Computer Vision

Module 5
Keypoint detection and keypoint descriptors

Krzysztof Krawiec
http://www.cs.put.poznan.pl/kkrawiec/

Poznan University of Technology, 2021/2022

Module outline

1. Introduction
2. Keypoint detection
3. Keypoint/feature descriptors (selected topics)
4. Feature matching

Keypoints (key points)

Also known as *features* (recall the earlier discussion on the ambiguity of this term).

Keypoint = a point (location in the image; in practice, a point with its neighborhood, *patch*) with the following characteristics:

1. has an expected characteristic (e.g. a corner, T-junction), or/and
2. is in some sense unique (usually within an image or scene).

Keypoint analysis typically involves two steps:

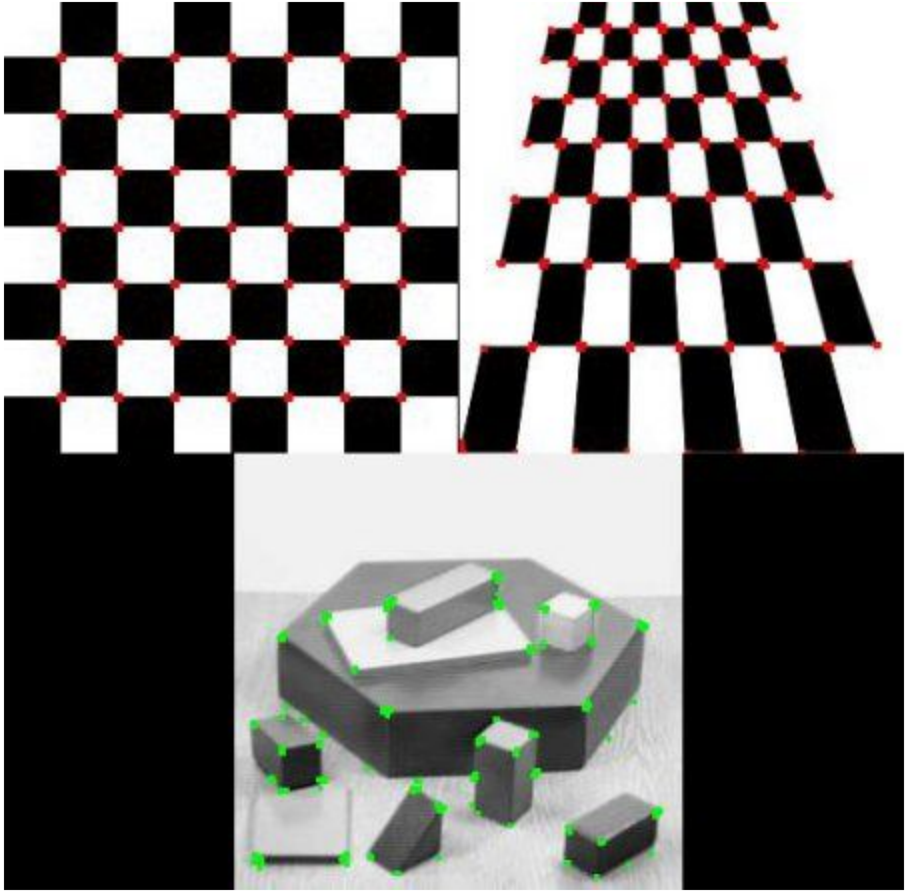
1. Detection (*feature detector*)
   Involves making a decision whether a location in the image is ‘characteristic enough’ to be considered a keypoint/feature.
2. Description (*feature descriptor*)
   Constructs a descriptor with desired properties (usually a vector).

Descriptors are then used for various purposes (e.g. matching).

Example

Images with keypoints (corners) detected using one of the popular detectors.

The descriptors of these keypoints capture the local characteristics of the image around a keypoint.



Applications

- Object detection
- Image indexing (for quick search for images containing certain features)
  - For instance: image databases, query by image content (QBIC)
- Identification of different images presenting the same object
  - E.g. different views/takes of the same scene.
- Image pair matching, including determination of the transformation matrix
  - *Stitching* of partially overlapping images (creation of panoramic images,)
- Detection of predefined markers
  - E.g. to support the navigation of a mobile robot.
- Tracking of objects in video sequences.
- Reconstruction of object’s shape based on feature points.

2. Keypoint detection

Keypoint detection

Desired detector characteristics:

- Invariance under selected transformations (mainly T and R)
- Replicability/robustness: detecting the same locations in similar images
- Speed of operation (frequent applications in video sequence analysis)

Keypoint detection is almost always based on a relatively small window (*patch, region of interest, aperture*), because:

- keypoints are by definition local,
- this facilitates parallelization of algorithms.

