University of Zurich[UZH] | ETH zürich

Institute of Informatics – Institute of Neuroinformatics

ROBOTICS & PERCEPTION GROUP

Lecture 06
Point Feature Detection and Matching
Part 2

Davide Scaramuzza

No lab exercise this afternoon

- The next exercise session will take place next week and will be about Stereo Vision

Course Schedule

Next lecture will be given by Dr. Guillermo Gallego

Date	Time	Description of the lecture/exercise	Lecturer
21.09.2017	10:15 - 12:00	01 – Introduction	Davide Scaramuzza
28.09.2017	10:15 - 12:00	02 - Image Formation 1: perspective projection and camera models	**Guillermo Gallego**
05.10.2017	10:15 - 12:00	03 - Image Formation 2: camera calibration algorithms	Davide Scaramuzza
	13:15 – 15:00	Exercise 1: Augmented reality wireframe cube	T. Cieslewski/H. Rebecq/A. Loquercio
12.10.2017	10:15 - 12:00	04 - Filtering & Edge detection	Davide Scaramuzza
	13:15 – 15:00	Exercise 2: PnP problem	T. Cieslewski/H. Rebecq/A. Loquercio
19.10.2017	10:15 - 12:00	05 - Point Feature Detectors 1: Harris detector	Davide Scaramuzza
	13:15 – 15:00	Exercise 3: Harris detector + descriptor + matching	T. Cieslewski/H. Rebecq/A. Loquercio
26.10.2017	10:15 - 12:00	06 - Point Feature Detectors 2: SIFT, BRIEF, BRISK	Davide Scaramuzza
02.11.2017	10:15 - 12:00	07 - Multiple-view geometry 1	**Guillermo Gallego**
	13:15 – 15:00	Exercise 4: Stereo vision: rectification, epipolar matching, disparity, triangulation	T. Cieslewski/H. Rebecq/A. Loquercio
09.11.2017	10:15 - 12:00	08 - Multiple-view geometry 2	Davide Scaramuzza
	13:15 – 15:00	Exercise 5: Eight-point algorithm and RANSAC	T. Cieslewski/H. Rebecq/A. Loquercio
16.11.2017	10:15 - 12:00	09 - Multiple-view geometry 3	Davide Scaramuzza
	13:15 – 15:00	Exercise 6: P3P algorithm and RANSAC	T. Cieslewski/H. Rebecq/A. Loquercio
23.11.2017	10:15 - 12:00	10 - Dense 3D Reconstruction (Multi-view Stereo)	Davide Scaramuzza
	13:15 – 15:00	Exercise 7: Intermediate VO Integration	T. Cieslewski/H. Rebecq/A. Loquercio
30.11.2017	10:15 - 12:00	11 - Optical Flow and Tracking (Lucas-Kanade)	Davide Scaramuzza
	13:15 – 15:00	Exercise 8: Lucas-Kanade tracker	T. Cieslewski/H. Rebecq/A. Loquercio
07.12.2017	10:15 - 12:00	12 – Place recognition	Davide Scaramuzza
	13:15 – 15:00	Exercise 9: Recognition with Bag of Words	T. Cieslewski/H. Rebecq/A. Loquercio
	10:15 - 12:00	13 – Visual inertial fusion	Davide Scaramuzza
14.12.2017	13:15 – 15:00	Exercise 10: Pose graph optimization and Bundle adjustment	T. Cieslewski/H. Rebecq/A. Loquercio
21.12.2017	10:15 - 12:00	14 - Event based vision + **lab visit and live demonstrations**	Davide Scaramuzza
	13:15 – 15:00	Exercise 11: final VO integration	T. Cieslewski/H. Rebecq/A. Loquercio

Outline

- Automatic Scale Selection
- The SIFT Detector (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?
  - Possible solution: rescale the patch!

Image 1

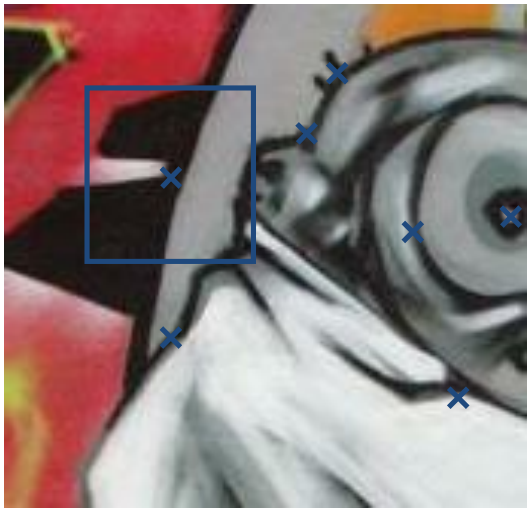
Image 2

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!

Image 1

Image 2



Scale changes

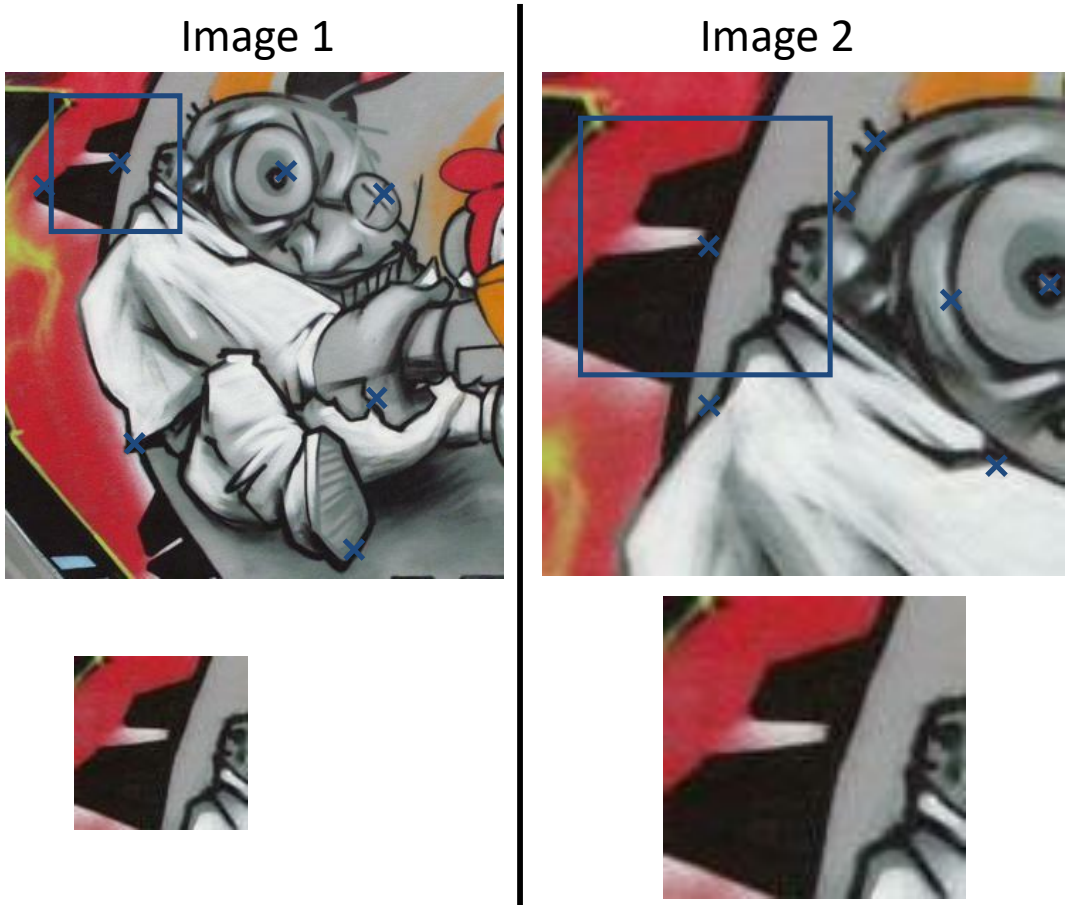
- How can we match image patches corresponding to the same feature but belonging to images taken at different scales?
  - Possible solution: rescale the patch!

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!



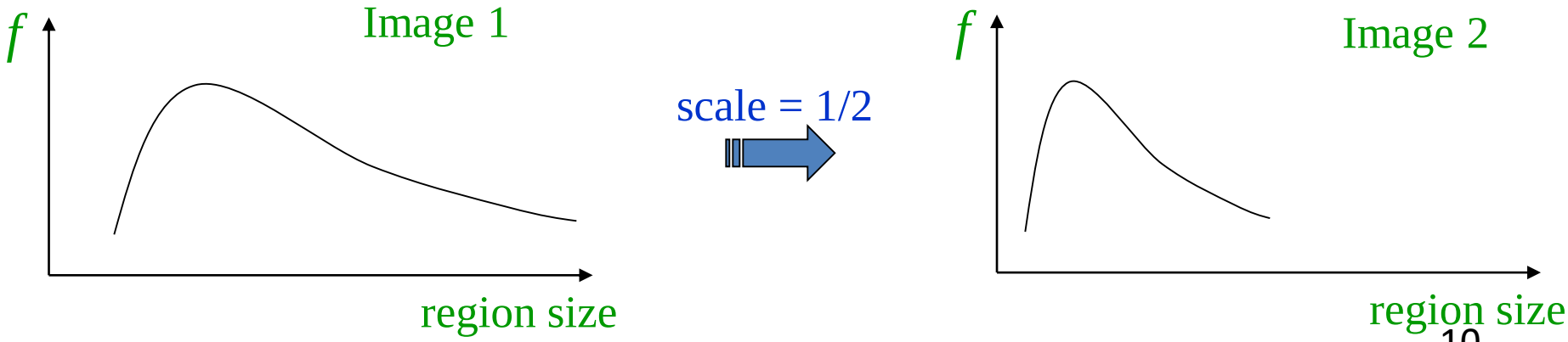
Scale changes

- Scale search is time consuming (needs to be done individually for all patches in one image)
  - Complexity would be $(NM)^2$ (assuming that we have N features per image and M scale levels for each image)
- Possible solution: assign each feature its own "scale" (i.e., size).
  - What's the optimal scale (i.e., size) of the patch?

Automatic Scale Selection

- Solution:
  - Design a function on the image patch, which is "scale invariant" (i.e., which has the same value for corresponding regions, even if they are at different scales)

    - For a point in one image, we can consider it as a function of region size (patch width)



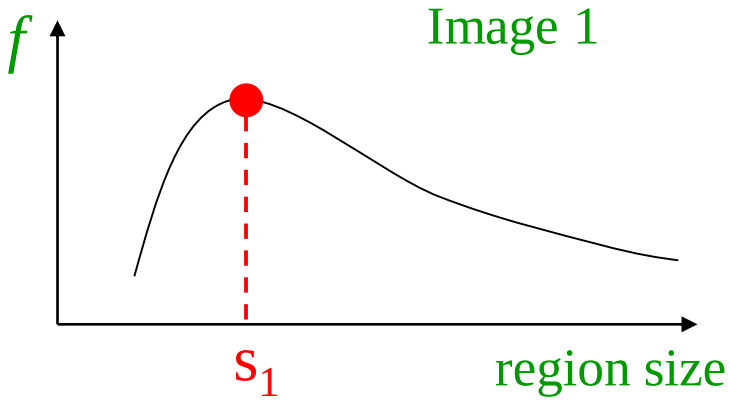
Automatic Scale Selection

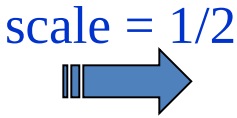
- Common approach:

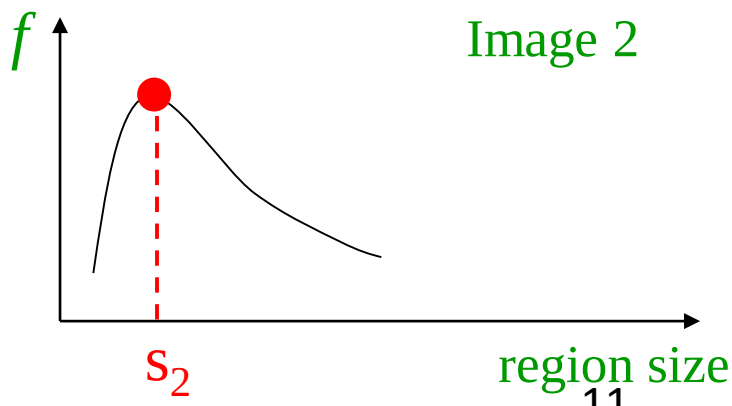
  Take a local maximum or minima of this function

  Observation: region size, for which the maximum or minima is achieved, should be *invariant* to image scale.

Important: this scale invariant region size is found in each image independently!



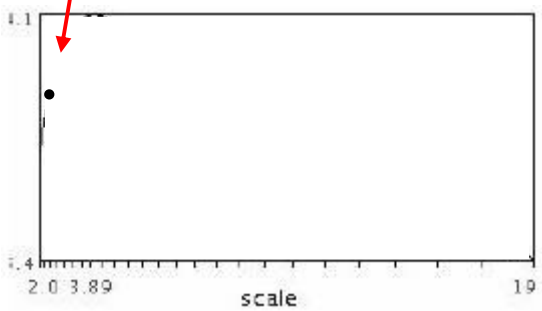




Automatic Scale Selection

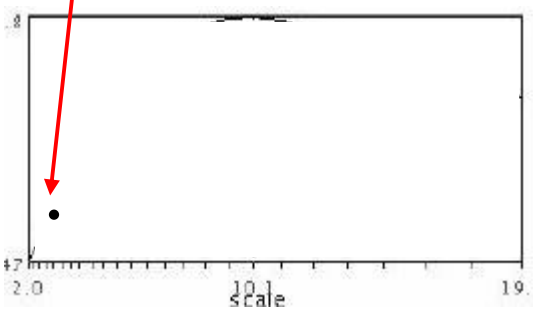
- Function responses for increasing scale (scale signature)

Image 1



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

Image 2

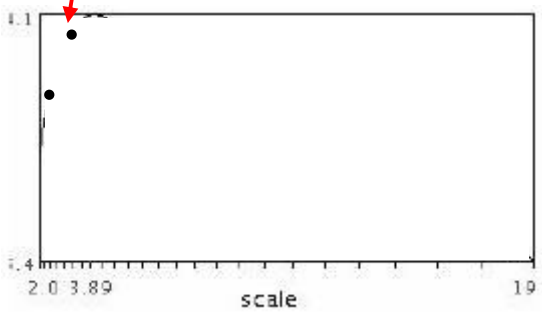


$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

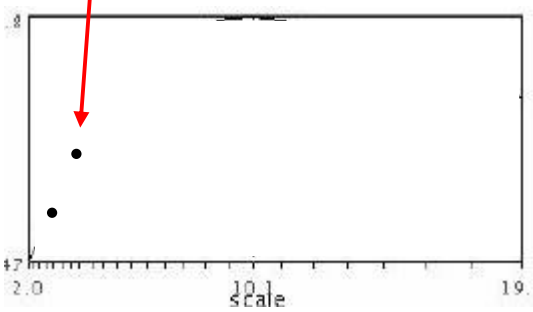
- Function responses for increasing scale (scale signature)

