



UNIVERSITÀ DEGLI STUDI DI PADOVA

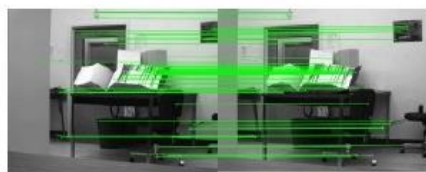
An overview of existing features

Stefano Ghidoni





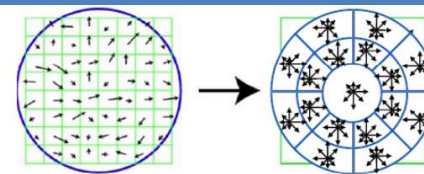
- SIFT-based features
- Features for analyzing shapes
- Binary features
- Feature comparison



(a) The left and right image of a stereo pair.



(b) All image rotated 30°, cropped and scaled by a factor 2 in respect to the original image on the left side.



(a) image gradients

(b) keypoint descriptor

Figure 4.19 The gradient location-orientation histogram (GLOH) descriptor uses log-polar bins instead of square bins to compute orientation histograms (Mikolajczyk and Schmid 2005).

PCA-SIFT

SURF

GLOH



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Principal Component Analysis (PCA)

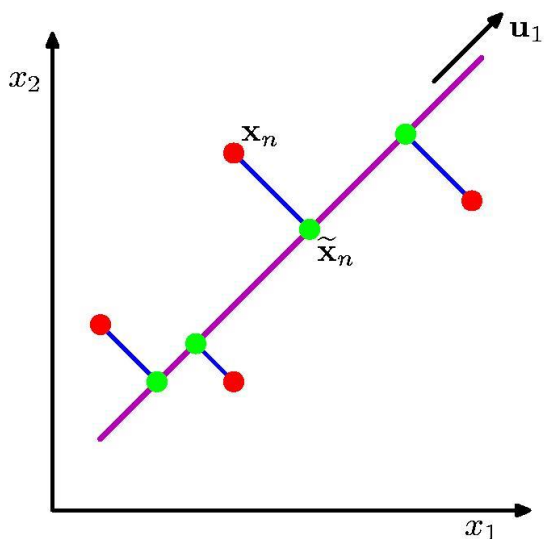
IAS-LAB

- What is PCA?



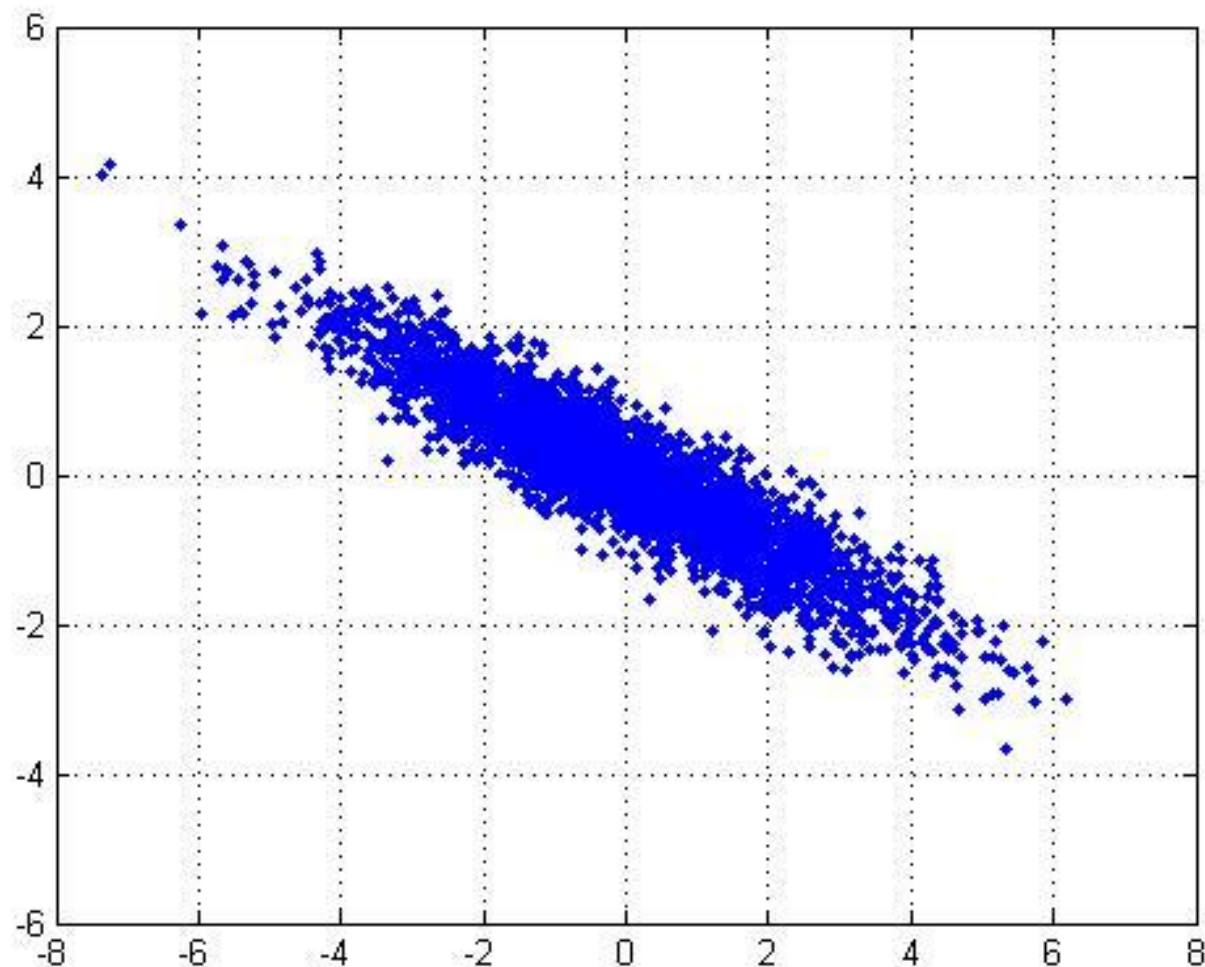
- What is PCA?
- A dimensionality reduction technique
- Reducing dimensions can be useful to
 - Work on more compact representations
 - Reduce computational workload
 - Highlight the most important information hiding the details

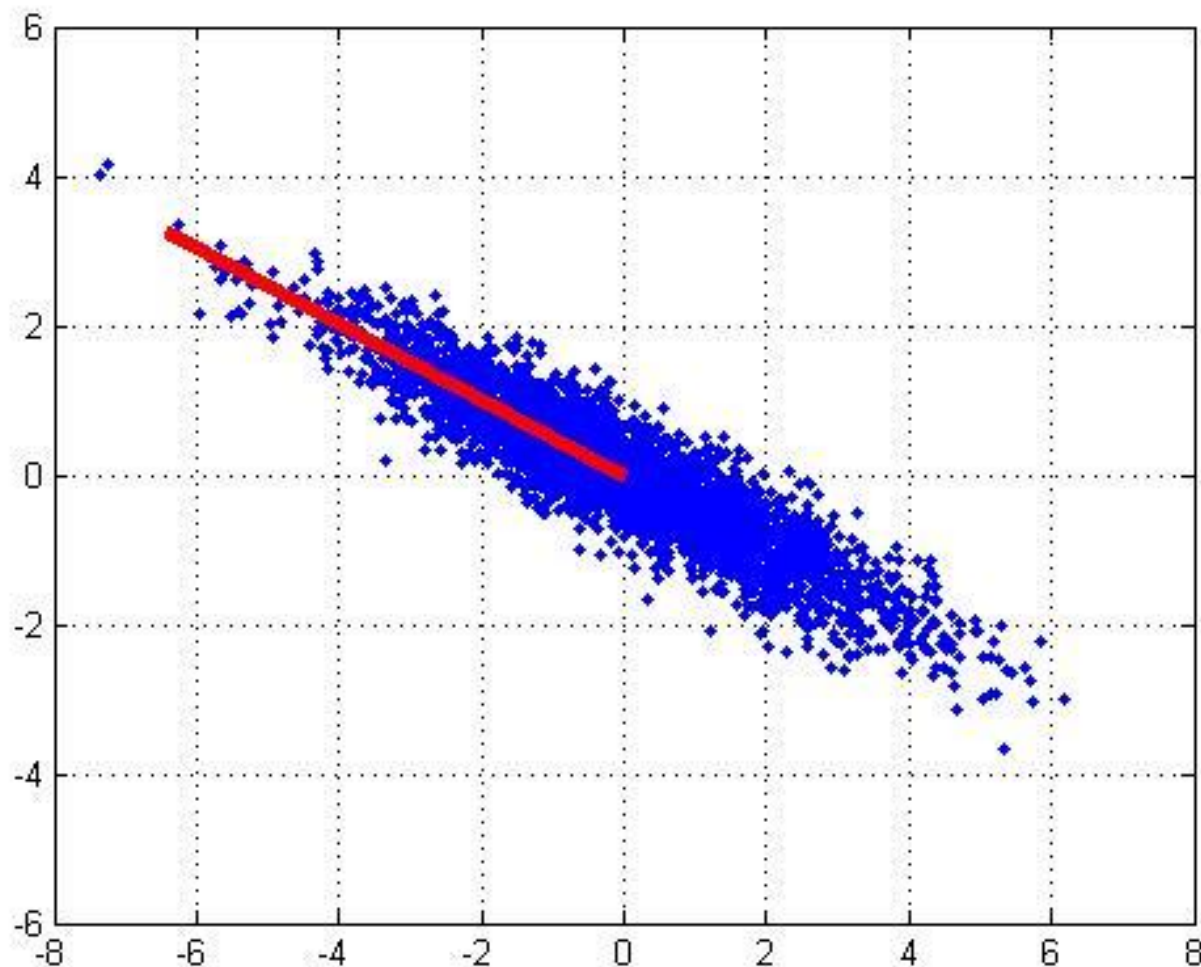
- Recall: PCA is an orthogonal projection of data onto a lower dimensional linear space that
 - Maximizes variance of projected data (purple line)
 - Minimizes mean squared distance between original data points and their projection (sum of blue lines)