Nevertheless: Keypoint detection often uses windows that are significantly larger than those used in convolution and morphological processing.

Keypoint detection

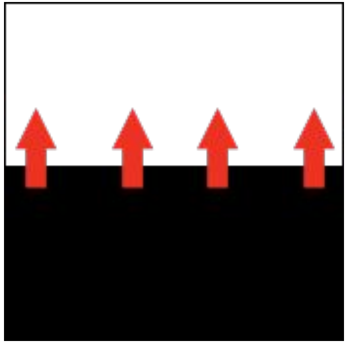
Keypoint detectors are typically designed to detect:

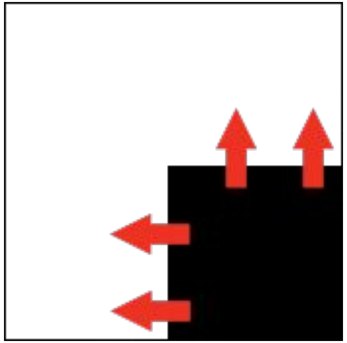
- Corners: a junction (including T-junction) or intersection of edges*:
  - Advantages: location can be precisely determined.
  - Disadvantages: hardly unique in a broader context (corners are very common, especially in images of man-made objects with many edges).
- Blobs: groups of adjacent points with relatively uniform characteristics that changes slowly in space
  - Advantages: can be more unique than corners (in terms of the changes in the spatial distribution of brightness).
  - Disadvantages: lower precision of location.

* Edges themselves are hardly ever considered as a target for keypoint detectors, as they are not unique/characteristic enough.

Detection of edges and corners

1. Edges: significant changes in brightness within the window when moving along *one* direction.
2. Corners: significant changes in brightness within the window when moving in *any* direction.
3. No edges nor corners: no significant changes when moving the window.





How to assess the directionality of changes?

Sum of squares of brightness differences (SSD).

For an image I, a window P, and displacement vector (Δx, Δy):

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

Question: how to avoid the expensive calculation of SSD for all possible (discrete) displacement vectors (Δx, Δy), individually for each image point?

Efficient approximation of the SSD

$$f(a) + \frac{f'(a)}{1!}(x-a)$$

Using Taylor series, the SSD can be approximated as:

$$I(x + \Delta x, y + \Delta y) \approx I(x, y) + I_x(x, y)\Delta x + I_y(x, y)\Delta y$$

where I_x and I_y are the derivatives of the brightness I w.r.t. x and y.

- The vector (I_x, I_y) is the gradient, calculated using the Sobel or Scharr filter.
- We disregard the higher order derivatives expressions of the Taylor series.

Then the estimate of the SSD for a window/path P can be expressed as:

$$SSD(\Delta x, \Delta y) \approx \sum_{x,y \in P} (I_x(x, y)\Delta x + I_y(x, y)\Delta y)^2$$

This in turn can be written in matrix notation as:

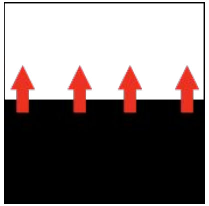
$$SSD(\Delta x, \Delta y) \approx \sum [\Delta x \quad \Delta y] \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

Examining special cases

Matrix M for special cases:

Horizontal edge: only the vertical component of the gradient is non-zero

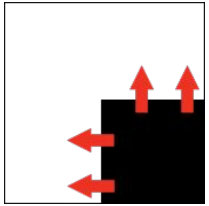
$$M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$



(and analogously for a vertical edge, of course)

Corner: both horizontal and vertical components of the gradient are non-zero

$$M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$



Rotations

The symmetry of matrix M allows for its decomposition:

$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

where R is the rotation matrix, and λ_1 and λ_2 are the eigenvalues of the matrix M.

If λ_1 and λ_2 are large, the location features a corner with high probability, and this does not depend on the rotation angle of the corner.

The Harris detector (1988) uses the criterion:

$$c = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 = det(M) - k \cdot trace^2(M) > thr$$

Shi-Tomasi (1994) detector uses the criterion:

$$c = \min(\lambda_1, \lambda_2) > thr$$

The complete algorithm

Input: image I

1. Calculate the derivatives I_x and I_y of I, i.e. the gradient (e.g., Schaar filter)
2. Calculate the squares of I_x and I_y and the pixelwise product $I_x I_y$
3. Perform local aggregation of the above values by convolving the above images within a window of size P with:
   a. An averaging filter (summing, mask weights 1), or
   b. A Gaussian filter (lowers the impact of the off-center pixels of the patch, which is desirable as per the assumed locality of the detector)
4. Calculate the 'cornerness' criterion c according to the Shi-Tomasi or Harris formula
5. Locate the points in the image where c is greater than a threshold.
   a. Desired: determining multiple/all *local* maxima.
   b. Non-maximum suppression (NMS)* can also be used.