Image 1



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

Image 2

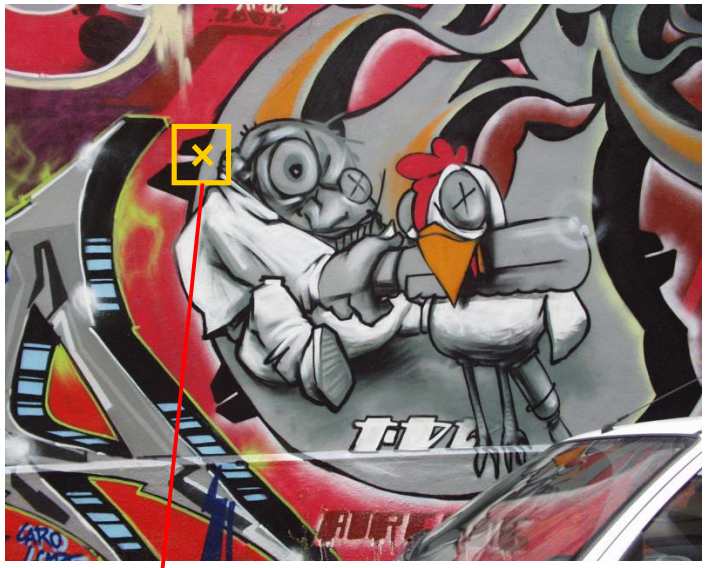


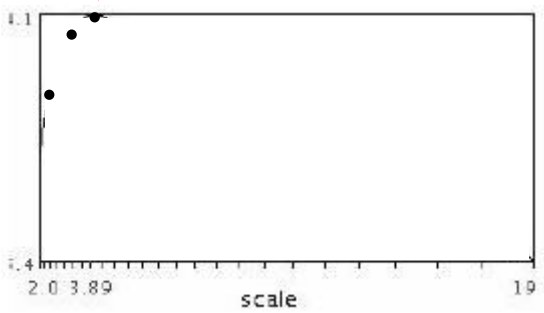
$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

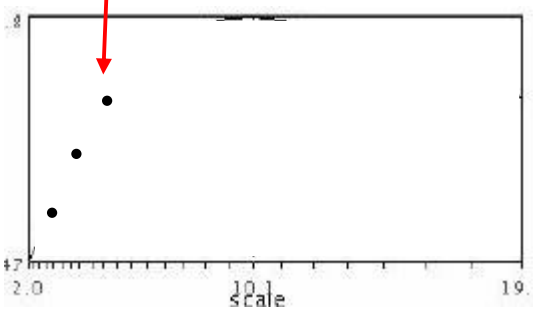
Image 1





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

Image 2

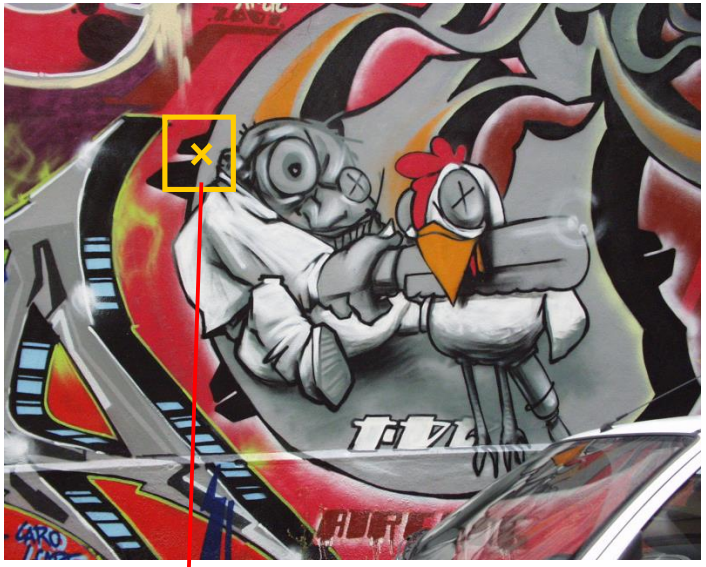


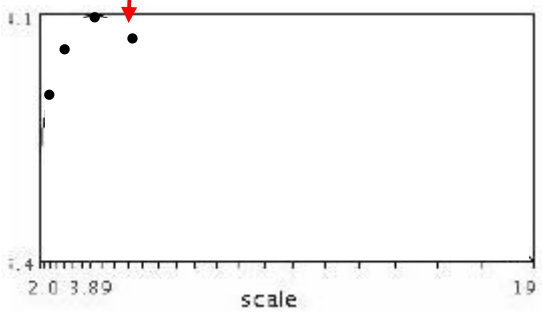
$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

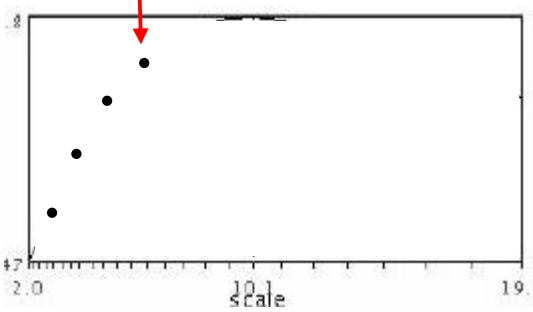
Image 1





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

Image 2

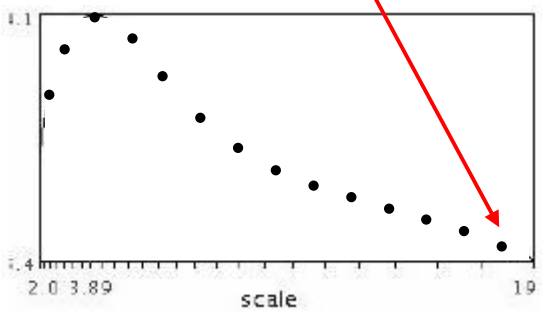


$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

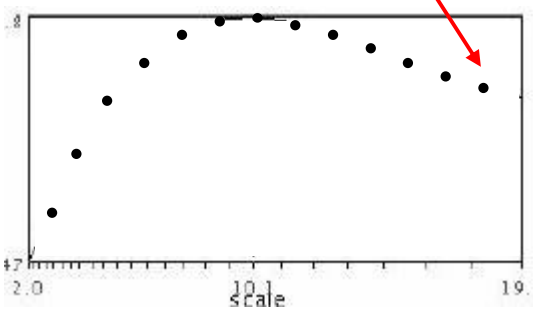
- Function responses for increasing scale (scale signature)

Image 1



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

Image 2

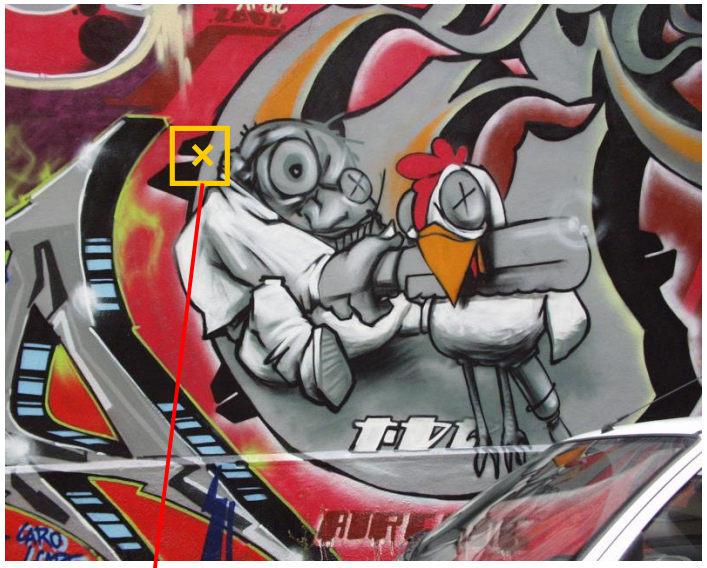


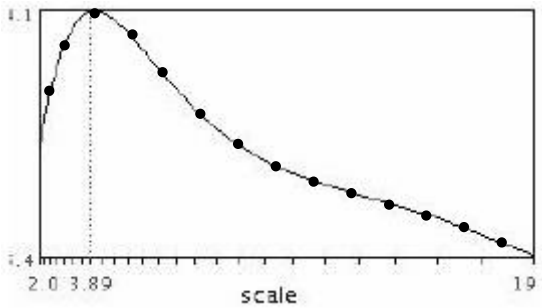
$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

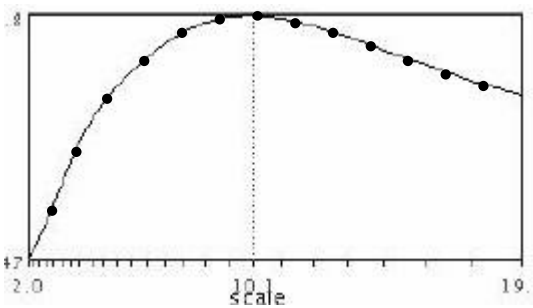
Image 1





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

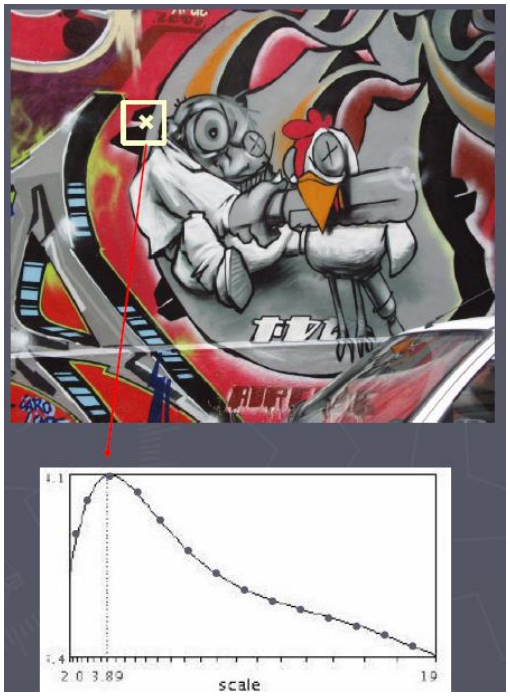
Image 2

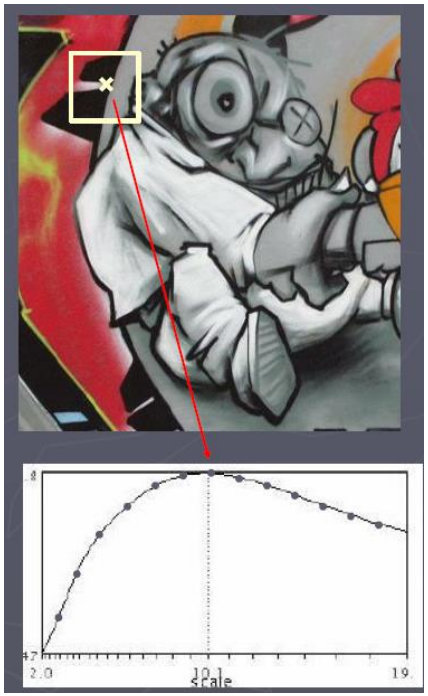


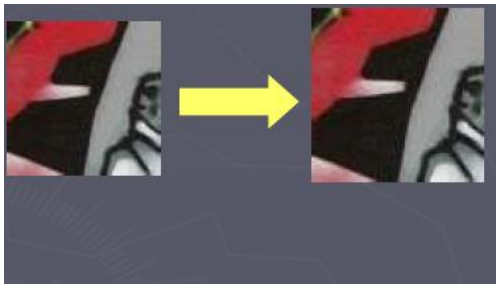
$f(I_{i_1 \ldots i_m}(x',\sigma'))$

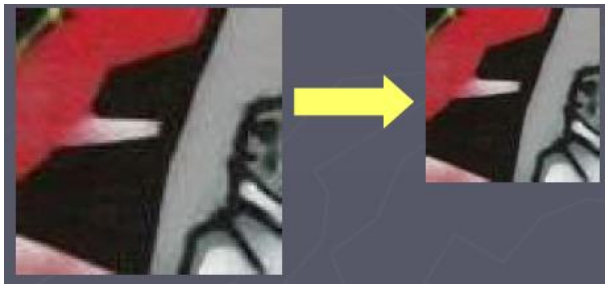
Automatic Scale Selection

- When the right scale is found, the patch must be normalized









Automatic Scale Selection

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

- What if there are multiple peaks?

- **Sharp, local intensity changes** are good regions to monitor in order to identify the scale

  ⇨ **Blobs and corners** are the **ideal locations**!

Automatic Scale Selection

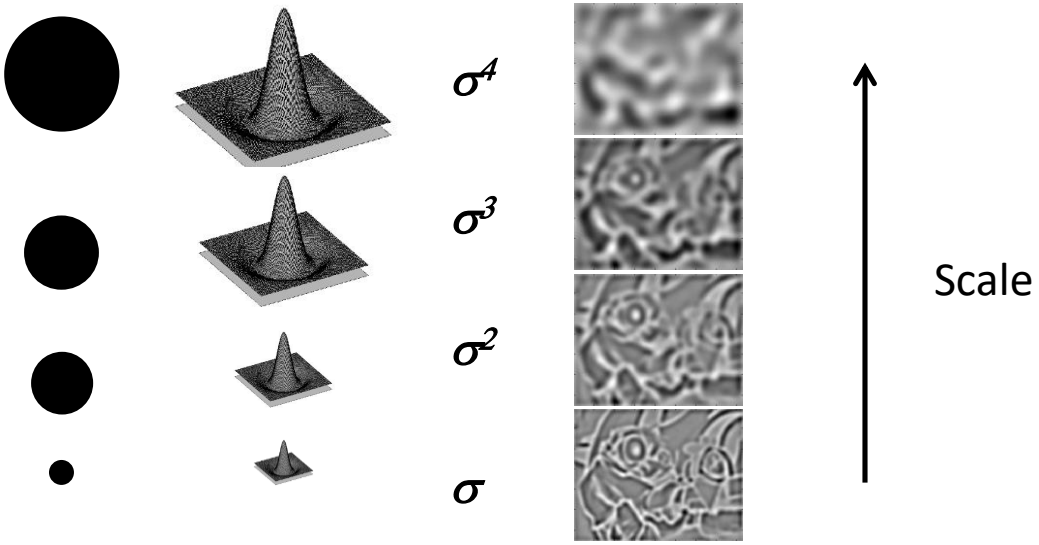
- Function for determining scale: convolve image with kernel to identify sharp intensity discontinuities

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

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

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

- Correct scale is found as local maxima or minima across consecutive smoothed images



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

Automatic Scale Selection

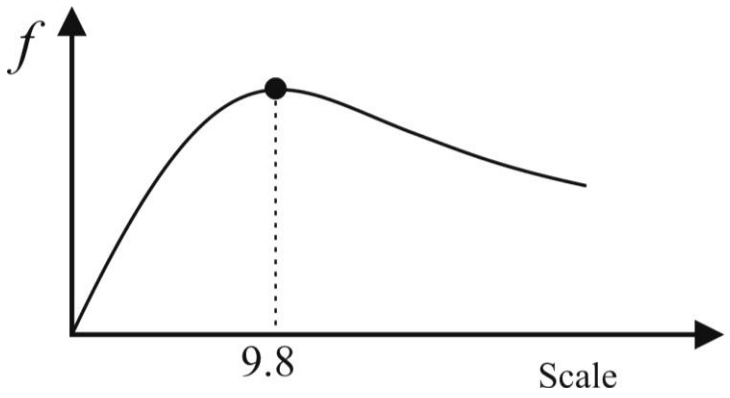
- Function for determining scale: convolve image with kernel to identify sharp intensity discontinuities

