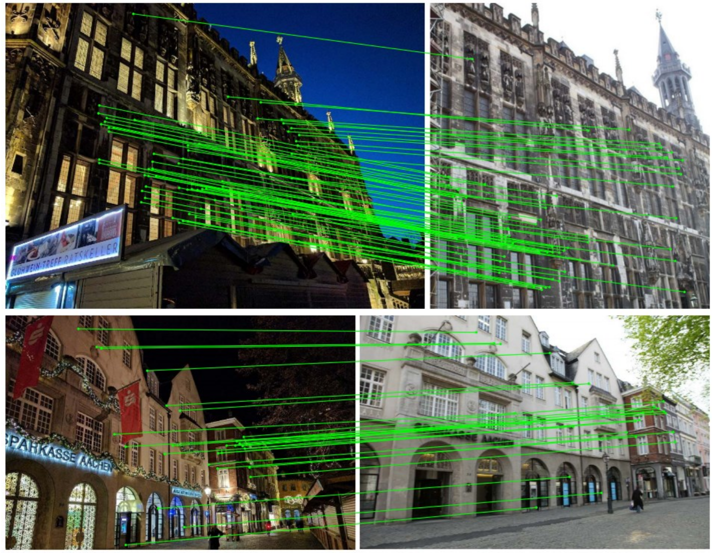
LIFT vs SIFT

https://youtu.be/hhxAttChmCo

"SuperPoint" Detector and Descriptor

- **Joint learning** of keypoint **location and descriptor**. **Self-supervised**.
- Trained on **synthetic images** and **fined tuned** on **real images**
- **Detector less accurate than SIFT and LIFT,** but **descriptor outperforms SIFT and LIFT**
- **But slower than SIFT and LIFT**





Detone, Malisiewicz, Rabinovich. SuperPoint: Self-Supervised Interest Point Detection and Description. CVPRW 2018. [PDF](PDF).

Recap Table

Detector	Localization Accuracy of the detector	Descriptor that can be used	Efficiency	Relocalization & Loop closing
Harris	++++	Patch SIFT/LIFT BRIEF ORB BRISK FREAK	+++ + ++++ ++++ +++ ++++	+ +++++ +++ ++++ +++ ++++
Shi-Tomasi	++++	Patch SIFT BRIEF ORB BRISK FREAK	++ + ++++ ++++ +++ ++++	+ +++++ +++ ++++ +++ ++++
FAST	++++	Patch SIFT/LIFT BRIEF ORB BRISK FREAK	++++ + ++++ ++++ +++ ++++	+ +++++ +++ ++++ +++ ++++
SIFT	+++	SIFT	+	++++
SURF	+++	SURF	++	++++
SuperPoint	++	SuperPoint	+	+++++

Summary (things to remember)

- Similarity metrics: NCC (ZNCC), SSD (ZSSD), SAD (ZSAD), Census Transform
- Point feature detection
  - Properties and invariance to transformations
    - Challenges: rotation, scale, view-point, and illumination changes
  - Extraction
    - Moravec
    - Harris and Shi-Tomasi
      - Rotation invariance
  - Automatic Scale selection
  - Descriptor
    - Intensity patches
      - Canonical representation: how to make them invariant to transformations: rotation, scale, illumination, and view-point (affine)
    - Better solution: Histogram of oriented gradients: SIFT descriptor
  - Matching
    - (Z)SSD, SAD, NCC, Hamming distance (last one only for binary descriptors)
      ratio 1^{st} /2^{nd} closest descriptor
  - Depending on the task, you may want to trade off repeatability and robustness for speed: approximated solutions, combinations of efficient detectors and descriptors.
    - Fast corner detector: FAST;
    - Keypoint descriptors faster than SIFT: SURF, BRIEF, ORB, BRISK

Readings

- Ch. 7.1 of Szeliski book, 2nd Edition
- Chapter 4 of Autonomous Mobile Robots book: link
- Ch. 13.3 of Peter Corke book

Understanding Check

Are you able to answer:

- How does automatic scale selection work?
- What are the good and the bad properties that a function for automatic scale selection should have or not have?
- How can we implement scale invariant detection efficiently? (show that we can do this by resampling the image vs rescaling the kernel).
- What is a feature descriptor? (patch of intensity value vs histogram of oriented gradients). How do we match descriptors?
- How is the keypoint detection done in SIFT and how does this differ from Harris?
- How does SIFT achieve orientation invariance?
- How is the SIFT descriptor built?
- What is the repeatability of the SIFT detector after a rescaling of 2? And for a 50 degrees viewpoint change?
- Illustrate the 1st to 2nd closest ratio of SIFT detection: what's the intuitive reasoning behind it? Where does the 0.8 factor come from?
- How does the FAST detector work? What are its pros and cons compared with Harris?