Lec. 5

Feature Detectors: SURF, SIFT

Assist. Prof. Dr. Saad Albawi

Scale Invariant Feature Transform (SIFT)

Interest Points

- it has a clear, preferably mathematically well-founded, definition
- it has a well-defined position in image space
- the local image structure around the interest point is rich in terms of local information contents, such that the use of interest points simplify further processing in the vision system
- it is stable under local and global perturbations in the image domain as illumination/brightness variations, such that the interest points can be reliably computed with high degree of reproducibility
- optionally, the notion of interest point should include an attribute of scale, to make it possible to compute interest points from real-life images as well as under scale changes

Feature Descriptors

Why need feature descriptors?

- Keypoints give only the positions of strong features.
- To match them across different images, have to characterize them by extracting feature descriptors.

What kind of feature descriptors?

- Able to match corresponding points across images accurately.
- Invariant to scale, orientation, or even affine transformation.
- Invariant to lighting difference.

ADVANTAGES OF INVARIANT LOCAL FEATURES?

Invariant to translation, rotation, scale changes

Robust or covariant to out-of-plane (≈affine) transformations

Robust to lighting variations, noise, blur, quantization

Locality: Features are local, therefore robust to occlusion and clutter.

Efficiency: Close to real-time performance.

- **Requirements**

- Region extraction needs to be repeatable (reliably finds the same interest points under different viewing conditions) and accurate
- We need a sufficient number of regions to cover the object.

Scale Invariant Feature Transform

- SIFT is an algorithm that finds interest point
- inspired by Harris corner detection
- the algorithm works the following way:

1. detection of extremes in scale-space representation
2. adjustment of the position of interest points
3. assignment of orientation to the interest points
4. construction of the descriptor of interest point

SIFT: Scale Invariant Feature Transform; transform image data into scale-invariant coordinates relative to local features

- Invariances:
  - Scaling
  - Rotation
  - Illumination
  - Deformation

Scale-Invariant Feature Transform

- Generates image features, "keypoints"
  - invariant to image scaling and rotation
  - partially invariant to change in illumination and 3D camera viewpoint
  - many can be extracted from typical images
  - highly distinctive

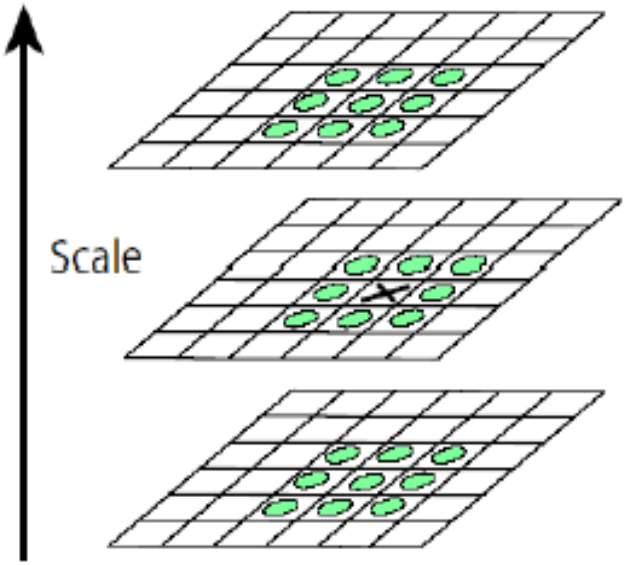
SIFT [Lowe]

SIFT Algorithm Stages

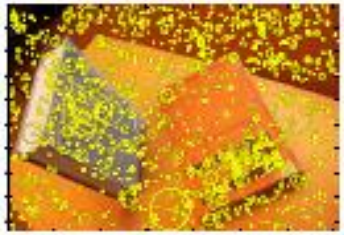
- Scale-space Extrema Detection
  - Uses difference-of-Gaussian function
- Keypoint Localization
  - Sub-pixel location and scale fit to a model
- Orientation assignment
  - 1 or more for each keypoint
- Keypoint descriptor
  - Created from local image gradients

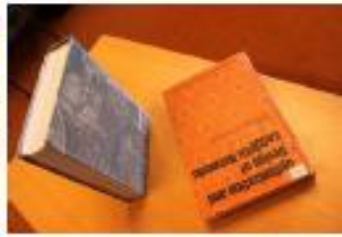
Extrema Detection

- Keypoint must be a minima or maxima of its 8 neighbors at it's scale and the 9 neighbors above and 9 below.



FINDING "KEYPOINTS"(CORNERS)

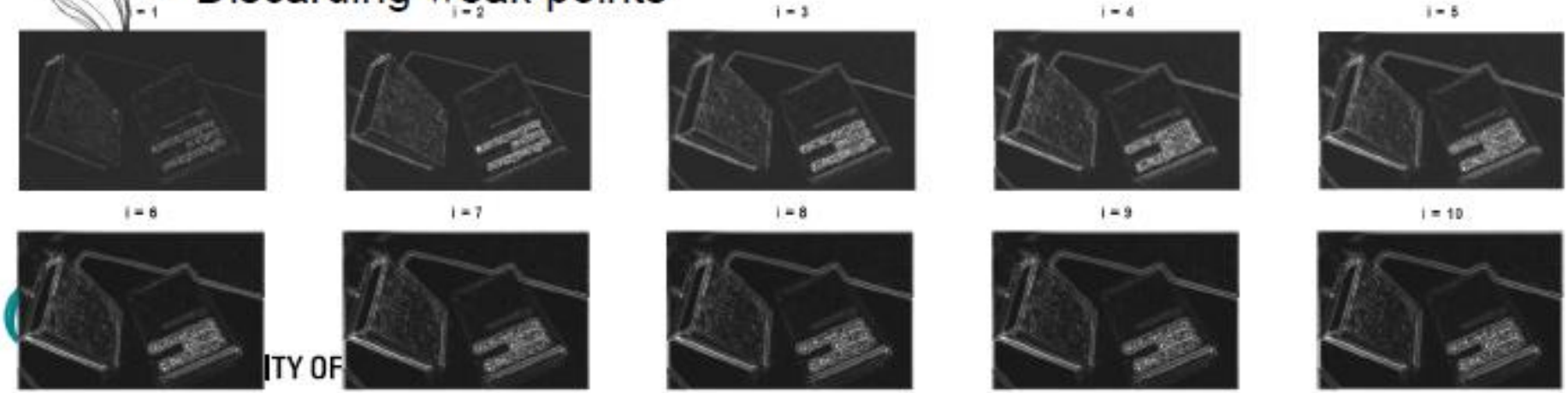




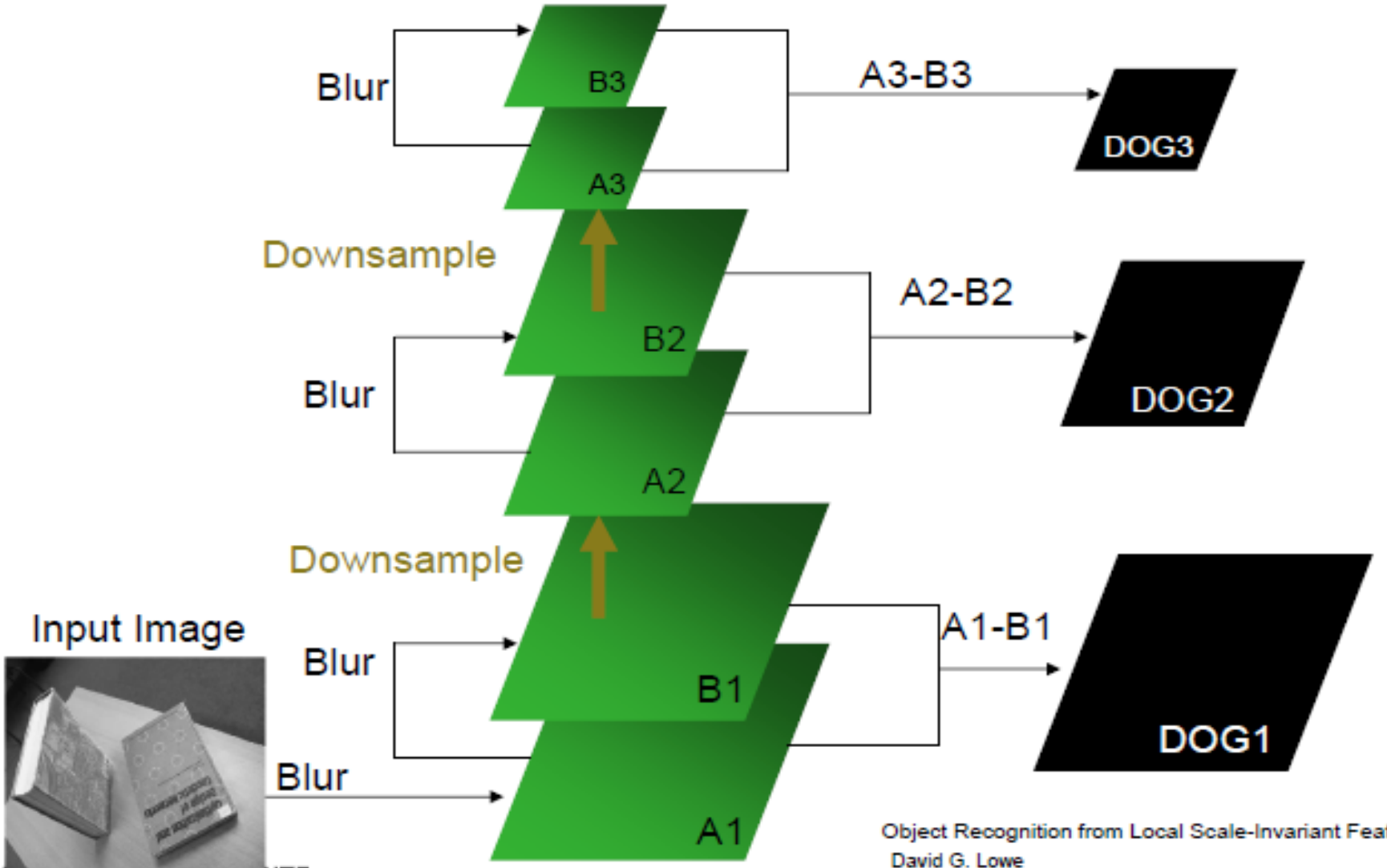
Idea: Find keypoints, but scale invariant

Approach:

- Run linear filter (diff. of Gaussians) at different resolutions (scale) of image pyramid
- Build difference of Gaussian DoG pyramid
- Locate extremas of DoG pyramid (xi,yi,σi)
- Refine the keypoint localization
- Discarding weak points



DIFFERENCE OF GAUSSIAN PYRAMID

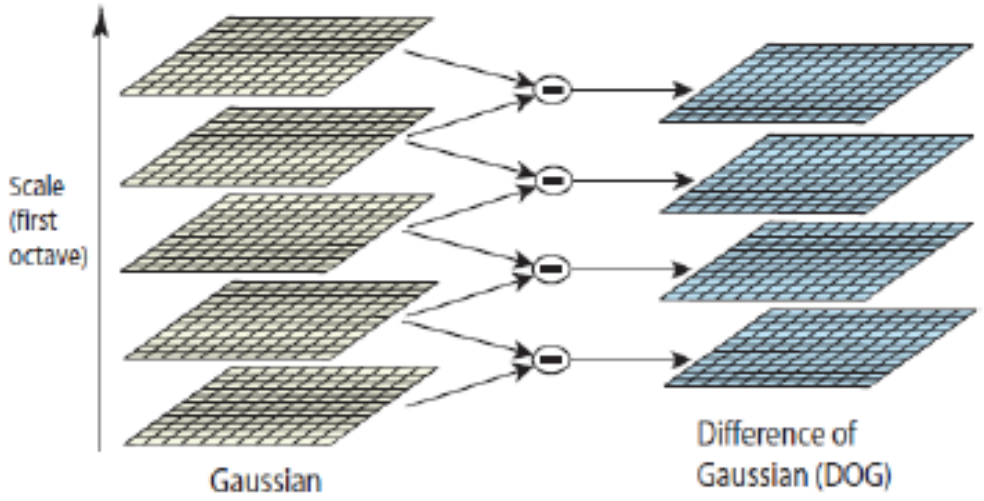


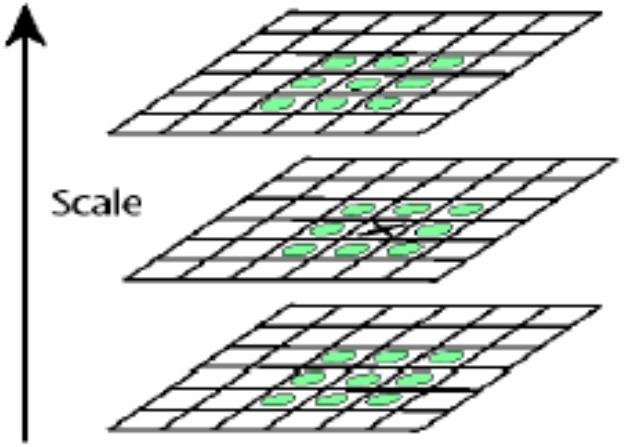
Object Recognition from Local Scale-Invariant Features
David G. Lowe
Presented by Ashley L. Kapron