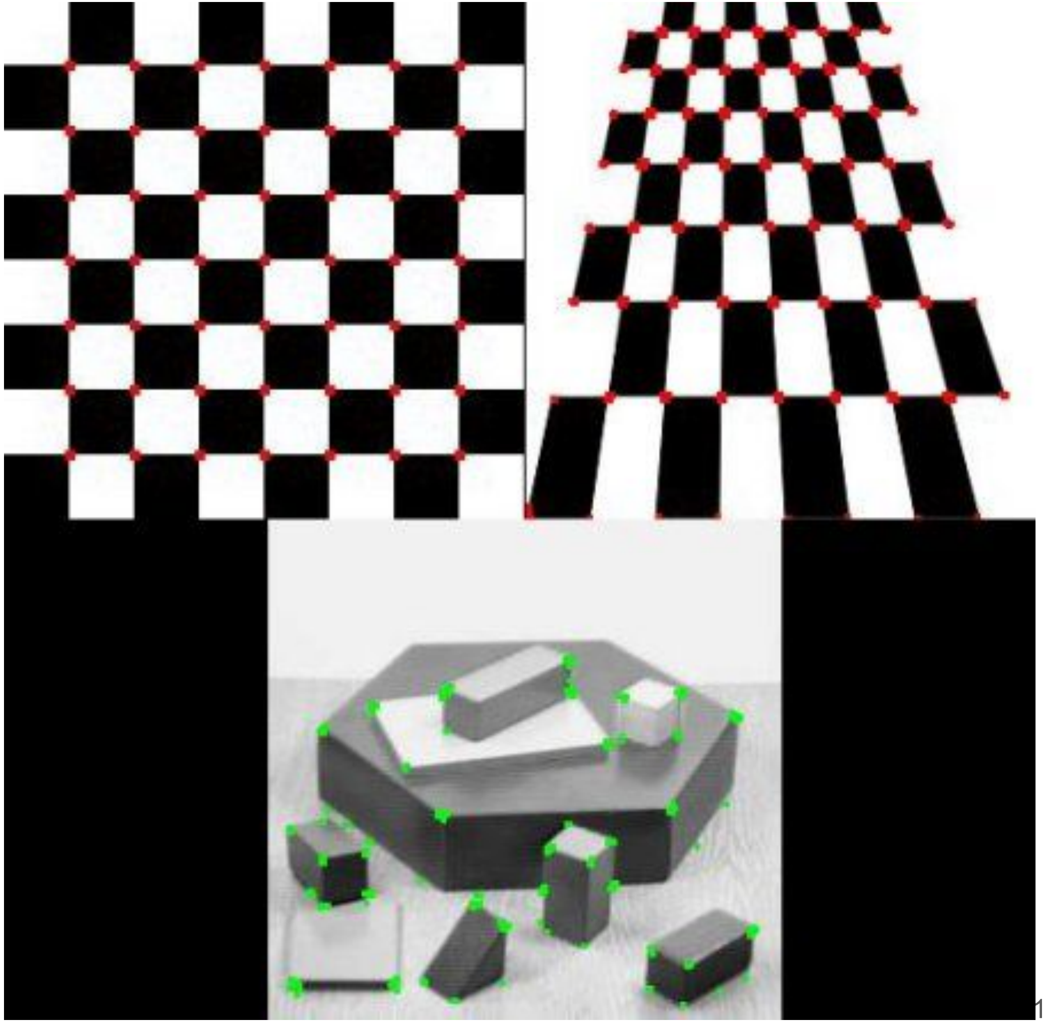
Parameters: window size P; size of the gradient filter mask.

* We mean here the pixel-wise NMS, not the bounding-box NMS used elsewhere.

Example

The effect of the Harris detector on two synthetic images and a real image.

The ‘corner points’ of a checkerboard are sometimes called *saddle points*.



Corner detection: final remarks

- Invariant under rotation
- Some degree of invariance under scaling
  - Because at high magnification, individual pixels can be treated as edges
- Can be relatively easily generalized to corner detection at <u>subpixel accuracy</u>
  - The method attempts to move (at sub-pixel resolution) the center point of the window while maintaining a high gradient.
- In addition to <u>non-maximum suppression</u>, more global elimination of similar detections can also be used.

  A simple (though not necessarily very efficient) algorithm:
  - Sort the detections (points) descending by the value of the criterion c
  - Iterate through the list, and for each successive point p, remove from the rest of the sorted list other detections p' whose Euclidean distance from p is smaller than some threshold r.

