Fundamentals of Computer Vision

Unit 7: Feature Description

Jorge Bernal

Index

1

Introduction

Introduction

Definition of feature:

- Piece of information that is useful to solve a given task
- Interesting part of the image

Types of features:

- **Global:** global properties of the whole image
  - Mean grey level, mean colour, main colours, histogram
- **Local:** properties of a part of the image with their own entity
  - Points, edges, regions

Introduction

- How can we find local features?
  - Feature detection/extraction algorithms

- Detection/Extraction: locate the position of the feature
- **Description: measures that are taken from the detected feature that allow us to distinguish it or compare with others**

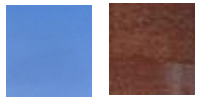
Introduction

- Why do we use features?
  - They have been used with success in several disciplines and applications:
    - Edge detection associated to roads in aerial images
    - Quality control
    - Polyp Detection
  - Interest points play a key role for certain Applications:
    - Tracking
    - 3D reconstruction
  - They are a first step to achive a robust image representation:
    - Object recognition
    - Scene classification
    - Texture analysis
    - Image search

Introduction

- Why do we use features?
  - They have been used with success in several disciplines and applications:
    - Edge detection associated to roads in aerial images
    - Quality control
    - Polyp Detection
  - Interest points play a key role for certain Applications:
    - Tracking
    - 3D reconstruction
  - They are a first step to achive a robust image representation:
    - Object recognition
    - Scene classification
    - Texture analysis
    - Image search

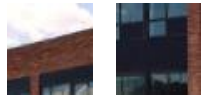
Introduction: why do we need feature descriptors



A

B

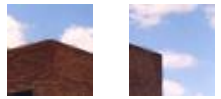
A and B are flat **surfaces** and they are spread over a lot of area. It is **difficult to find the exact** location of these patches.



C

D

C and D are **edges**. You can find an approximate location, but exact location is still difficult. An edge is therefore better feature compared to flat area, but **not good enough.**

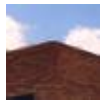


E

F

E and F are **corners**. They can be easily found. Because wherever you move this patch, it will look different, then it can be considered as **good features**.

Introduction: why do we need feature descriptors

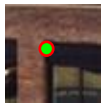


(12,43,54)

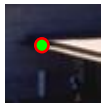
(0,73,14)

(12,43,0)

(12,43,54)



(1,0,54)

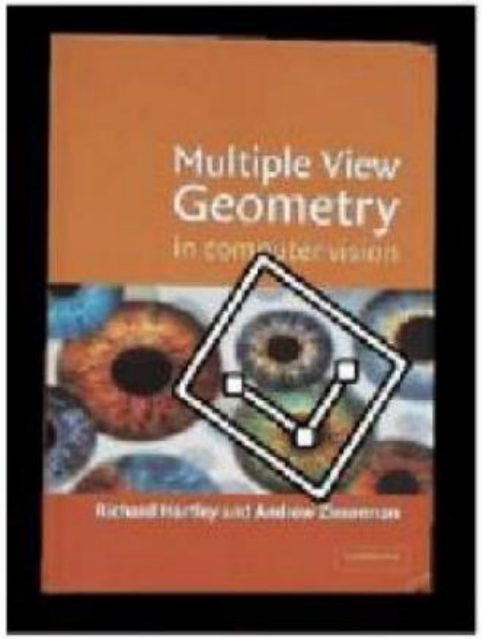


(1,3,5)

(12,23,74)

Introduction

- Desirable properties:
  - Shift invariant
  - Scale invariant
  - Rotation invariant
  - Contrast invariant



Introduction

- General approach:
  - Determine the scale where a feature is most stable
  - Determine the orientation of the small patch surrounding this feature point at this scale
  - Characterize a small patch tilted in this orientation and scale with a descriptor
  - The descriptor should be invariant to intensity shift or scaling, and small position shift

2

Shape Descriptors

Shape Descriptors

- Shape is, indeed, an important visual cue and it is one of the most used to describe image content.
- But it also has its complications because we can not forget that when we project 3-D objects into a 2-D image we are losing one whole dimension of information, so the 2-D representation of an object gives only a partial representation of it.
- It is highly affected by noise, defects, occlusion or deformation, which can make harder the process of object recognition.

Shape Descriptors

- Assumption: the object whose shape we want to describe has been segmented (or, at least, somehow separated) from the image so we have a binary image patch which contains the object.
- Two groups:
  - Contour-based: we extract features only from the contour of the shape.
  - Region-based: we use the whole shape region.
- Each group is also subdivided into structural approaches (representing the shape by segments/sections) or global approaches (represent the shape as a whole).

Shape Descriptors

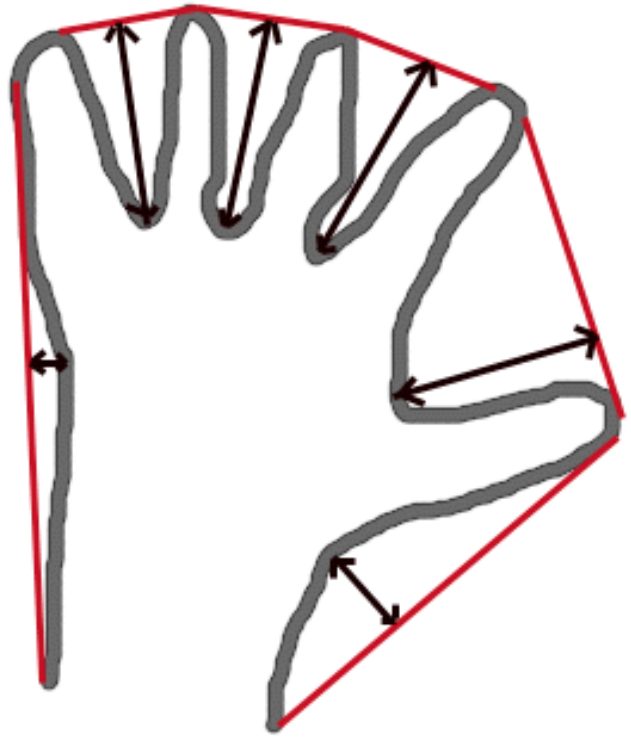
- Contour-based Shape Descriptors (Global):
  - Simple descriptors:
    - Area
    - Eccentricity
    - Axis orientation
    - Radius of the principal axis

Shape Descriptors

- Contour-based Shape Descriptors (Global):
  - Convex Hull:
    - One region is considered as convex only if by taking any two points of it the segment that binds them is inside the region. Convex Hull is defined as the minimal convex region.
    - Before dividing the contour in segments, it is smoothed to avoid some non-desired effects such as hysterical responses to noise. At last the whole shape will be represented as a chain of concavities.