DIFFERENCE OF GAUSSIAN PYRAMID

Detect maxima and minima of difference-of-Gaussian in scale space by

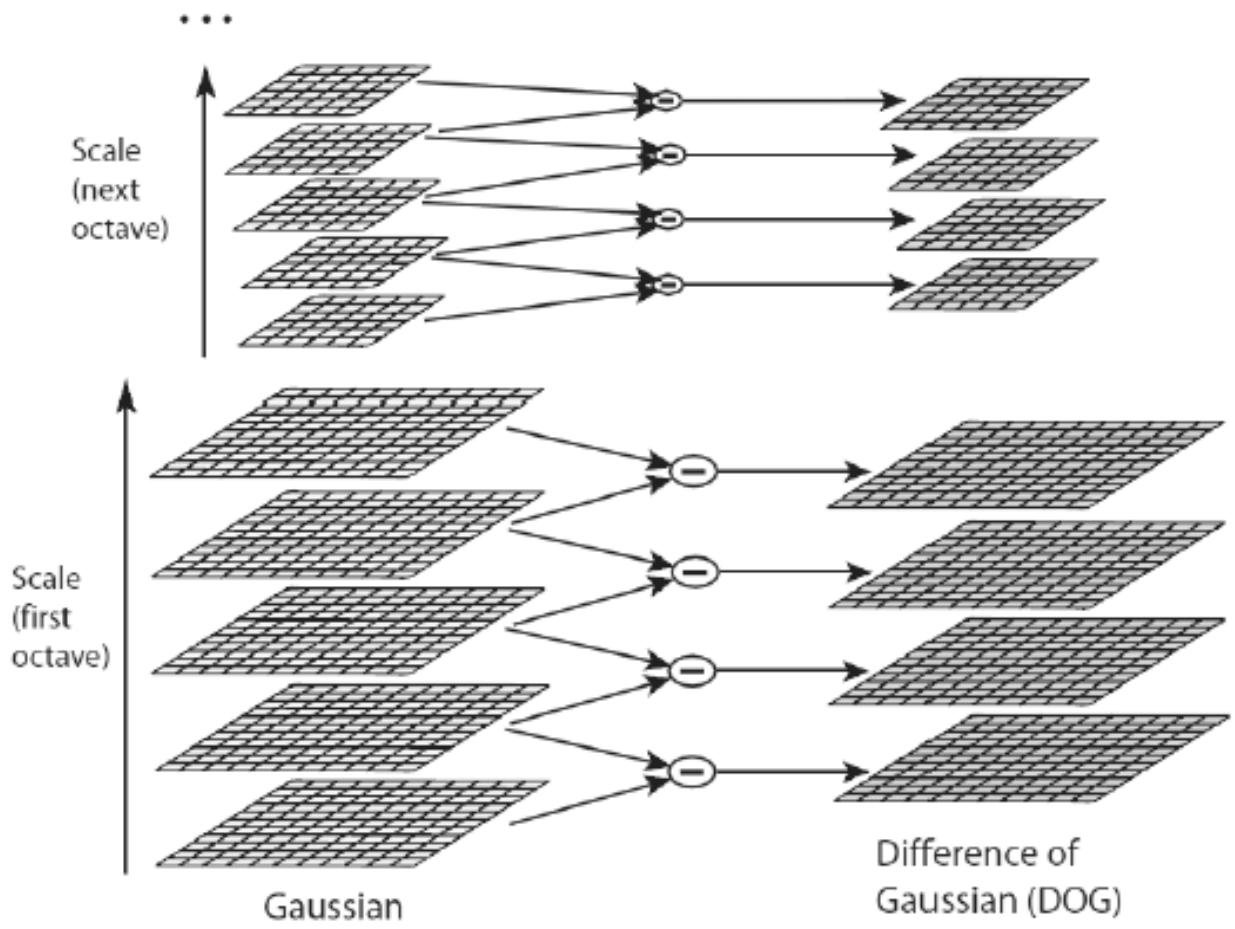
- comparing each pixel in the pyramid to its 26 neighbors (8 neighbors at the same level of the pyramid)



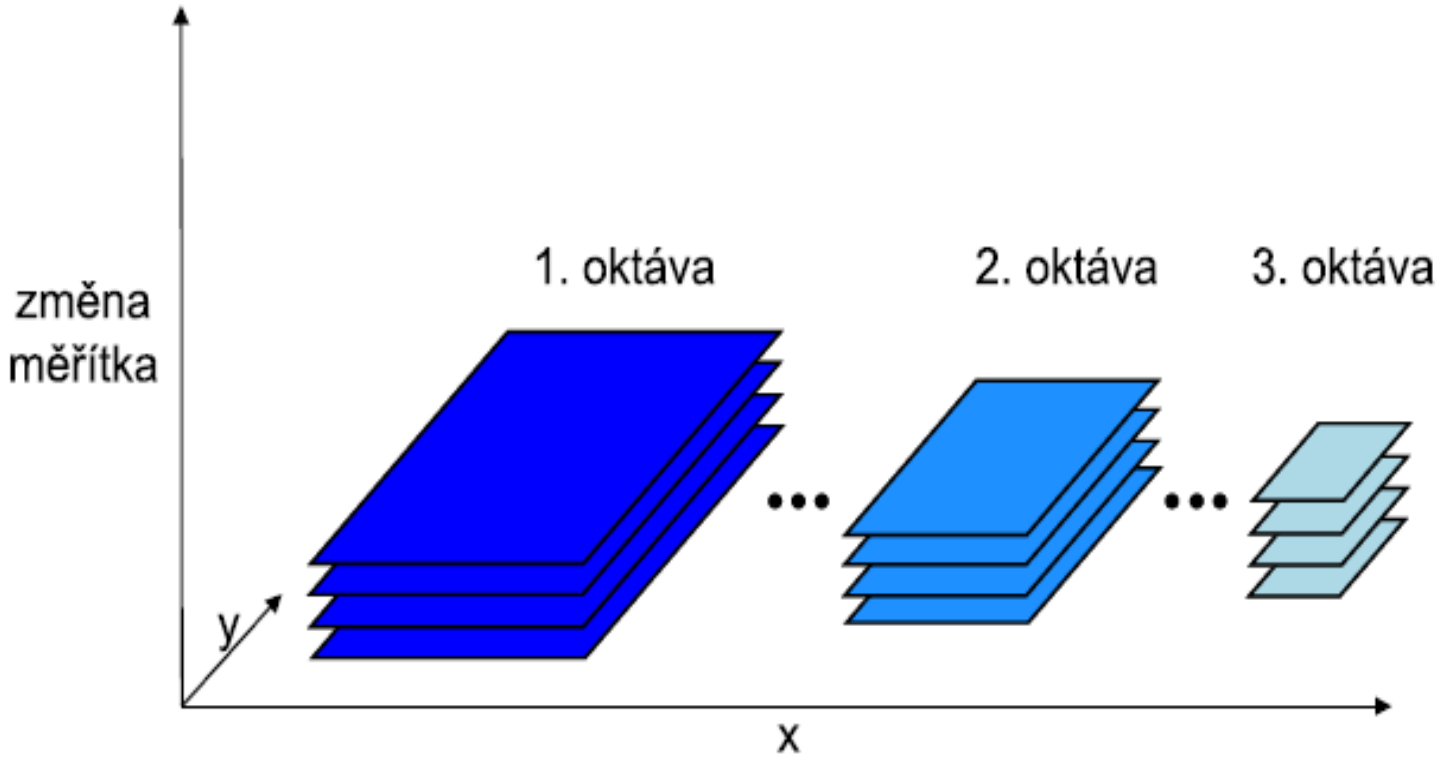


Maxima and minima in a (3*3*3) neighborhood

This is followed by a refining and filtering steps.



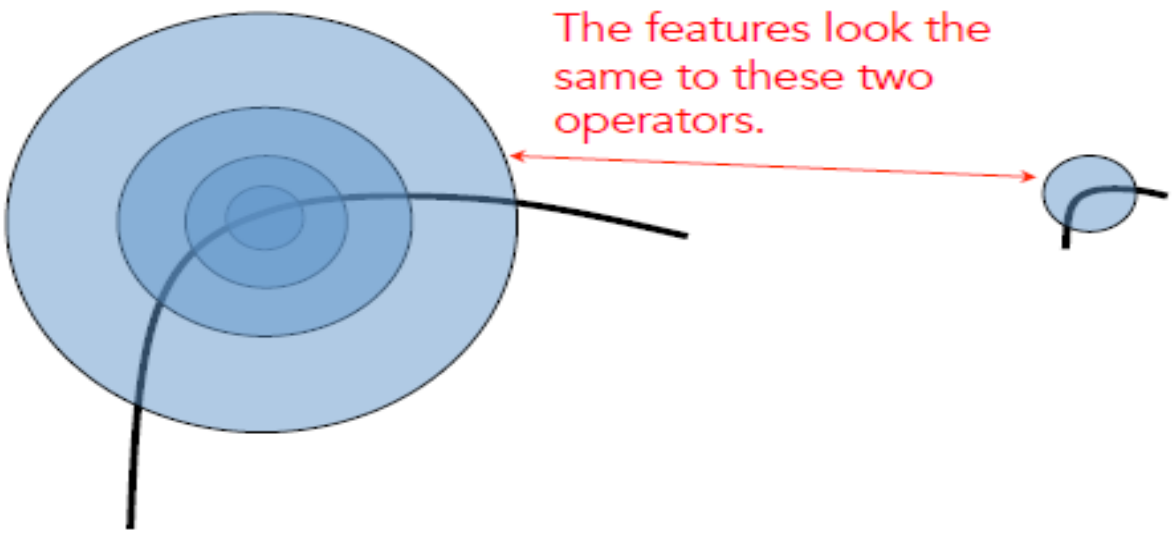
- Have 3 different scales within each octave (doubling of σ).
- Successive DOG images are subtracted to produce D.
- D images in a lower octave are downsampled by factor of 2.



Obrázek: Scale-space representations

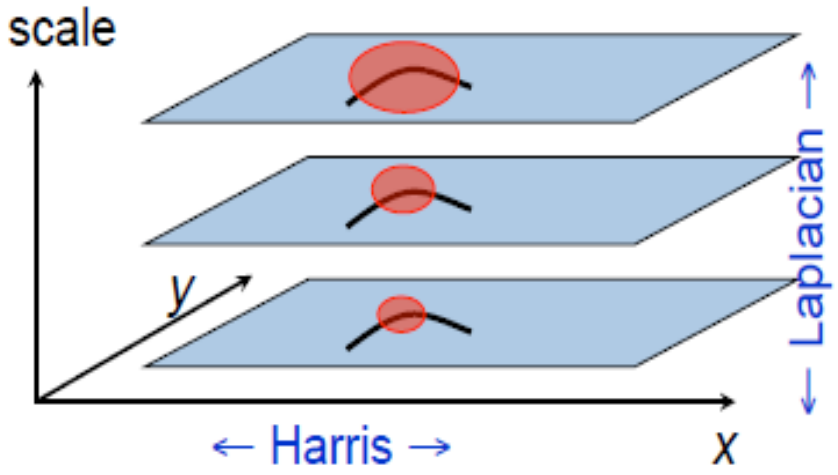
Scale Invariant Detection

- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images

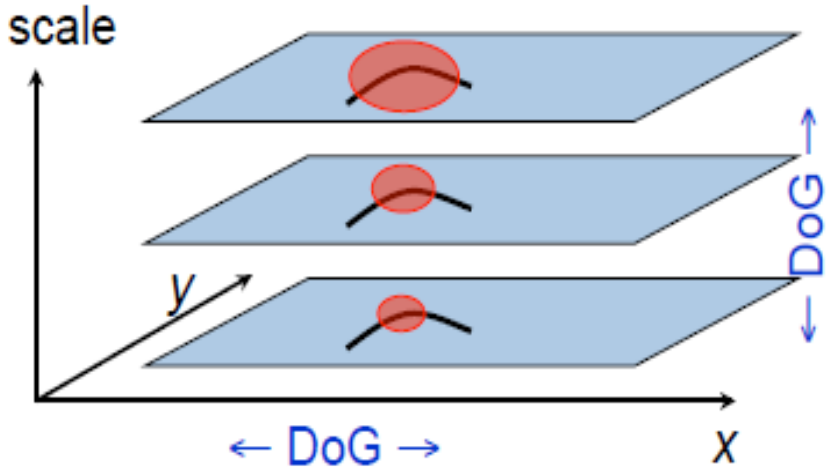


Scale Invariant Detectors

- Harris-Laplacian[1]
  Find local maximum of:
  - Harris corner detector in space (image coordinates)
  - Laplacian in scale



- SIFT (Lowe)[2]
  Find local maximum of:
  - Difference of Gaussians in space and scale



DISCARDING LOW-CONTRAST KEYPOINTS

- The elimination strategy of weak points is simply by testing against a threshold:

$$|D(x_i', y_i', \sigma_i')| > threshold \text{ (0.03-Lowe)}$$

After scale space extremas are detected

SIFT algorithm discards low contrast keypoints

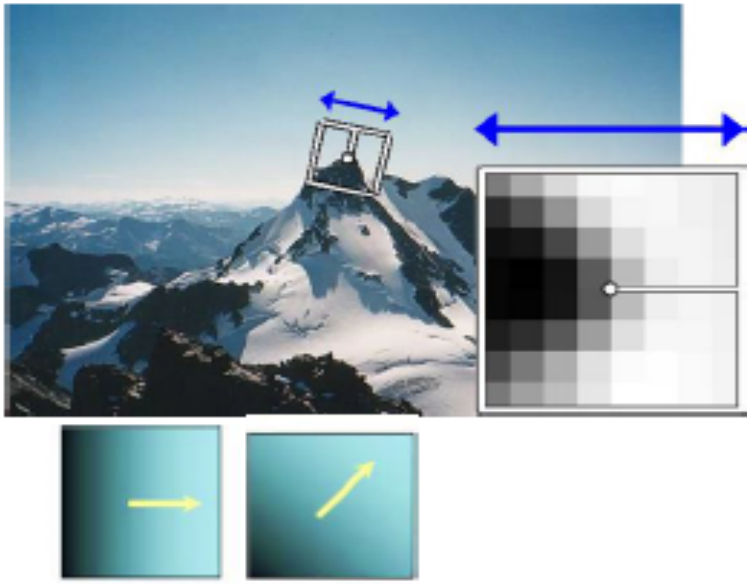
WHAT IS NEXT AFTER DETECTION?