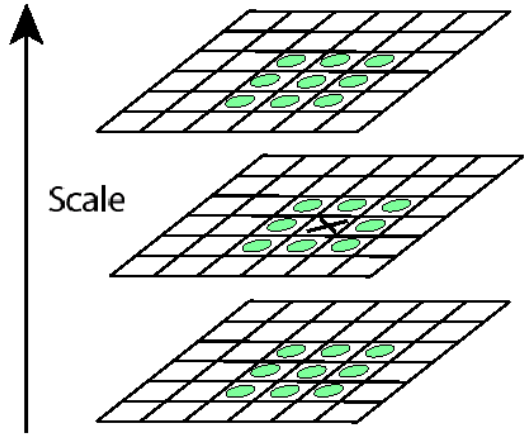
$$f = \text{Kernel} * \text{Image}$$

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

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

- Correct scale is found as local maxima or minima across consecutive smoothed images





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

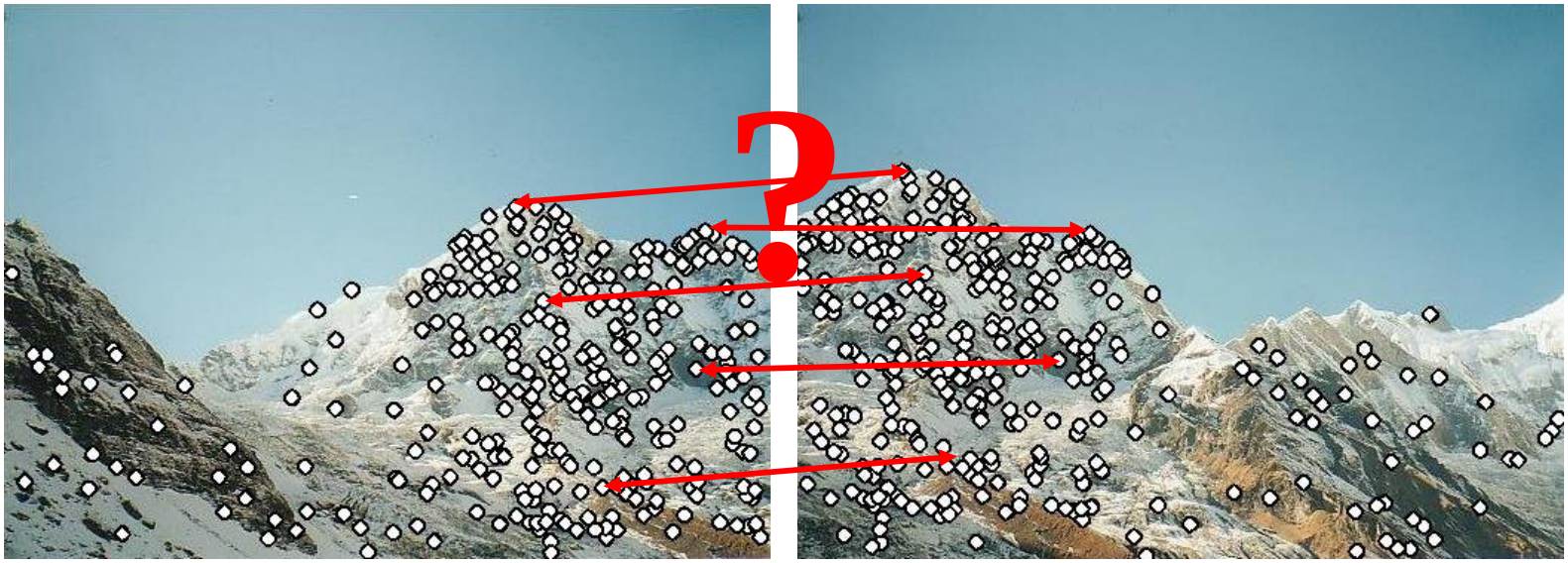
Main questions

- What points are distinctive (i.e., *features*, *keypoints, salient* points), such that they are *repeatable? (i.e.,* can be re-*detected* from other views)
- How to *describe* a local region?
- How to establish *correspondences*, i.e., compute matches?

Feature descriptors

- We know how to detect points
- Next question:

How to *describe* them for matching?



- **Simplest descriptor**: **intensity values** within a squared patch
- Alternative: **Census Transform** or **Histograms of Oriented Gradients** (like in SIFT, see later)
- Then, descriptor matching can be done using **Hamming Distance (Census)** or **(Z)SSD, (Z)SAD, or (Z)NCC**

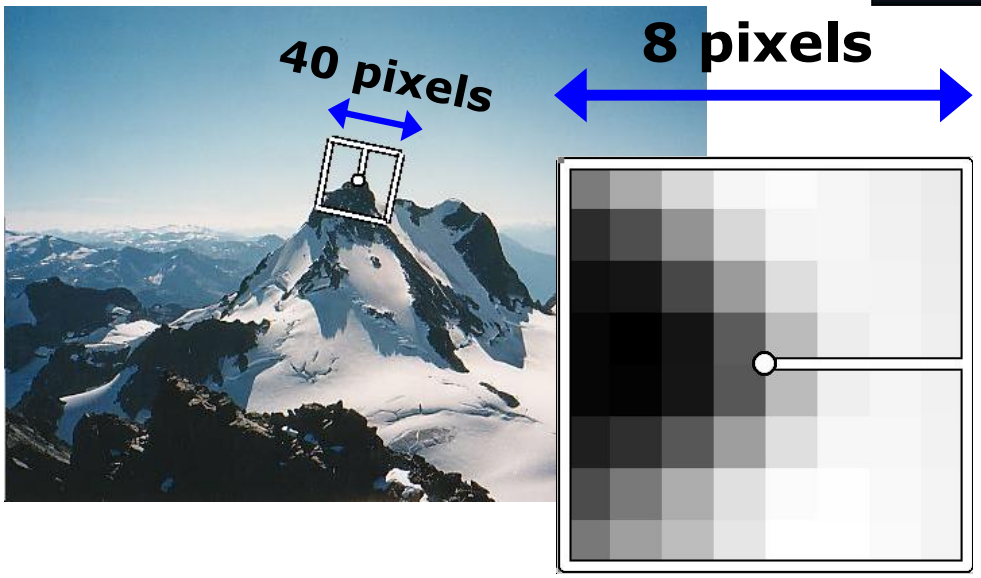
Feature descriptors

- We'd like to find the same features regardless of the **transformation** (*rotation, scale, view point,* and *illumination)*
  - **Most feature methods** are designed to be invariant to
    - 2D translation,
    - 2D rotation,
    - Scale
  - Some of them can also handle
    - **Small view-point invariance** (e.g., SIFT works up to about 60 degrees)
    - **Linear illumination changes**

How to achieve invariance

Step 1: Re-scaling and De-rotation

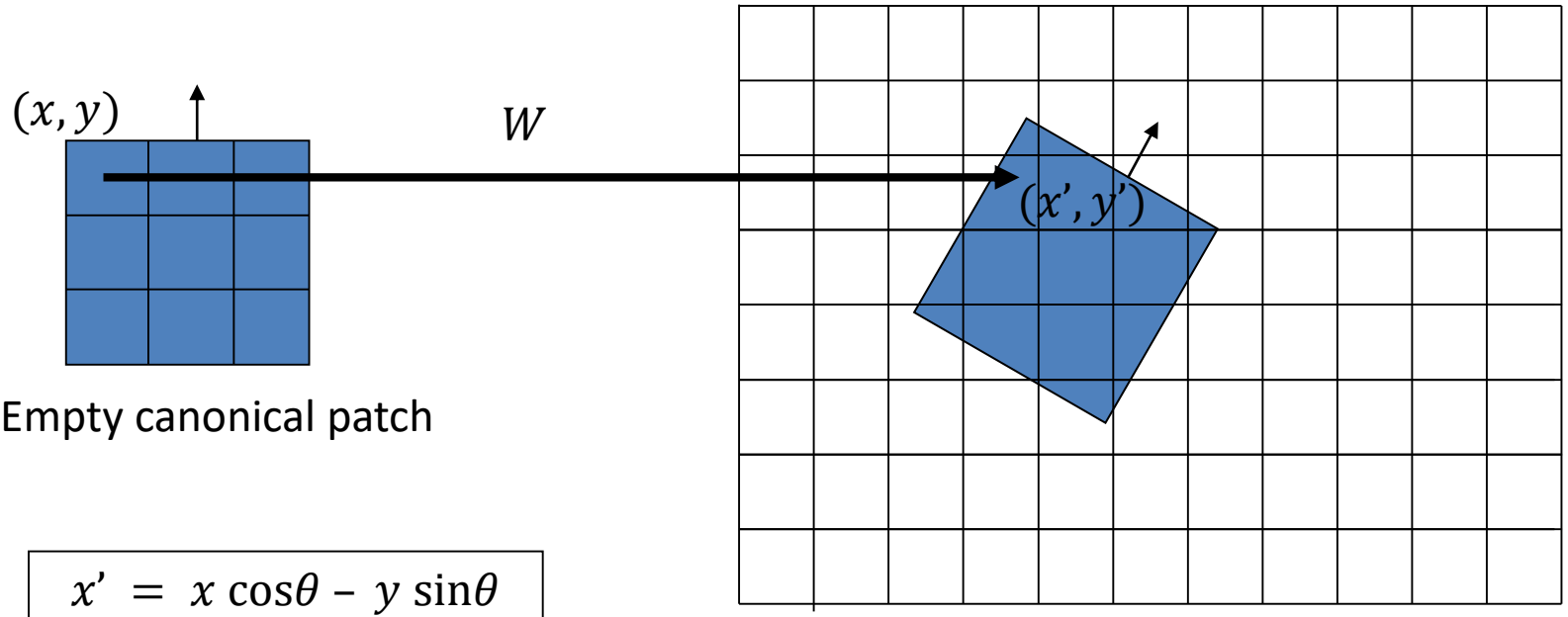
- **Find correct scale** using LoG operator
- **Rescale the patch** to a default size (e.g., 8x8 pixels)
- **Find local orientation**
  - Dominant direction of gradient for the image patch (e.g., Harris eigenvectors)
- **De-rotate patch through "patch warping"**
  - This puts the patches into a **canonical orientation**



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 detected 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:
  - use *nearest neighbor*
  - or *bilinear interpolation*

Example 1: Rotational warping



Empty canonical patch

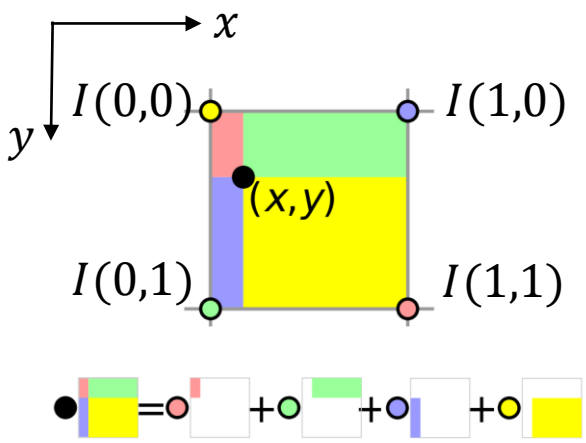
Patch detected in the image

$$W \quad \boxed{\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{aligned}}$$

counterclockwise rotation

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.



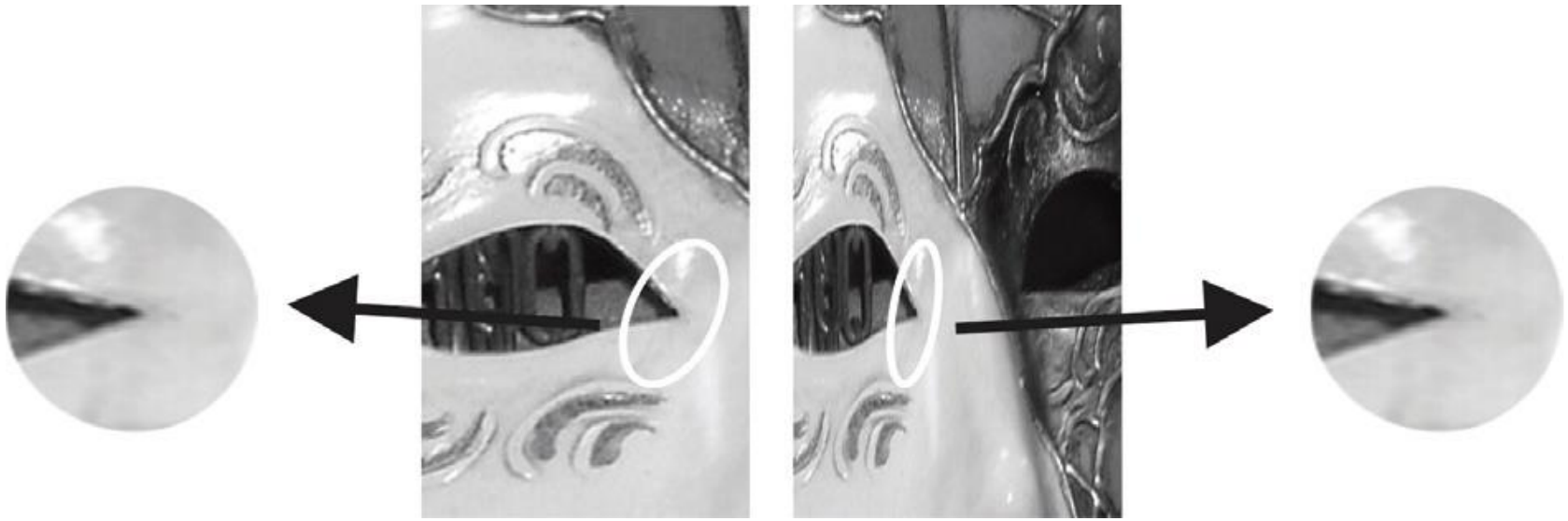
$$
\begin{aligned}
&I(x,y) = \\
&I(0,0)(1-x)(1-y) + \\
&I(0,1)(1-x)(y) + \\
&I(1,0)(x)(1-y) + \\
&I(1,1)(x)(y)
\end{aligned}
$$

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.