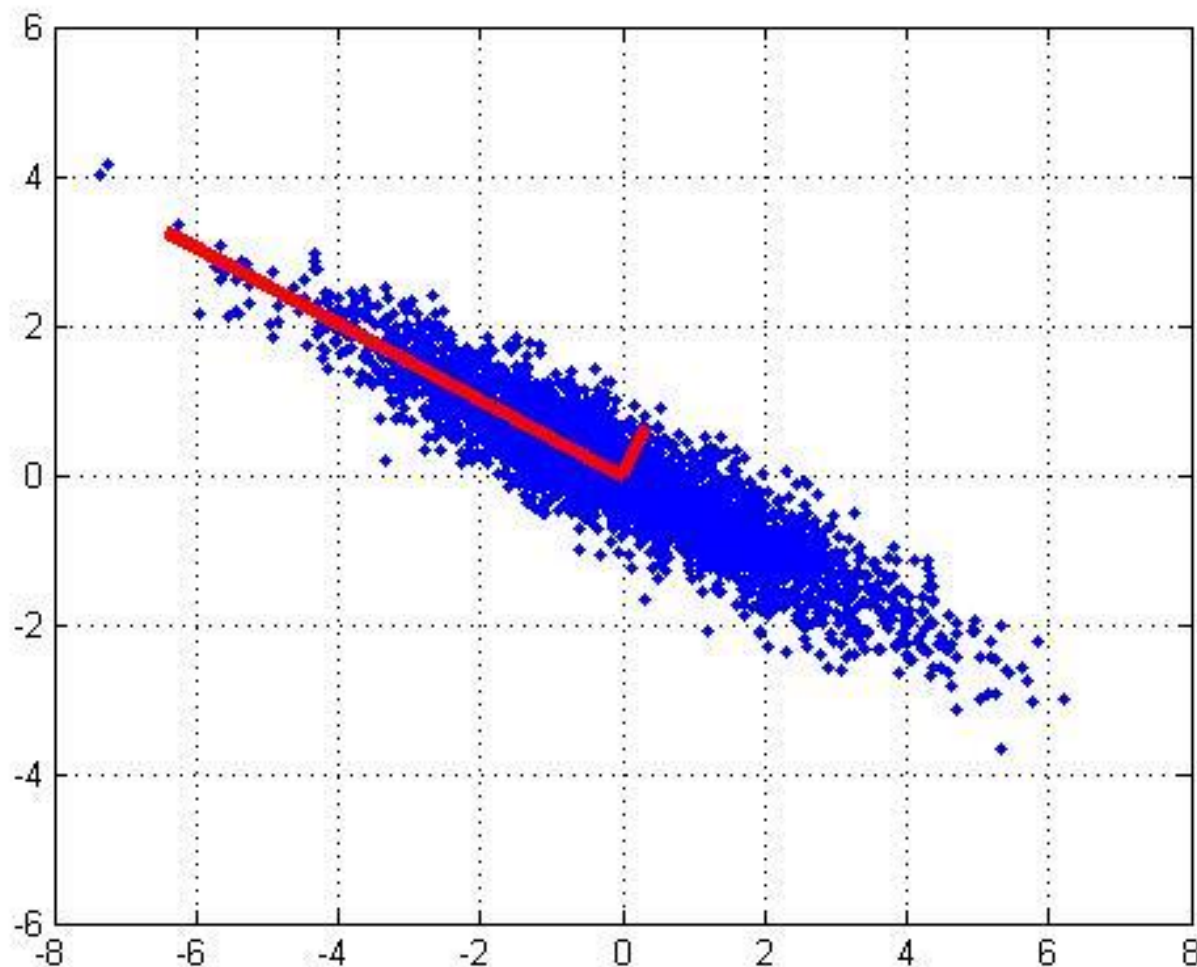
- Principal component #1 points in the direction of largest variance
- Each other principal component
 - Is orthogonal to the previous ones
 - Points in the direction of largest variance of the residual subspace



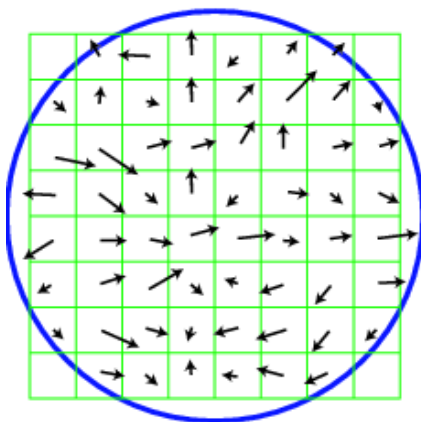




Example: 2nd axis



- Steps 1-3: same as SIFT
- Modify step 4 (descriptor calculation)
- Refer to a 41×41 patch at the given scale, centered around the keypoint, normalized to a canonical direction





- SIFT: weighted histograms
- PCA-SIFT: concatenate horizontal and vertical gradients (39×39) into a long vector
 - Length: 2 directions $\times 39 \times 39 = 3042$
- Normalize the vector to unit norm
- Reduce the vector using PCA
 - E.g., from 3042 to 36

SURF

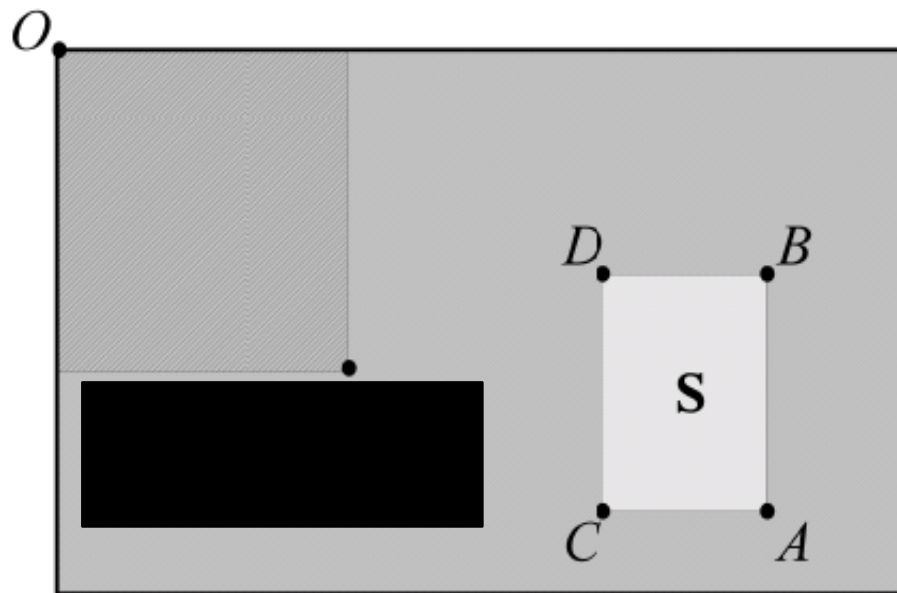


- Speeded Up Robust Features (SURF)
- Speed-up computations by fast approximation of hessian matrix and descriptors using integral images

