

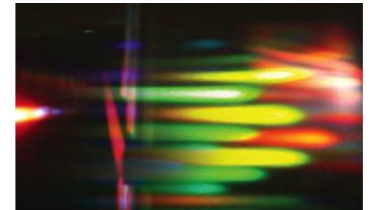


COMP70058 Computer Vision

Lecture 10 – Feature Descriptors and Matching

Stamatia (Matina) Giannarou, PhD
The Hamlyn Centre for Robotic Surgery

stamatia.giannarou@imperial.ac.uk



The Hamlyn Centre
for Robotic Surgery

Contents

- Image Processing with Local Features
- Feature Descriptors
- SIFT
- SURF
- Shape Context descriptor
- Feature Descriptor Distance
- Feature Matching



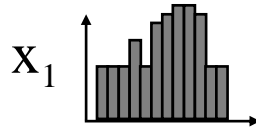
Image Processing with Local Feature

1. **Detection** - Find a set of distinctive key points.

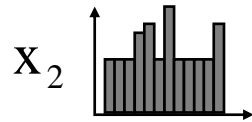


2. **Description** - Extract feature descriptor around each interest point as vector.

$$\mathbf{x}_1 = [x_1^{(1)}, \dots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \dots, x_d^{(2)}]$$



3. **Matching** - Compute distance between feature vectors to find correspondence.

$$d(\mathbf{x}_1, \mathbf{x}_2) < T$$



Feature Descriptors

Note for good image features, we require **locality** (small areas are less sensitive to view-dependent deformations), **invariance** (scale, orientation and deformation), **repeatability** (same points can be repetitively identified as required for image sequence tracking), and **distinctiveness** (to ensure high sensitivity and specificity).

- Most feature descriptors can be thought of as templates, histograms (counts), or combinations
- The ideal descriptor should be
 - Robust and Distinctive
 - Compact and Efficient
- Most available descriptors focus on edge/gradient information
 - Capture texture information
 - Color rarely used



SIFT Descriptor

SIFT – Scale Invariant Feature Transform

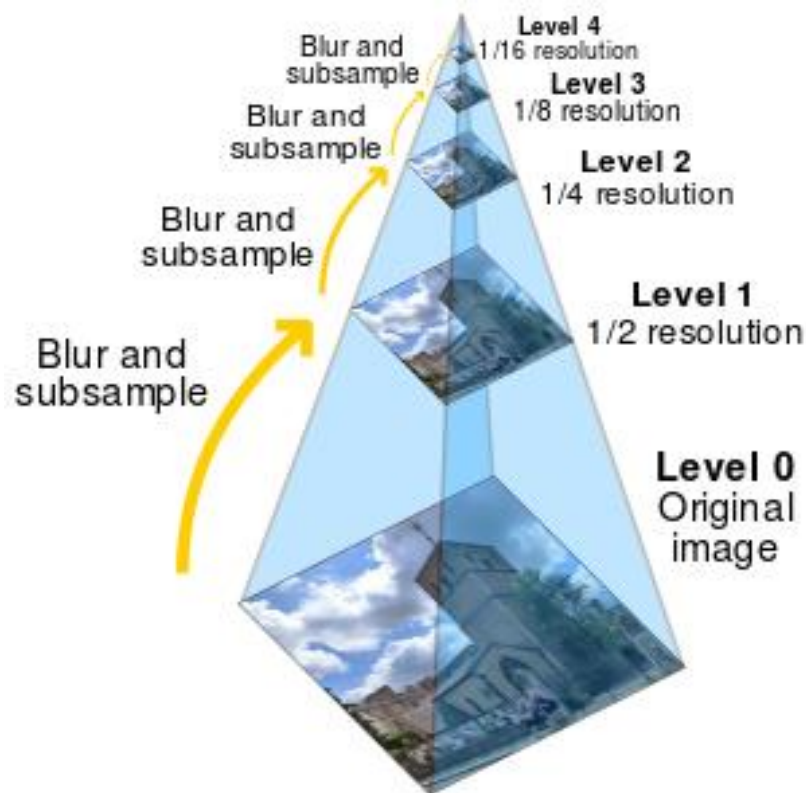
Step 1: Scale-space Extrema Detection -

Detect interesting points using DOG.

Step 2: Keypoint Localization – Determine location and scale at each candidate location, and select them based on stability.

Step 3: Orientation Estimation – Use local image gradients to assign orientation to each localized keypoint. Preserve theta, scale and location for each feature.

Step 4: Keypoint Descriptor - Extract local image gradients at the selected scale around the keypoint and form a representation invariant to local shape distortion and illumination