Example 2: Affine Warping

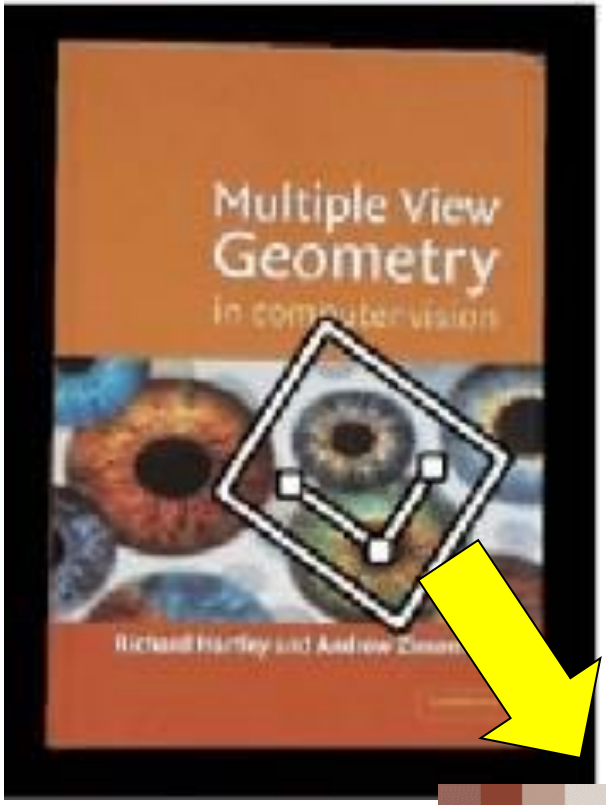
Affine warping (to achieve slight view-point invariance)

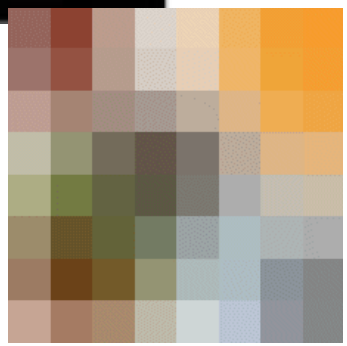
- The second moment matrix *M* can be used to identify the two directions of fastest and slowest change of intensity around the feature.
- Out of these two directions, an elliptic patch is extracted at the scale computed by with the LoG operator.
- The region inside the ellipse is normalized to a circular one

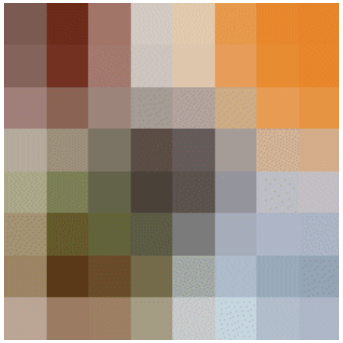


How to achieve invariance

Example: de-rotation, re-scaling, and affine un-warping







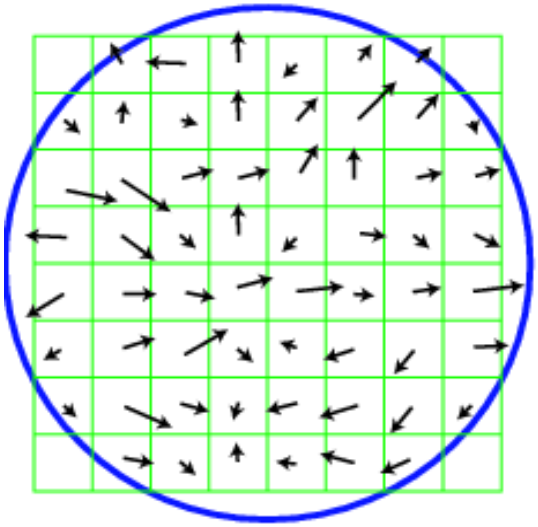
Feature descriptors

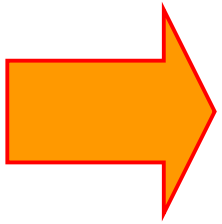
- Disadvantage of patches as descriptors:
  - If not warped, very small errors in rotation, scale, and view-point will affect matching score significantly
  - Computationally expensive (need to unwarp every patch)

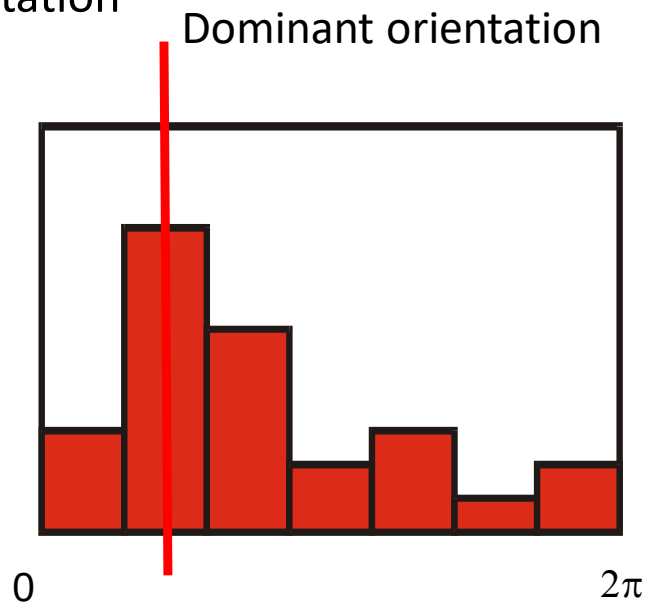
- Better solution **nowadays**: build descriptors from Histograms of Oriented Gradients (HOGs)

HOG descriptor (Histogram of Oriented Gradients)

- Compute a histogram of orientations of intensity gradients
- Peaks in histogram: dominant orientations
- **Keypoint orientation = histogram peak**
  - If there are multiple candidate peaks, **construct a different keypoint for each such orientation**
- **Rotate patch** according to this angle
  - This puts the patches into a canonical orientation







Rotation and Scale Normalization

- Rotate the window to standard orientation
- Scale the window size based on the scale at which the point was found

Outline

- Automatic Scale Selection
- The SIFT Detector (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 [IJCV, 2004] (now at Google)
- Descriptor computation:
  - Divide patch into 4×4 sub-patches = 16 cells
  - Compute HOG (8 bins, i.e., 8 directions) for all pixels inside each sub-patch
  - Concatenates all HOGs into a single 1D vector:
    - Resulting SIFT descriptor: 4×4×8 = 128 values
  - Descriptor Matching: SSD (i.e., Euclidean-distance)

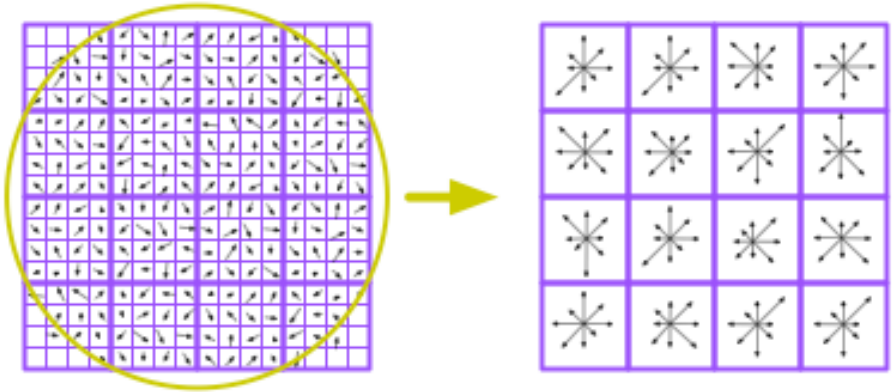


Image gradients

Keypoint descriptor

Intensity Normalization

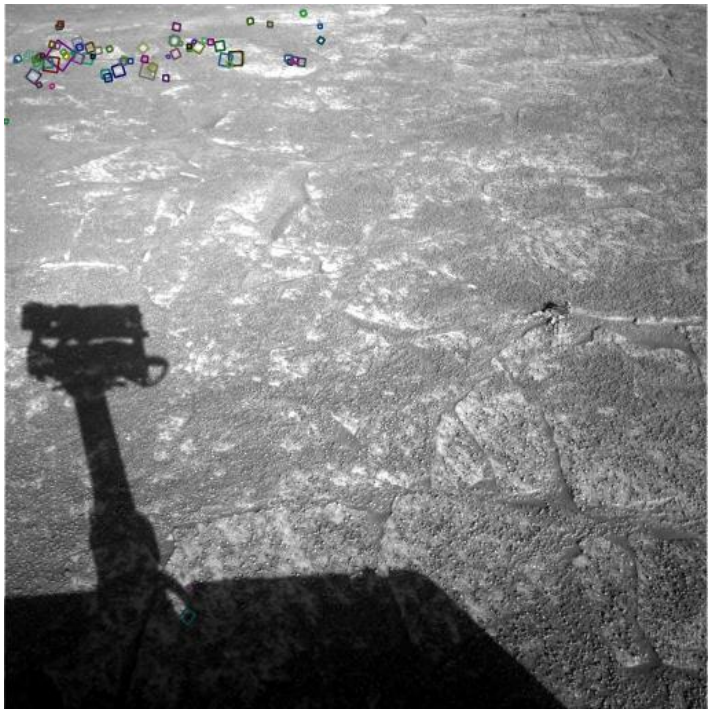
- The descriptor vector $\boldsymbol{v}$ is then normalized such that its l_2 norm is 1:

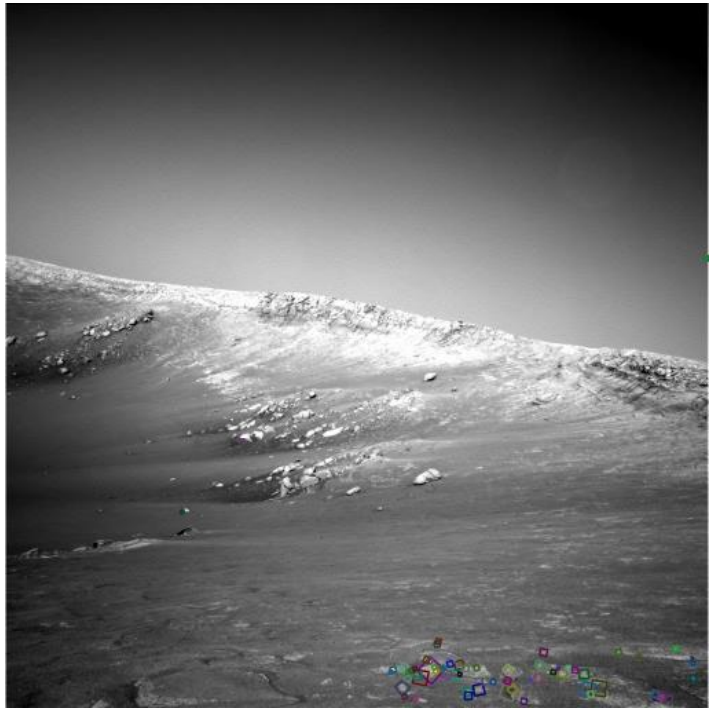
$$\overline{\boldsymbol{v}} = \frac{\boldsymbol{v}}{\sqrt{\sum_i^n v_i^2}}$$

- This guarantees that the descriptor is invariant to linear illumination changes (the descriptor is already invariant to additive illumination because it is based on gradients).

SIFT matching robustness

- Can handle changes in viewpoint (up to 60 degree out-of-plane rotation)
- Can handle significant changes in illumination (low to bright scenes)
- Expensive: 10 fps
- Original SIFT code (binary files): http://people.cs.ubc.ca/~lowe/keypoints

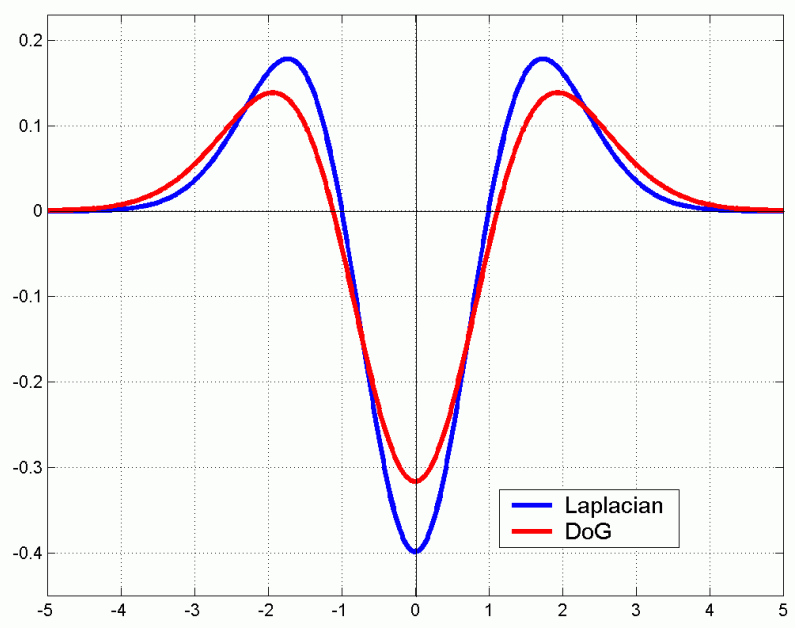




Scale Invariant Feature Detection

Difference of Gaussian (DoG) kernel instead of Laplacian of Gaussian (computationally cheaper)

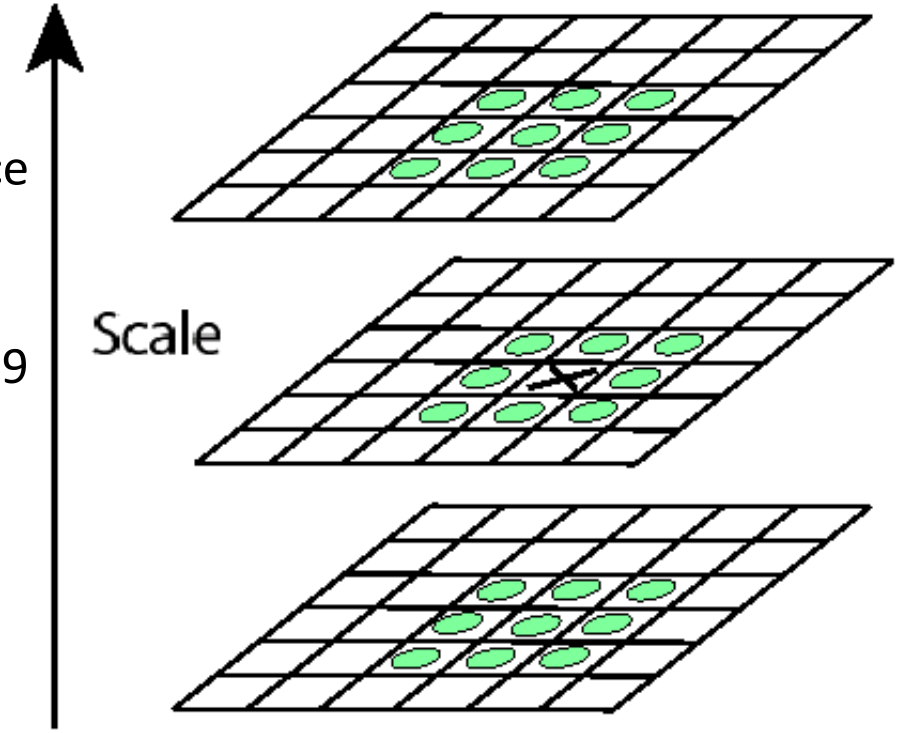
$$LOG \approx DoG = G_{k\sigma}(x, y) - G_{\sigma}(x, y)$$



SIFT detector (location + scale)

SIFT keypoints: local extrema (i.e., maxima and minima) in **both space** and **scale** of the DoG images