- SURF uses the integral image $I_{\Sigma}(x, y)$
 - Each pixel represents the sum of all pixels in the rectangle between $(0,0)$ and that pixel

$$I_{\Sigma}(x_i, y_i) = \sum_{i=0}^{x_i-1} \sum_{j=0}^{y_i-1} I(i, j)$$

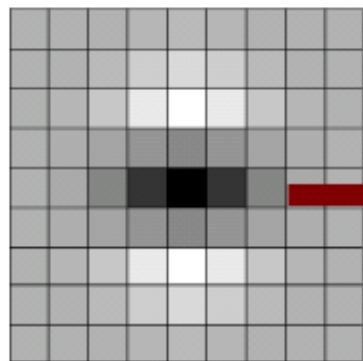
- $S=A+D-B-C$



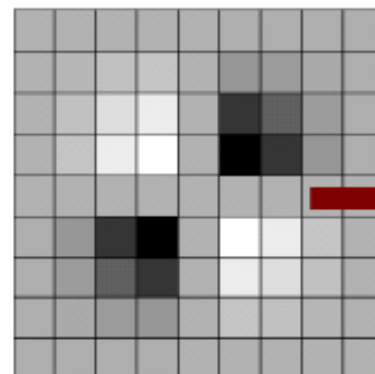
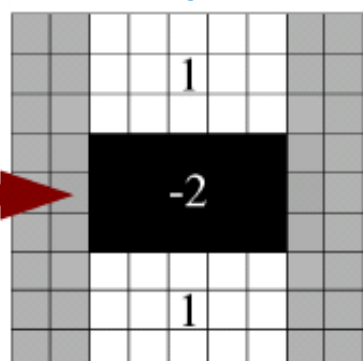
- A box filter is composed of black and white rectangles
 - White: positive weight
 - Black: negative weight
- Box filters exploit the integral image
 - Calculation is constant-time irrespective of the filter size

DERIVATIVE FILTER (actually LOG)

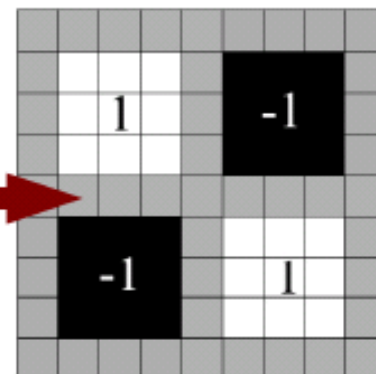
BOX FILTER APPROX DERIVATIVE



L_{yy}



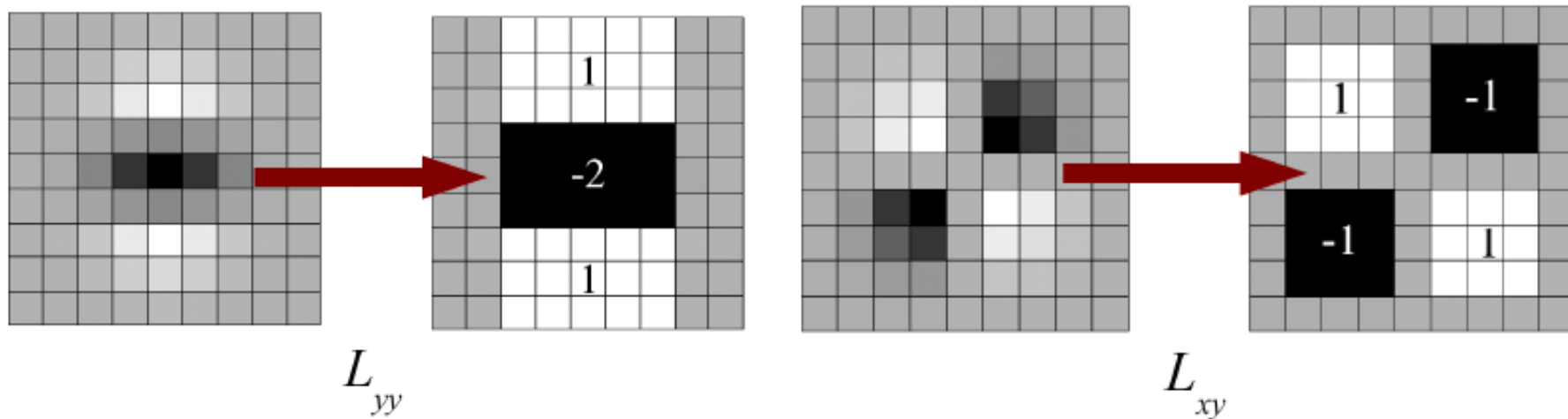
L_{xy}



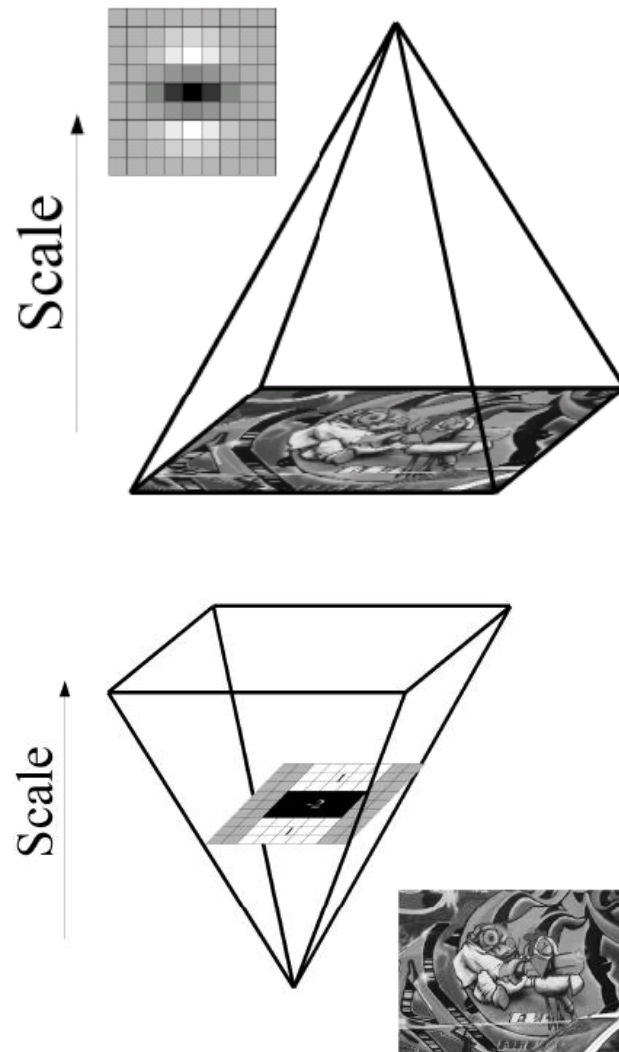
- Keypoint detection: compute the Hessian matrix in scale space

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

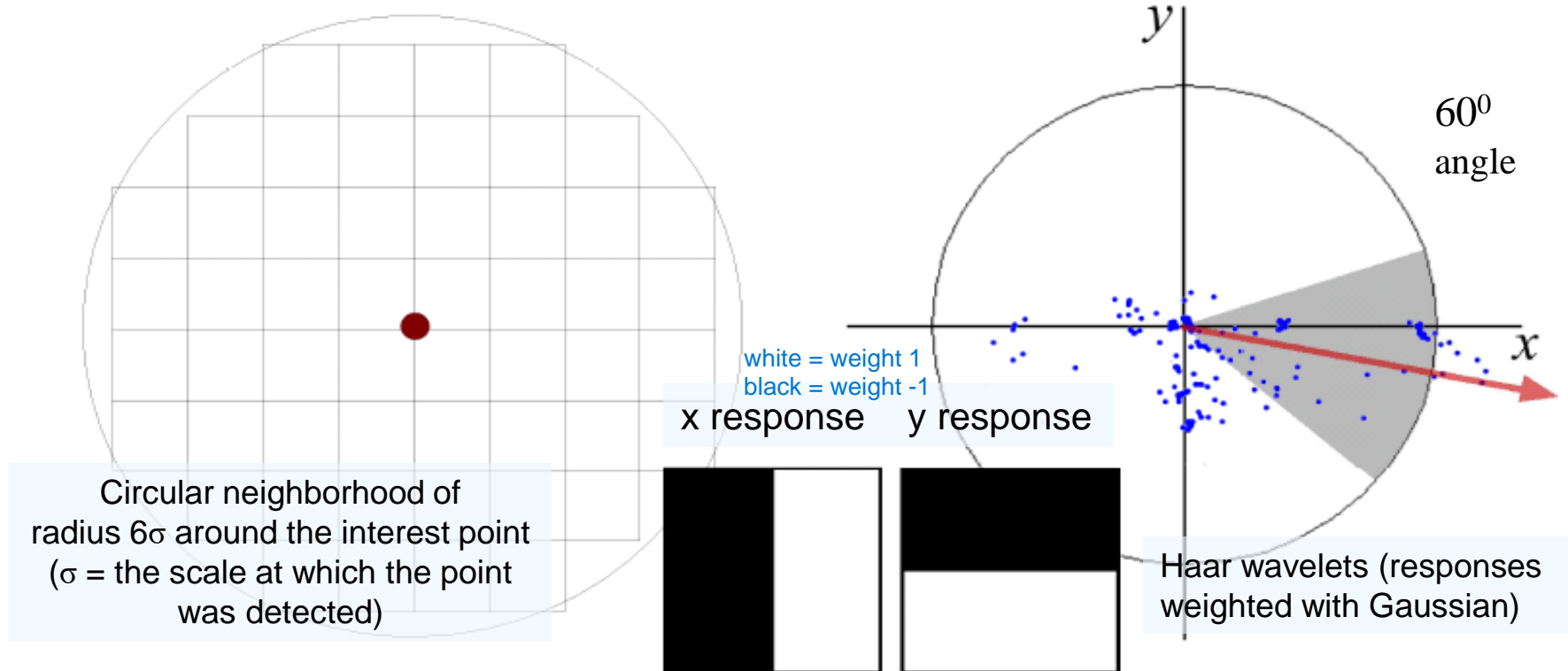
- Approximate L using box filters (fast calculation)
- Find the maximum of the determinant