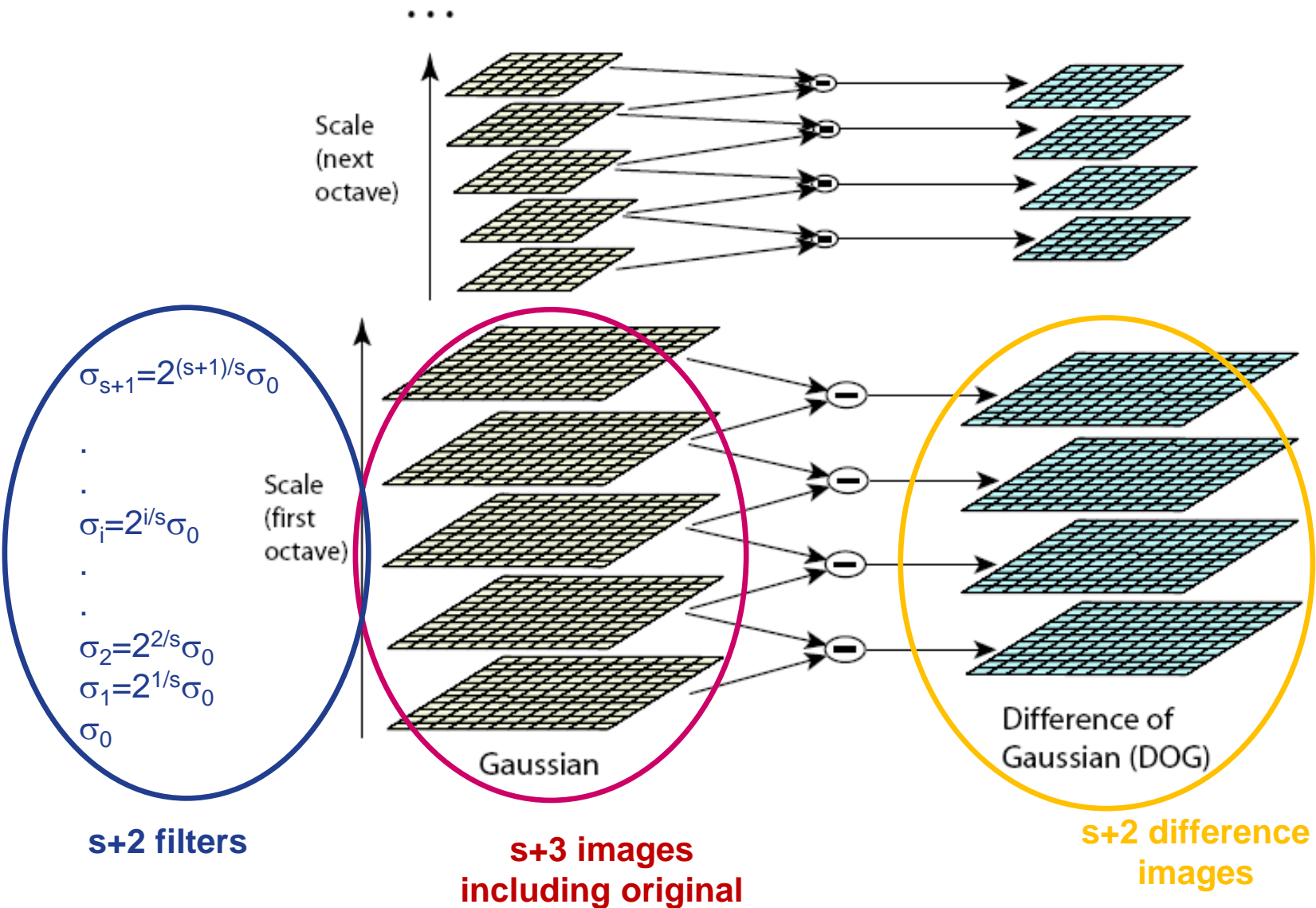


<https://cs.brown.edu/courses/csci1430/>

SIFT Descriptor - Extrema Detection

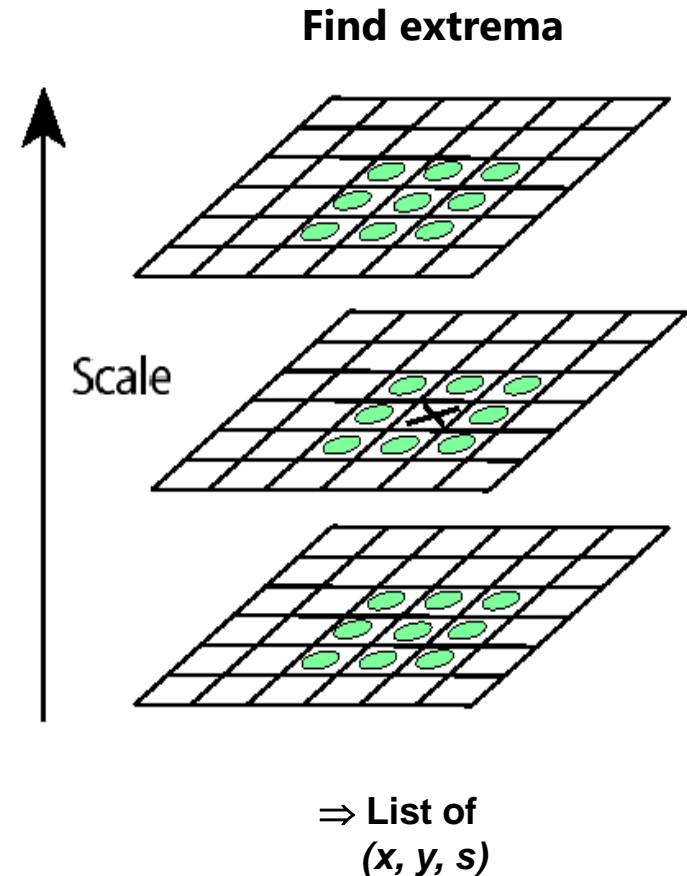
- Scale space is separated into octaves
 - Octave 1 uses scale σ
 - Octave 2 uses scale 2σ
- In each octave, the initial image is repeatedly **convolved** with Gaussians to produce a set of scale space images
- Adjacent Gaussians are **subtracted** to produce the DOG
- Once a complete octave has been processed, the Gaussian image that has twice the initial value of σ (it will be 2 images from the top of the stack) is **resampled** by taking every second pixel in each row and column to start the next level

SIFT Descriptor - Extrema Detection



SIFT Descriptor - Extrema Detection

- Increasing σ increases robustness, but it is also computationally expensive. $\sigma = 1.6$ a good tradeoff
- It has been shown experimentally that the highest repeatability is obtained when sampling 3 scales per octave
- Detect maxima and minima of Difference-of-Gaussian in scale space
- Each point is compared to its 26 neighbors in the current image and in the scales above and below



SIFT Descriptor - Keypoint Localization

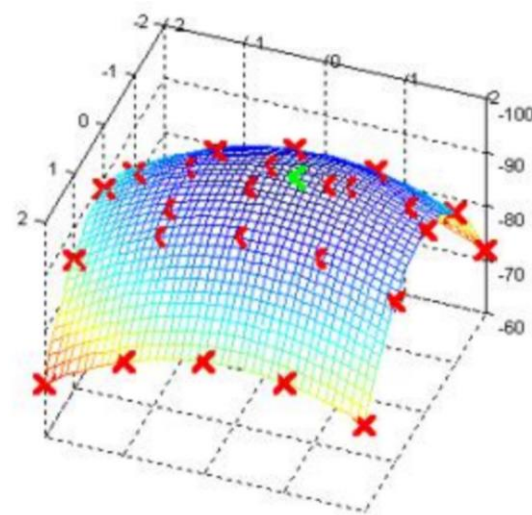
- For each candidate keypoint, interpolation of nearby data is used to accurately determine its position and scale
- This can be done by fitting a quadratic Taylor expansion of the Difference-of-Gaussian scale-space function shifted so that the origin is at the sample point

$$D(X) = D + \frac{\partial D}{\partial X} X + \frac{1}{2} X^T \frac{\partial^2 D}{\partial X^2} X$$

where $X = (x, y, \sigma)^T$

- The offset for the extremum location is determined by taking the derivative of this function with respect to X and setting it to zero

$$0 = \frac{\partial D}{\partial X} + \frac{\partial^2 D}{\partial X^2} \hat{X}$$



SIFT Descriptor - Keypoint Localization

- This equation results in a 3x3 linear system
- Derivatives can be approximated by finite differences
- The solution using Newton's method is:

$$\hat{X} = -\frac{\partial^2 D^{-1}}{\partial X^2} \frac{\partial D}{\partial X}$$

- If the offset X is greater than 0.5 in any dimension, it means that the extremum lies closer to a different sample point and the process is repeated