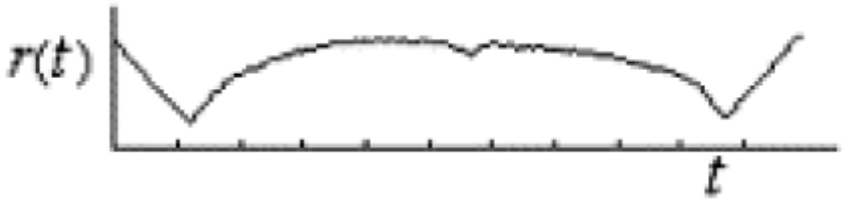
Shape Descriptors

- Contour-based Shape Descriptors (Global):
  - Convex Hull:



Shape Descriptors

- Contour-based Shape Descriptors (Global):
  - Shape signature:
    - This method represents the shape of an object by means of an uni-dimensional function which is extracted from the points belonging to the contour of the shape.
    - There are several possible Shape Signatures such as the centroidal profile, shape radii, complex coordinates, distance to the centroid, tangent or accumulative angle, curvature, arc length, etc.



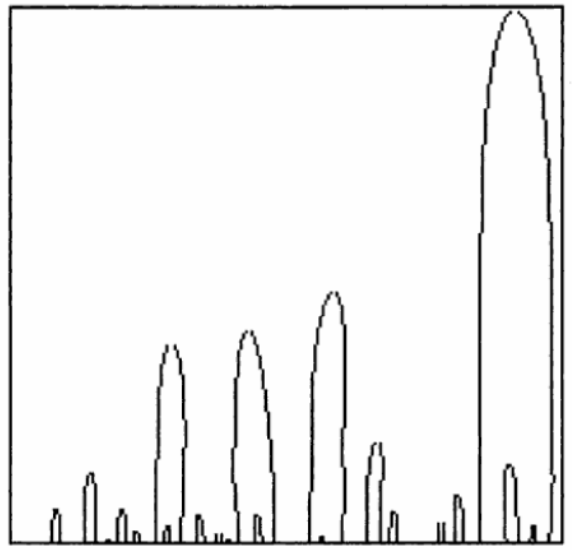
Shape Descriptors

- Contour-based Shape Descriptors (Global):
  - Shape signature:
    - Shape Signatures are usually normalized for translation and scale invariance.
    - In order to compensate for orientation changes, shift matching is needed to find the best matching between two shapes. Most of the signature matching is normalized to shift matching in 1-D space, however, some signature matching requires shift matching in 2-D space, such as the matching of centroidal profiles.

    ☹ Shape Signatures are sensitive to noise, and slight changes in the boundary can cause large errors in matching.

Shape Descriptors

- Contour-based Shape Descriptors (Global):
  - Curvature scale space:
    - Several arch-shaped contours representing the inflection points of the shape
    - The maxima of these contours are used to represent a shape

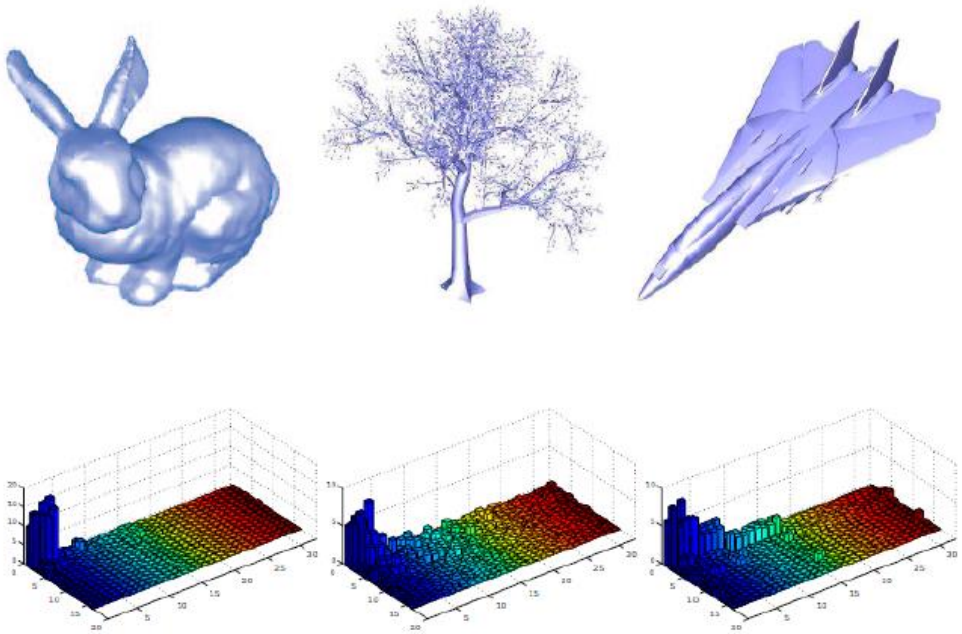


Shape Descriptors

- Contour-based Shape Descriptors (Global):
  - Wavelet Transform:
    - Hierarchical descriptor of planar curves which is able to decompose a curve in several scales.
    - Higher scale components convey more general information and those with lower scale have more local and detailed information.
    - The descriptor also has some interesting properties like multi-resolution representation, is invariant and it keeps the unicity, stability and spatial location properties