After the detection of points,

How to *describe* them for matching?

Rotation Invariant Descriptors

Descriptors characterize the local neighborhood of a point in a vector. This locality gives robustness to illumination changes.



Find local orientation

– Dominant direction of gradient for the Image patch

Rotate patch according to this angle

– This puts the patches into a canonical orientation.

How to achieve invariance in image matching

Two steps:

1. Make sure your feature *detector* is invariant
   - Harris is invariant to translation and rotation
   - Scale is trickier
     - common approach is to detect features at many scales using a Gaussian pyramid (e.g., MOPS)
     - More sophisticated methods find "the best scale" to represent each feature (e.g., SIFT)
2. Design an invariant feature *descriptor*
   - A descriptor captures the intensity information in a region around the detected feature point
   - The simplest descriptor: a square window of pixels
     - What's this invariant to?
   - Let's look at some better approaches…

Scale Invariant Feature Transform (SIFT) [Low04].

Convolve input image I with Gaussian G of various scale σ:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{16}$$

where

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right). \tag{17}$$

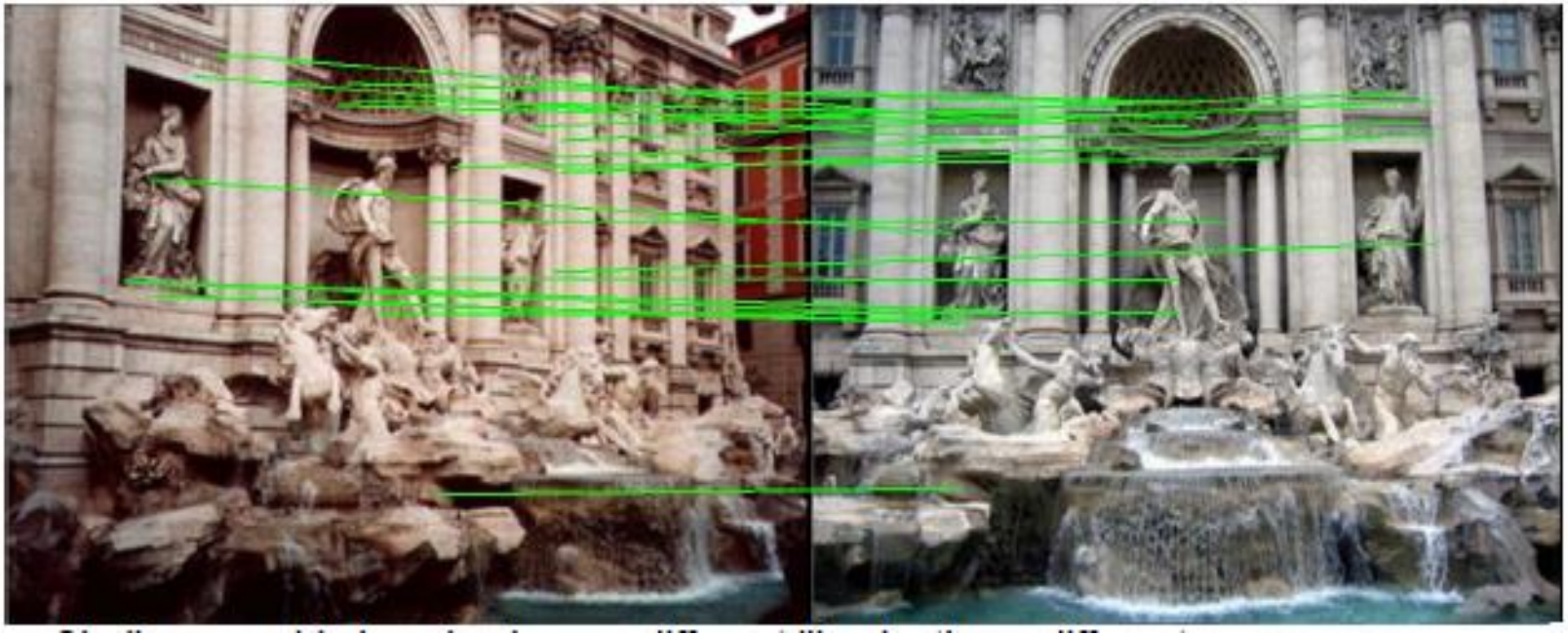
This produces L at different scales.

To detect stable keypoint, convolve image I with difference of Gaussian:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned} \tag{18}$$

MAJOR APPLICATIONS

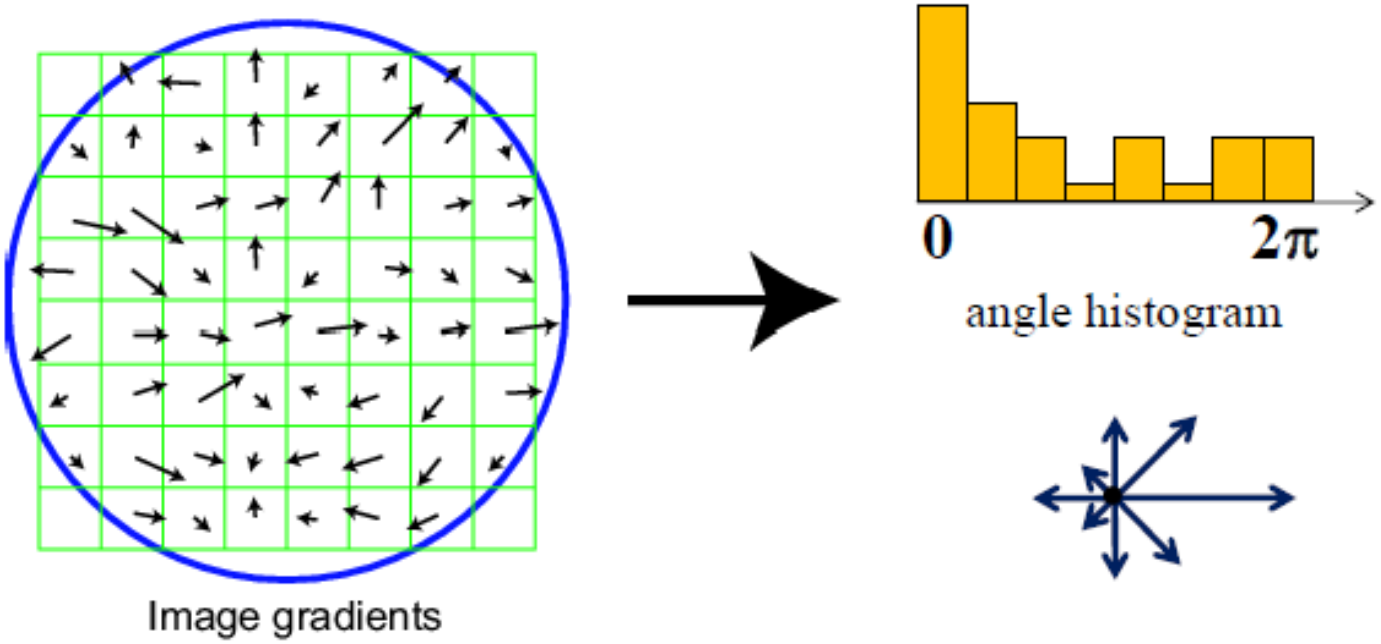
- Wide baseline stereo
- Motion tracking
- Panoramas and mosaicking
- 3D reconstruction
- Recognition



Scale Invariant Feature Transform

Basic idea:

- Take 16x16 square window around detected interest point (8x8 shown below)
- Compute edge orientation (angle of the gradient minus 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations (8 bins)



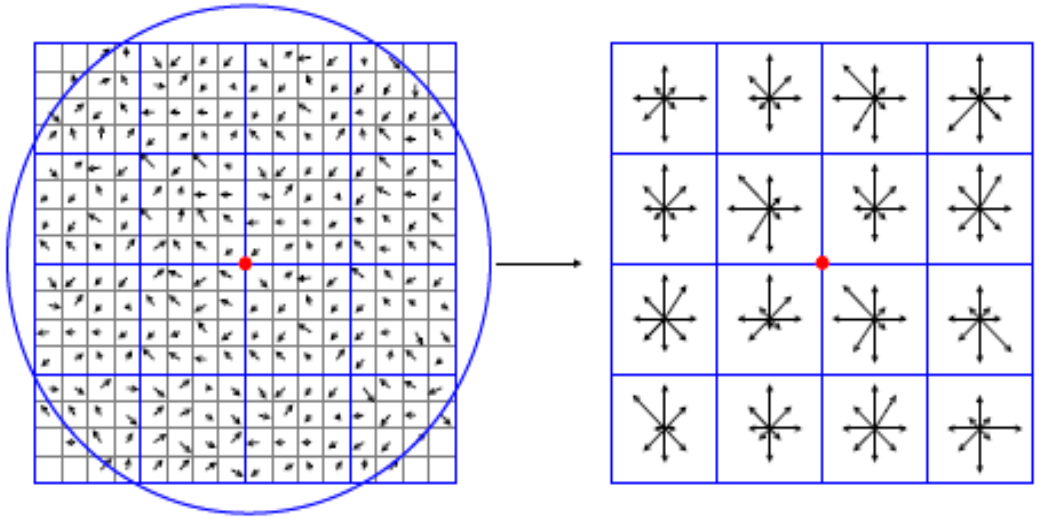
ORIENTATION

The summarized procedure is lustrated as follows:

- Compute the smoothed image
- Compute the gradient magnitude and orientation in neighborhood of (xi', yi')
- Compute the histogram of orientations
- The highest peak in the histogram is assigned for orientation

The Key-point descriptor

- a description should be independent on geometric and brightness transformations
- the neighborhood of the key-point is divided into 4x4 regions
- in each region the gradients are computed
- the orientations of the gradients are then rotated to align with the dominant direction
- they are concatenated into a 128-dimensional feature vector



gradienty v okolí

finální deskriptor

SIFT DESCRIPTORS

The methodology is implemented as follows:

- Compute image gradient over 16x16 array of locations centered at (xi, yi) in the Gaussian smoothed image at the scale of the keypoint.
- Create an array of orientation histograms by computing the gradient orientation with respect to the keypoint gradient orientation.
- Compute the orientation histogram of 8 orientations (45^o covering) in a 4x4 pixel block which results in 128 vector descriptor.
- Compute the weights of contribution of each pixel in the orientation histogram according to its closeness to the keypoint center.
- Normalize to unit length to reduce the effect of illumination change

SIFT DESCRIPTORS

1- Compute image gradient over 16x16 array of locations centered at (xi, yi) in the Gaussian smoothed image at the scale of the keypoint.

2- Create an array of orientation histograms by computing the gradient orientation with respect to the keypoint gradient orientation.

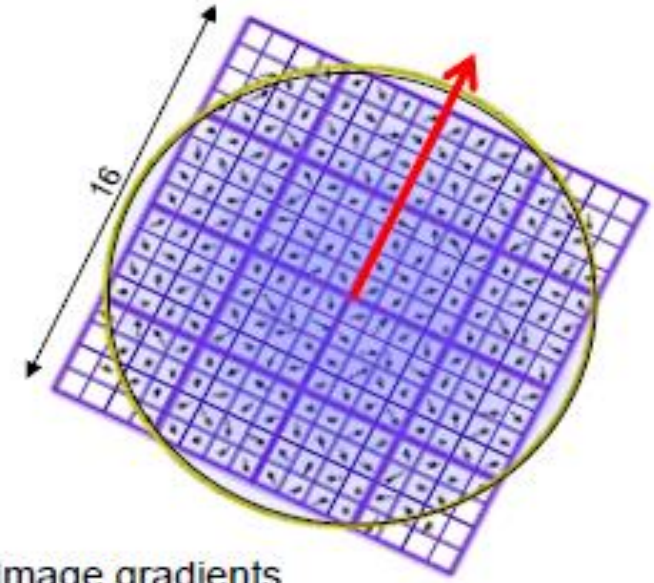
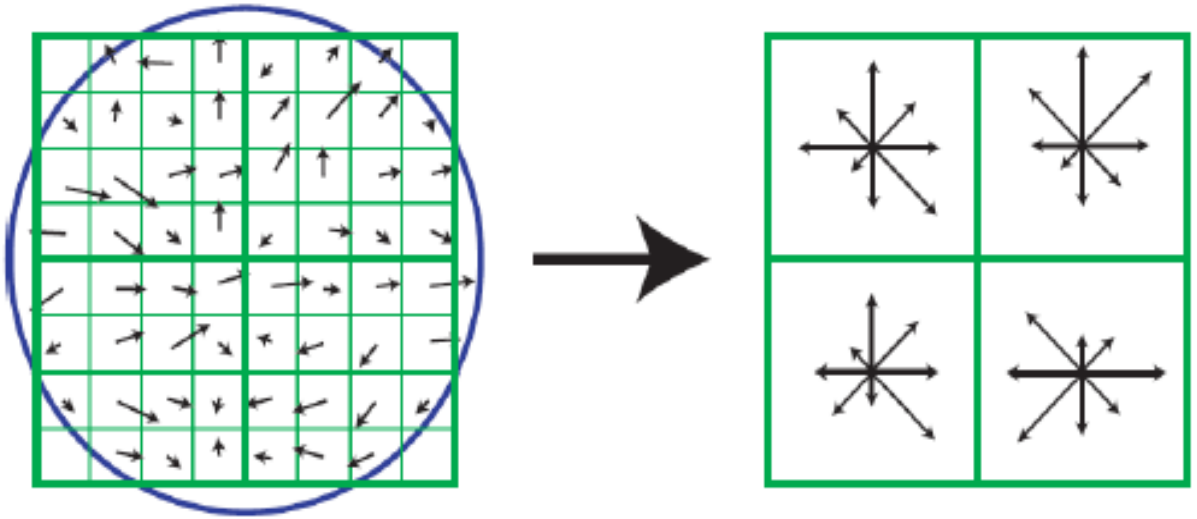