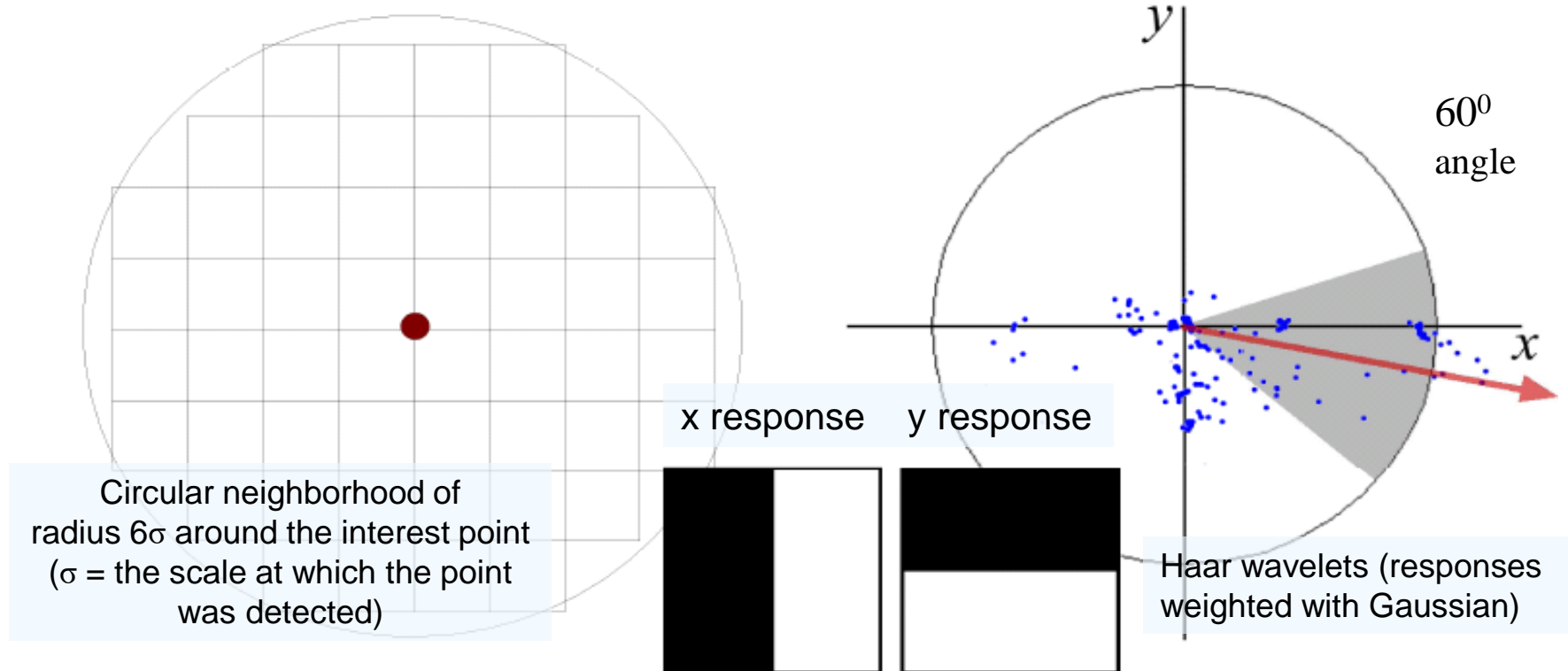
- SIFT: images are smoothed with a gaussian and subsampled to move along the pyramid
- SURF: use filters of increasing size
 - Computational time does not depend on filter size using the integral image!



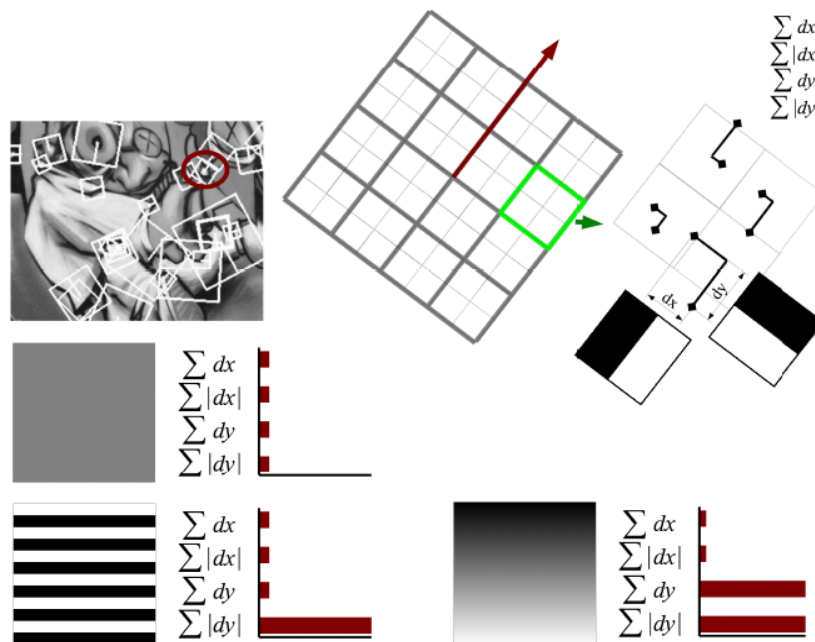
- Consider a neighborhood of size $6s$
 - s being the scale where the keypoint was found
- Evaluate gradients in x and y using the Haar wavelets
 - Responses smoothed using a gaussian



- Plot responses in a 2-dimensional space
- Sum horizontal (d_x) and vertical (d_y) response and determine the orientation
- Quantize in bins of 60°



- Consider 16 regions (same as SIFT)
- For each region:
 - Sum the response for d_x and d_y
 - Sum the modules for d_x and d_y
- Vector normalization
- 4 elements per region
 - Feature size: $4 \times 16 = 64$





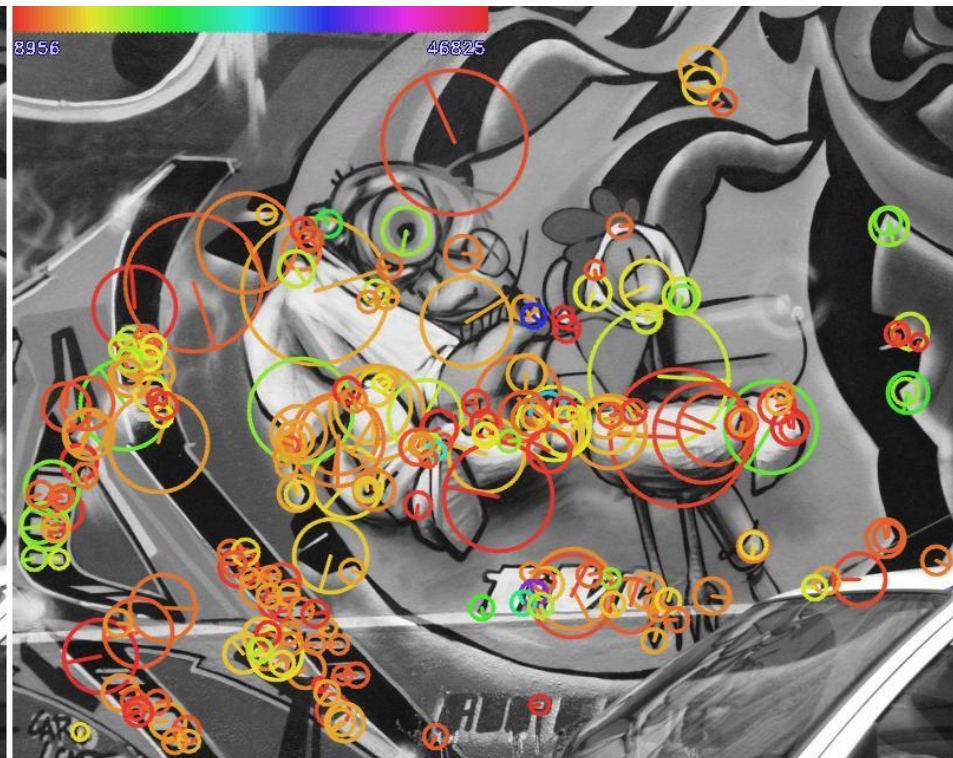
- Faster than SIFT
- Less robust to illumination and viewpoint changes WRT SIFT
- Both are patented