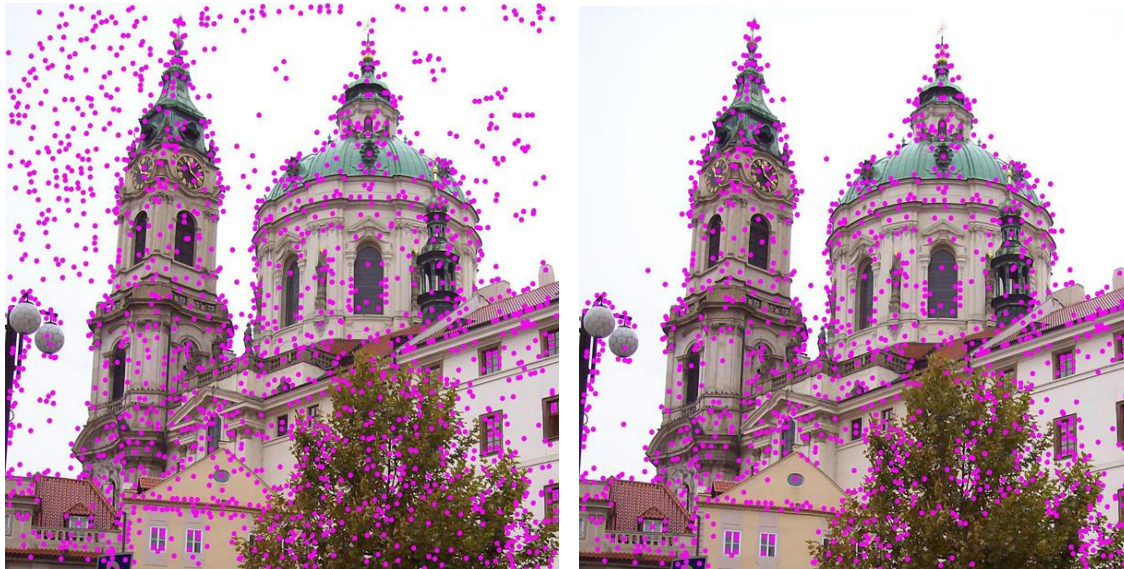
$$\begin{bmatrix} \frac{\partial^2 D}{\partial \sigma^2} & \frac{\partial^2 D}{\partial \sigma y} & \frac{\partial^2 D}{\partial \sigma x} \\ \frac{\partial^2 D}{\partial \sigma y} & \frac{\partial^2 D}{\partial y^2} & \frac{\partial^2 D}{\partial y x} \\ \frac{\partial^2 D}{\partial \sigma x} & \frac{\partial^2 D}{\partial y x} & \frac{\partial^2 D}{\partial x^2} \end{bmatrix} \begin{bmatrix} \sigma \\ y \\ x \end{bmatrix} = - \begin{bmatrix} \frac{\partial D}{\partial \sigma} \\ \frac{\partial D}{\partial y} \\ \frac{\partial D}{\partial x} \end{bmatrix}$$

$$\begin{aligned} \frac{\partial D}{\partial \sigma} &= \frac{D_{k+1}^{i,j} - D_{k-1}^{i,j}}{2} \\ \frac{\partial^2 D}{\partial \sigma^2} &= \frac{D_{k-1}^{i,j} - 2D_k^{i,j} + D_{k+1}^{i,j}}{1} \\ \frac{\partial^2 D}{\partial \sigma y} &= \frac{(D_{k+1}^{i+1,j} - D_{k-1}^{i+1,j}) - (D_{k+1}^{i-1,j} - D_{k-1}^{i-1,j})}{4} \end{aligned}$$

SIFT Descriptor - Keypoint Localization

- Some of the detected keypoints lie along an edge, or they don't have enough contrast. In both cases, they are not as useful as features and should be eliminated
- If the DoG value at an extrema is less than a threshold (e.g 0.03), the keypoint is rejected

$$|D(\hat{\mathbf{x}})| < 0.03$$



https://en.wikipedia.org/wiki/Scale-invariant_feature_transform

SIFT Descriptor - Keypoint Localization

- A keypoint located on an edge in the image will have a large principle curvature across the edge and a low one along it. On the contrary, a well-defined keypoint will have a large principle curvature in both directions. Finding these principal curvatures amounts to solving for the eigenvalues of the second-order Hessian matrix.

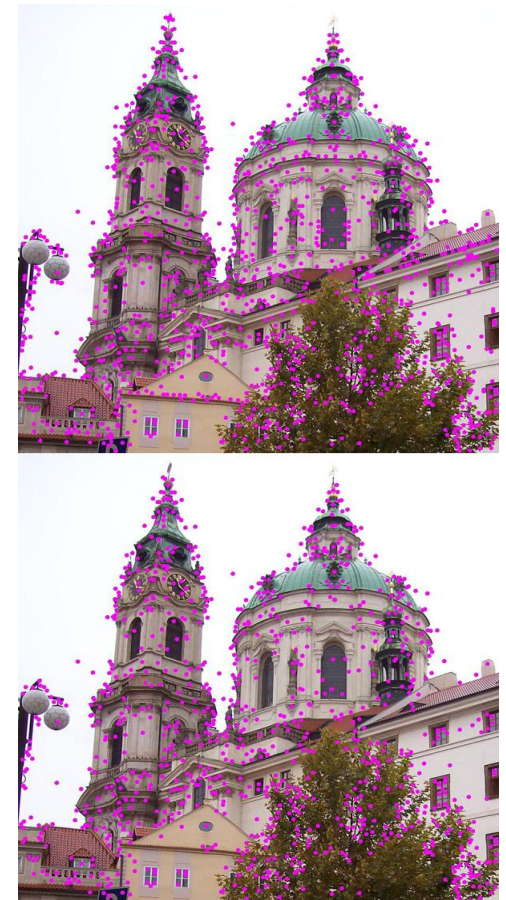
$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

$$\text{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$\text{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

$$\frac{\text{Tr}(\mathbf{H})^2}{\text{Det}(\mathbf{H})} = \frac{(\alpha + \beta)^2}{\alpha\beta} = \frac{(r\beta + \beta)^2}{r\beta^2} = \frac{(r + 1)^2}{r} > (r_{th} + 1)^2 / (r_{th})$$

If the above ratio for a candidate keypoint is larger than a threshold ($r_{th}=10$), the keypoint is poorly localized and hence rejected.



SIFT Descriptor

- Each keypoint is characterised by location, scale, orientation
- A descriptor is computed for the local image window around each keypoint

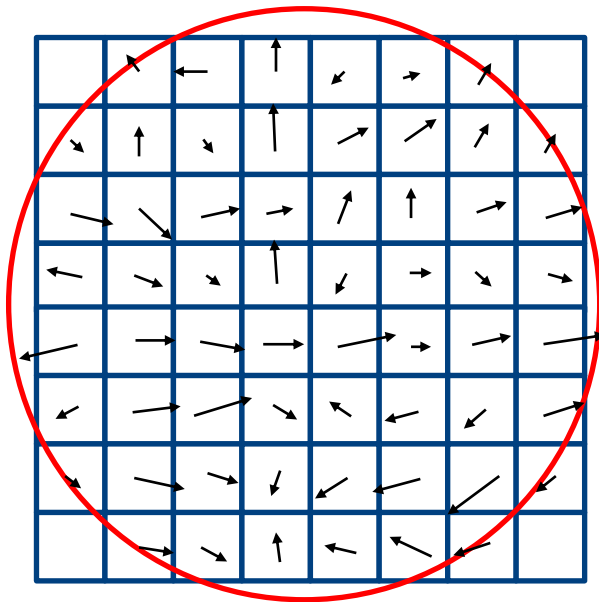
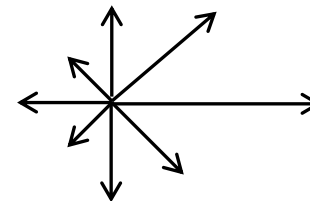
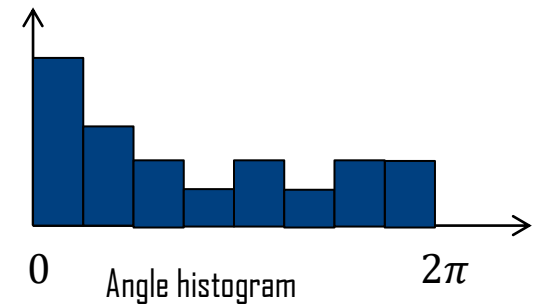
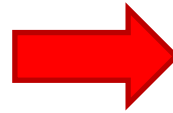