Image gradients

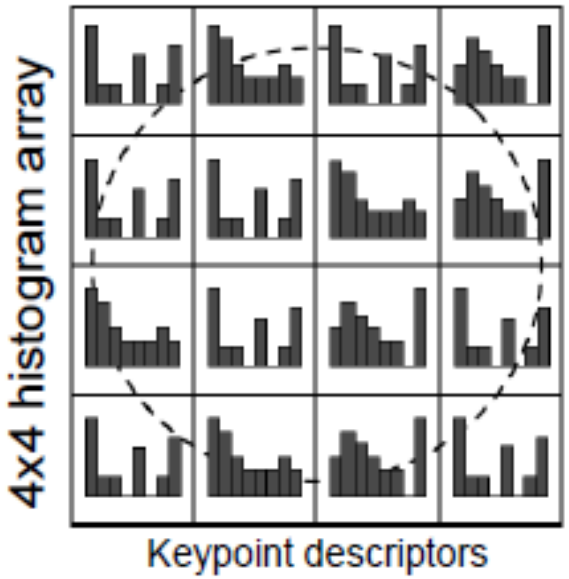
SIFT Descriptors

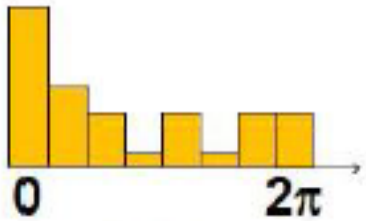


- Compute gradient magnitude and orientation in 16×16 region around keypoint location at the keypoint's scale.
- Coordinates and gradient orientations are measured relative to keypoint orientation to achieve orientation invariance.
- Weighted by Gaussian window.
- Collect into 4×4 orientation histograms with 8 orientation bins.
- Bin value = sum of gradient magnitudes near that orientation.

SIFT DESCRIPTORS

- Compute the orientation histogram of 8 orientations (45º covering) in a 4x4 pixel block which results in 128 vector descriptor



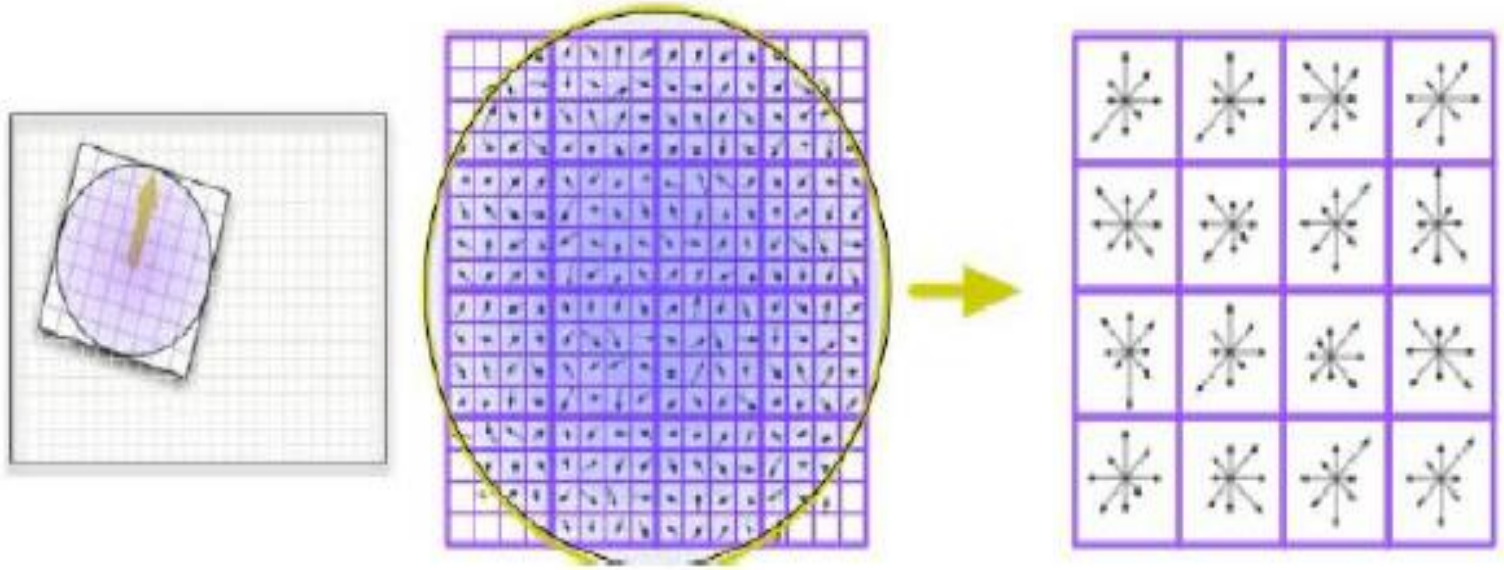


8 bins

The result: 128 dimensions feature vector.

SIFT Feature Calculation

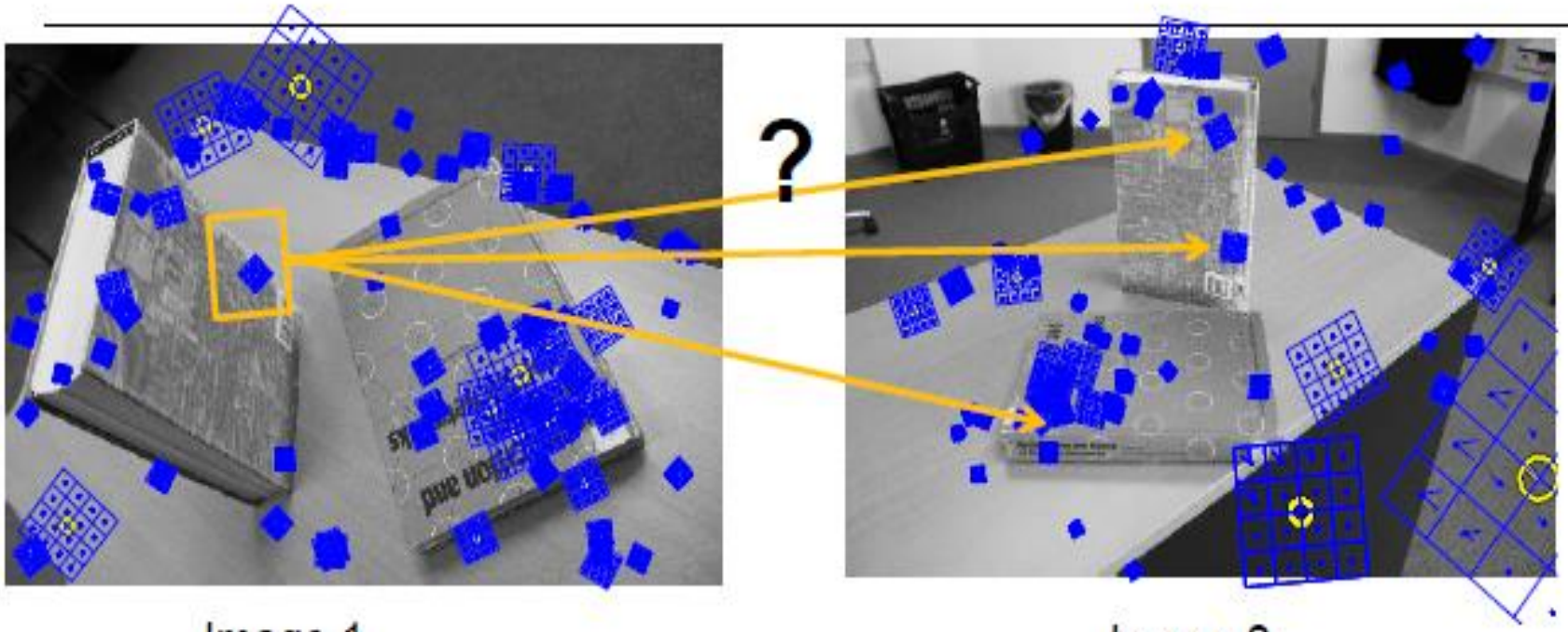
- Take the region around a keypoint according to its scale
- Rotate and align with the previously calculated orientation
- 8 orientation bins calculated at 4x4 bin array
- 8 x 4 x 4 = 128 dimension feature



Illumination Issues

- SIFT is a 128 dimensional vector
- For robustness to illumination, normalize the feature to unit magnitude
- To cater for image saturations, truncate the feature to 0.2
- Renormalize to unit magnitude
- SIFT properties
  - Repeatable keypoints
  - Scale invariant
  - Rotation invariant
  - Robust to viewpoint
  - Robust to illumination changes

MATCHING LOCAL FEATURES

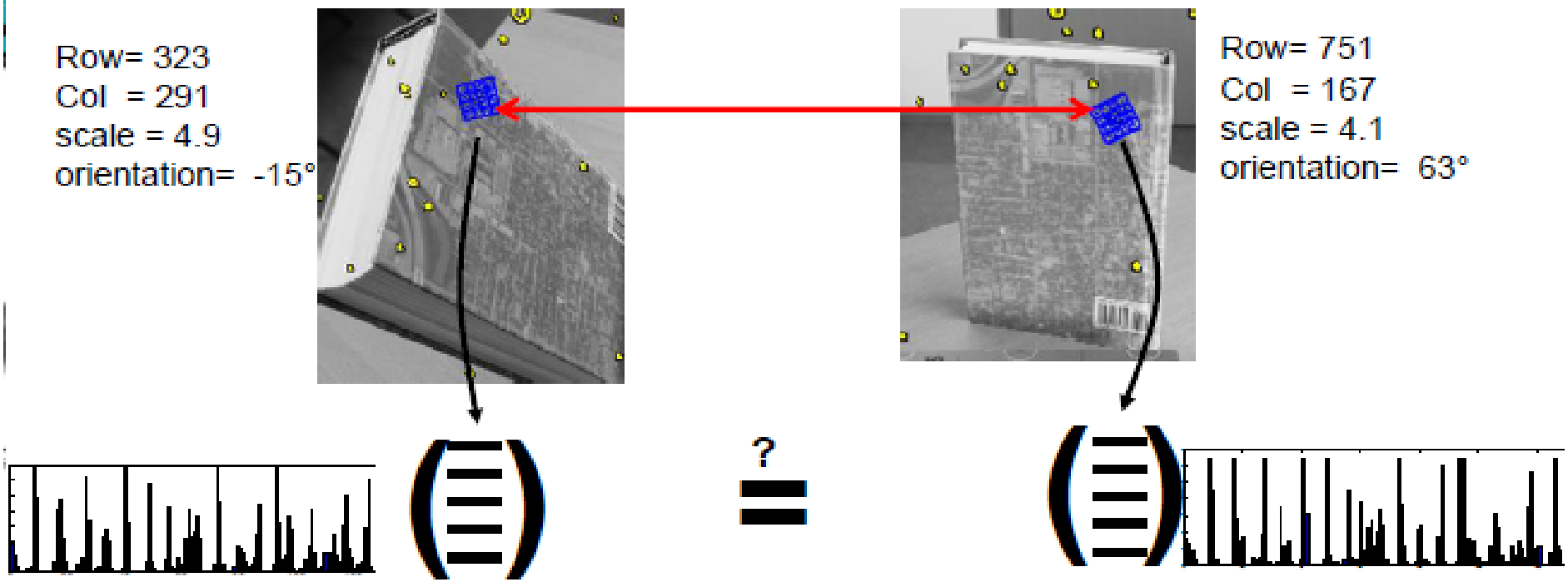


To generate **candidate matches**, find patches that have the most similar appearance (e.g., lowest SSD)

Simplest approach: compare them all, take the closest (within a threshold distance)

MATCHING DESCRIPTORS

- Hypotheses are generated by **approximate nearest neighbor** matching of each feature to vectors in the database.