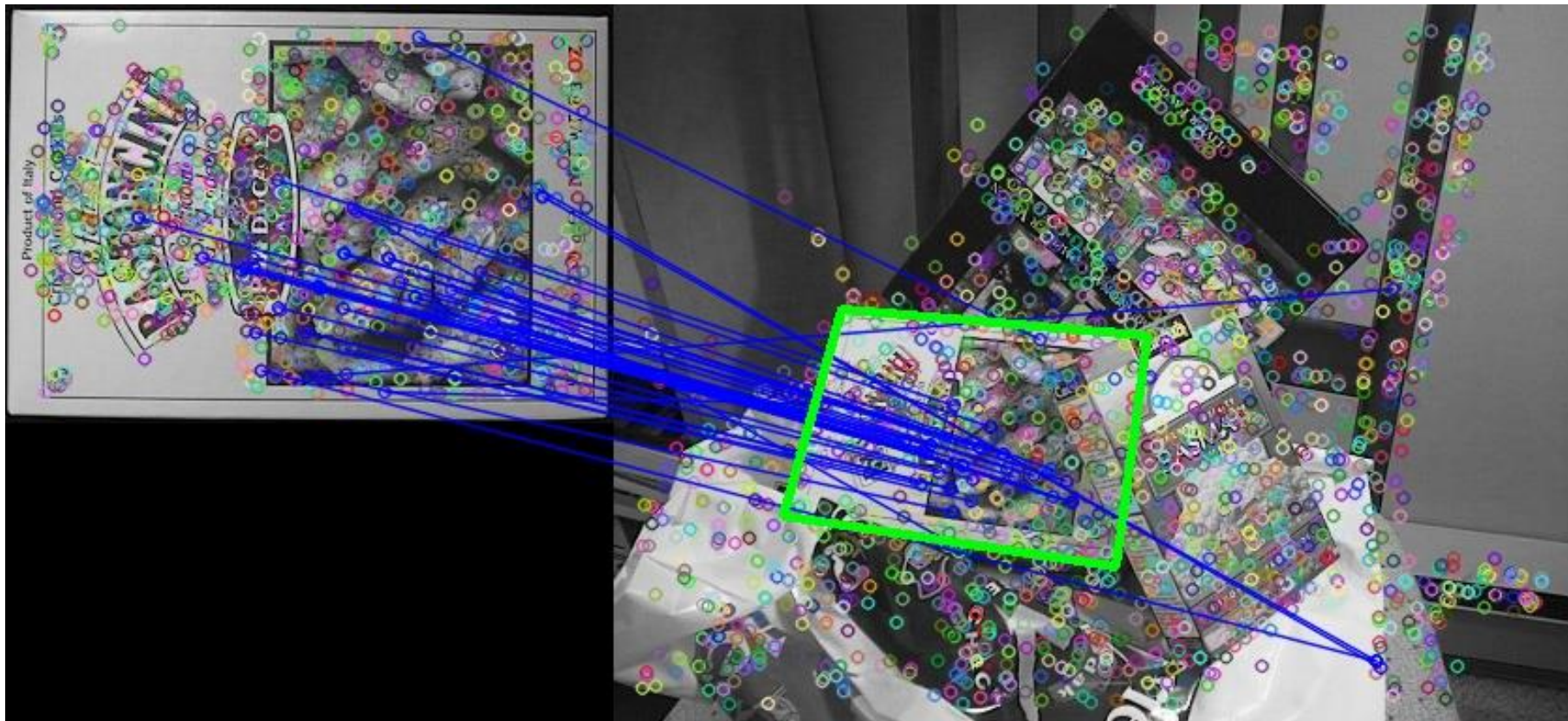
UNIVERSITÀ
DEGLI STUDI
DI PADOVA

SURF – example

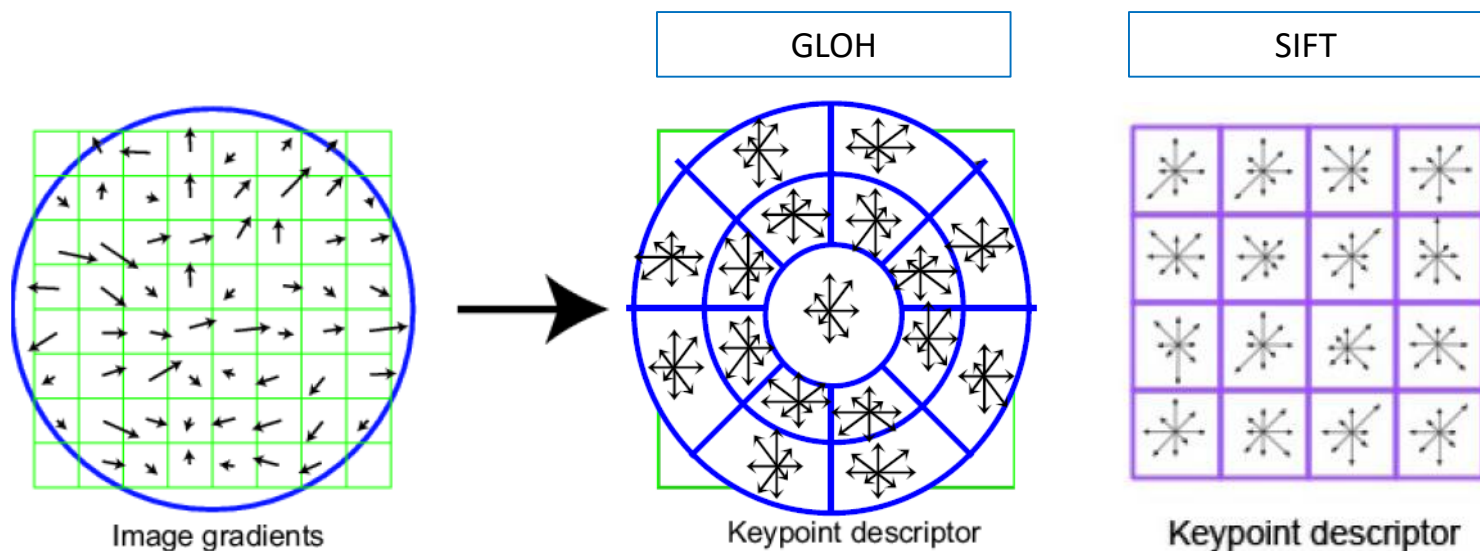
IAS-LAB



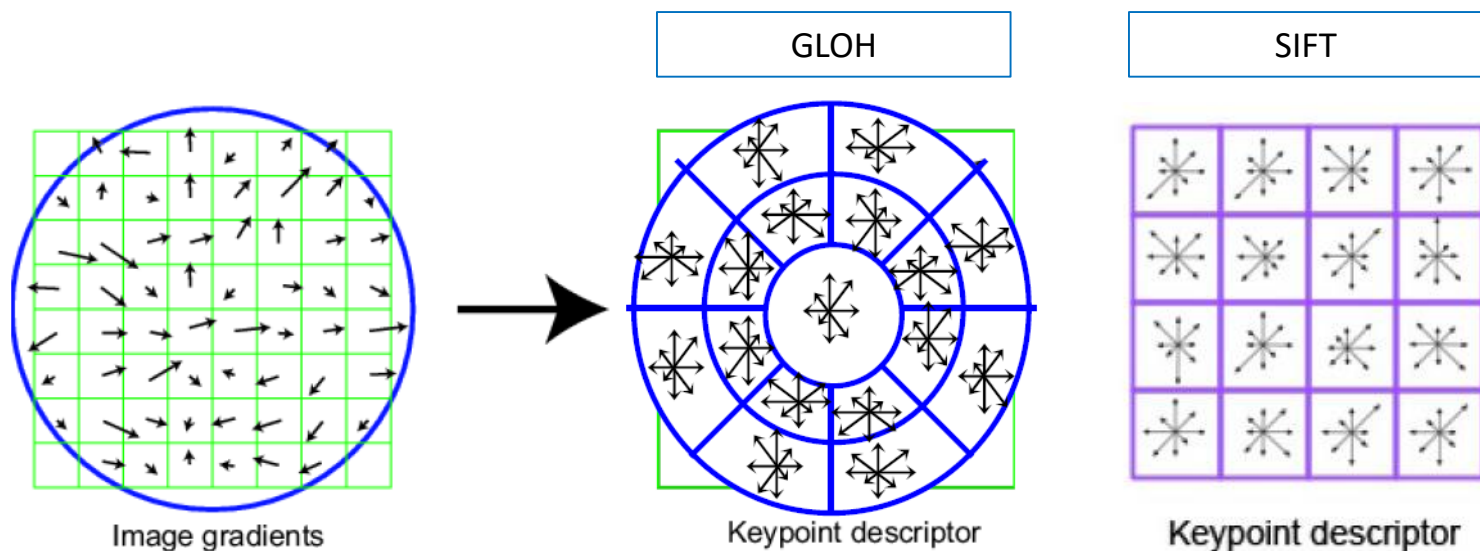
Surf – matching



- Gradient Location-Orientation Histogram
- Obtained (again) modifying the 4th step of the SIFT algorithm
- Computes SIFT descriptor using a log-polar location grid