Image gradients



SIFT Descriptor

- The Basic Idea
 - 16×16 window around interest point (8×8 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

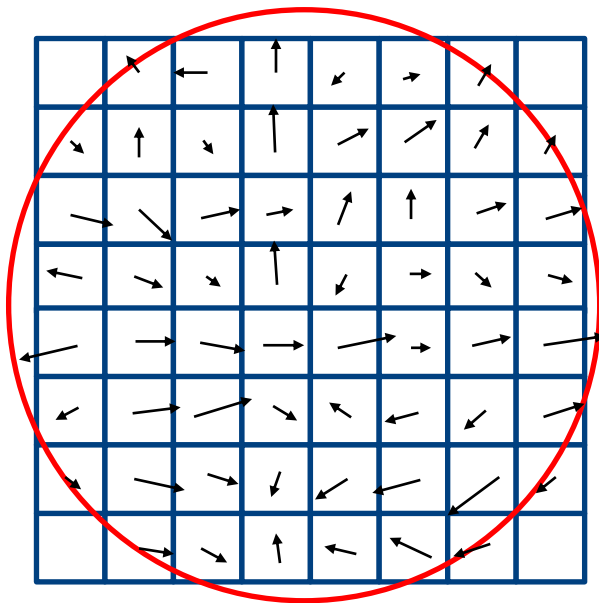
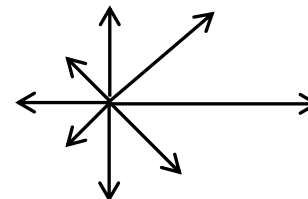
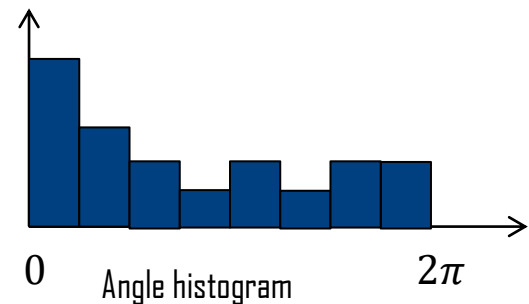
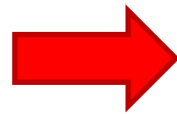


Image gradients



SIFT Descriptor

- Full Version
 - Divide the 16×16 window into a 4×4 grid of cells (8×8 window and 2×2 grid shown below for simplicity)
 - Compute an orientation histogram for each cell
 - $16 \text{ cells} * 8 \text{ orientations} = 128 \text{ dimensional descriptor}$

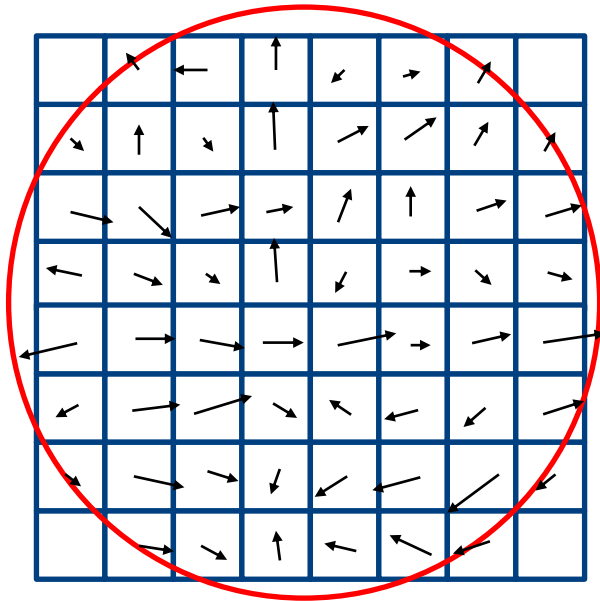
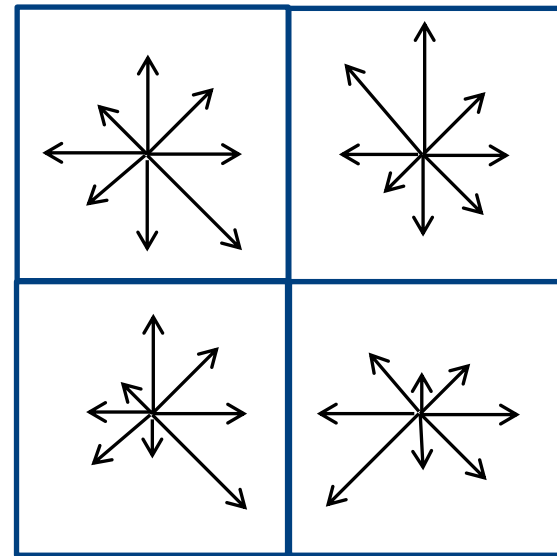