Feature matching

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

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

Measure difference as Euclidean distance between feature vectors:

$$d(\mathbf{u}, \mathbf{v}) = \left(\sum_i (u_i - v_i)^2 \right)^{1/2}$$

Several possible matching strategies:

- Return all feature vectors with d smaller than a threshold.
- Nearest neighbor: feature vector with smallest d.
- Nearest neighbor distance ratio:

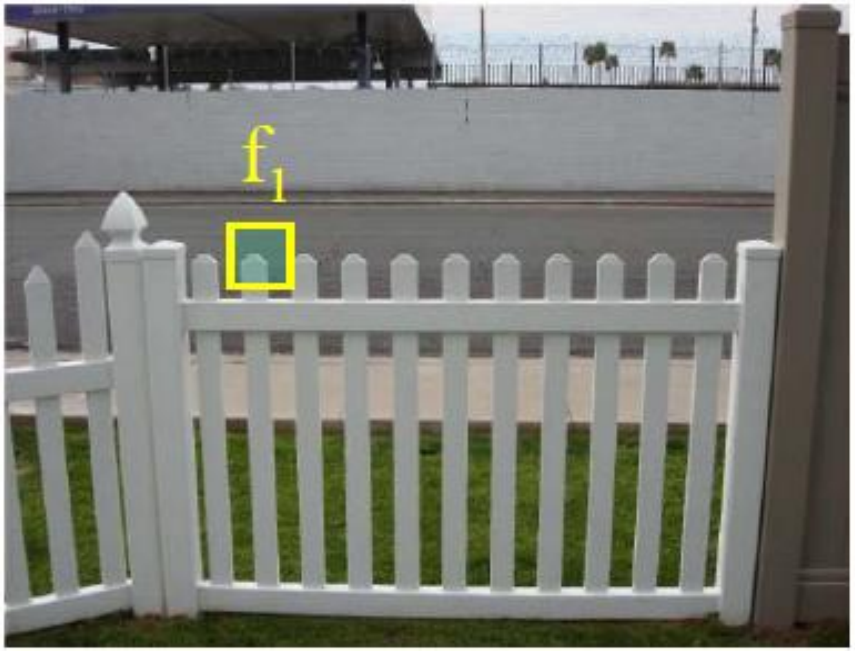
$$\text{NNDR} = \frac{d_1}{d_2}$$

  - d_1, d_2: distances to the nearest and 2nd nearest neighbors.
  - If NNDR is small, nearest neighbor is a good match.

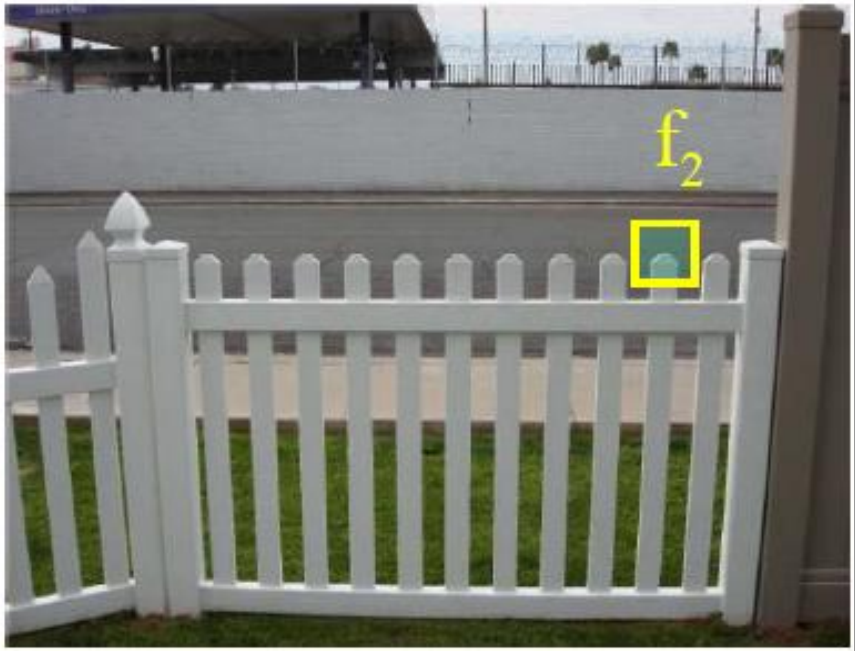
Feature distance: SSD

How to define the similarity between two features f_1, f_2?

- Simple approach is SSD(f_1, f_2)
  - sum of square differences between entries of the two descriptors
  - Doesn't provide a way to discard ambiguous (bad) matches



I_1

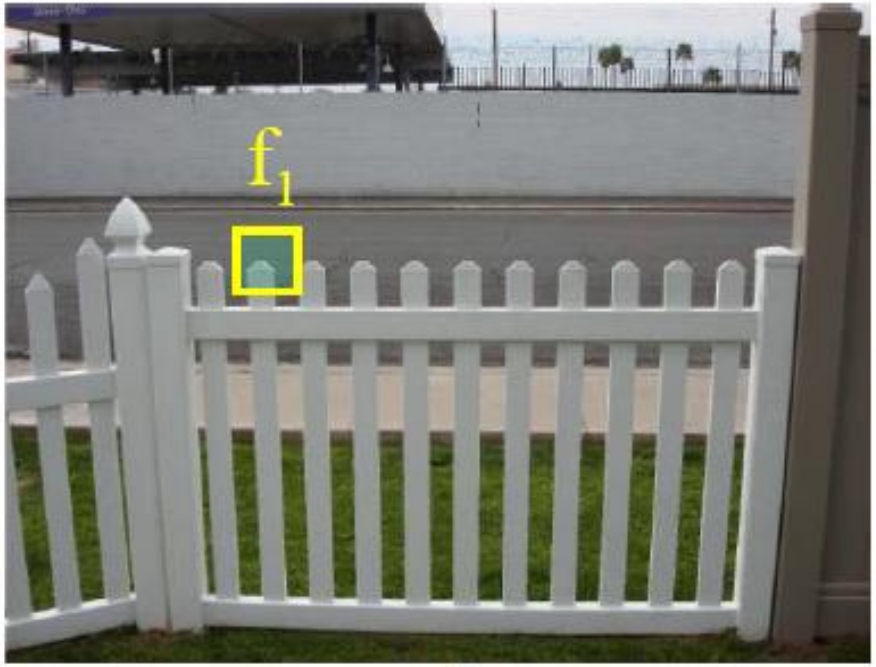


I_2

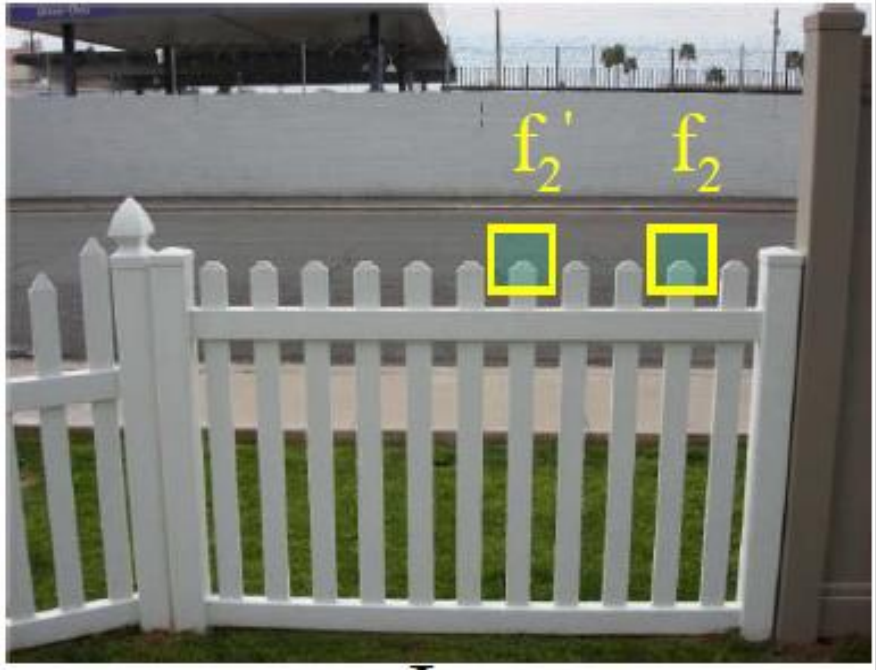
Feature distance: Ratio of SSDs

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

- Better approach: ratio distance = $SSD(f_1, f_2) / SSD(f_1, f_2')$
  - f_2 is best SSD match to f_1 in I_2
  - f_2' is 2nd best SSD match to f_1 in I_2
  - An ambiguous/bad match will have ratio close to 1
  - Look for unique matches which have low ratio

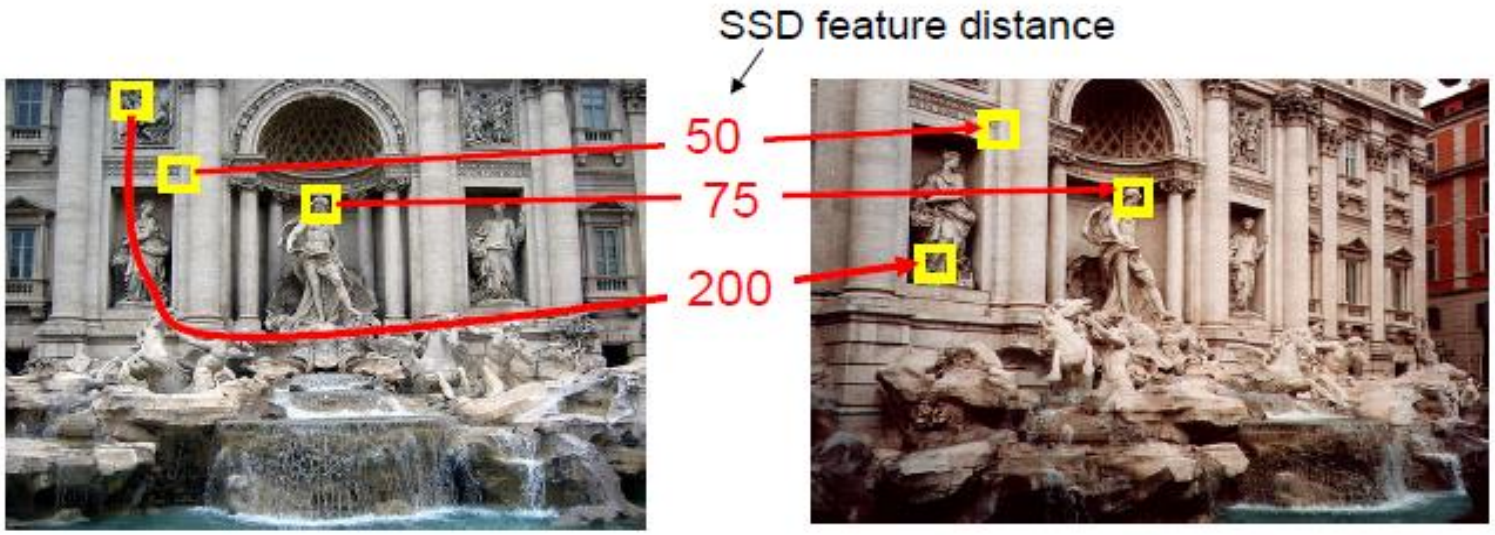


I_1



I_2

Image matching

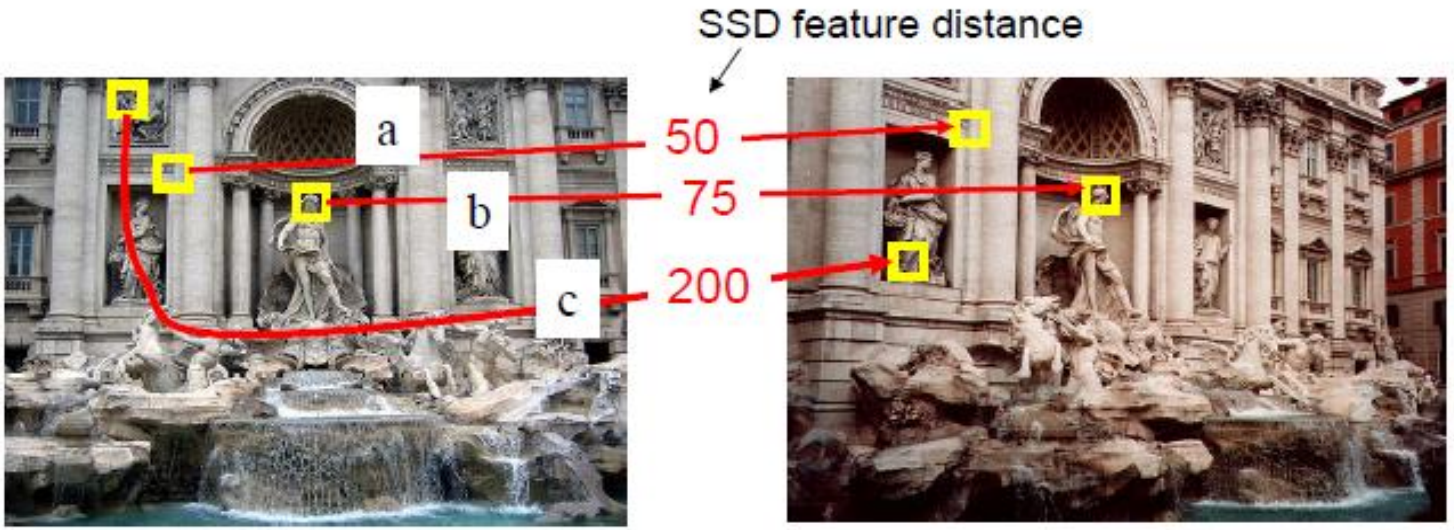


Suppose we use SSD
Small values are possible matches but how small?

Decision rule: Accept match if SSD < T
where T is a threshold

What is the effect of choosing a particular T?