Shape Descriptors

- Contour-based Shape Descriptors (Global):
  - Wavelet Transform:

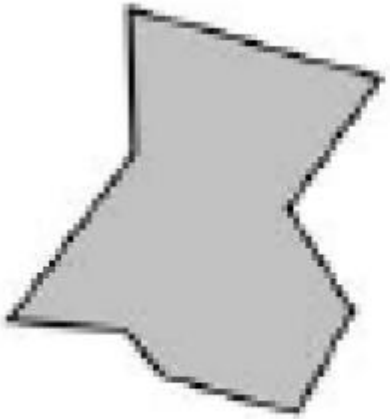


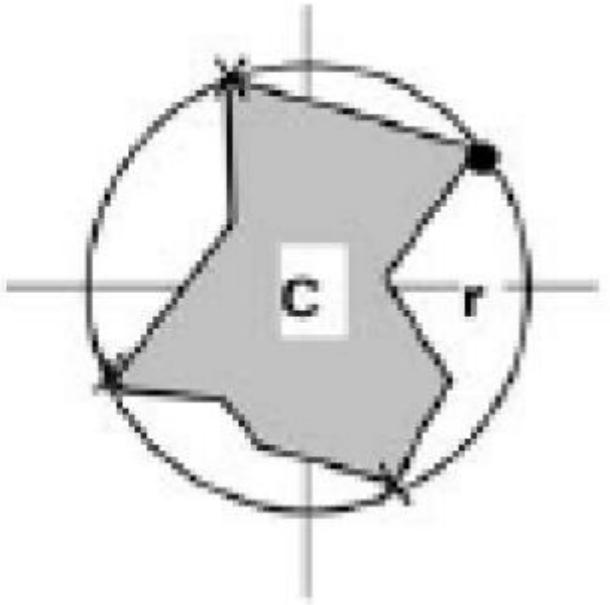
Shape Descriptors

- Contour-based Shape Descriptors (Global):
  - Minimum Boundary Circle-based:
    - Features are extracted from the minimum circle that surrounds the object (that is, the circle that touches its further borders)
    - We can obtain the following: center coordinates, radius, minimum circle crossing points, angle sequence of these crossing points, vertex angle sequence and angle sequence starting point.
    - Rotation, translation and scale invariant which makes it a good candidate

Shape Descriptors

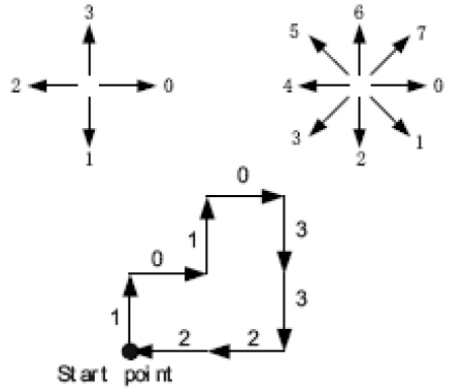
- Contour-based Shape Descriptors (Global):
  - Minimum Boundary Circle-based:





Shape Descriptors

- Contour-based Shape Descriptors (Structural):
  - Chain code:
    - Codifying lines into a determinate code.
    - The nodes that surround a certain point (or central node) are enumerated counter-clockwise in ascending order from inside to outside.
    - A chain will consist of an ordered link sequence. The inverse and the length of the chain can also be calculated.

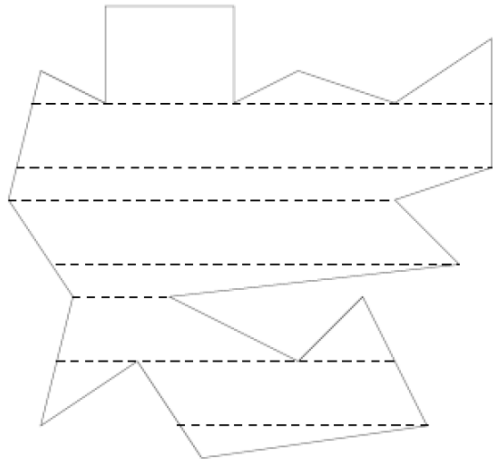


Shape Descriptors

- Contour-based Shape Descriptors (Structural):
  - Chain code:
    - ☺ Translation invariance, potential scale and rotation invariance
    - ☹ Depends on the starting point

Shape Descriptors

- Contour-based Shape Descriptors (Structural):
  - Polygon Decomposition:
    - The contour of the shape is divided in approximated segments or polygons, using as primitives the vertex of these polygons.
    - For each primitive the feature extracted consists of a chain of four elements (inner angle, distance to the next vertex and x and y coordinates).



Shape Descriptors

- Contour-based Shape Descriptors (Structural):
  - Polygon Decomposition:
    - ☹Not rotation, translation or scale invariant
    - Similarity can be calculated using Edit Distance

Shape Descriptors

- Contour-based Shape Descriptors (Structural):
  - Shape context:
    - It is intended to be a way of describing shapes oriented to measure shape similarity and recover of point correspondences
    - To get this, and by using polar coordinates, vectors from the chosen point to every point of the frontier/contour are calculated.
    - The length and orientation of these vectors are quantified so a histogram can be created and then used to represent this point. The histogram of each point is flattened and concatenated to be part of the context of the certain shape

Shape Descriptors

- Contour-based Shape Descriptors (Structural):
  - Shape context:

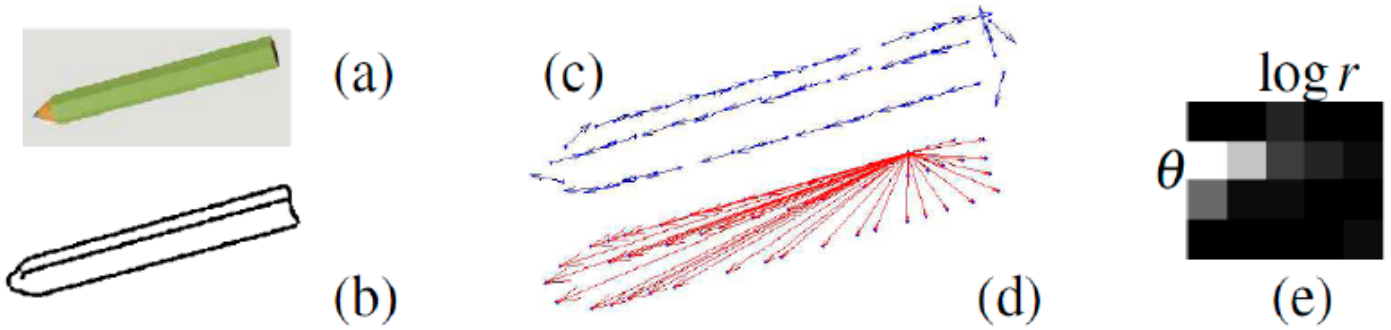


Figure 4.10: Example for deriving the shape context descriptor for the image of a pencil. (a) Input image of the pencil. (b) Contour of the pencil derived using the Canny operator. (c) Sampled points of the contour. (d) All vectors from one point to all the other sample points. (e) Histogram with four angle and five log-radius bins comprising the vectors depicted in (d)[61].

Shape Descriptors

- Contour-based Shape Descriptors (Structural):
  - Shape context:
    - ☺ Potential translation, scale and rotation invariance
    - ☹ Affected by noise

Shape Descriptors

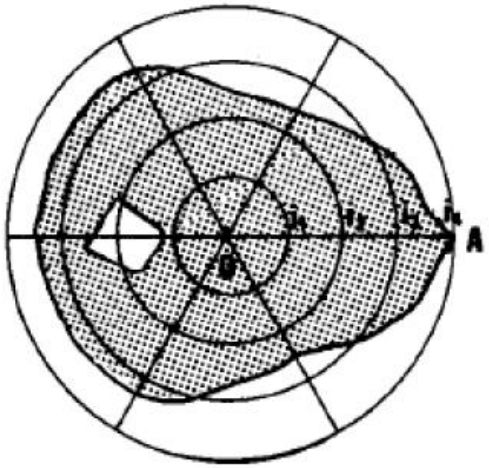
- Contour-based Shape Descriptors (Structural):
  - Chamfer:
    - Find a model within an image. First, we have to find weak and noisy edges and then remove them from the image.
    - Next, transformation distance of the remaining pixels has to be calculated.
    - The value of this pixel according to this transformation is proportional to the distance of this pixel to the nearest edge pixel.
    - This model is then shifted around the already transformed image and
    - in each shift position the sum of distances up to the model is calculated. The shift position with less accumulative sum is taken as the best correspondence to the model.

Shape Descriptors

- Region-based Shape Descriptors (Global):
  - Zernike Moments:
    - Zernike moments are rotation invariant and robust to noise, so if we use a descriptor based in these moments, it will inherit this characteristics.
    - The input image has to be binarized
    - ☺ Rotation invariant, robust to noise
    - ☹ Image coordinates have to be transformed, not effiient

Shape Descriptors

- Region-based Shape Descriptors (Global):
  - Shape matrix:
    - A shape is transformed into a matrix by polar quantization of the shape
    - Translation, rotation and scale invariant



(a)

	0	1	2	3	4
0	1	1	1	1	1
1	1	1	1	0	0
2	1	1	1	1	0
3	1	1	0	1	0
4	1	1	1	1	0
5	1	1	1	0	0

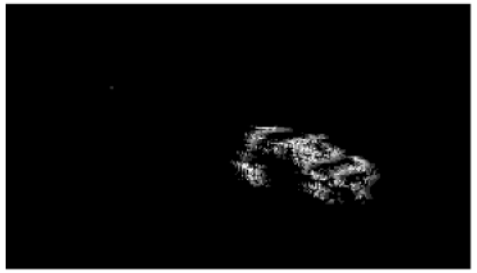
(b)

Shape Descriptors

- Region-based Shape Descriptors (Global):
  - Grid-based descriptor:
    - Grid of cells is superimposed over a shape and this grid is scanned from side to side, giving a bitmap as a result.
    - The cells that cover the shape of the object are set to 1 and the rest, to 0. The difference between the two shapes can be calculated as the number of cells in the grids which are covered by one shape and not the other and hence the sum of 1's in the result of the exclusive-or of the two binary numbers.
    - Not rotation or scale invariant (needs normalization)

Shape Descriptors

- Region-based Shape Descriptors (Global):
  - Grid-based descriptor:



The Sustained temporal change.

The grid with cellsize $\lambda = 4$.

The grid with cellsize $\lambda = 8$.