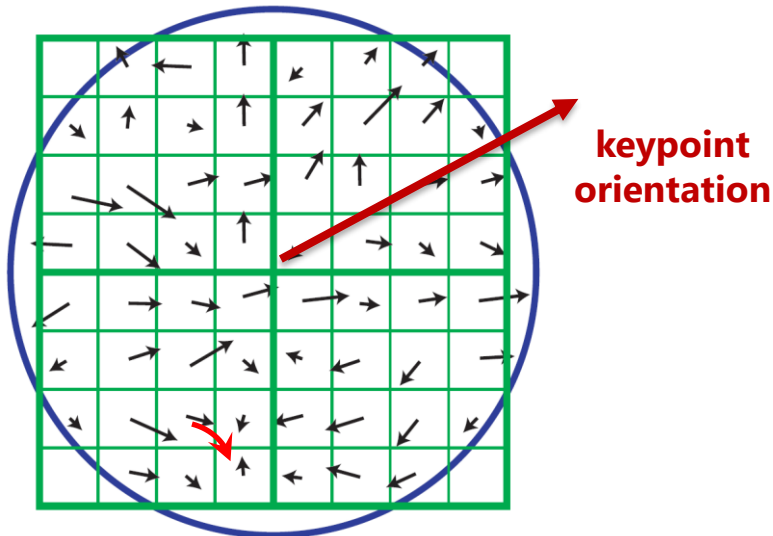
Image gradients



Keypoint descriptor

SIFT Descriptor - Invariance Properties

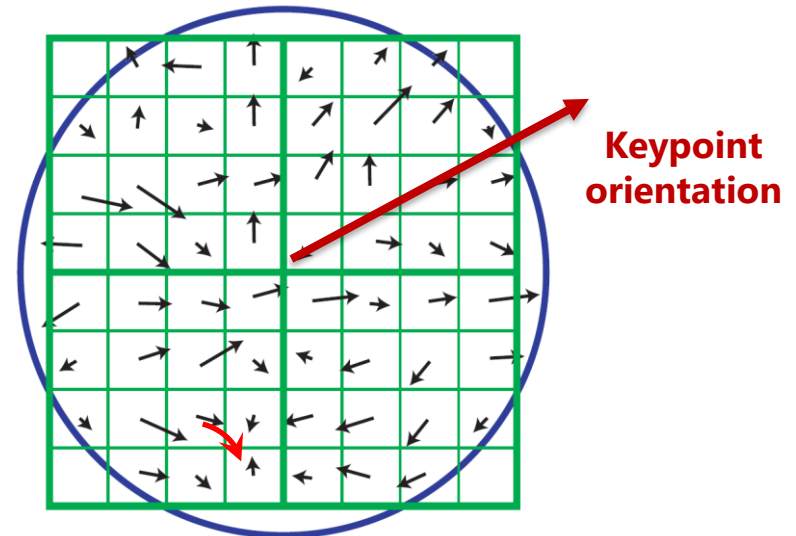
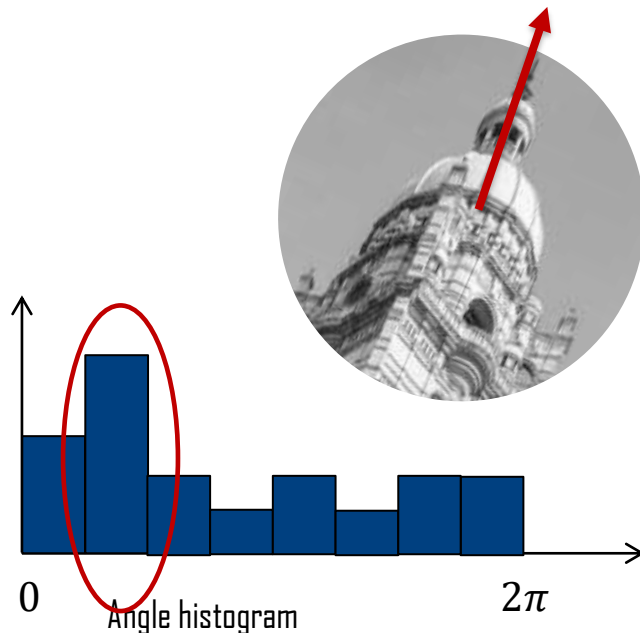
- To be robust to intensity value changes
 - Use gradient orientations
- To be scale-invariant
 - Estimate the scale using scale-space extrema detection
 - Scale the window size based on that scale at which the point was found
 - Calculate the gradient after Gaussian smoothing with this scale
- To be orientation-invariant
 - Rotate the gradient orientations using the dominant orientation in a neighbourhood.



SIFT Descriptor - Orientation Estimation

- To be orientation-invariant the descriptor of a keypoint can be computed relative to the dominant orientation in its neighbourhood

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$
$$\theta(x, y) = a \tan 2((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y)))$$



Properties of SIFT Descriptors for Matching

- Robust and can handle changes in viewpoint
 - Up to about 60 degrees out of plane rotation
- Can handle significant changes in illumination
- Fast and efficient, can run in real-time
- Other variations available, e.g., SURF, PCA SIFT, GLOH (gradient location-orientation histogram)

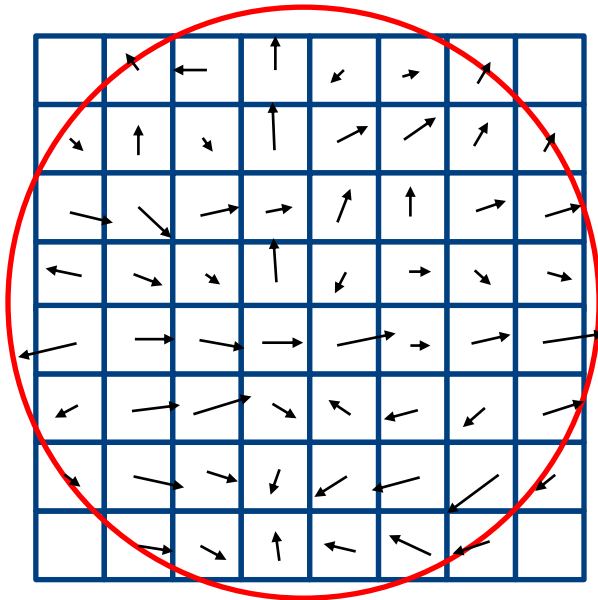
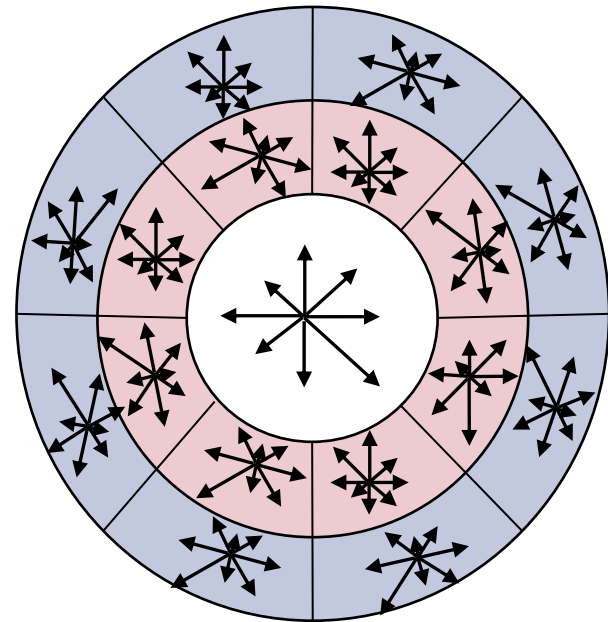


Image gradients

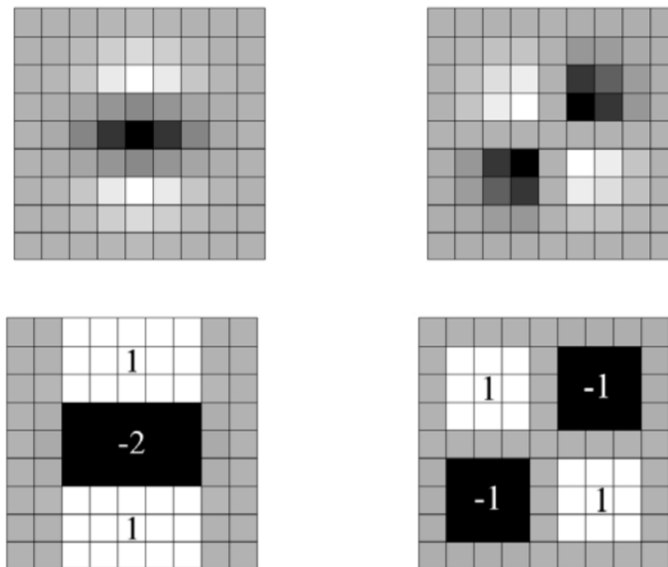


SURF: Speeded Up Robust Features

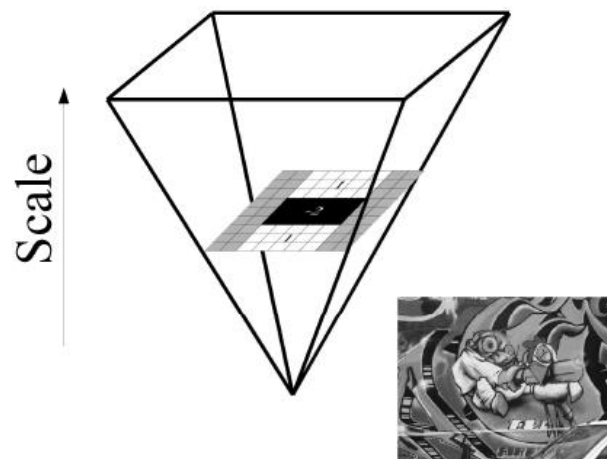
- The SURF feature detector works by applying an approximate Gaussian second derivative mask to an image at multiple scales and using the Hessian matrix to find points of interest

$$\det(H_{\text{approx}}) = D_{xx}D_{yy} - (0.9D_{xy})^2$$

- The method is very fast because of the use of integral images for image convolution
- Due to using integral images, filters of any size can be applied at high speed
- A keypoint is detected as a local maximum of the determinant of the Hessian matrix within a $3 \times 3 \times 3$ array similar to keypoint detection in the LoG or DoG scale space