Effect of threshold T



Decision rule: Accept match if SSD < T

Example: **Large T**

T = 250 ⇒ a, b, c are all accepted as matches

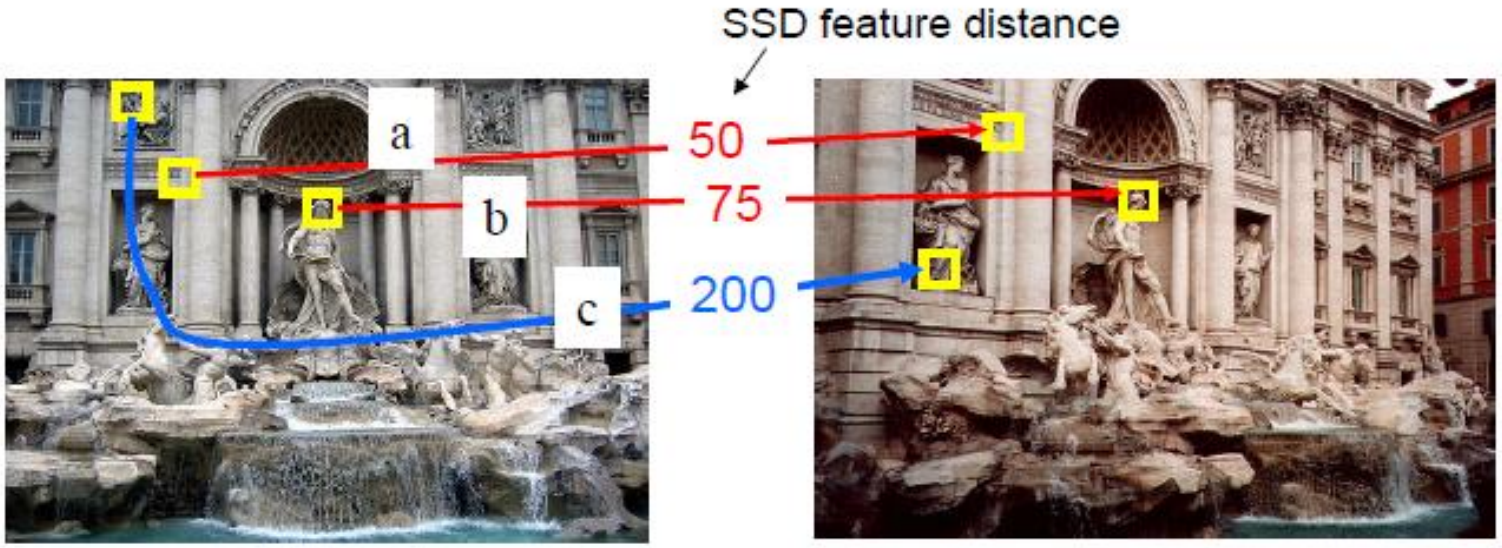
a and b are true matches ("**true positives**")

– they are actually matches

c is a false match ("**false positive**")

– actually not a match

Effect of threshold T



Decision rule: Accept match if SSD < T

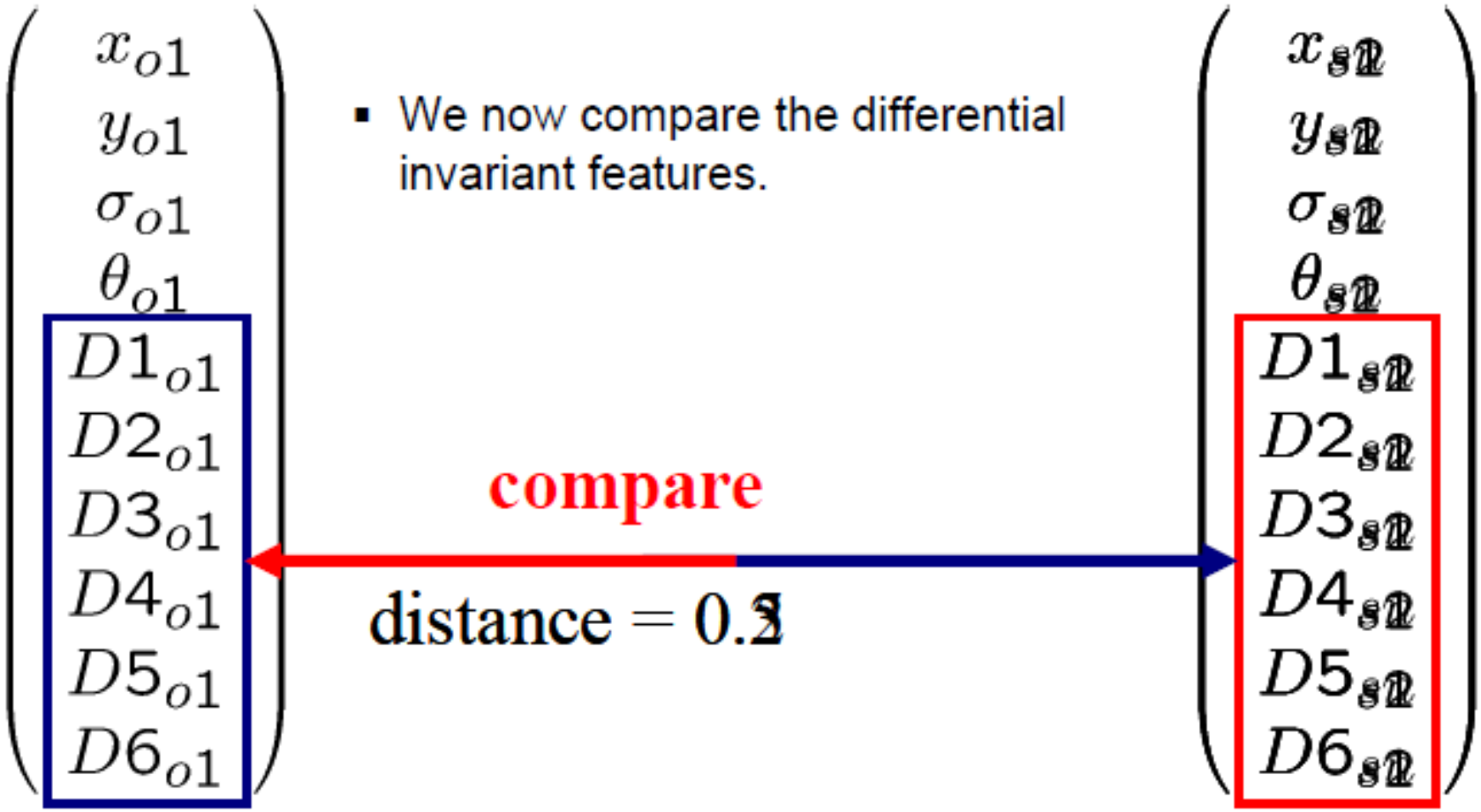
Example: **Smaller T**

T = 100 ⇒ only a and b are accepted as matches

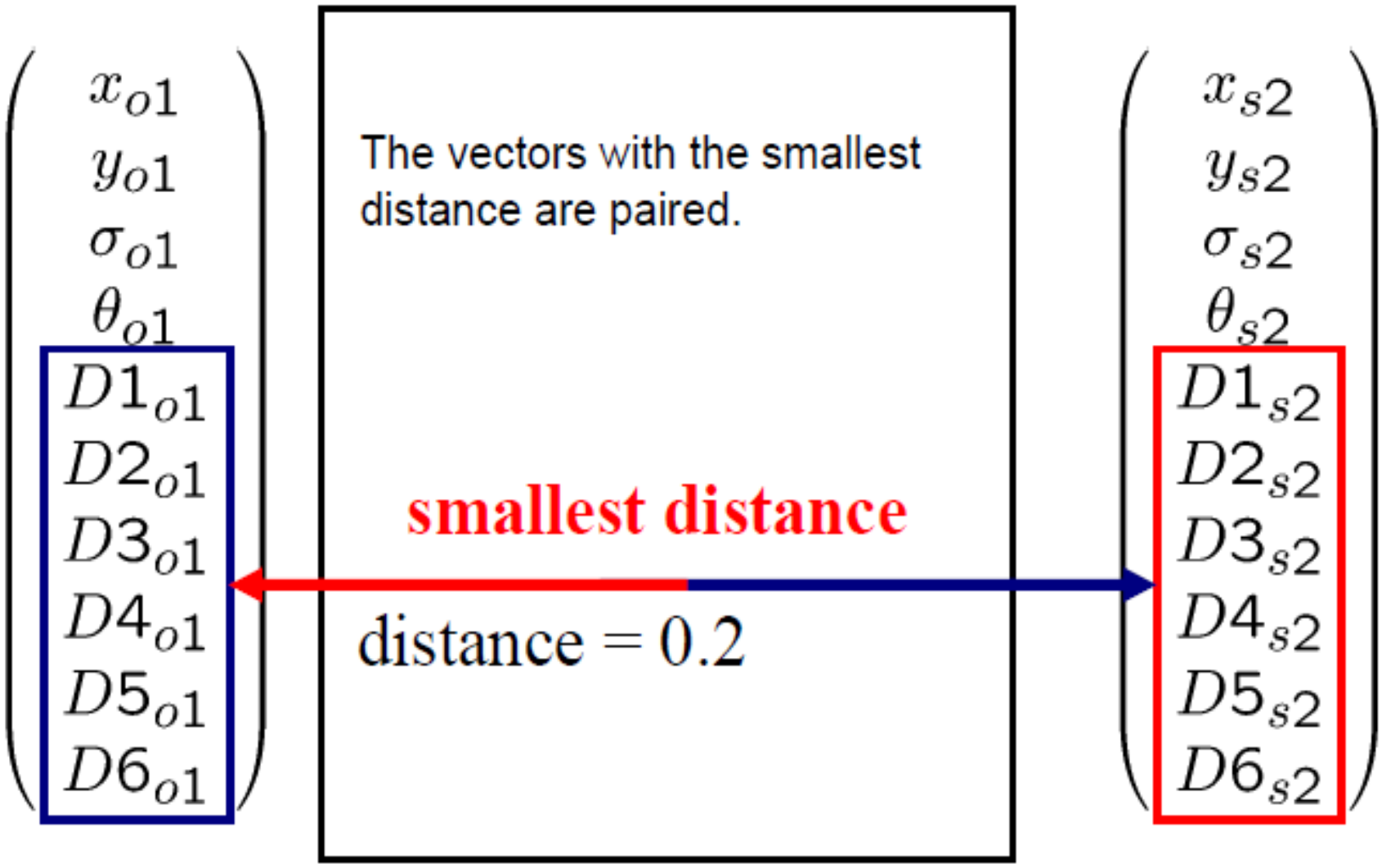
a and b are true matches ("true positives")
c is no longer a "false positive" (it is a "true negative")

- Some matches are correct, some are not.
- Can include other info such as color to improve match accuracy.
- In general, no perfect matching results.
- Feature matching methods can give false matches.
- Manually select good matches.
- Or use robust method to remove false matches:
  - True matches are consistent and have small errors.
  - False matches are inconsistent and have large errors.
- Nearest neighbor search is computationally expensive.
  - Need efficient algorithm, e.g., using k-D Tree.
  - k-D Tree is not more efficient than exhaustive search for large dimensionality, e.g., > 20.