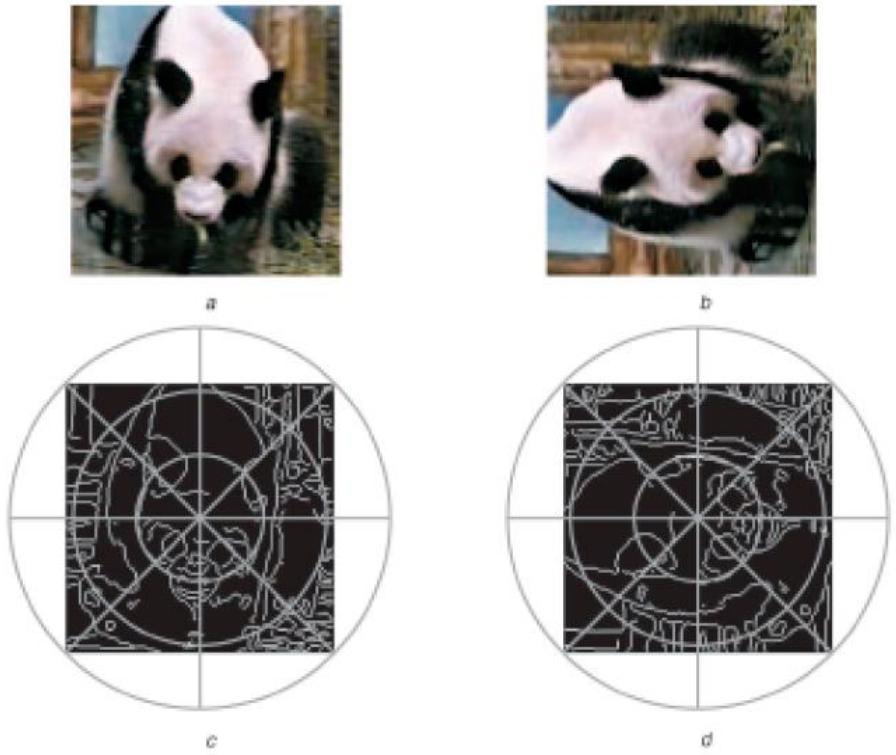
The grid with cellsize $\lambda = 16$.

Shape Descriptors

- Region-based Shape Descriptors (Global):
  - Angular Radial Partitioning:
    - Transforms the image data into a new structure that supports measurement of the similarity between images in an effective, easy and efficient manner with emphasis on capturing scale and rotation invariant properties.
    - Needs of the calculation of the normalized edge image
    - This image is partitioned into M (number of radial partitions) and N (number of angular partitions)

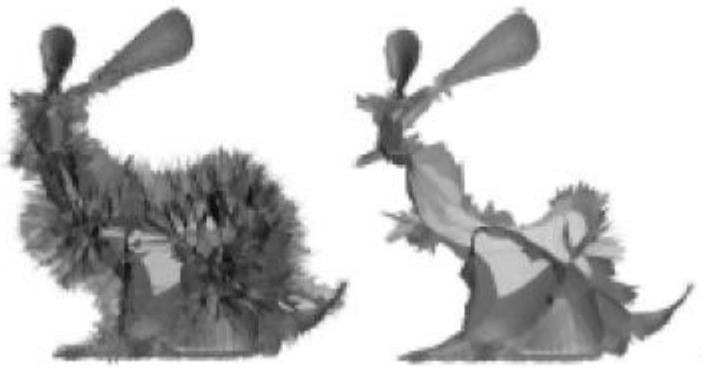
Shape Descriptors

- Region-based Shape Descriptors (Global):
  - Angular Radial Partitioning:



Shape Descriptors

- Region-based Shape Descriptors (Structural):
  - Skeleton:
    - Connected contour of medial lines along the different parts of an object.
    - One of the ways to calculate the Skeleton of an image is by using the Medial Axis Transform (MAT), where the Medial Axis is the geometric place where the centers of the maximum discs that fit into the shape coincide.
    - The Skeleton can be discomposed in segments and represented as a graph.



Shape Descriptors

Descriptor	Type	Rotation invariant	Translation invariant	Scale invariant
Shape signature	Contour-based Global	Not direct	After normalization	After normalization
Convex Hull	Contour-based Global	Yes	Yes	Yes
Wavelet Transform	Contour-based Global	No	No	Yes
Minimum Boundary Circle	Contour-based Global	Yes	Yes	Yes
Chain Code	Contour-based Structural	Possible	No	Yes
Polygons Decomposition	Contour-based Structural	No	No	No
Shape Context	Contour-based Structural	Possible	Possible	Possible
Chamfer	Contour-based Structural	No	Yes	Yes

Shape Descriptors

Descriptor	Type	Rotation invariant	Translation invariant	Scale invariant
Zernike Moments	Region-based Global	Yes	No	No
Shape Matrix	Region-based Global	Yes	Yes	Yes
ARP	Region-based Global	No	No	Yes
Grid-based	Region-based Global	Possible	No	No
Skeleton	Region-based Structural	Yes	No	Yes

2

Color Descriptors

Color Descriptors

- Color is the visual perceptual property corresponding in humans to the categories called red, yellow, blue and others.
- Color derives from the spectrum of light (distribution of light energy versus wavelength) interacting in the eye with the spectral sensitivities of the light receptors

Color Descriptors

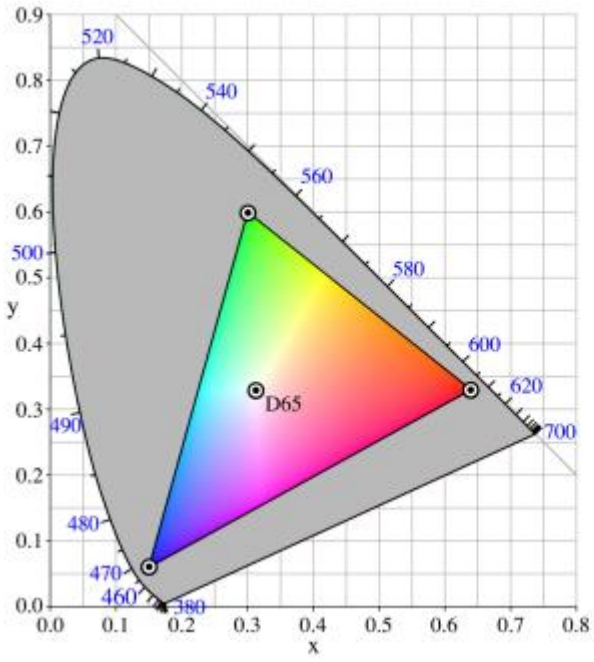
- Color spaces: RGB
  - Additive color model in which red, green, and blue lights are added together in various ways to reproduce a broad array of colors.
  - The name of the model comes from the initials of the three additive primary colors, red, green, and blue.