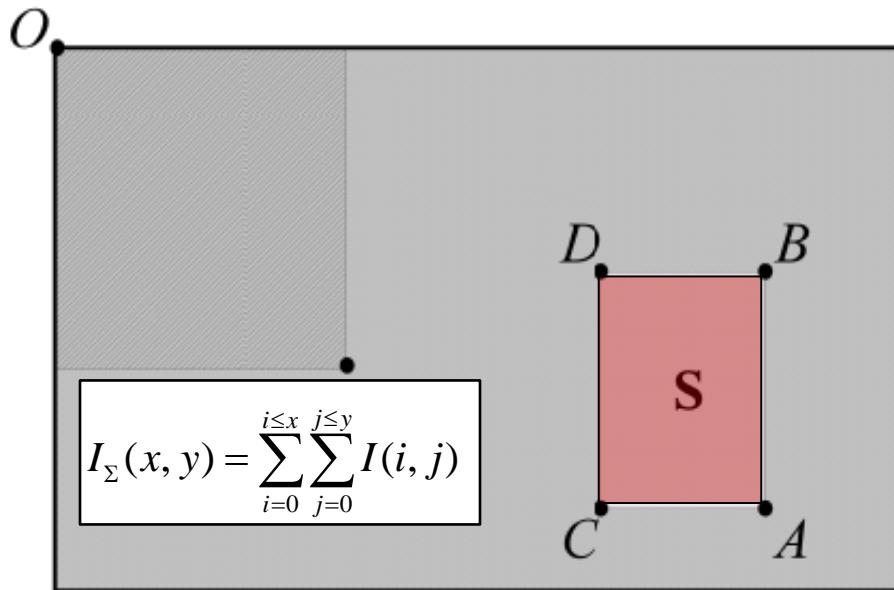


Gaussian second order partial derivatives in y-direction and xy-direction and their box filter approximations



SURF: Speeded Up Robust Features

- The integral image $I_{\Sigma}(x,y)$ of an image $I(x, y)$ represents the sum of all pixels in $I(x,y)$ of a rectangular region formed by $(0,0)$ and (x,y) .
- Using integral images, it takes only four array references to calculate the sum of pixels over a rectangular region of any size.



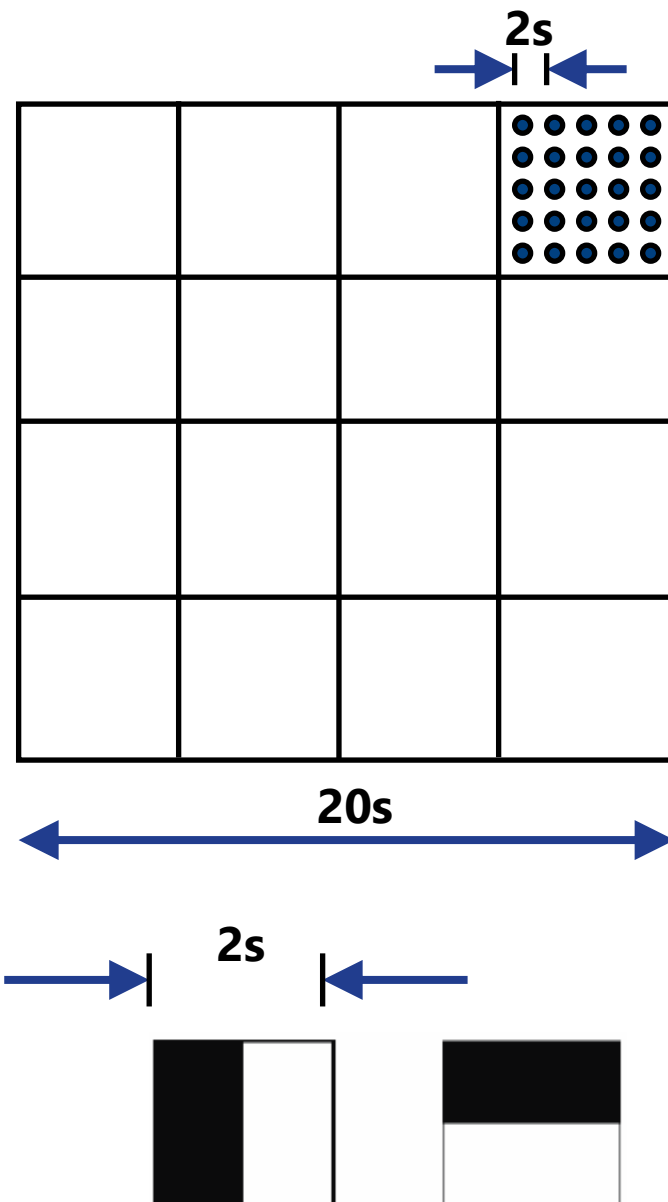
$$S = A - B - C + D$$

SURF: Speeded Up Robust Features

- To estimate the SURF descriptor of a keypoint:
 - Divide an image window of size $20 \times \text{scale}$ around the keypoint into 4×4 subregions
 - The Haar wavelet response d_x in the horizontal and d_y in the vertical direction is estimated in each subregion at 5×5 regularly spaced sample points
 - In each subregion:
 - estimate dx and dy at the 25 (5×5) points
 - sum over all 25 points to get 4 values

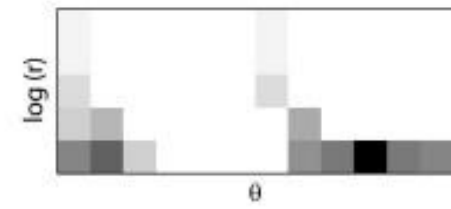
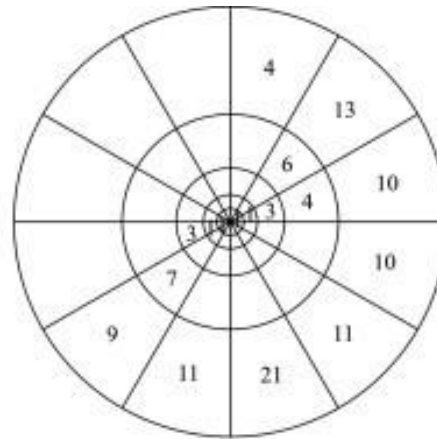
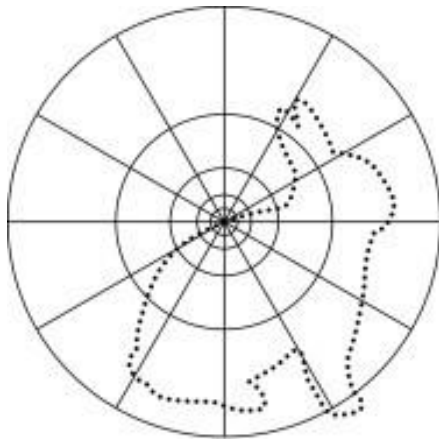
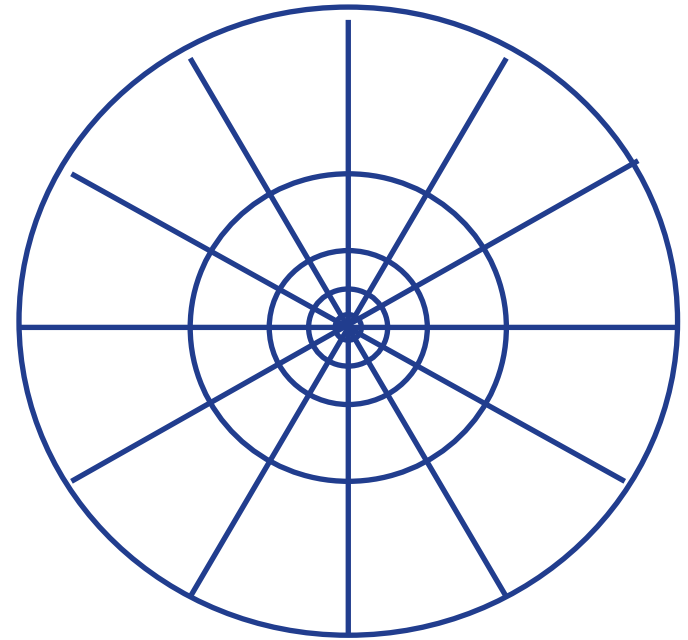
$$\text{SURF} = (dx, dy, |dx|, |dy|)$$