3. Keypoint descriptors

Characteristics of descriptors

Edges and corners are very simple image features.

- In fact, local features are often more sophisticated.
- Going beyond just edges and corners allows expanding the repertoire of features, and characterize image points in a ‘more unique’ way
    - This allows identifying them more reliably.

Characteristics of descriptors:

- A feature descriptor is a vector that characterizes the desirable properties of points of interest.
- Descriptors are usually designed to capture more refined features than edges, i.e. towards broadly defined 'blobs'.

Selected keypoint descriptors

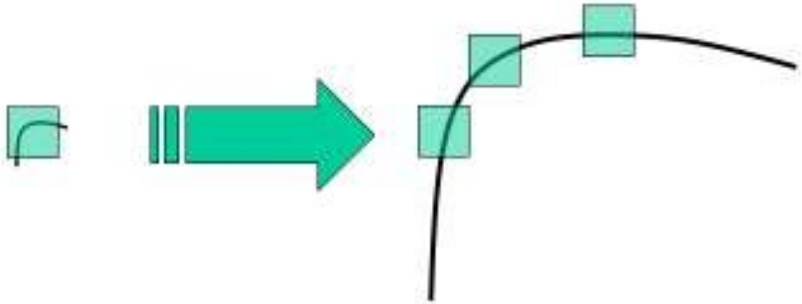
A wide range of diverse descriptors have been proposed.

- SIFT (Scale-Invariant Feature Transform)
- SURF (Speeded-Up Robust Features)
- FAST Algorithm for Corner Detection
- BRIEF (Binary Robust Independent Elementary Features)
- ORB (Oriented FAST and Rotated BRIEF)
- ...

There are too many of them to cover in detail, so the following slides present selected aspects/components of some descriptors.

Multiscale representation

Used in SIFT, among others.



Motivation: a fixed detector window size is insufficient to detect the pattern at a larger scale (see the example).

- A detector tuned for a small complex shape (left) may fail detecting it when it appears larger (right).

Solution: construct a 'pyramid' of input images with successively smaller resolutions.

For example, the SIFT detector uses a multi-scale representation to efficiently approximate the Laplacian of Gaussians (LoG) with the Difference of Gaussians (DoG).

Laplacian of Gaussian (LoG)

Conceptually, calculation of LoG involves two steps:

1. Smoothing of the image f with a Gaussian filter g (to de-noise and reduce small local brightness fluctuations).
2. Applying the Laplassian ∇^2 filtering to detect edges.

These two steps can be folded into one:

$$\nabla^2(f * g) = f * \nabla^2 g$$

which yields:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

or in short:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - r\right] e^{-r}$$

where r is the standardized radius (implicitly: the radius of the blob).

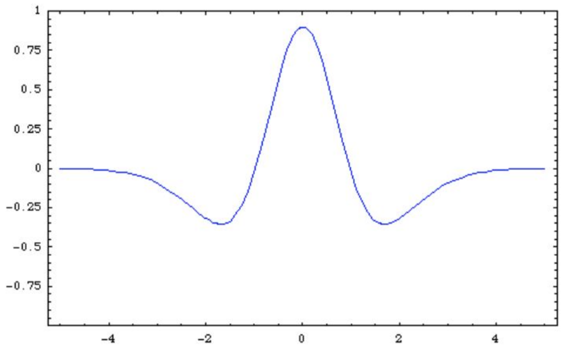
The Laplacian of Gaussian (LoG)

LoG returns

- a strongly positive response for dark blobs of radius $\sim\sqrt{2}\sigma$
- a strongly negative response for bright blobs of radius $\sim\sqrt{2}\sigma$

A number of detectors calculate LoG multiscale, for different values of σ, thus working in a three-dimensional space (X, Y, σ).

- One looks for points that are local maxima in this space,
  e.g., by literally comparing $LoG_\sigma(x,y)$
  at a given point with its 3^3-1=26 neighbors.



In practice, LoG is often approximated by the difference of Gaussian filters with different values of σ, the so-called **Difference of Gaussians** (DoG).

Histogram of Oriented Gradients (HoG)

1. Calculate the gradient (Sobel or Schaar)
2. Determine the direction d of the gradient at each image point
3. At each image point, create a histogram of the d in the local window (cell), discretizing the angle to n values (n: histogram length).
   a. The histogram is incremented by the length of the gradient vector.
4. Normalize histograms locally in groups of neighboring cells (blocks).

$$d(x, y) = \mathbf{arctan}(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x})$$

Variants and extensions of the method:

- Applying directly to multi-channel images;
  - The direction in step 2 is determined from the dominant channel.
- Using also the second derivative in step 2 as an additional feature.
- Using different normalizations in step 4.