Color Descriptors

- Color spaces: RGB
  - The main purpose of the RGB color model is for the sensing, representation, and display of images in electronic systems, such as televisions and computers,
  - It is based in human perception of colors.
  - ☹ It is device-dependent

Color Descriptors

- Color spaces: RGB

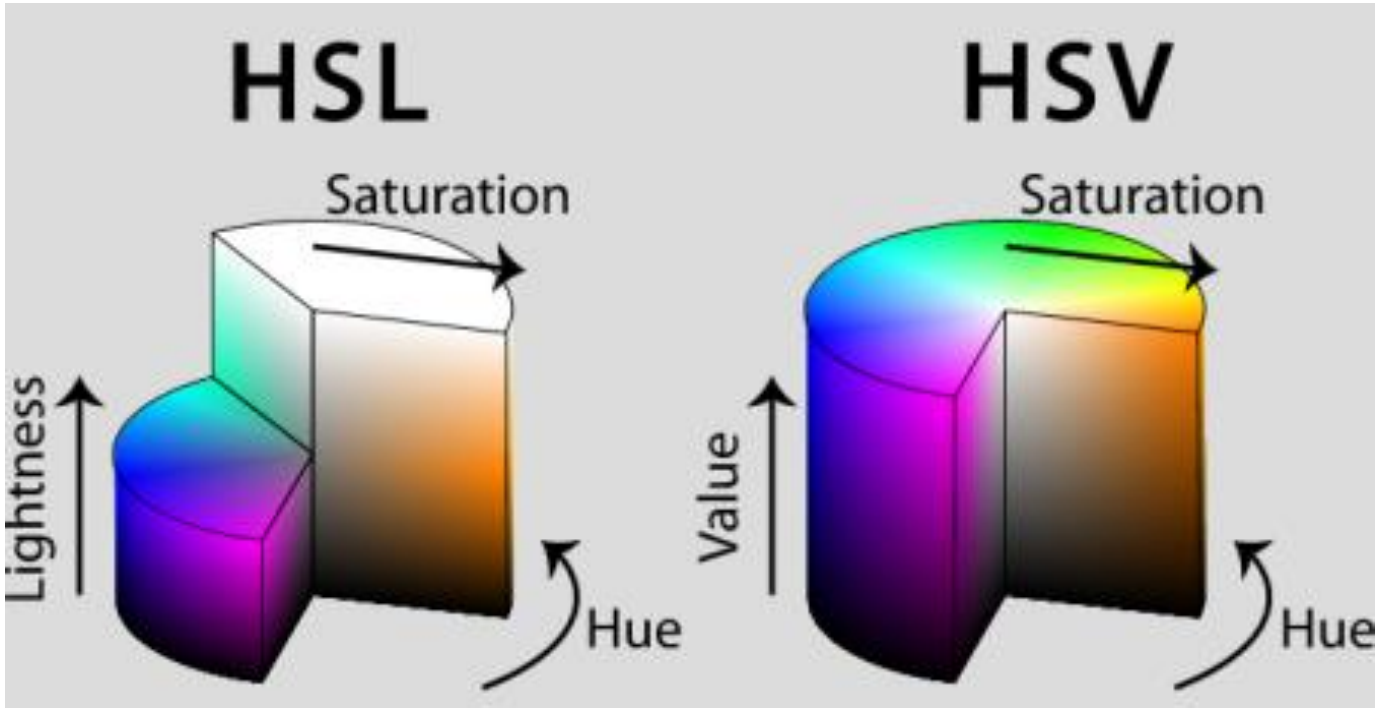


Color Descriptors

- Color spaces: HSV, HSL y HSI
  - Cylindrical-coordinate representation of points in an RGB color model, which rearranges the geometry of RGB in an attempt to be more perceptually relevant than the cartesian representation