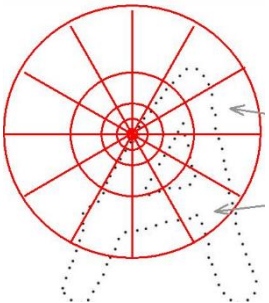
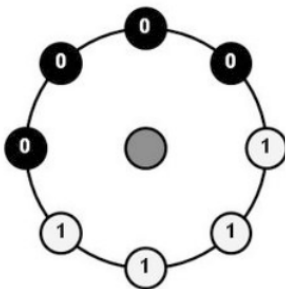
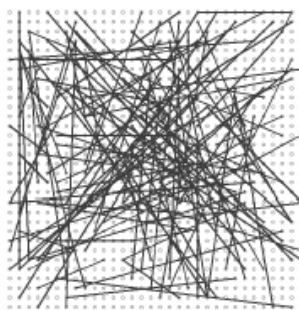
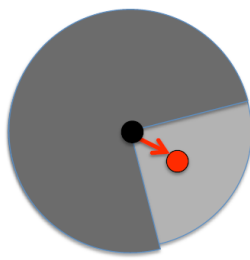


- 3 bins in radial direction (radii: 6, 11, 15)
- 8 bins in angular direction (central bin not subdivided)
- 16 orientations
- Overall: $(1+8+8) \times 16 = 272$ orientation bins
- PCA used for reducing dimensionality to 128

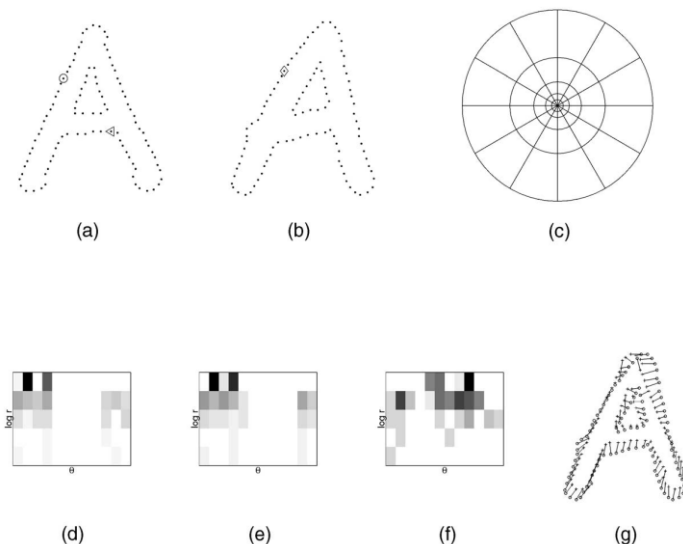
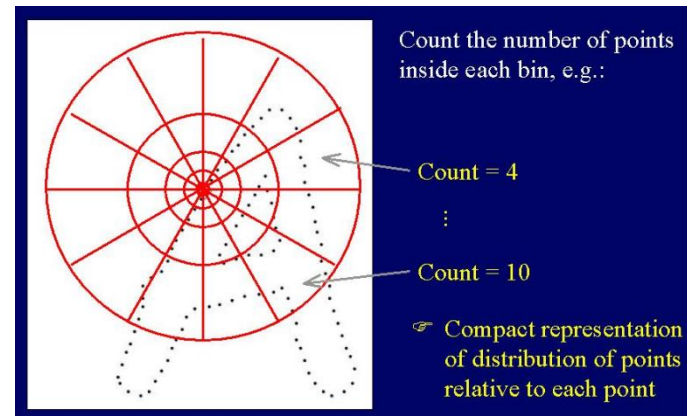


Other approaches



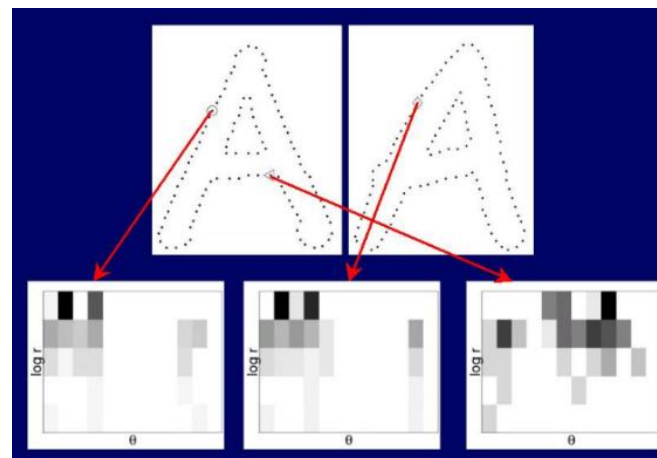
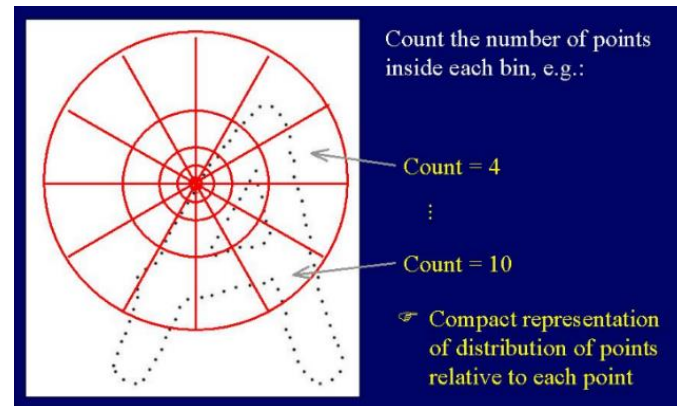
			
<i>Shape Context</i>	<i>LBP</i>	<i>BRIEF</i>	<i>ORB</i>

- **Shape context descriptor**
- 3D histogram of edge point locations and orientations
 - Edges often extracted using Canny
- Location quantized using log-polar coordinate system
 - 5 bins for distance
 - 12 bins for orientation
 - 60 combinations

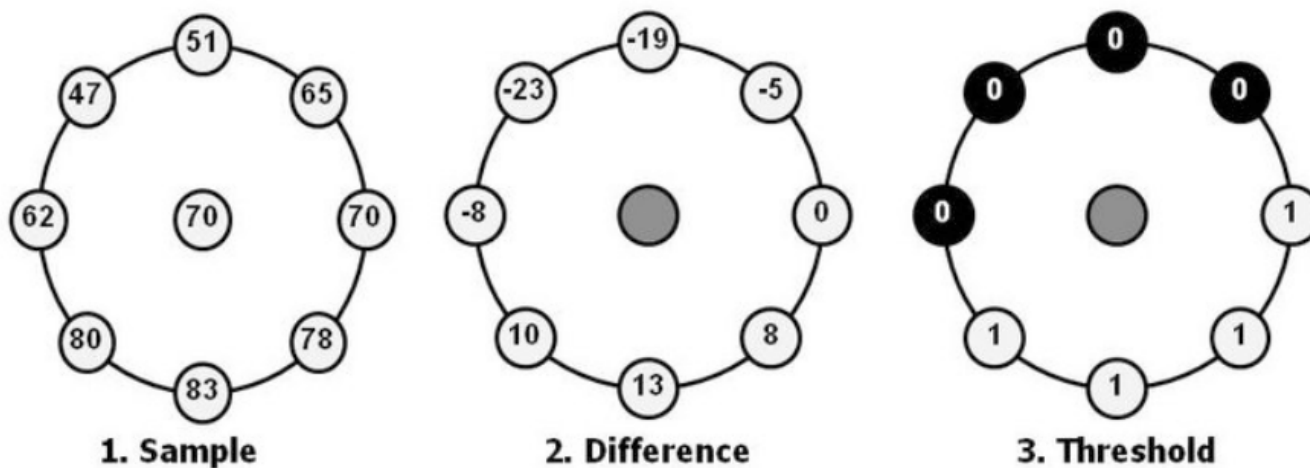


- Accumulate the distribution of edge points in the log-polar bins b_k

$$h_i(k) = \text{count}[q \neq p_i; (q - p_i) \in b_k]$$
- Translation invariance: positions are relative distances
- Scale invariance: obtained by normalizing all radial distances by the mean distance by point pairs
- Robust to deformations, noise, outliers – experimentally demonstrated
- Depends on rotations!
 - Can be added by referring the features WRT edge orientation