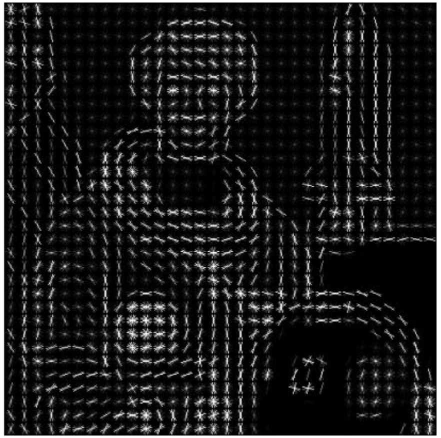
Histogram of Oriented Gradients (HoG): Example

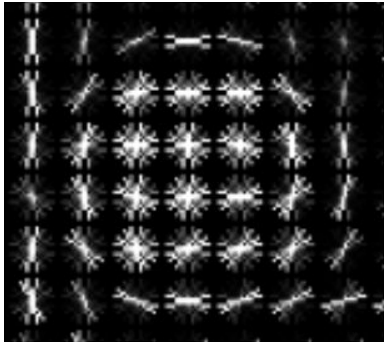
The right images visualize the HoG in 16x16 pixel <u>cells</u> (i.e. selected <u>patches*</u>).

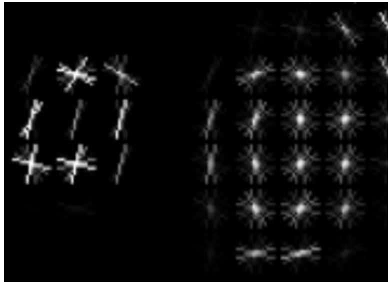
Frequency of a given gradient direction is reflected by brightness.

- Many details ⇒ roughly uniform histogram of relatively large values.
- No clear features ⇒ roughly uniform histogram of relatively low values.
- Edge ⇒ one dominant maximum in the histogram.

* visualizations show only cells, i.e. <u>non-overlapping</u> patches.



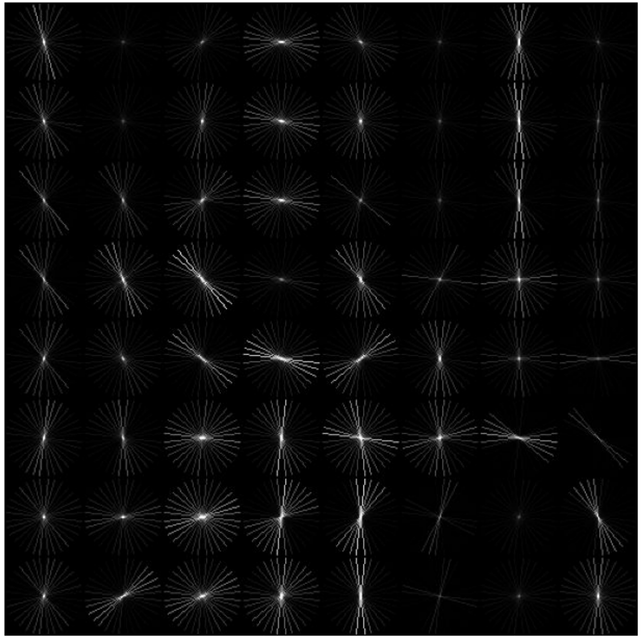




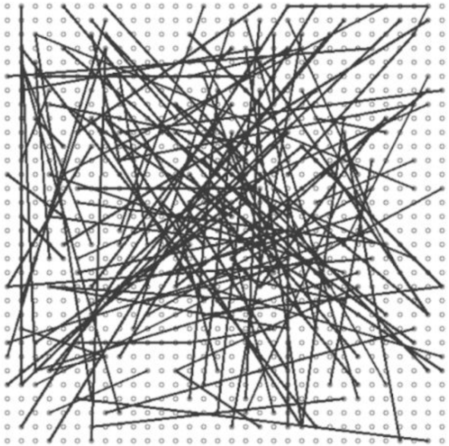
Histogram of Oriented Gradients (HoG): Example

Same input image, but:

- Cell size increased from 16x16 to 64x64
- The number of considered directions/angles (n) increased from 8 to 16