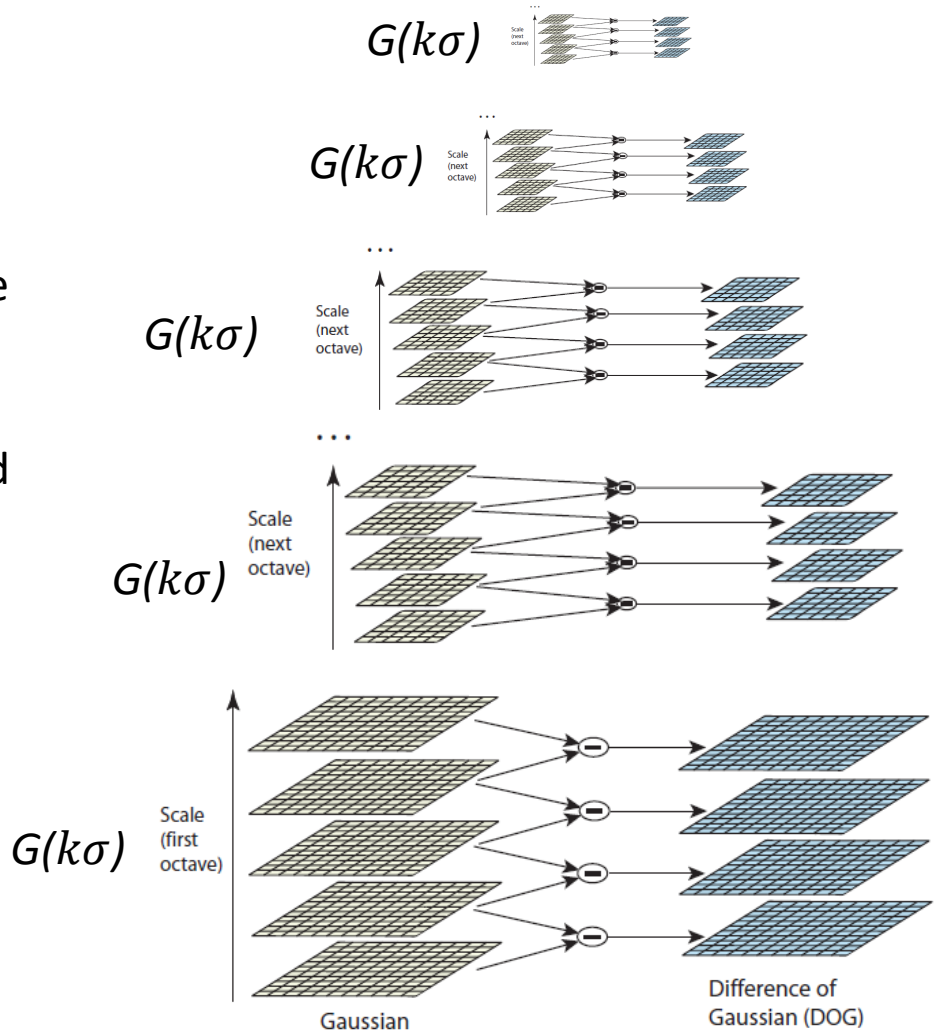
- Detect maxima and minima of difference-of-Gaussian in scale space
- Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below



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

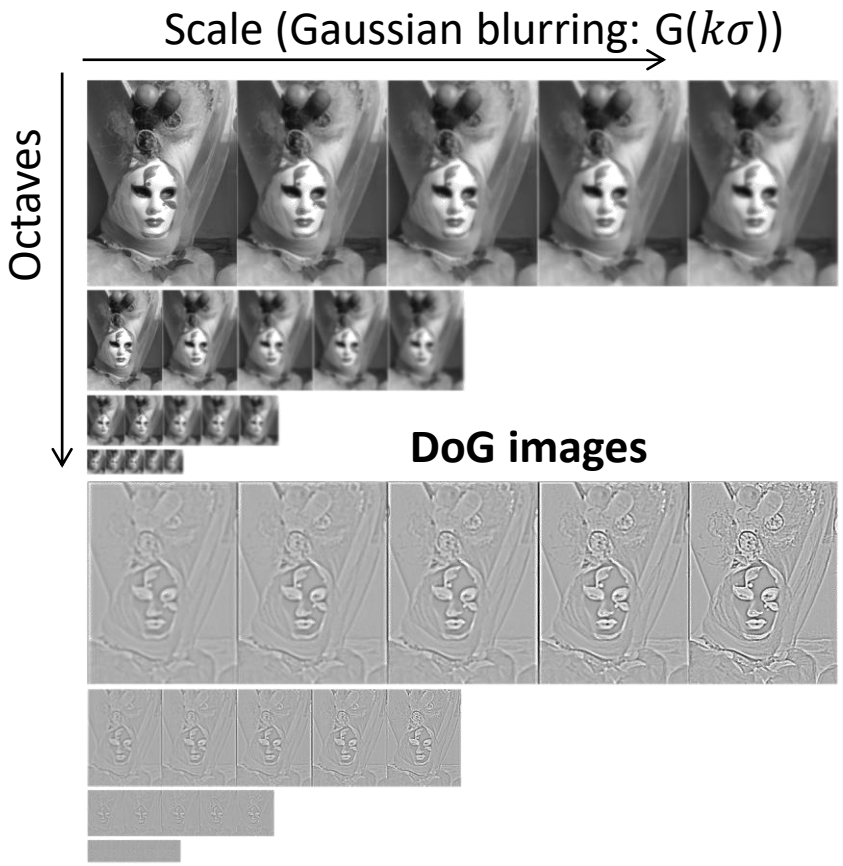
How it is implemented in practice

1. The initial image is **incrementally convolved with Gaussians** $G(k\sigma)$ to produce images separated by a constant factor k in scale space, shown stacked in the left column
   1. The initial Gaussian G(σ) has $\sigma = 1.6$
   2. k is chosen such that $k = 2^{1/s}$, where s is an integer (typically s =3)
   3. For efficiency reasons, when k reaches 2, the image is downsampled by a factor of 2 and then the procedure is repeated again up to 4 or 6 octaves (pyramid levels)
2. **Adjacent image scales are then subtracted** to produce the *Difference-of-Gaussian* (DoG) images



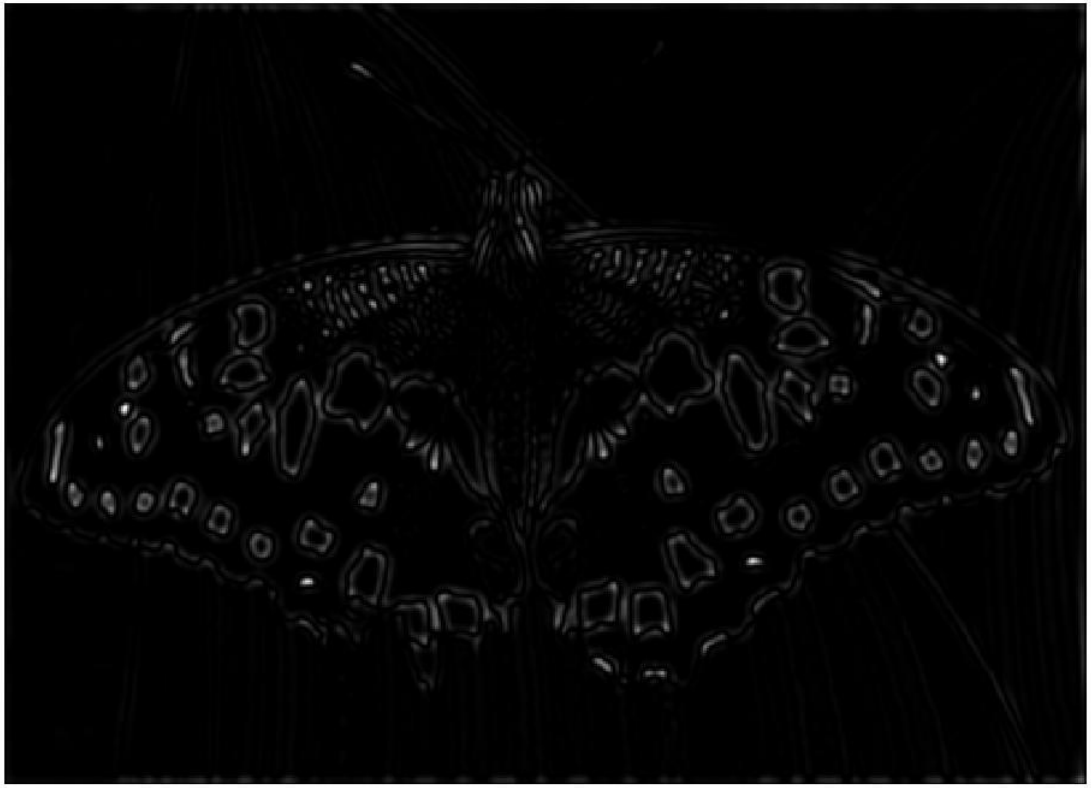
How it is implemented in practice

1. The initial image is **incrementally convolved with Gaussians** $G(k\sigma)$ to produce images separated by a constant factor k in scale space, shown stacked in the left column
   1. The initial Gaussian $G(\sigma)$ has $\sigma = 1.6$
   2. k is chosen such that $k = 2^{1/s}$, where s is an integer (typically $s = 5$)
   3. For efficiency reasons, when k reaches 2, the image is downsampled by a factor of 2 and then the procedure is repeated again up to 4 or 6 octaves (pyramid levels)
2. **Adjacent image scales are then subtracted** to produce the *Difference-of-Gaussian* (DoG) images



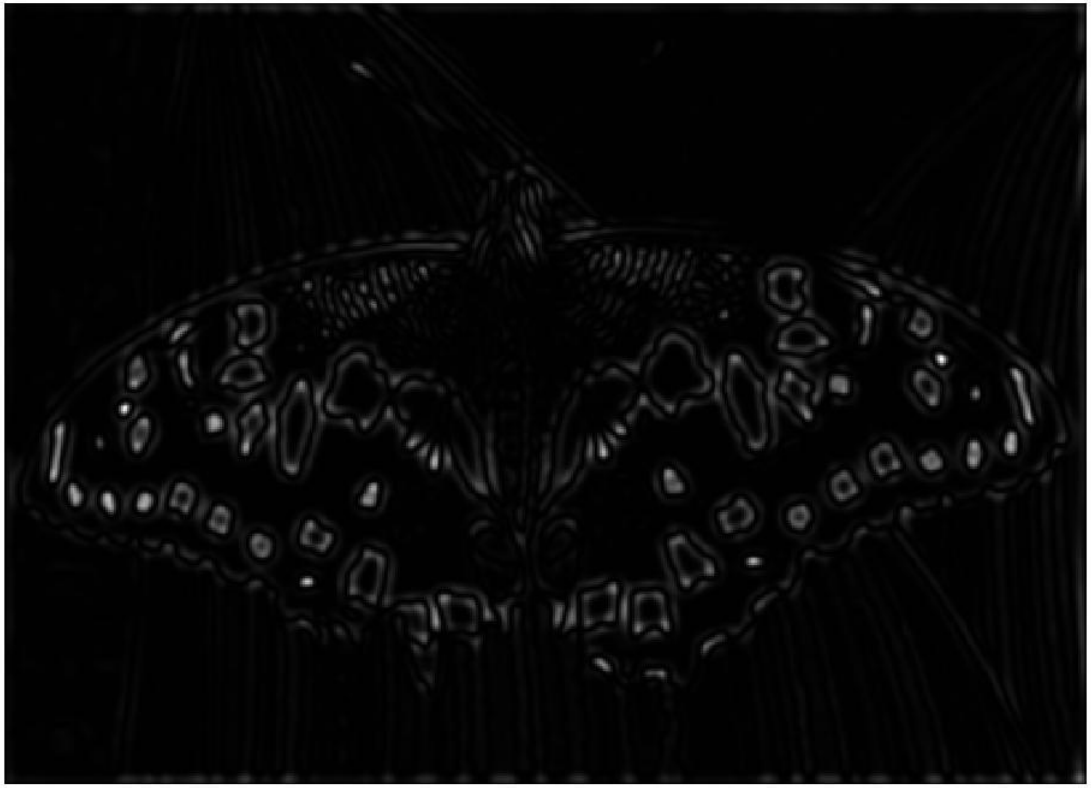
Scale-space detection: Example

DoG Images example



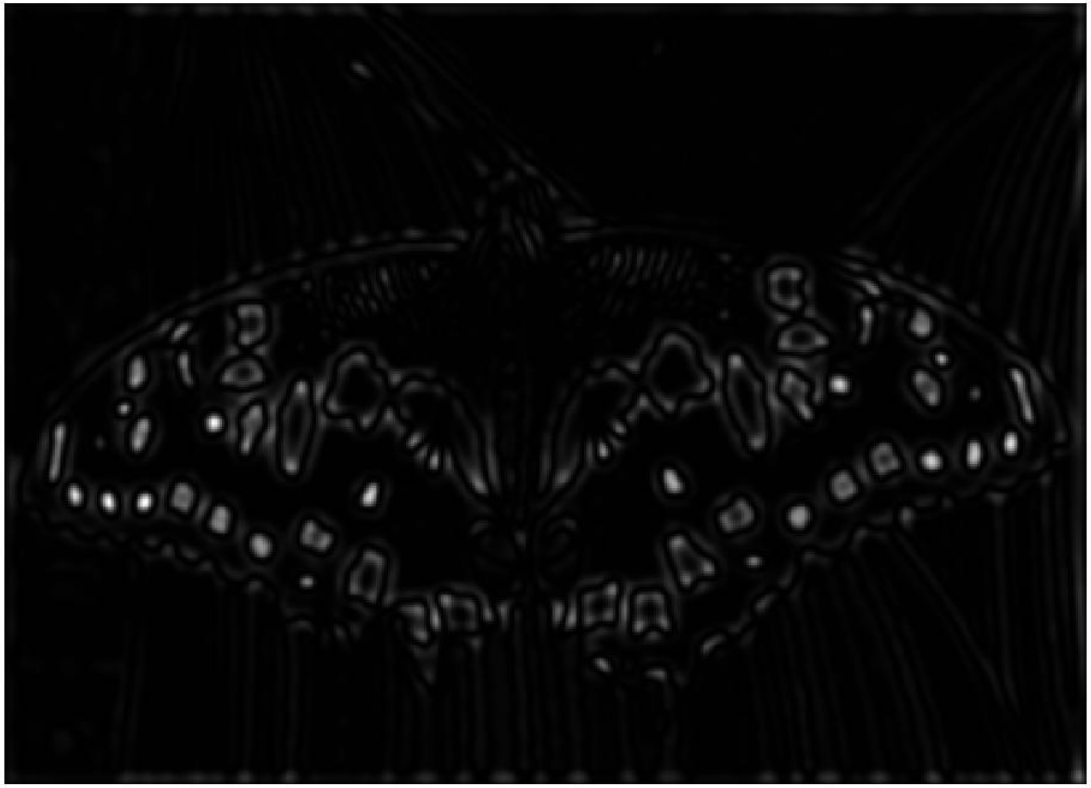
$G(k\sigma) - G(\sigma)$ with increasing σ starting from $\sigma = 1.6$ and $s = 6$ number of scales per octave