Color Descriptors

- Color spaces: HSV, HSL y HSI

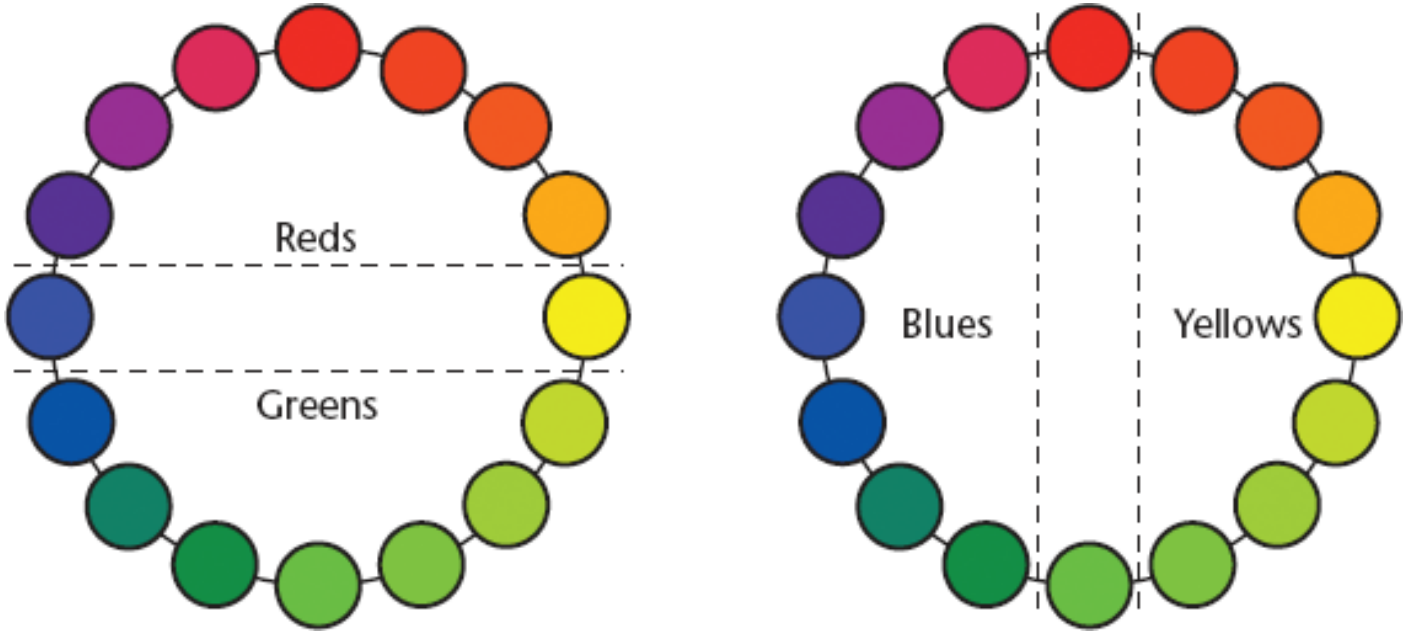


Color Descriptors

- Color spaces: Lab Colour Space
  - Colour-opponent Space with dimension L for lightness and *a* and *b* for the color-opponent dimensions, based on non-linearly compressed CIE XYZ Color Space coordinates.
  - It is designed to approximate the way humans see and understand colors: It aspires to perceptual uniformity, and its L component closely matches human perception of lightness.

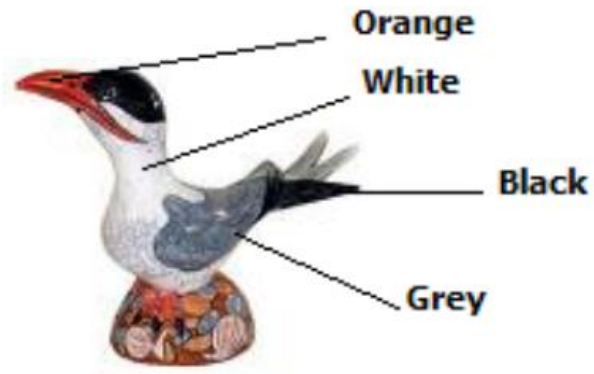
Color Descriptors

- Color spaces: Lab Colour Space



Color Descriptors

- Color descriptor: Dominant Color
  - This method consists of finding out the main color in the area of the patch.
  - This descriptor could be used to differentiate between large areas in the image, but it is very noise sensitive



Color Descriptors

- Color descriptor: Mean Grey Value
  - This method consists of describing a region according to its mean grey value.
  - It is not used as a primary descriptor, but it is useful to discard some regions.

Color Descriptors

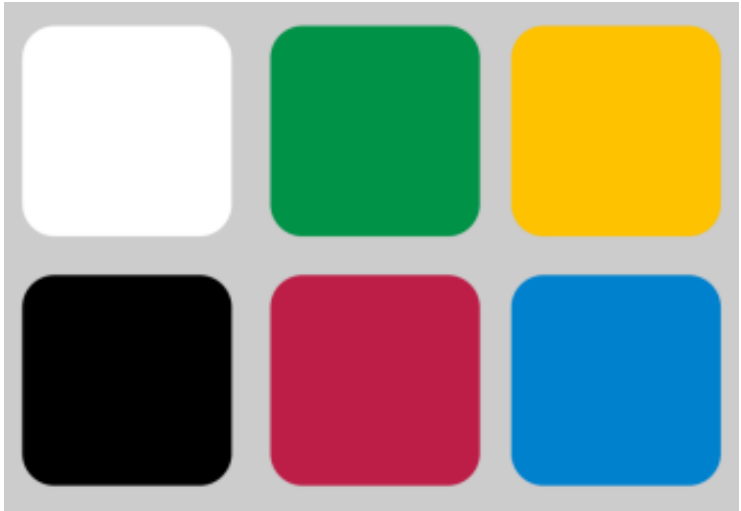
- Color descriptor: MPEG-7 Color Descriptor
  - *Scalable color descriptor:*
    - Generic descriptor, part of MPEG-7, that consists of a Color Space, color quantification and histogram descriptors.
    - It lets us specify color histograms with a variable number of bins and non-uniform quantification of several color spaces.
    - 256-bins histogram providing 16 different values for H, and 4 different values for S and V.
      - Normally, H is defined in the range [0, 360], while S and V are defined in [0, 1]

Color Descriptors

- Color descriptor: MPEG-7 Color Descriptor
  - *Color structure descriptor:*
    - Expresses local color structure in an image by using an 8×8 structural element.
    - It counts the number of times that a color is contained inside the structural element while this element scans the image.
    - A histogram of color structures can be defined in the following way:
      - the value of each bin represents the number of structural elements in the image that contains one or more pixels with a certain color.

Color Descriptors

- Color descriptor: Opponent Color SIFT
    - Extension of SIFT in Opponent Color Space
    - Three different channels:



Color Descriptors

- Color descriptor: Normalized RGB Histogram
  - This method increases the global contrast of some images [104], especially when usable image data are represented by near contrast values.
  - By using this adjustment, intensity values can be scattered along the histogram and it let us gain a major local contrast without having incidence in global contrast

Color Descriptors

- Color descriptor: Color Constant Color Indexing
    - Identifies an object by comparing its colors to the colors of each object stored in a database (taking also into account the total area covered by each color).
    - The different areas associated to each color are calculated and the histograms of both images are compared (the comparison is between the original image and the ones from the database).

Color Descriptors

- Color descriptor: Color Based Object Recognition
    - Similar to CCCI but in this case, this method also tries to select an appropriate color model and corrects reflectance effects by using white or colored illuminance.