MATCHING OF DESCRIPTORS



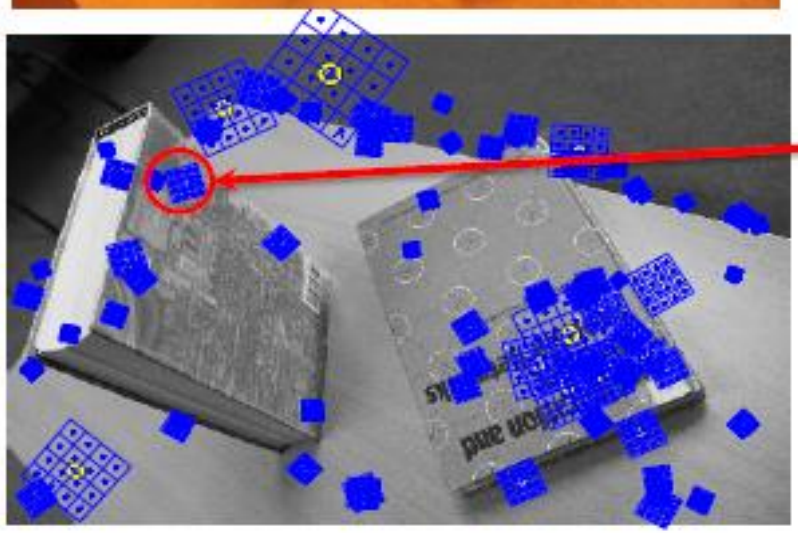
MATCHING OF DESCRIPTORS

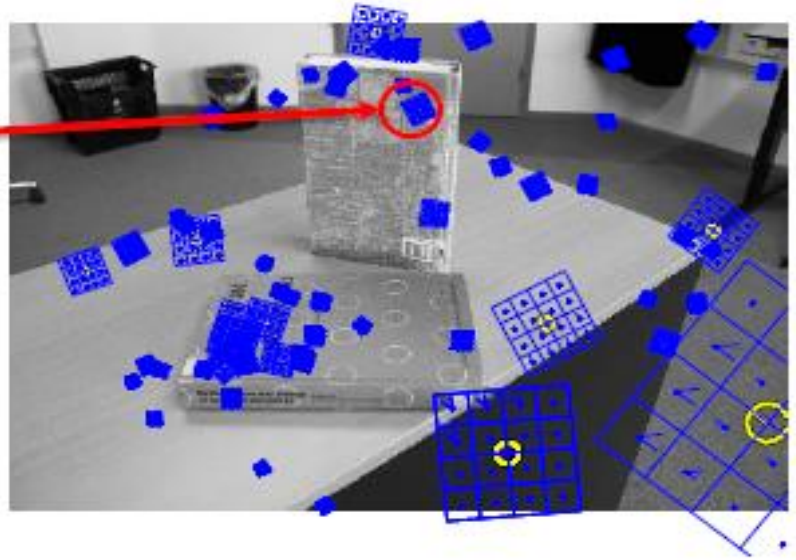


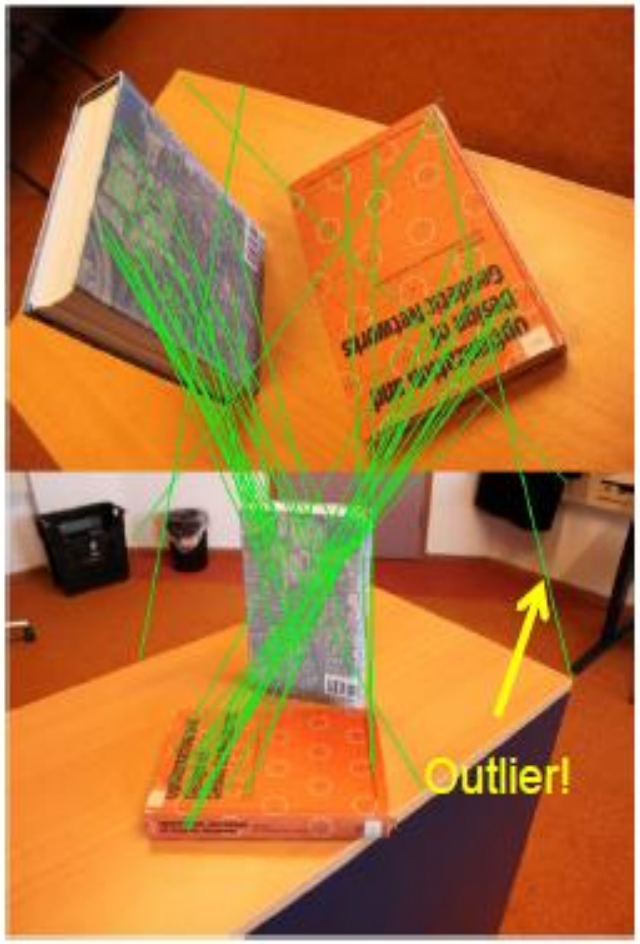
EXAMPLE

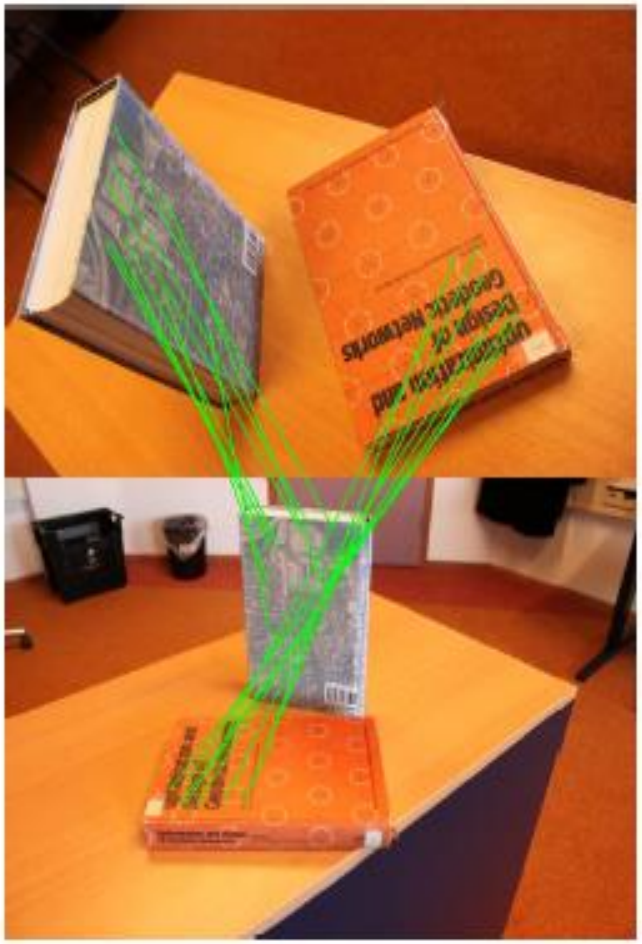
1st image

2nd image









AFTER RANSAC

Properties of SIFT-based matching

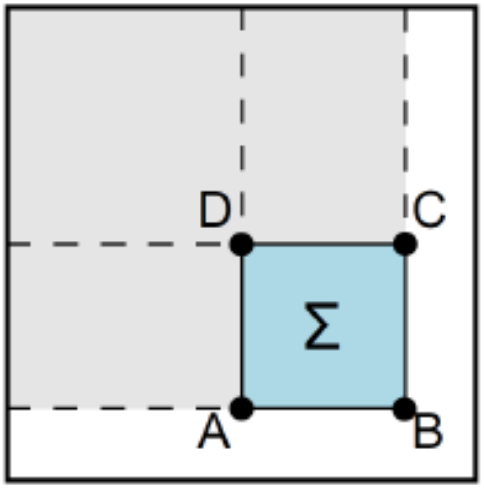
Extraordinarily robust matching technique

- Can handle changes in viewpoint
  - Up to about 60 degree out of plane rotation
- Can handle significant changes in illumination: Sometimes even day vs. night (below)
- Fast and efficient — can run in real time
- Lots of code available:

Speeded Up Robust Features (SURF)

- inspired by SIFT with real-time capabilities
- the DoG images and computing of Hess matrix is integrated into computing the determinant of Hess matrix
- this approach is using the integral image

$$I_{\Sigma}(x,y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i,j) \tag{19}$$



SURF – SPEEDED UP ROBUST FEATURE

The goal is to develop both a detector and descriptor, which is faster than SIFT, while not sacrificing performance.
How?
By reducing the descriptor's dimension and complexity

- ✓ Fast interest point **detection**
- ✓ Distinctive interest point **description**
- ✓ Speeded-up descriptor **matching**
- ✓ Invariant to common image transformations:
  - Image rotation
  - Scale changes
  - Illumination change
  - Small change in Viewpoint

METHODOLOGY

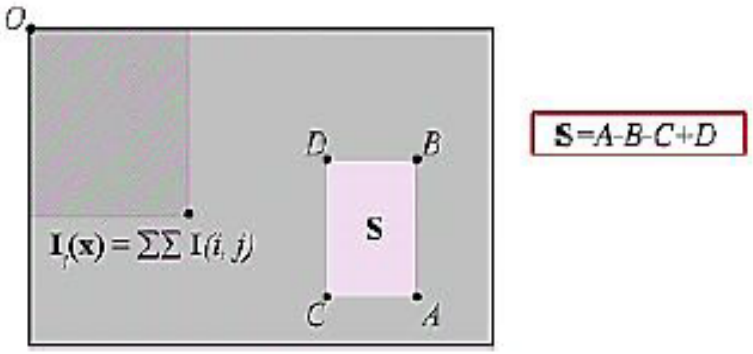
- **Integral image**
- Much of the performance increase in SURF can be attributed to the use of an intermediate image known as the *Integral Image*. The integral image is computed and is used to speed up the calculation of any upright rectangular area. Given an input image *I*, and point (x, y) the integral image $I_{\sum(x,y)}$ is calculated by the sum of the values between the point and the origin. Formally, this can be defined by the formula:

$$I_{\sum(x,y)} = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} I(x, y)$$

.

INTEGRAL IMAGE

- If we wanted to compute the sum of intensity of a rectangle area bounded by vertices A, B, C and D , the sum of pixel intensities is calculated according to the integral image by: $A - B - C + D$ and offers a speeded calculation for large areas



Area computation using integral images

DETECTION

1- Hessian

The SURF detector (for finding maxima or minima) is based on the determinant of the Hessian matrix and if it is positive definite. Hessian matrix needs to compute the second order partial derivative of a function. For images, this can be done by convolution with an appropriate kernel.

The Hessian matrix applied on image is:

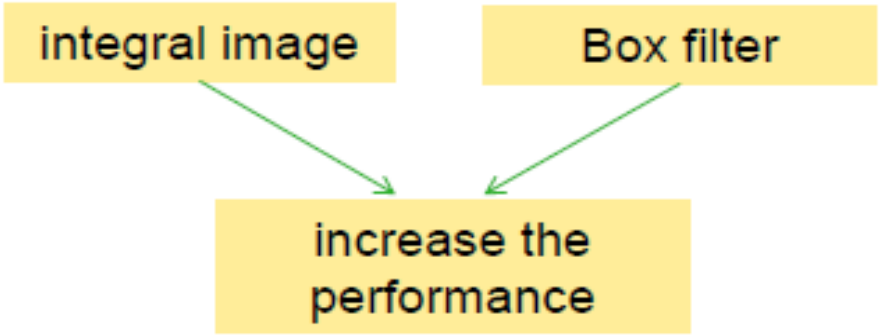
$$H = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{bmatrix}$$

→ For every image point

Where L_{xx} L_{yy} and L_{xy} are the second derivatives or the Laplacian of Gaussian of the image.

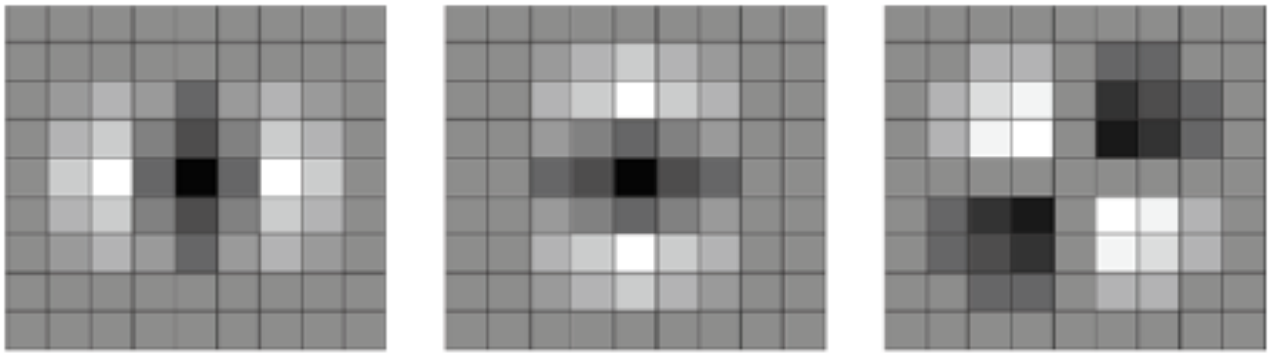
LAPLACIAN OF GAUSSIANS - APPROXIMATION

- In SIFT the Laplacian of Gaussian is approximated by the DoG . In a similar manner, Bay et al. (2008) proposed an approximation to the Laplacian of Gaussians by using box filter representations of the respective kernels.

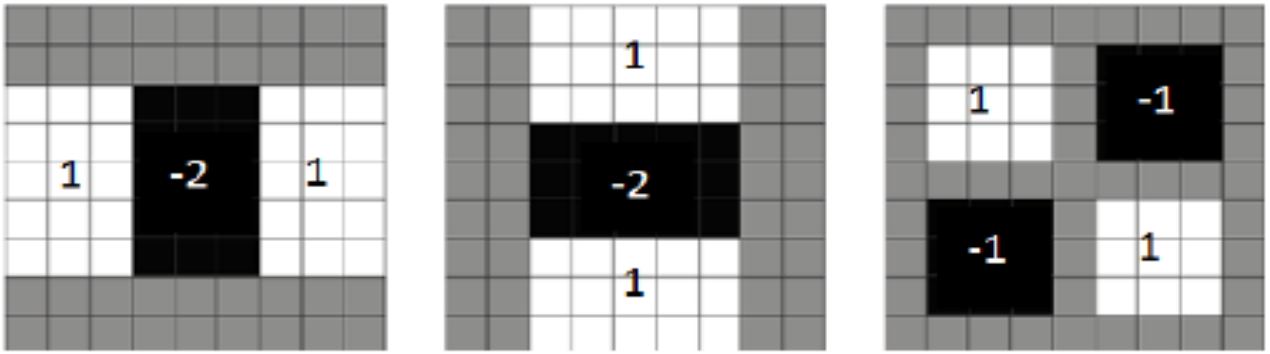


BOX FILTERS

the 2nd derivative L_{xx} L_{yy} and L_{xy} .

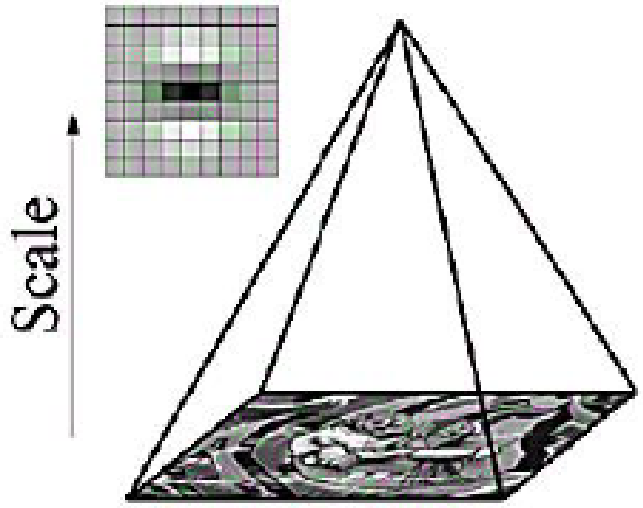


- the weighted box filter approximation



SCALE ANALYSIS WITH CONSTANT IMAGE SIZE

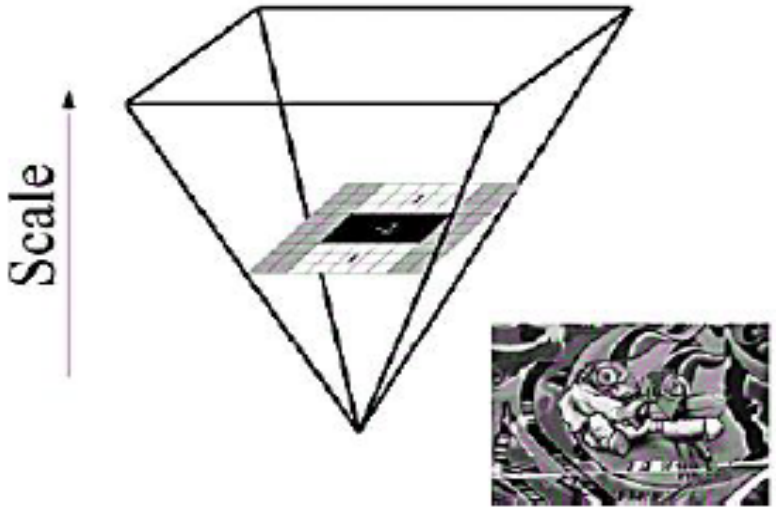
- In SIFT each layer, of an image pyramid, relies on the previous, and images need to be resized which is not computationally efficient.