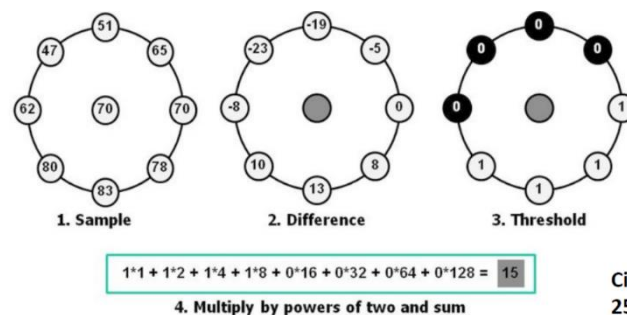
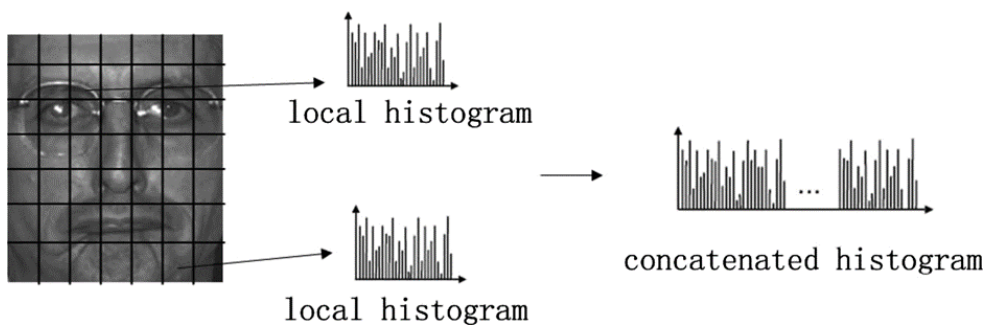
- **LBP descriptor**
- First proposed for texture recognition in 1994
- Compare the central pixel with surrounding samples and build a binary sequence of signs of differences



$$1 \cdot 1 + 1 \cdot 2 + 1 \cdot 4 + 1 \cdot 8 + 0 \cdot 16 + 0 \cdot 32 + 0 \cdot 64 + 0 \cdot 128 = 15$$

- Select a pixel and compare it with its 8 neighbors
- Follow the pixels along a circle: if it is $>$ center, assign 1, 0 otherwise
- Compute the histogram of such values
- Histogram normalization (optional)
- A whole image can be analyzed divided into subwindows (histograms are concatenated)

$$\tau(x, x_i) = \begin{cases} 1 & \text{if } i(x) > i(x_i) \\ 0 & \text{if } i(x) \leq i(x_i) \end{cases} \quad f(x) = \sum_{i=1}^8 2^{i-1} \tau(x, x_i)$$





- Binary Robust Independent Elementary Features
- Based on:

- Gaussian smoothing
- Pairs of pixels compared inside a window

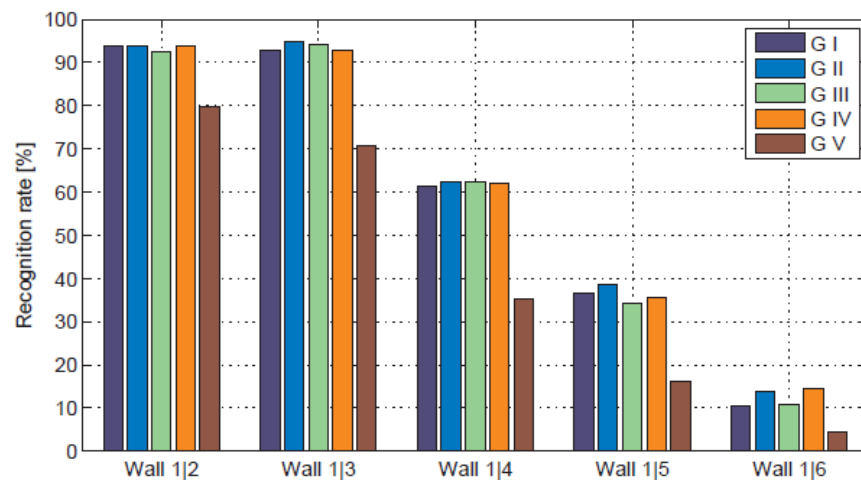
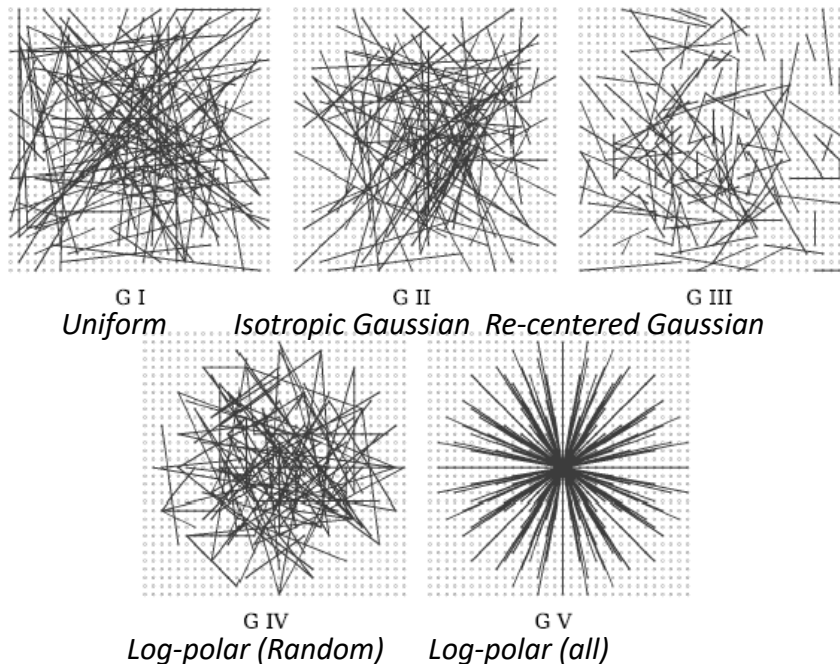
$$\tau(\mathbf{p}; x, y) = \begin{cases} 1 & \text{if } p(x) < p(y) \\ 0 & \text{otherwise} \end{cases}$$