- This results in a descriptor vector of length $4 \times 16 = 64$
- 3 times faster than SIFT



Shape Context Descriptor

- The Shape Context descriptor estimates a 2D histogram of edge point locations and orientations
- The location is quantized into 5 bins using a log-polar coordinate system
- The orientation is quantized in 12 bins
- To estimate the descriptor, the number of points inside each bin are counted
- Log-polar binning provides more precision for nearby points, more flexibility for farther points



Common Detectors/Descriptors

← → ↻ ⚠ Not secure | robots.ox.ac.uk/~vgg/research/affine/ ☆ M

Overview

This page is focused on the problem of detecting affine invariant features in arbitrary images and on the performance evaluation of region detectors/descriptors.

Affine Covariant Regions

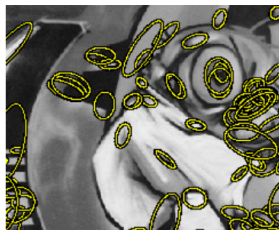


Image 1

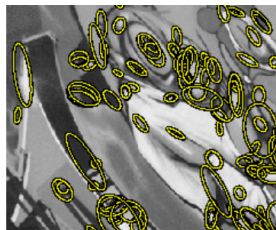


Image 2

Publications

Region detectors

- *Harris-Affine & Hessian Affine*: [K. Mikolajczyk](#) and [C. Schmid](#), Scale and Affine invariant interest point detectors. In IJCV 60(1):63-86, 2004. [PDF](#)
- *MSER*: [J. Matas](#), [O. Chum](#), [M. Urban](#), and [T. Pajdla](#), Robust wide baseline stereo from maximally stable extremal regions. In BMVC p. 384-393, 2002. [PDF](#)
- *IBR & EBR*: [T. Tuytelaars](#) and [L. Van Gool](#), Matching widely separated views based on affine invariant regions. In IJCV 59(1):61-85, 2004. [PDF](#)
- *Salient regions*: [T. Kadir](#), [A. Zisserman](#), and [M. Brady](#), An affine invariant salient region detector. In ECCV p. 404-416, 2004. [PDF](#)
- *All Detectors - Survey*: [T. Tuytelaars](#) and [K. Mikolajczyk](#), Local Invariant Feature Detectors - Survey. In CVG, 3(1):1-110, 2008. [PDF](#)

Region descriptors

- *SIFT*: [D. Lowe](#), Distinctive image features from scale invariant keypoints. In IJCV 60(2):91-110, 2004. [PDF](#)

Performance evaluation

- [K. Mikolajczyk](#), [T. Tuytelaars](#), [C. Schmid](#), [A. Zisserman](#), [J. Matas](#), [F. Schaffalitzky](#), [T. Kadir](#) and [L. Van Gool](#), A comparison of affine region detectors. In IJCV 65(1/2):43-72, 2005. [PDF](#)
- [K. Mikolajczyk](#), [C. Schmid](#), A performance evaluation of local descriptors. In PAMI 27(10):1615-1630. [PDF](#)

Software

- [Region detectors](#) - Linux binaries for detecting affine covariant regions.
- [Region descriptors](#) - Linux binaries for computing region descriptors.

<http://www.robots.ox.ac.uk/~vgg/research/affine/index.html>

Common Detectors/Descriptors

← → ↻ epfl.ch/labs/cvlab/research/descriptors-and-keypoints/ ☆ M

EPFL About Education Research Innovation Schools Campus **Coronavirus Info** Q EN

☰ Browse 🏠 > IC Machine Learning (ML) Computer vision Signal & image processing Artificial Intelligence (AI) Computer graphics Imaging and vision > Laboratories > CVLAB > Research >

Image Descriptors and Keypoint Detection

Image Descriptors and Keypoint Detection

3D Object Tracking

Deformable Surface Modeling

Unmanned Aerial Vehicles

Tracking and Modeling People

Modeling the Brain

Machine Learning for Biomedical Imaging

Domain Adaptation

Garment Simulation

Crowd Counting

Completed Projects

Image Descriptors and Keypoint Detection

We propose deep-learning solutions to extract image features and keypoint locations for applications such as image matching and 3D reconstruction. Opposed to hand-crafted features, such as SIFT, invariances are learned from example images.

Learned Local Features

- [LIFT: Learned Invariant Feature Transform](#)

Keypoint Detection

- [TILDE: A Temporally Invariant Learned DETector](#)

<https://www.epfl.ch/labs/cvlab/research/descriptors-and-keypoints/>

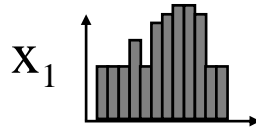
Image Processing with Local Feature

1. **Detection** - Find a set of distinctive key points.