SIFT - subsampling pyramid

FILTER IMAGE PYRAMIDS

(Bay et al., 2008) proposed that since the processing time of the kernels used in SURF is size invariant according to the used box filters, the scale-space can be created by applying kernels of increasing size to the original image. This allows for multiple layers of the scale-space pyramid to be processed simultaneously and contradicts the need to subsample the image hence providing performance increase.

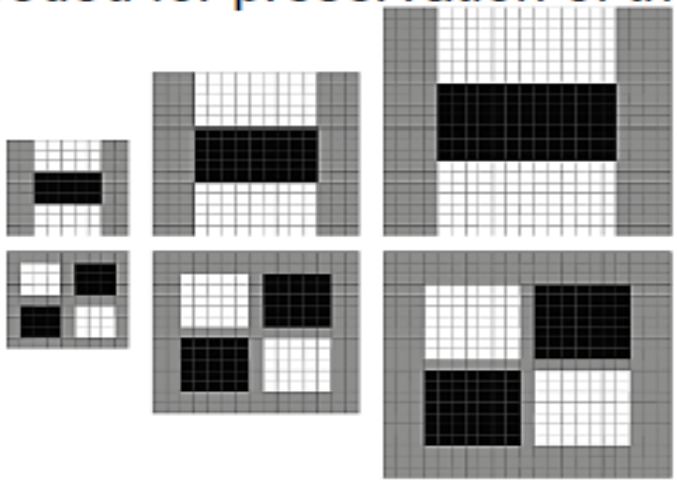


SURF – scale space pyramid

SCALE - SPACE

- The scale-space is divided into a number of octaves, where an octave refers to a series of response maps of covering a doubling of scale.
- In SURF the lowest level of the scale - space is obtained from the output of the 9*9 filters.
- These filters correspond to a real valued Gaussian with a scale of 1.2 values
- Increment of six or higher needed for preservation of the filter structure

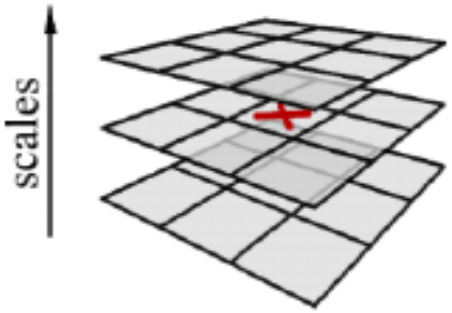
Filters Dyy (top) and Dxy (bottom) for successive scale levels (9*9,15*15 and 21*21).



LOCALIZATION

The task of localizing the scale and rotation, the invariant interest points in the image can be divided into three steps:

- Filtering with a threshold to keep strong points.
- Non-maximum suppression and interpolation to find a set of candidate points just like the SIFT method.



The construction of the scale space starts with the 9*9 filter, which calculates the blob response of the image for the smallest scale. Then, filters with sizes 15*15, 21*21,and 27*27 are applied,

Interpolating the nearby data, to find the location in both space and scale to sub-pixel accuracy. This is done by fitting a 3D quadratic

DESCRIPTION

The SURF descriptor of each interest point detected by using the integral images in conjunction with filters known as Haar wavelets in order to increase robustness and decrease computation time.

- Haar wavelets are simple filters which can be used to find gradients in the x and y directions.

Haar wavelets, the left filter computes the response in the x-direction and the right the y-direction

DESCRIPTION

Orientation Task

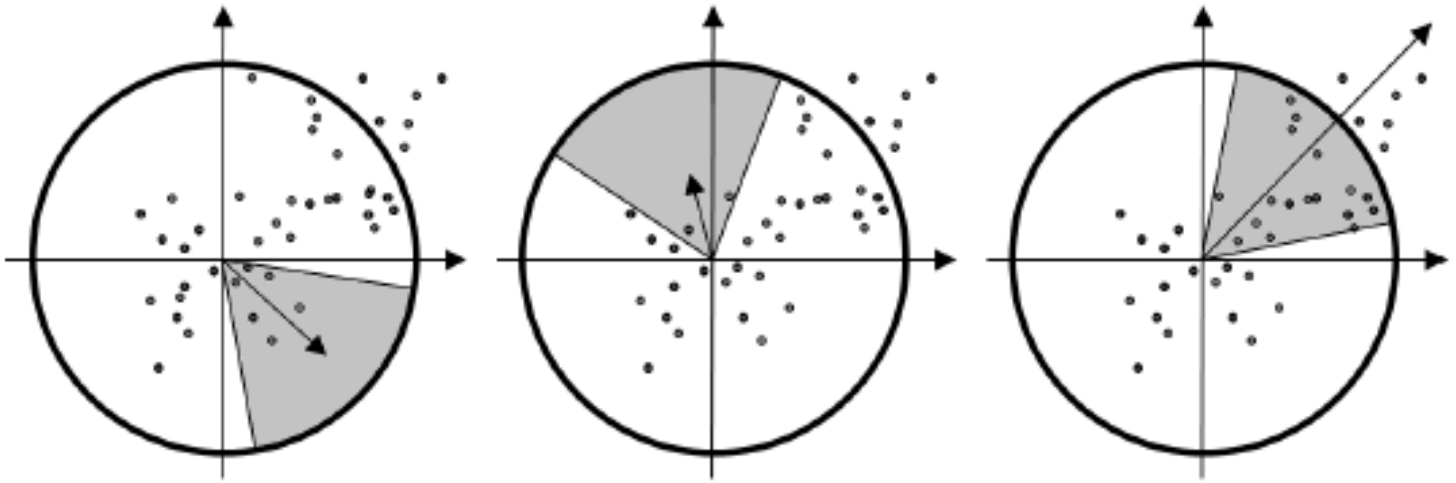
- In order to achieve rotation invariance each detected interest point is assigned a re-producible orientation. To determine the orientation, Haar wavelet responses of size $4s$ are calculated for a set pixel within a radius $6s$ of the detected interest point,
- s refers to the scale at which the point was detected.

To find the dominant orientation:

- The Haar wavelet responses are represented as vectors
- Sum all responses within a sliding orientation window covering an angle of 60°. The two summed response yield a new vector.

ORIENTATION TASK

- The longest vector is the dominant orientation and neglects the others.

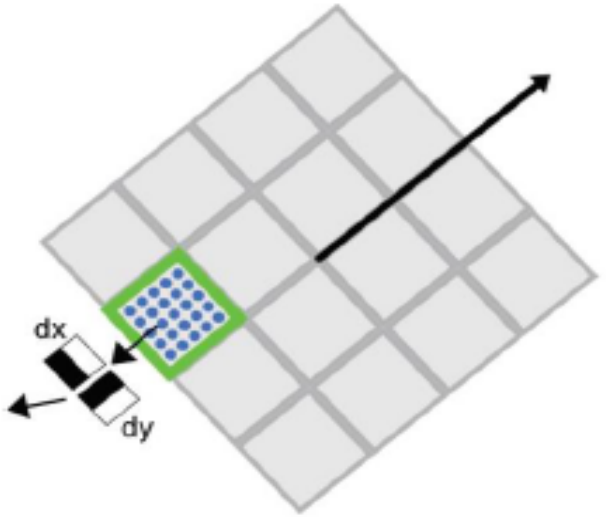


dominant orientation

DESCRIPTOR COMPONENTS

The descriptor component starts by constructing a square window around the interest point of $4 * 4$ square sub-regions with $5 * 5$ regularly spaced sample points inside. This square window is oriented along the dominant direction. Then, calculate Haar wavelet response d_x and d_y for these 25 points

$$\begin{bmatrix} \sum dx \\ \sum dy \\ \sum |dx| \\ \sum |dy| \end{bmatrix}$$



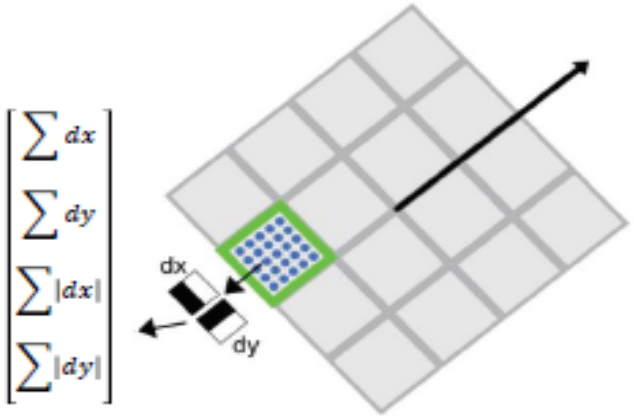
DESCRIPTOR COMPONENTS

- Weight the response with a Gaussian kernel centered at the interest point
- Sum the response over each sub-region for d_x and d_y
- extract the sum of absolute value of the responses → **feature vector of length 64**

$$v_{subregion} = \left[\sum dx, \sum dy, \sum |dx|, \sum |dy|\right]$$

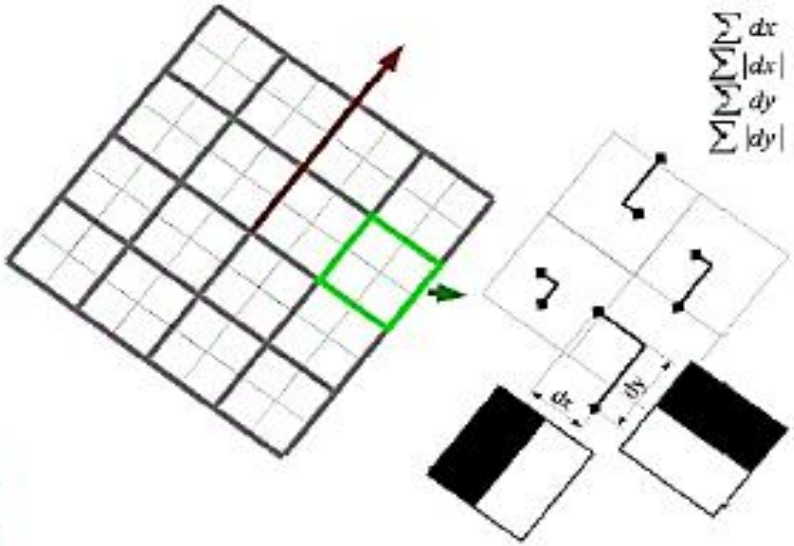
Feature vector of length = $v_{subregion} * 4 * 4 = 64$

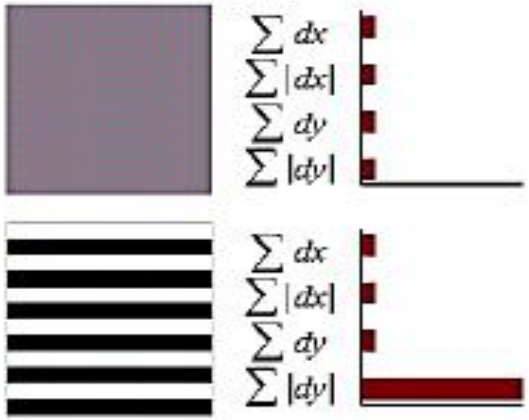
- Normalize the vector into unit length to be invariant to contrast.

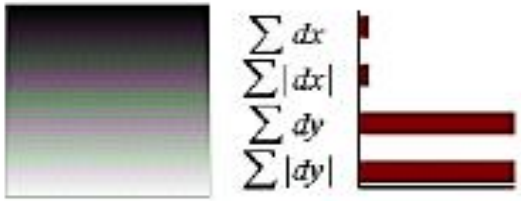


DESCRIPTOR COMPONENTS

- Description







COMPARE SURF TO SIFT

- *SURF:* Fast-Hessian detector + SURF descriptor
- *SIFT:* DOG detector + SIFT descriptor
- SURF is good at
  - handling serious blurring
  - handling image rotation
- SURF is poor at
  - handling viewpoint change
  - handling illumination change
- SURF describes image faster than SIFT by 3 times.
- SURF is not as well as SIFT on invariance to illumination change and viewpoint change (Bay et al., 2008)

Sample detected SURF keypoints (without non-maximal suppression):

(a) (b)

(a) Low threshold gives many cluttered keypoints.

(b) Higher threshold gives fewer keypoints, but still cluttered.

Sample detected SURF keypoints.
With adaptive non-maximal suppression, keypoints are well spread out.

(a)

(b)

(a) Top 100 keypoints.
(b) Top 200 keypoints