DoG Images example



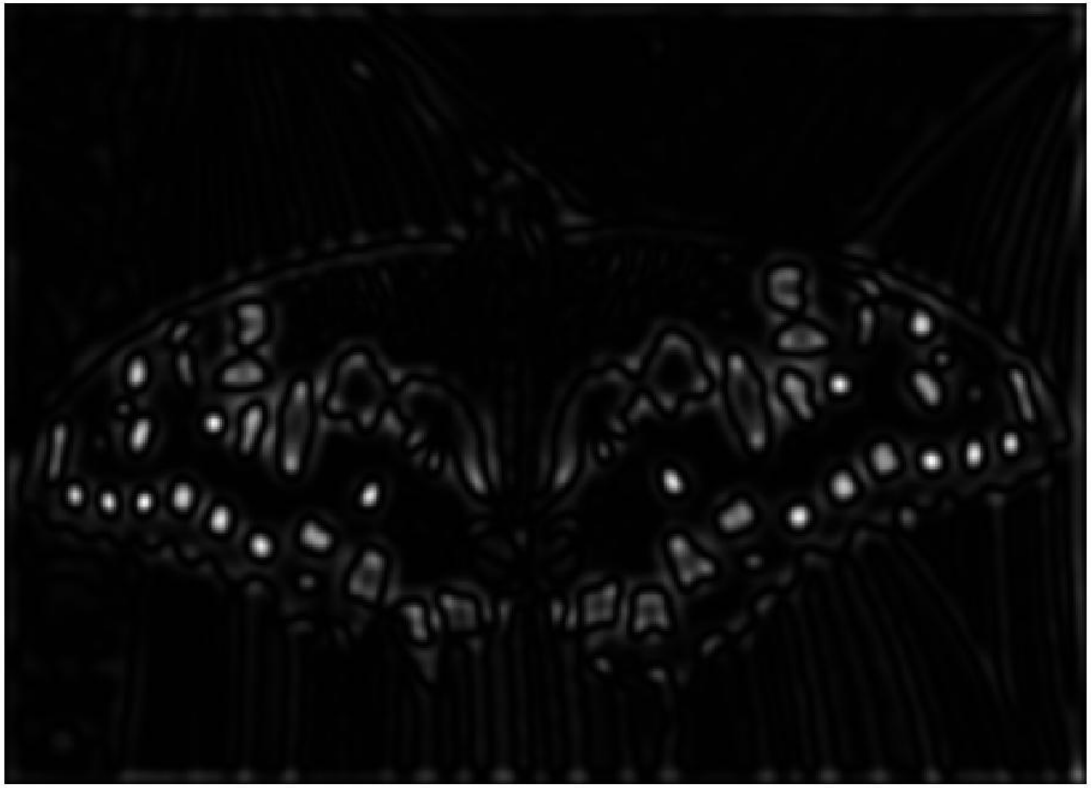
$G(k\sigma) - G(\sigma)$ with increasing σ starting from $\sigma = 1.6$ and $s = 6$ number of scales per octave

DoG Images example



$G(k\sigma) - G(\sigma)$ with increasing σ starting from $\sigma = 1.6$ and $s = 6$ number of scales per octave

DoG Images example

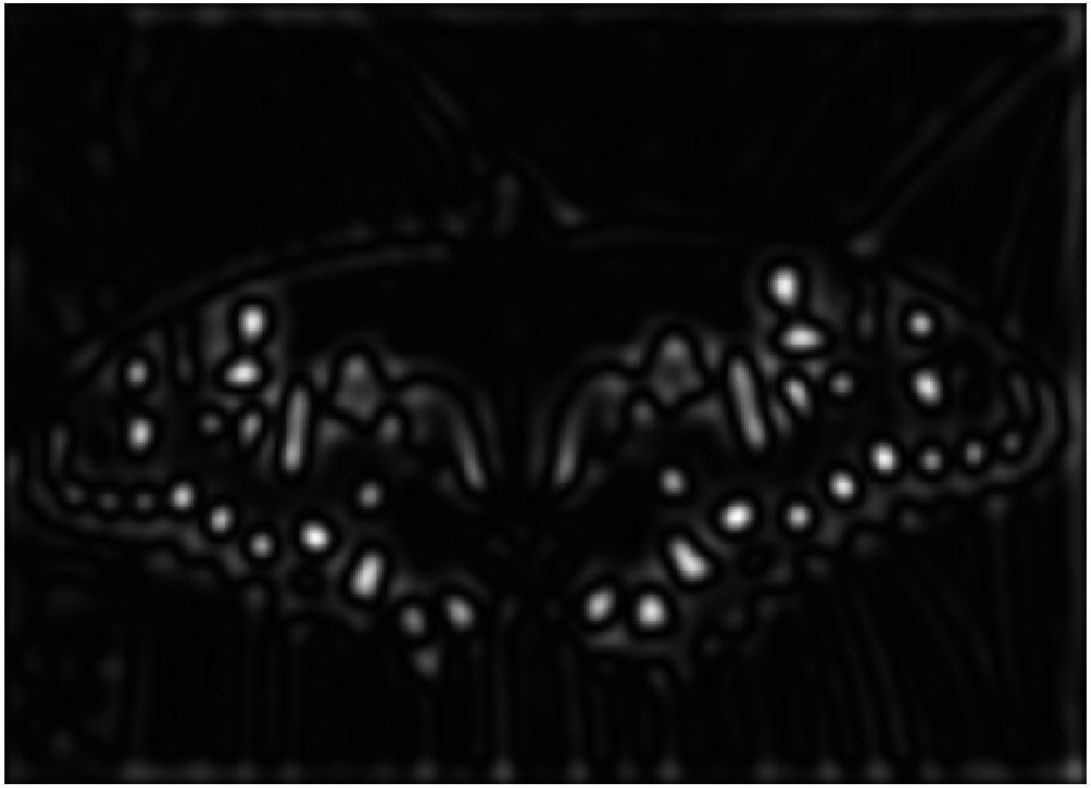


$G(k\sigma) - G(\sigma)$ with increasing σ starting from $\sigma = 1.6$ and $s = 6$ number of scales per octave

DoG Images example

$G(k\sigma) - G(\sigma)$ with increasing σ starting from $\sigma = 1.6$ and $s = 6$ number of scales per octave

DoG Images example



$G(k\sigma) - G(\sigma)$ with increasing σ starting from $\sigma = 1.6$ and $s = 6$ number of scales per octave

DoG Images example

$G(k\sigma) - G(\sigma)$ with increasing σ starting from $\sigma = 1.6$ and $s = 6$ number of scales per octave

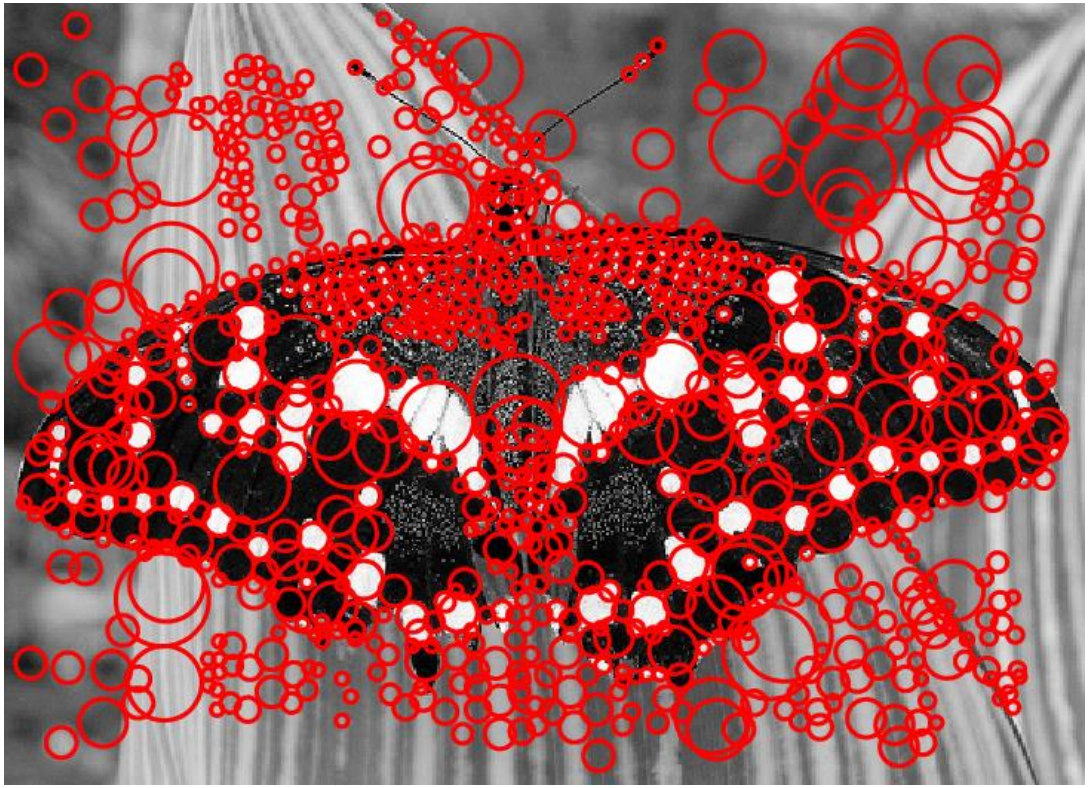
DoG Images example

$G(k\sigma) - G(\sigma)$ with increasing σ starting from $\sigma = 1.6$ and $s = 6$ number of scales per octave

DoG Images example

$G(k\sigma) - G(\sigma)$ with increasing σ starting from $\sigma = 1.6$ and $s = 6$ number of scales per octave

All DoG local maxima and minima



SIFT Features: Summary

- **SIFT: Scale Invariant Feature Transform** [Lowe, IJCV 2004]
- An approach to **detect and describe** regions of interest in an image.
  - NB: SIFT detector = DoG detector
- SIFT features are **reasonably invariant** to changes in **rotation, scaling**, and **changes in viewpoint** (up to 60deg) **and illumination**
- **Real-time but still slow (10 Hz on an i7 laptop)**
  - Expensive steps are the scale detection and descriptor extraction

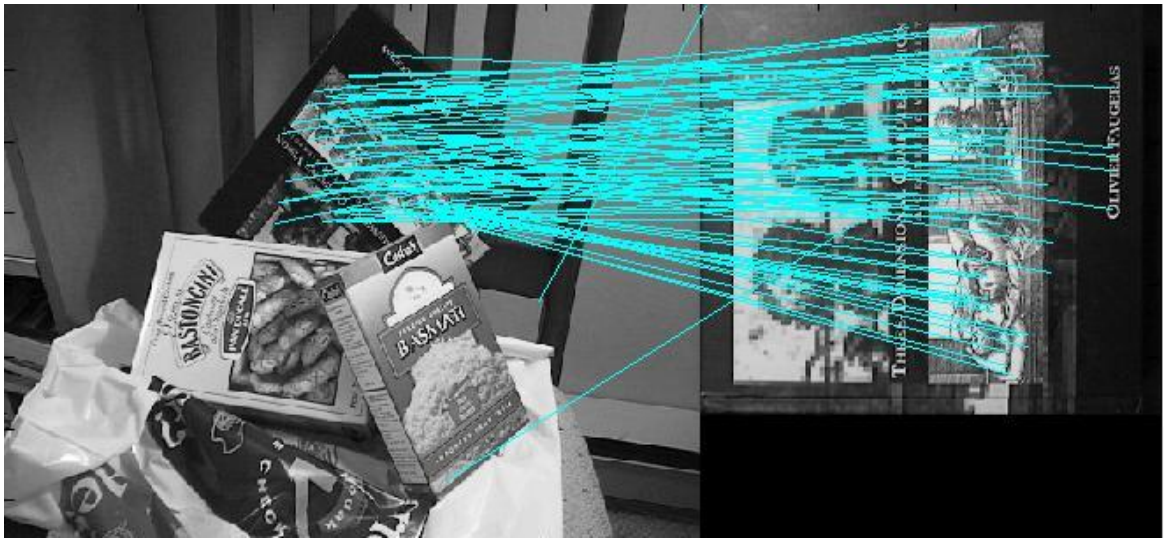
SIFT Demo

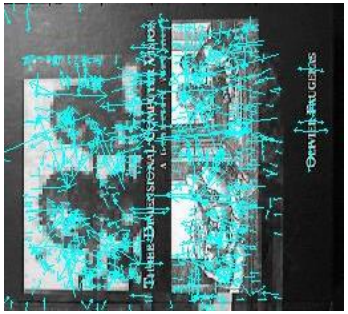
- Download original SIFT binaries and Matlab function from : http://people.cs.ubc.ca/~lowe/keypoints

```
>>[image1, descriptor1s, locs1] = sift('scene.pgm');
>>showkeys(image1, locs1);

>>[image2, descriptors2, locs2] = sift('book.pgm');
>>showkeys(image2, locs2);

>>match('scene.pgm','book.pgm');
```