BRIEF Descriptor



Binary Robust Independent Elementary Features

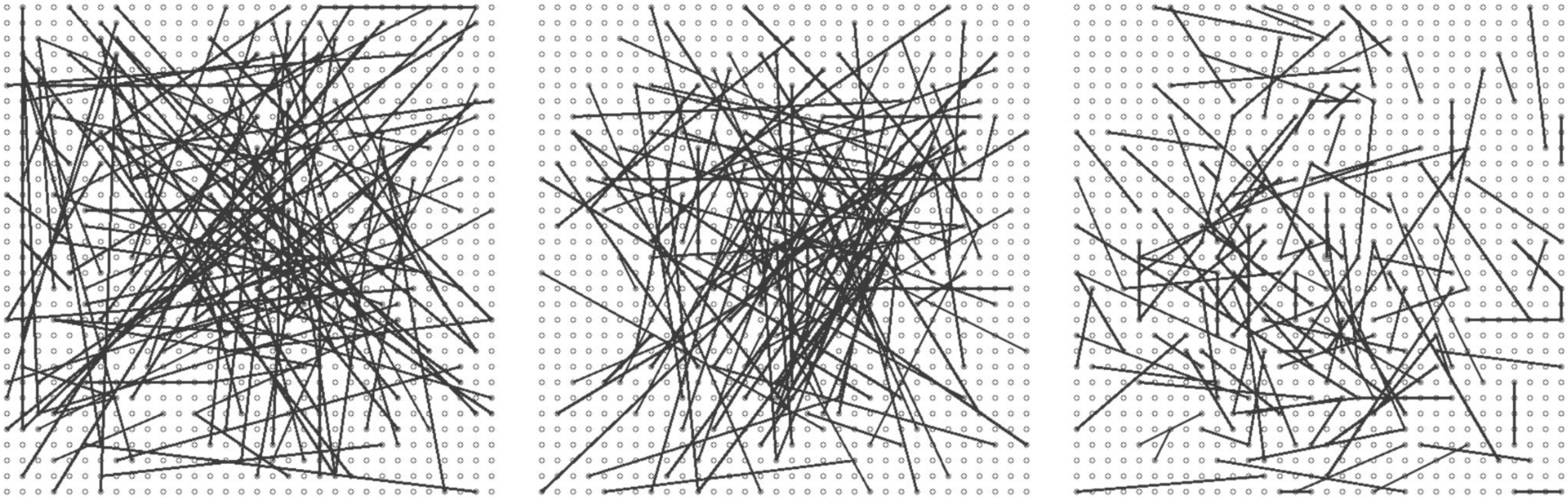
1. Sample at random the list L of 256 pairs $((x_1, y_1), (x_2, y_2))$ of coordinates within the neighborhood/patch size.
   a. Important: this list remains fixed, i.e. it is a fixed property of this particular instance of the BRIEF descriptor.
2. Blur the image (e.g. with the Gaussian filter).
3. For each keypoint (or keypoint candidate)
   a. For each $((x_1, y_1), (x_2, y_2)) \in L$
      i. Retrieve the values (p_1, p_2) from, respectively, locations (x_1, y_1) and (x_2, y_2)
      ii. If $p_1 < p_2$, set the corresponding descriptor bit to 1, otherwise to 0
   b. The result (per keypoint) is binary vector of 256 elements.

Properties:

- Very fast.
- Easy to compare/match due to binary nature: Hamming distance.
- Sensitive to scaling and rotation.

BRIEF Descriptor

Examples of different samplings of random point pairs in a window (list L).



BRIEF: Computing a Local Binary Descriptor Very Fast, M. Calonder; V. Lepetit; M. Özuysal; T. Trzcinski; C. Strecha et al., IEEE Transactions on Pattern Analysis and Machine Intelligence. 2012. Vol. 34, num. 7, p. 1281-1298. DOI : 10.1109/TPAMI.2011.222.

4. Feature matching

Feature matching

One of the main uses of feature descriptors: essential for efficient finding/locating of objects in images.

The task: for two given sets of descriptors (D_1,D_2) extracted from a pair of images, find their most likely pairing (in terms of the adopted quality metric).

The convenience: The feature matching problem is (essentially) independent of the feature description problem (i.e., on the way in which the feature descriptors have been obtained).

- As a result, the matching algorithms are typically generic.

The challenge: Computational complexity.