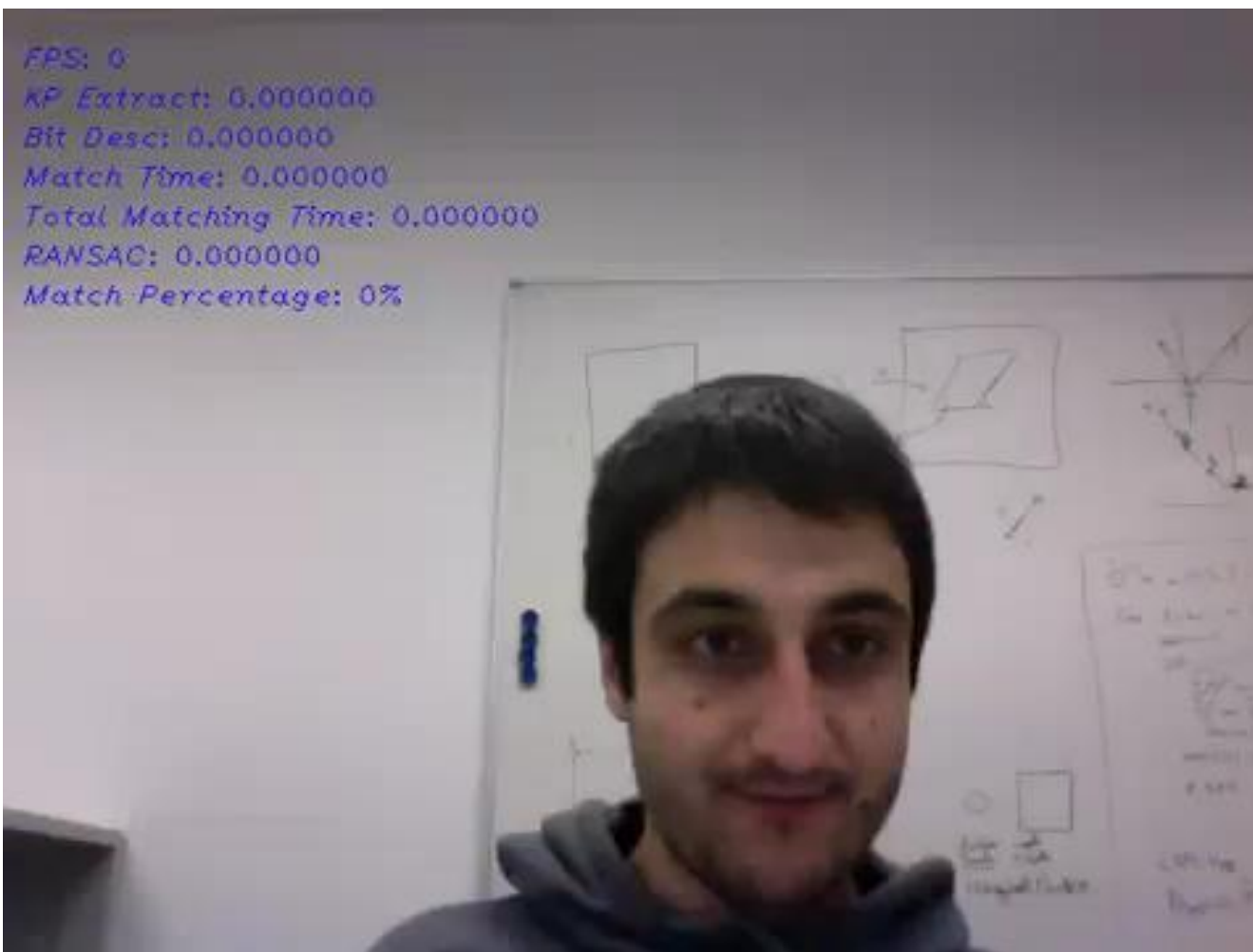
- Build a vector with comparison output

$$f(\mathbf{p}) = \sum_i 2^{i-1} \tau(\mathbf{p}; x, y)$$

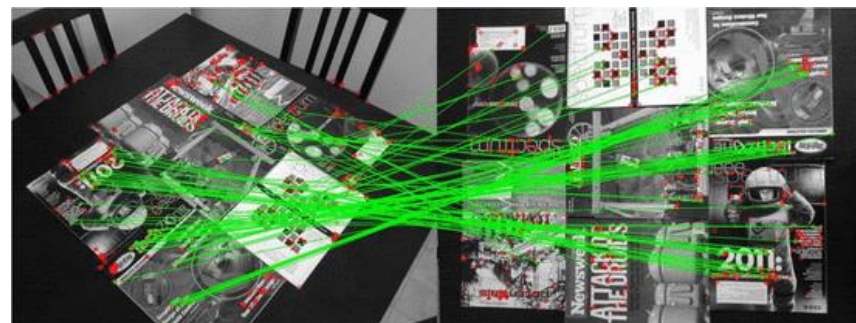
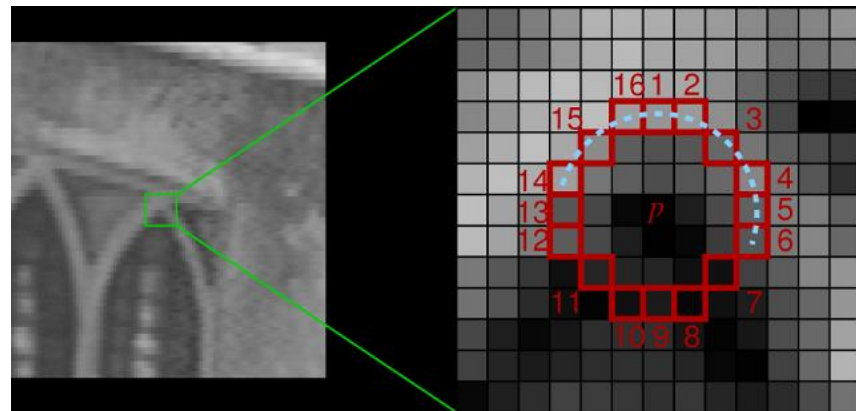
- Vectors compared using the Hamming distance (XOR)
- **Homework: check the documentation***

- Fixed sampling pattern of 128, 256 or 512 pairs
- Random pattern provides the best performance
 - Best solution: isotropic gaussian distribution

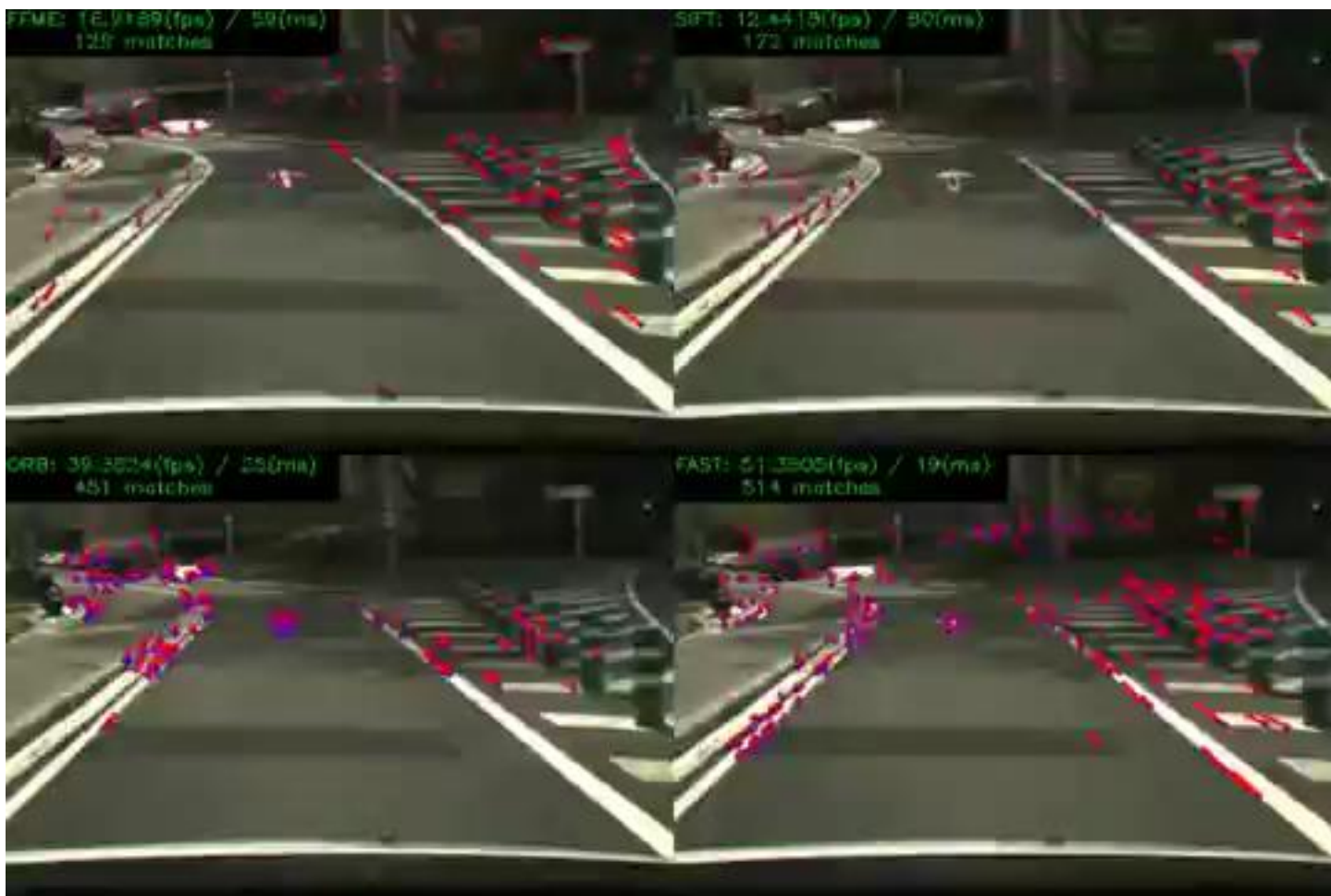




- Oriented FAST and Rotated BRIEF
- FAST corner detector
- Add rotation invariance to BRIEF
- Orientation assignment based on the intensity centroid WRT the central pixel
- **Homework: check the documentation***



- Comparison of SIFT, ORB and FAST





UNIVERSITÀ DEGLI STUDI DI PADOVA

An overview of existing features

Stefano Ghidoni