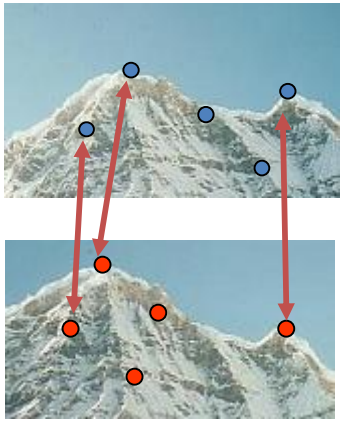


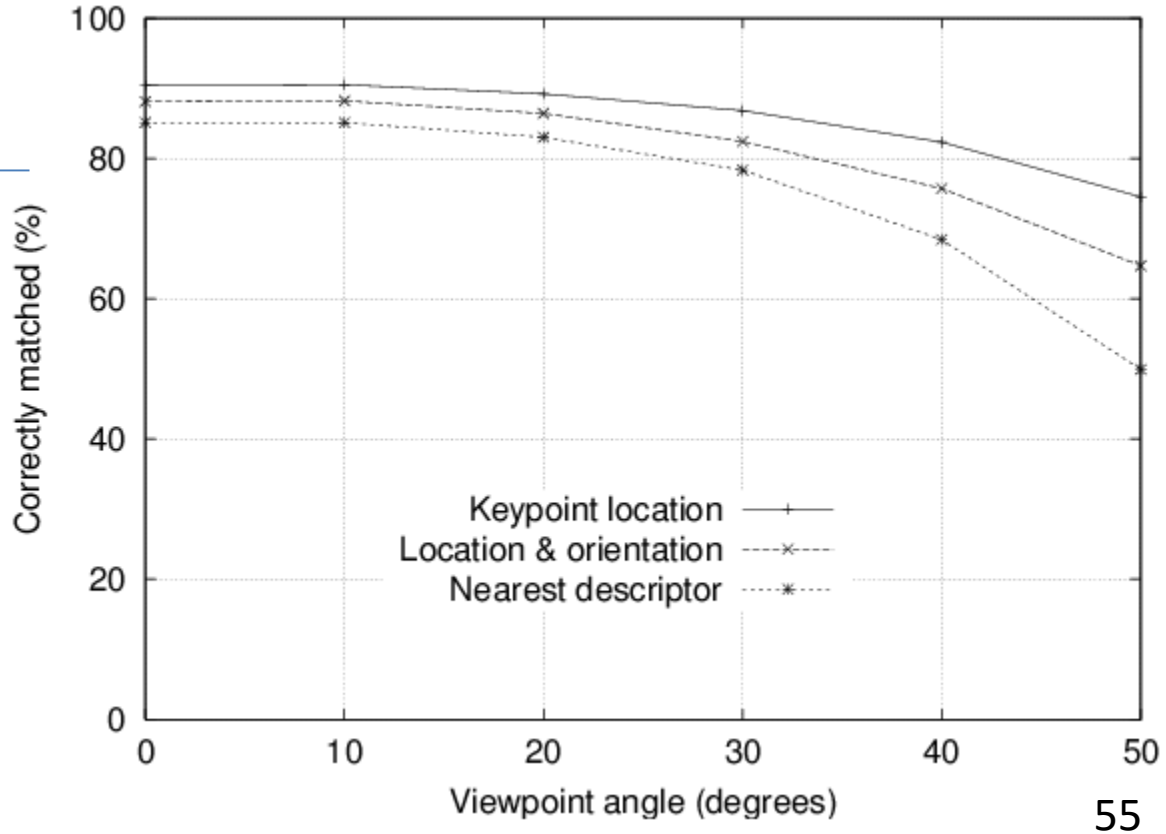


SIFT repeatability vs. viewpoint angle

Repeatability=

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



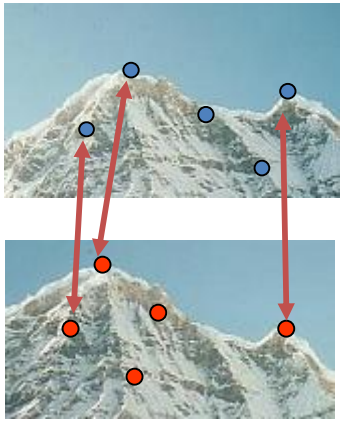


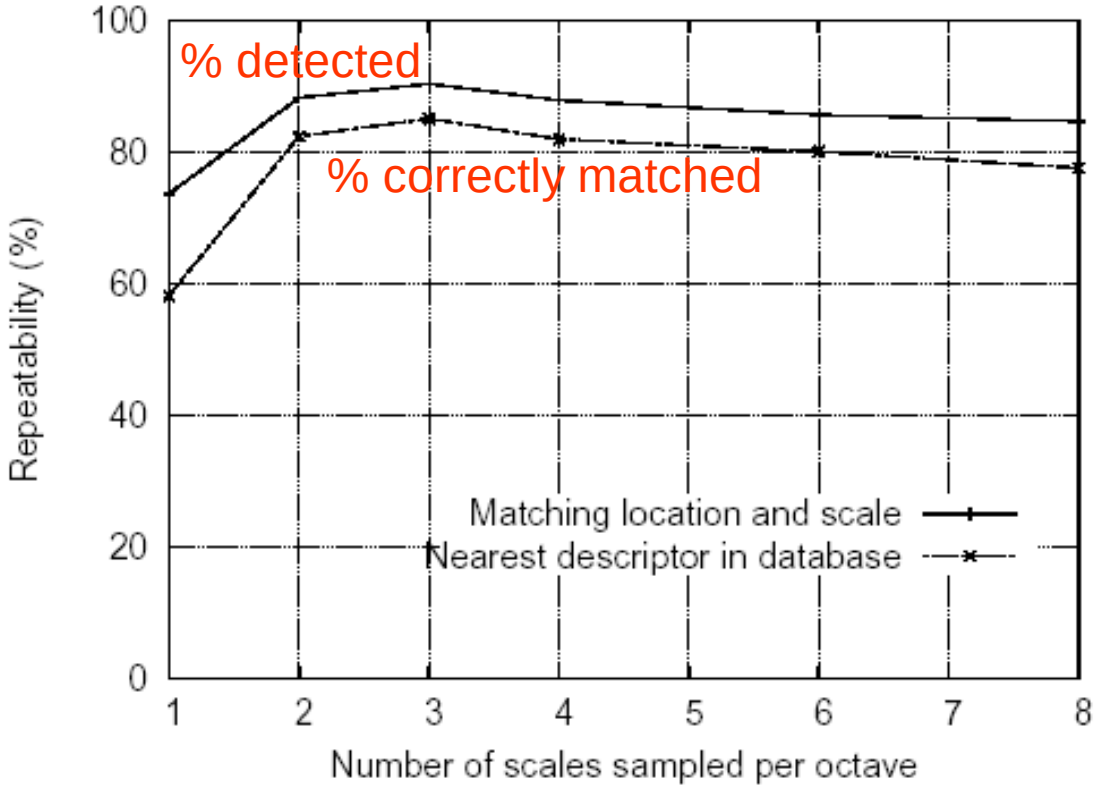
SIFT repeatability vs. Scale

The highest repeatability is obtained when sampling 3 scales per octave!

Repeatability=

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





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, but the results continue to improve up to a 4x4 array of histograms with 8 orientations. After that, adding more orientations or a larger descriptor can actually hurt matching by making the descriptor more sensitive to distortion.

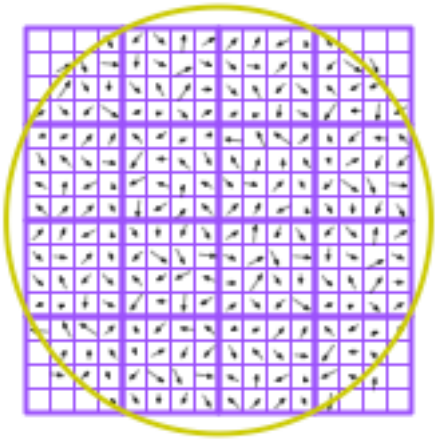


Image gradients

4x4 HOGs

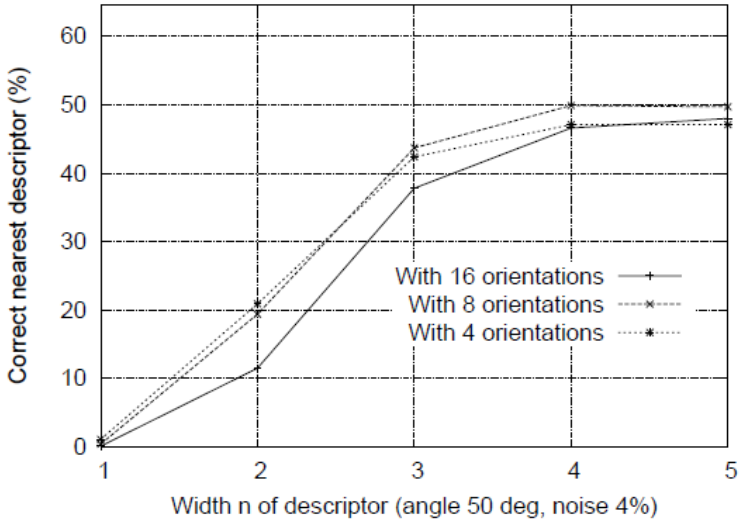
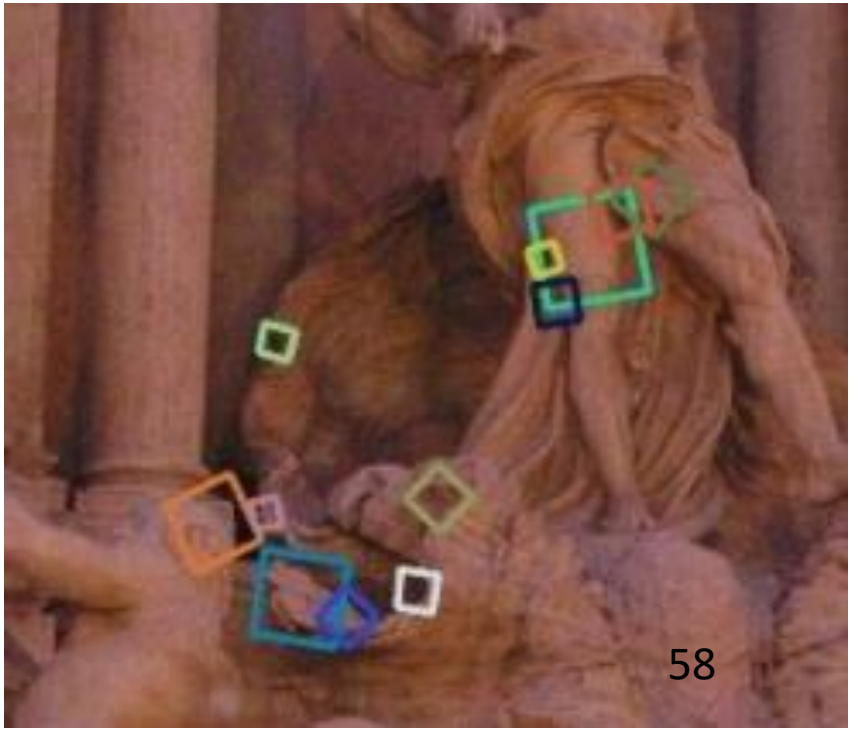


Figure 8: This graph shows the percent of keypoints giving the correct match to a database of 40,000 keypoints as a function of width of the $n \times n$ keypoint descriptor and the number of orientations in each histogram. The graph is computed for images with affine viewpoint change of 50 degrees and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

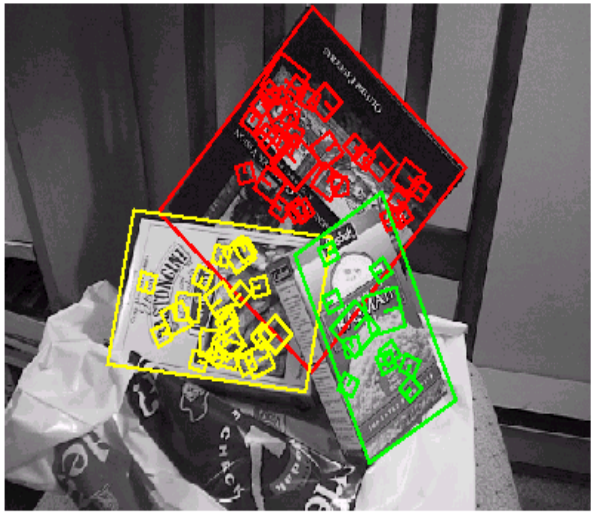
How many parameters are used to define a SIFT feature?

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



SIFT for Object recognition

- Can be simply implemented by returning as best object match the one with the largest number of correspondences with the template (object to detect)
- 4 or 5 point RANSAC can be used to remove outliers (see next lectures)



SIFT for Panorama Stitching

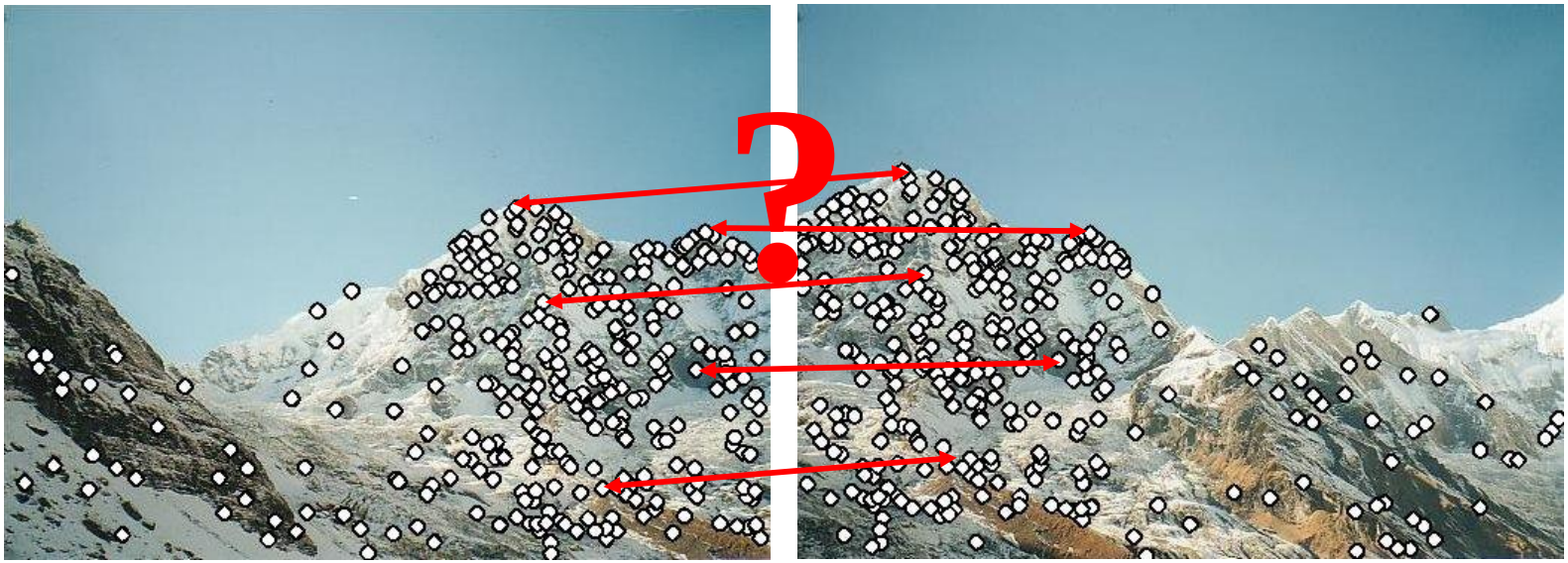
AutoStitch: http://matthewalunbrown.com/autostitch/autostitch.html

[M. Brown and D. G. Lowe. Recognising Panoramas. ICCV 2003]

Main questions

- What points are distinctive (i.e., *features*, *keypoints, salient* points), such that they are *repeatable? (i.e.,* can be re-*detected* from other views)
- How to *describe* a local region?
- 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, SAD, NCC or Hamming distance for binary descriptors (e.g., Census, BRIEF, BRISK)
  2. **Brute-force matching:**
     1. Test all the features in I_2
     2. Take the one at min distance

- **Issues with closest descriptor**: can give good scores to very ambiguous (bad) matches (curse of dimensionality)
- **Better approach**: compute ratio of distances to 1st to 2nd closest match

$$d(f_1) \,/\, d(f_2) \;<\; Threshold\ (usually\ 0.8)$$

  - $d(f_1)$ is the distance of the closest neighbor
  - $d(f_2)$ is the distance of the 2nd closest neighbor

Distance ratio: Explanation

- In SIFT, the nearest neighbor is defined as the keypoint with minimum Euclidean distance. However, many features from an Image 1 may not have any correct match in Image 2 because they arise from background clutter or were not detected in the Image 1.
- An effective measure is obtained by comparing the distance of the **closest neighbor** to that of 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 second-closest match as providing an estimate of the density of false matches within this portion of the feature space and at the same time identifying specific instances of feature ambiguity.

SIFT Feature matching: distance ratio

The SIFT paper recommends to use a threshold on 0.8. Where does this come from?