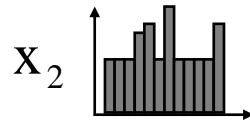


2. **Description** - Extract feature descriptor around each interest point as vector.

$$\mathbf{x}_1 = [x_1^{(1)}, \dots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \dots, x_d^{(2)}]$$



3. **Matching** - Compute distance between feature vectors to find correspondence.

$$d(\mathbf{x}_1, \mathbf{x}_2) < T$$



Feature Descriptor Distance

Two feature descriptors \mathbf{x} and \mathbf{y} can be compared using any of the following distances:

Quadratic distance

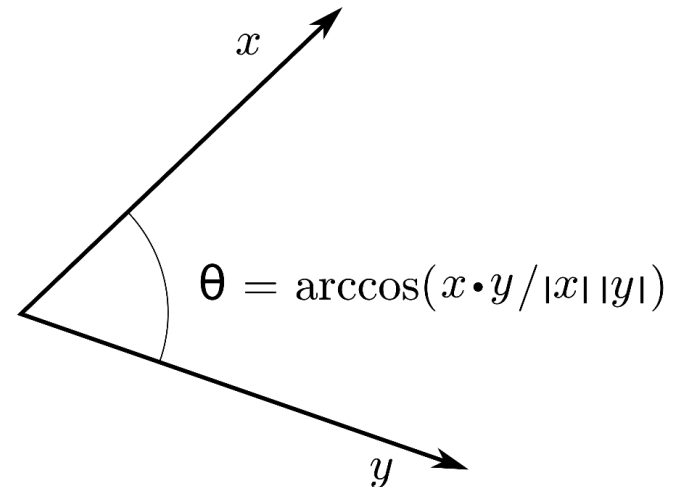
$$D(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^n |x_i - y_i|^2 \right)$$

x^2 distance

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n \frac{(x_i - y_i)^2}{x_i + y_i}$$

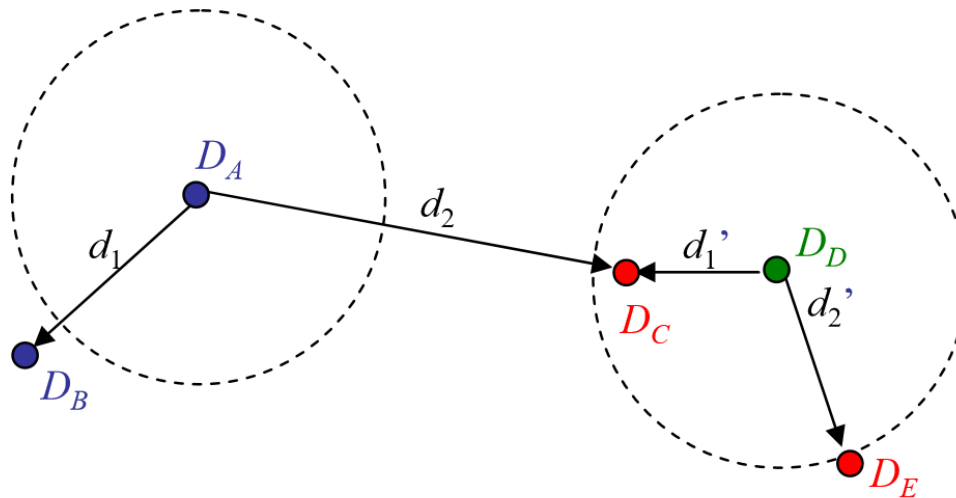
Cosine similarity

$$D(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} = \frac{x_1 y_1 + \dots + x_n y_n}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}}$$



Feature Matching

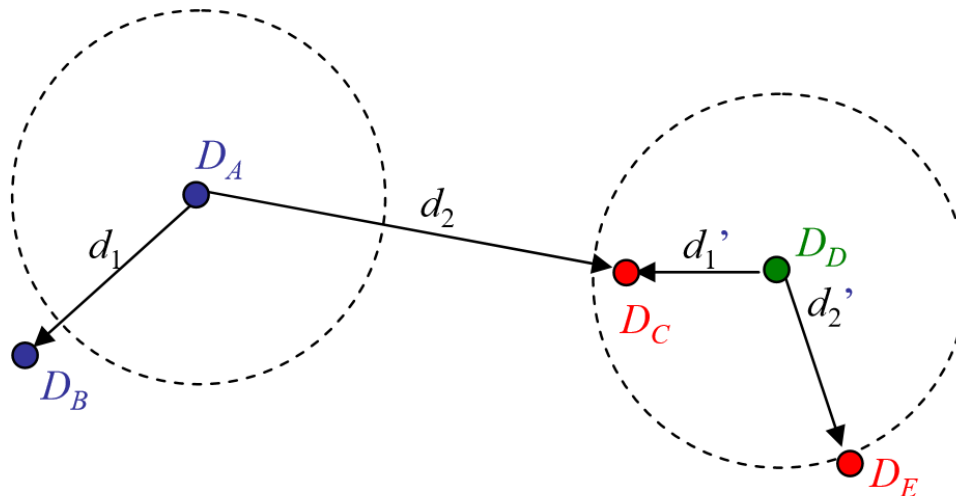
- Estimate the distance of keypoints in the feature space, e.g., Euclidean distance between 128-dim SIFT descriptors
- The simplest matching strategy is to set a **threshold** and to return all matches from other images within this threshold
- A **high threshold** results in too many incorrect matches being returned (false positives) while a **low threshold** results in too many correct matches being missed (false negatives)
- The problem with using a fixed threshold is that it is difficult to set



At a fixed distance threshold (dashed circles), descriptor D_A fails to match D_B and D_D incorrectly matches D_C and D_E .

Feature Matching

- A better strategy is to match a keypoint to its **nearest neighbour** in the feature space
- Since some features may have no matches (e.g., due to occlusion), a **threshold** can still be used to reduce the number of false positives
- Non-distinctive features could have lots of close matches, only one of which is correct



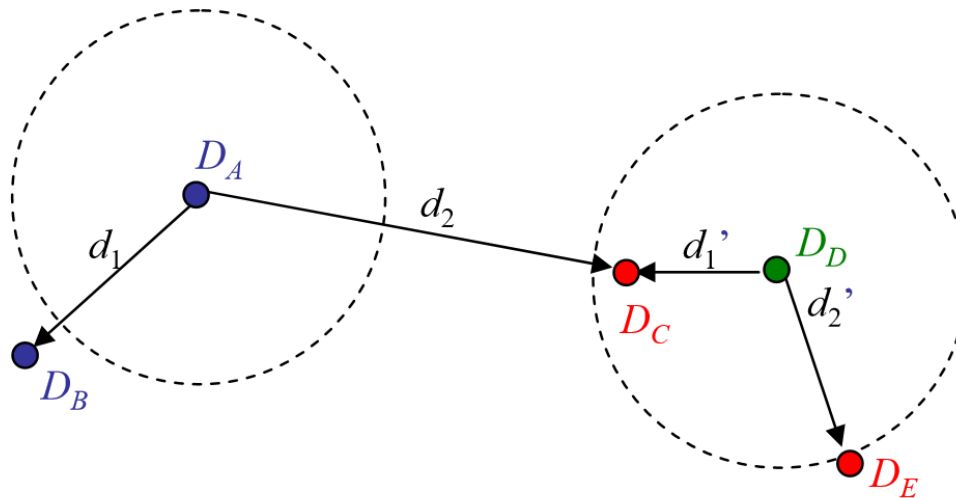
Using the nearest neighbour, descriptor D_A correctly matches D_B but D_D incorrectly matches D_C .

Feature Matching

- Another feature matching approach is the **Nearest Neighbour Distance Ratio**
- Estimate the distance of a feature vector to its Nearest Neighbour (d_1) and its second Nearest Neighbour (d_2)

$$NNDR = \frac{d_1}{d_2} = \frac{|D_A - D_B|}{|D_A - D_C|}$$

- If $d_1 \approx d_2$ $NNDR \approx 1$ the matching is ambiguous and should be rejected
- If $d_1 \ll d_2$ $NNDR \rightarrow 0$ the matching is correct



Using NNDR for matching, descriptor correctly matches D_A to D_B and correctly rejects matches for D_D .

Conclusions

- Image Processing with Local Features
- Feature Descriptors
- SIFT
- SURF
- Shape Context descriptor
- Feature Descriptor Distance
- Feature Matching