Feature matching: Exhaustive algorithm

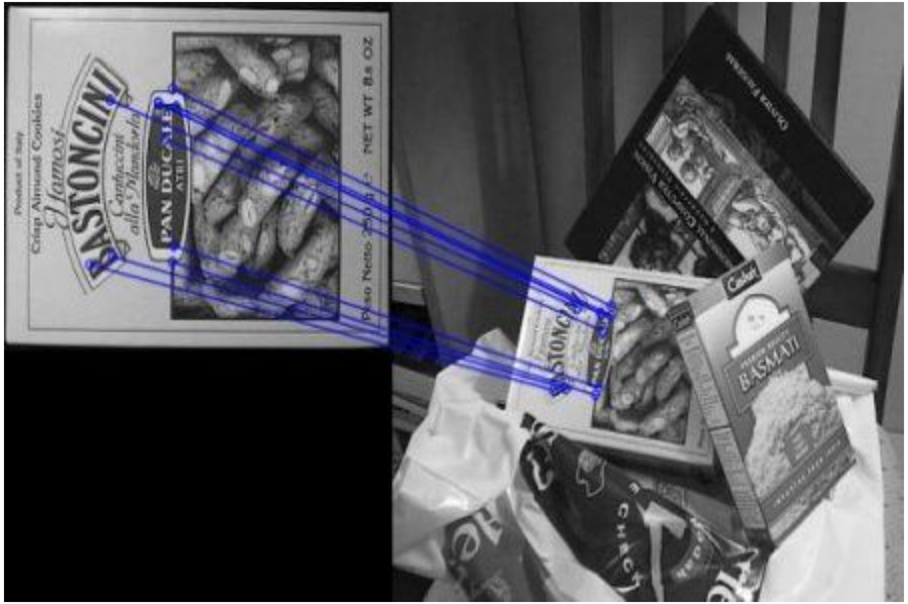
Also known as *exact/brute-force matching*

For each descriptor in D_1, finds the closest one in D_2 in terms of the accepted metric (usually Euclidean; sometimes Hamming distance, e.g. for discrete descriptors such as BRIEF).

<u>The symmetric version</u>: returns only the pairs of descriptors (d_1,d_2) for which it holds simultaneously:

$$d_2 = \min_{d \in D_2} ||d - d_1||$$
$$d_1 = \min_{d \in D_1} ||d - d_2||$$

Example on the right.



The problem of ambiguous pairings

In practice, it often happens that for a descriptor d_1 in D_1 there exists more than one very similar counterpart in D_2.

- The maximum-based criterion (see previous slide) will always choose the closer (more similar) descriptor.
- The result: false positive matches.

Countermeasure: distance ratio (Lowe 2004). For a given descriptor d_1 in D_1:

- Calculate the ratio of distances of the *two closest matches* in D_2 (smaller distance in the numerator).
- Accept d_1 only if the quotient does not exceed a given threshold (often recommended value: 0.7).

This technique is especially recommended for discrete descriptors (e.g., BRIEF).

Feature matching: Other algorithms

The exhaustive search method is

- computationally expensive (quadratic complexity as a function of the number of keypoints),
- only locally optimal: does not guarantee minimization of a global criterion, such as the sum of distances between all pairs of descriptors.

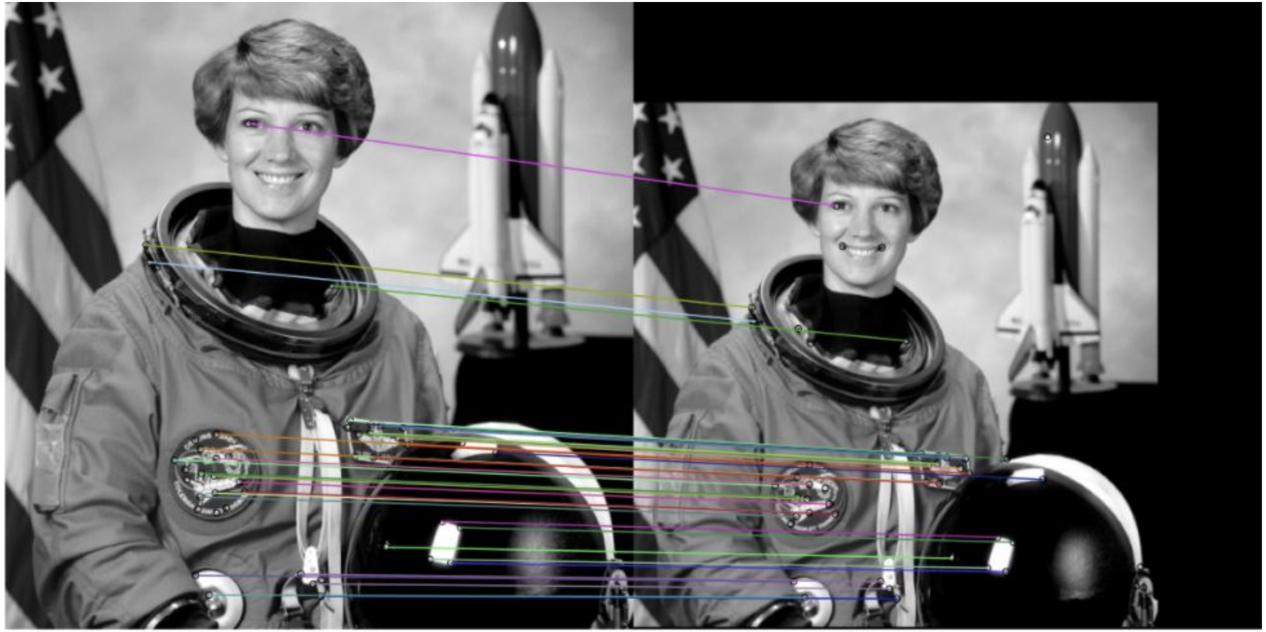
Other matching algorithms:

- Use different variants of fast (approximate) nearest neighbor search (in descriptor space).
  - In particular: They are often based on k-d tree structures, i.e. trees (usually binary) spanning the dimensions of the search space (here: number of dimensions = length of descriptors)
  - May involve multiple random sampling and estimation, e.g. RANSAC.
- E.g. Fast Library for Approximate Nearest Neighbors in the OpenCV library.

Example: BRIEF

Effect of the BRIEF descriptor in the presence of scaling (symmetric matching).

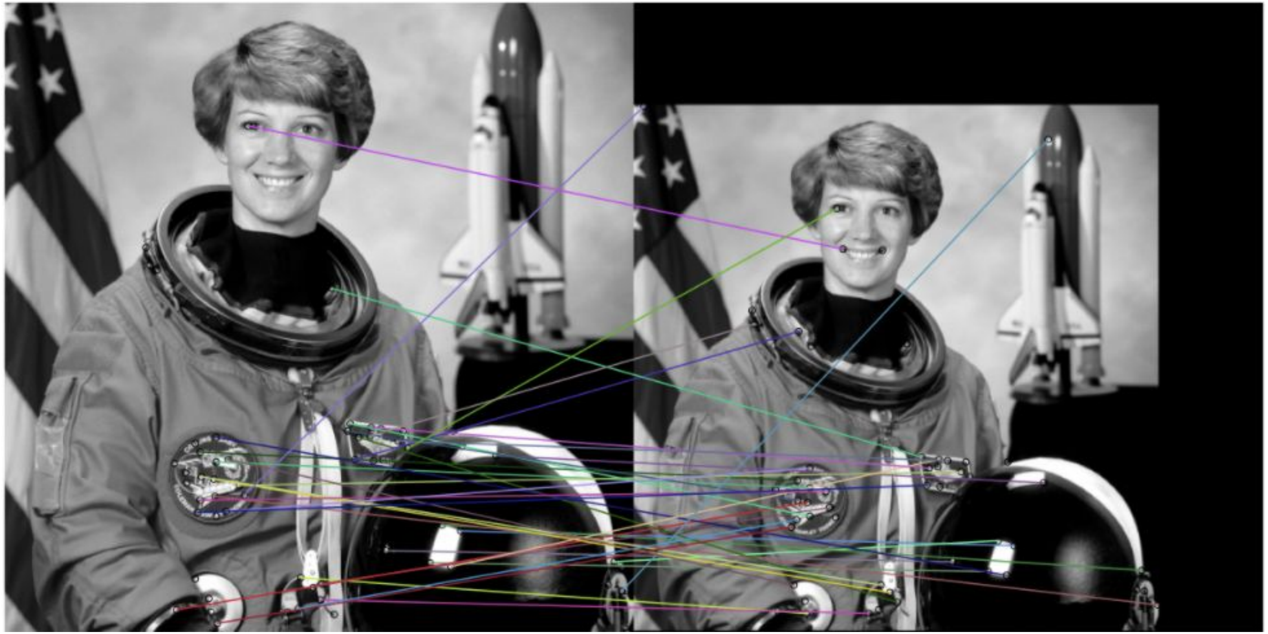
- Descriptor applied <u>only to the keypoints detected with the Harris detector</u>.



Example: BRIEF

Effect of the BRIEF descriptor in the presence of scaling (symmetric matching).

- Descriptor applied to all image points.
- Conclusion: preselection provided by the detector is essential.



Applications of feature matching

- Detecting objects in a scene
  E.g. Detect an object in the scene if the required percentage of descriptors of the [known] prototype are found in the image.
- Determining the homography, i.e., the bijective transformation (isomorphism) of the projection space (or, more precisely, of the resulting set of descriptors).
  - The result: a transformation matrix

Knowledge of the transformation matrix allows (depending on the context):

- Estimating the position of an object in the scene (pose estimation)
- Estimating the camera position relative to the scene (camera calibration).

Module summary

Module outline

1. Introduction
2. Keypoint detection
3. Keypoint/feature descriptors (selected topics)
4. Feature matching