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

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

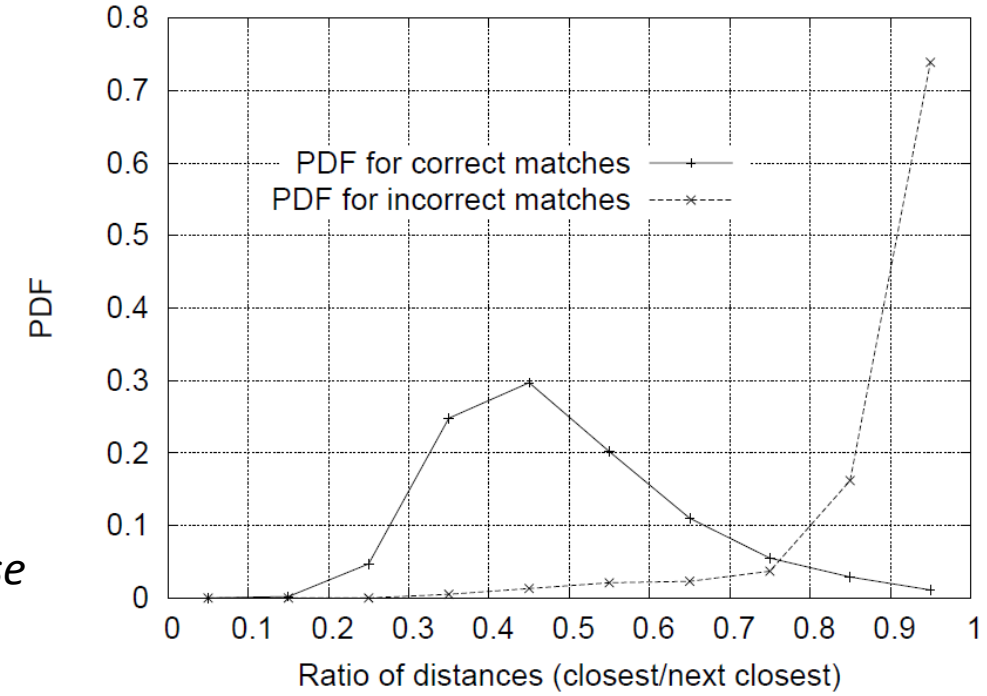


Figure 11: The probability that a match is correct can be determined by taking the ratio of distance from the closest neighbor to the distance of the second closest. Using a database of 40,000 keypoints, the solid line shows the PDF of this ratio for correct matches, while the dotted line is for matches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

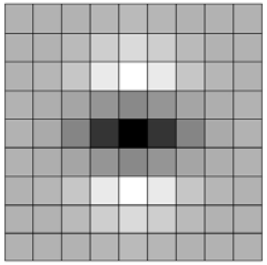
Outline

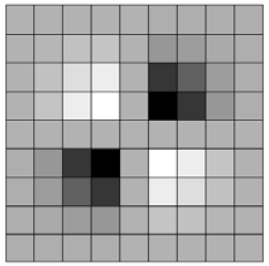
- Automatic Scale Selection
- The SIFT Detector (blob detector) and Descriptor
- Other corner and blob detectors and descriptors

SURF [Bay et al., ECCV 2006]

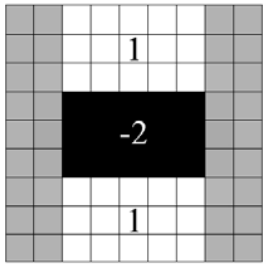
- ***S***peeded ***U***p ***R***obust ***F***eatures
- Based on ideas similar to **SIFT**
- Approximated computation for detection and descriptor
- Results comparable with SIFT, plus:
  - Faster computation
  - Generally shorter descriptors

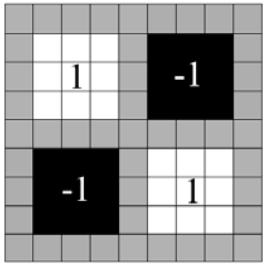
Gaussian second order partial derivatives





Approximation using **box filter**

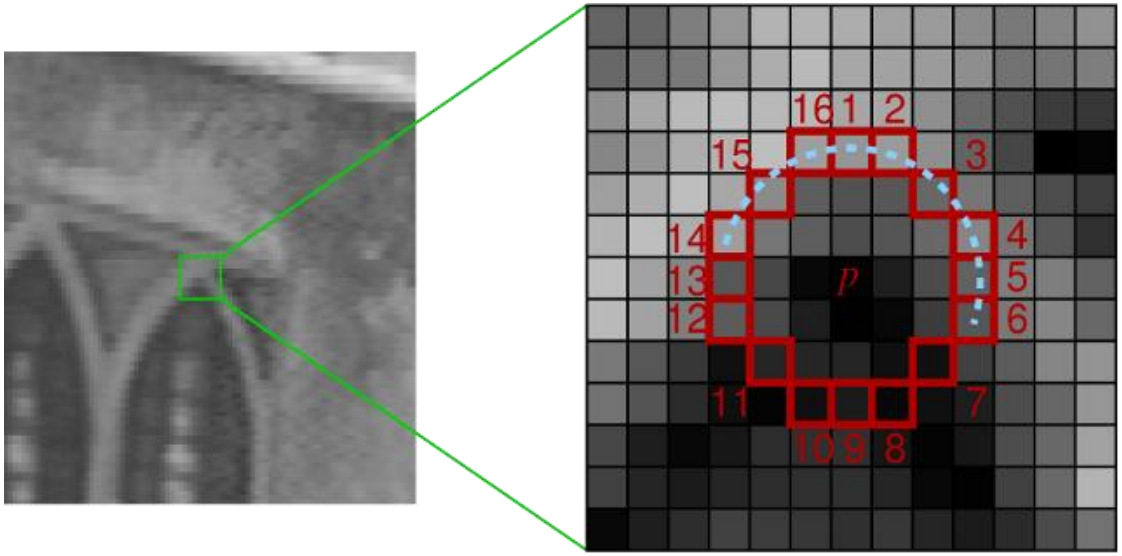




Bay, Tuytelaars, Van Gool, " Speeded Up Robust Features ", European Conference on Computer Vision (ECCV) 2006

FAST detector [Rosten et al., ICCV'05]

- **FAST**: **F**eatures from **A**ccelerated **S**egment **T**est
- Studies intensity of pixels on circle around candidate pixel $\boldsymbol{C}$
- $\boldsymbol{C}$ is a FAST corner **if** a set of $\boldsymbol{N}$ contiguous pixels on circle are:
  - all brighter than $\boldsymbol{intensity\_of(C) + theshold}$, or
  - all darker than $\boldsymbol{intensity\_of(C) + theshold}$



- Typically tests for **9** contiguous pixels in a **16**-pixel circumference
- **Very fast detector** - in the order of 100 Mega-pixel/second

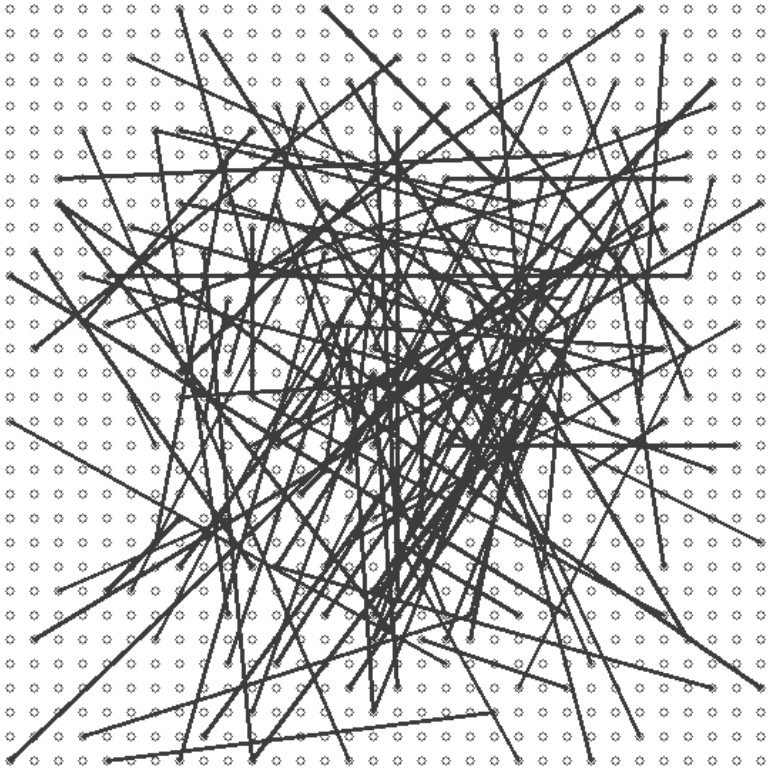
Rosten, Drummond, Fusing points and lines for high performance tracking, International Conference on Computer Vision (ICCV), 2005

BRIEF descriptor [Calonder et. al, ECCV 2010]

- **B**inary **R**obust **I**ndependent **E**lementary **F**eatures
- Goal: high speed (in description and matching)

- **Binary** descriptor formation:
  - Smooth image
  - **for each** detected keypoint (e.g. FAST),
  - ***sample*** 256 intensity pairs $({p_1}^i, {p_2}^i)$ $(i = 1, \dots, 256)$ within a squared patch around the keypoint
  - Create an empty 256-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 by **machine learning**) only once; then, the same pattern is used for all patches
- Pros: **Binary descriptor**: allows **very fast** Hamming distance matching: count the number of bits that are different in the descriptors matched
- Cons: **Not scale/rotation invariant**

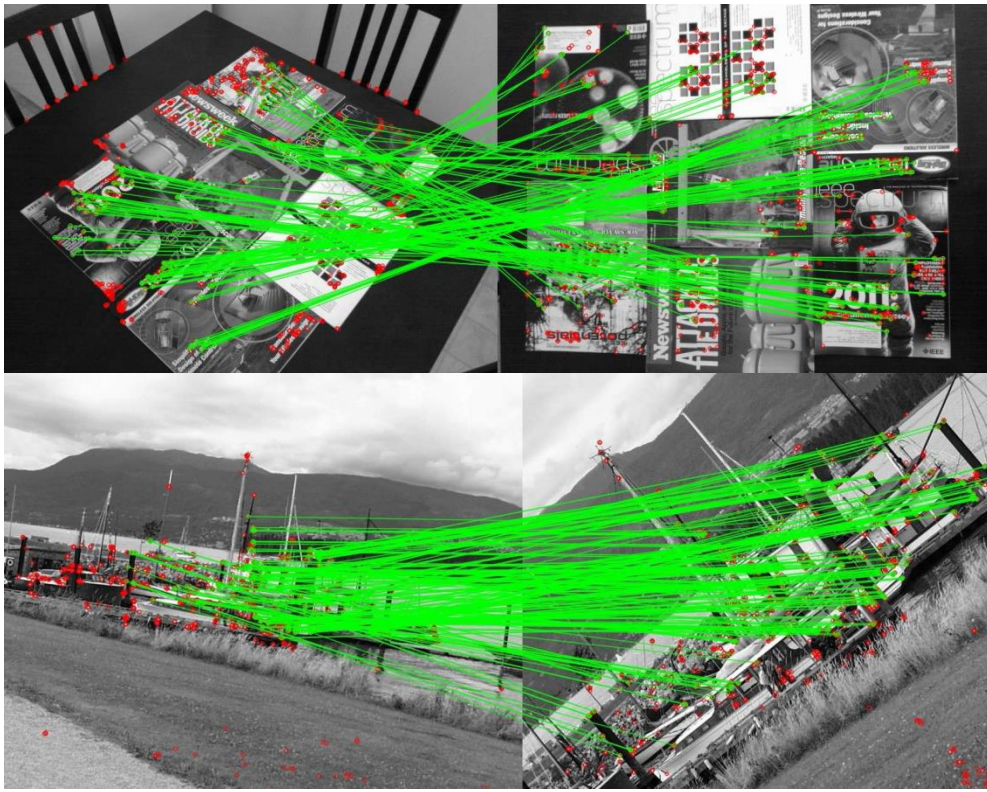


Pattern for intensity pair samples – generated randomly

Calonder, Lepetit, Strecha, Fua, BRIEF: Binary Robust Independent Elementary Features, ECCV'10]

ORB descriptor [Rublee et al., ICCV 2011]

- ***O***riented FAST and ***R***otated ***B***RIEF
- Keypoint detector based on **FAST**
- **BRIEF** descriptors are *steered* according to keypoint orientation (to provide rotation invariance)
- Good Binary features are learned by minimizing the correlation on a set of training patches.



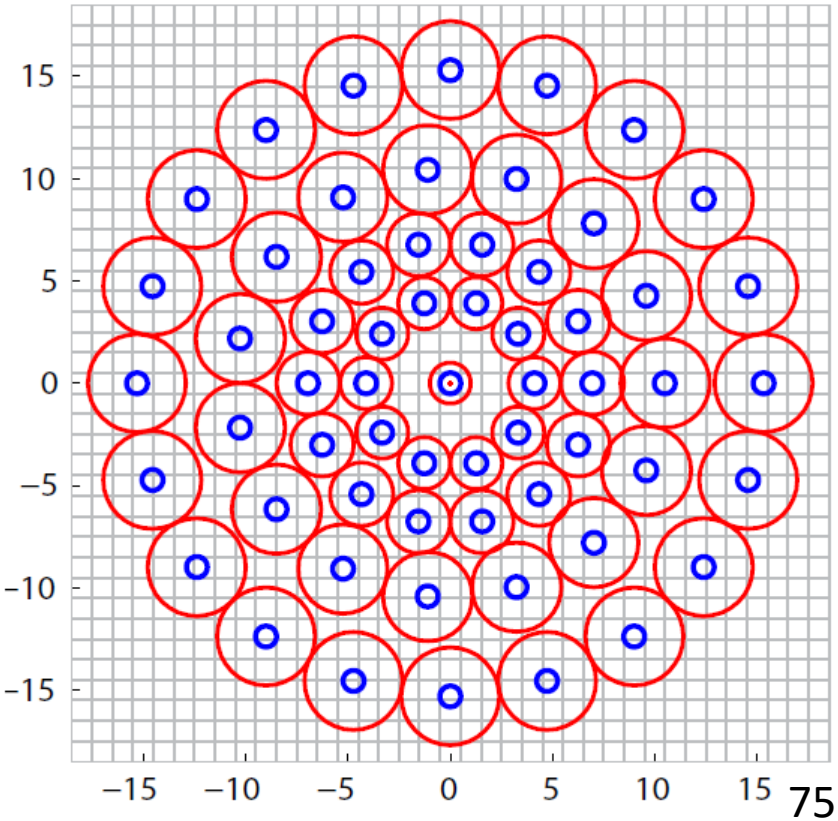
Rublee,Rabaud, Konolige, Bradski, (2011). "ORB: an efficient alternative to SIFT or SURF" (PDF). IEEE International Conference on Computer Vision (ICCV).

BRISK descriptor

[Leutenegger, Chli, Siegwart, ICCV 2011]

- **B**inary **R**obust **I**nvariant **S**calable **K**eypoints
- Detect corners in scale-space using FAST
- Rotation and scale invariant

- **Binary**, formed by pairwise intensity comparisons (like BRIEF)
- **Pattern** defines intensity comparisons in the keypoint neighborhood
- **Red circles**: size of the smoothing kernel applied
- **Blue circles**: smoothed pixel value used
- Compare short- and long-distance pairs for orientation assignment & descriptor formation
- Detection and descriptor speed: ~10 times faster than SURF
- Slower than BRIEF, but scale- and rotation- invariant



Recap Table

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

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

Reading

- Ch. 4.1 and Ch. 8.1 of Szeliski book
- Ch. 4 of Autonomous Mobile Robots book
- Ch. 13.3 of Peter Corke book