3

Texture Descriptors

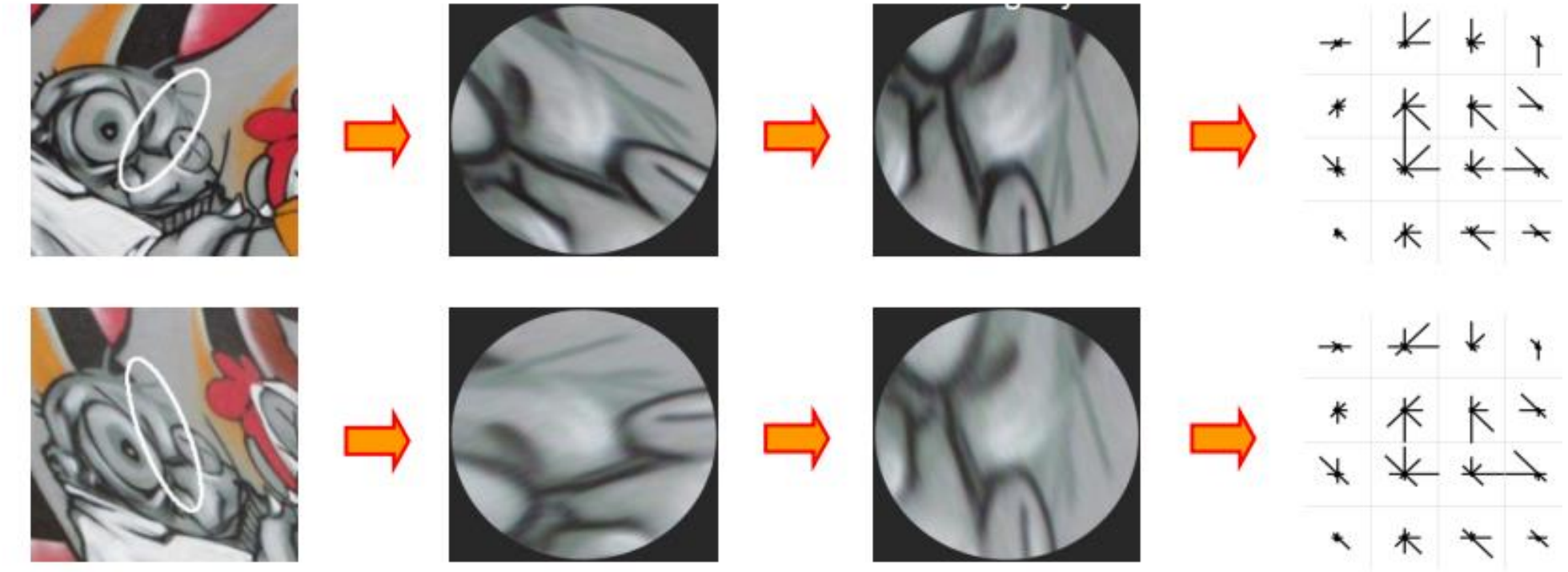
Texture Descriptors

- Texture:
  - used as a synonym for surface structure and it has been described by five different properties in the psychology of perception: coarseness, contrast, directionality, line-likeness and roughness.
  - In 3-D computer graphics, a texture is a digital image applied to the surface of a three-dimensional model by texture mapping to give the model a more realistic appearance.
  - In image processing, every digital image composed of repeated elements is called a texture

Texture Descriptors

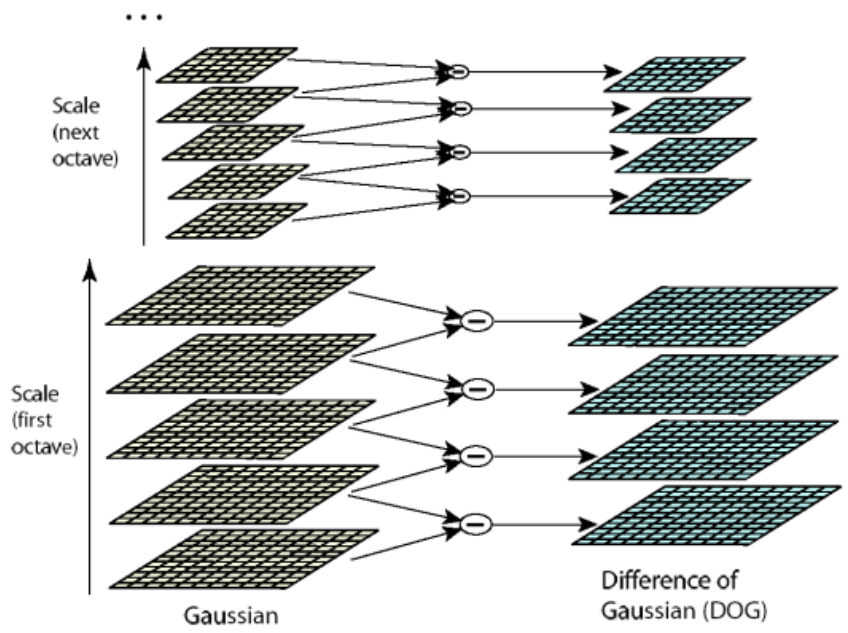
- SIFT (Scale Invariant Feature Transform):
  - It is a technique for detecting salient, stable feature points in an image.
  - For every such point, it also provides a set of “features” that “characterize/describe” a small image region around the point. These features are invariant to rotation and scale.
  - They are particularly useful to detect salient, stable points in two or more images and determine correspondences between them.
  - SIFT features must not change with:
    - Object position/ pose
    - Scale
    - Illumination
    - Image artifacts

Texture Descriptors



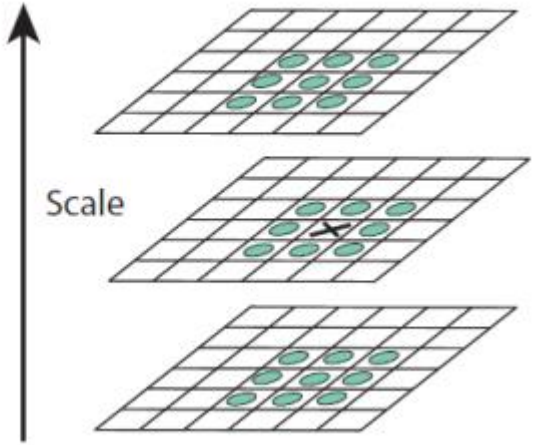
Texture Descriptors

- SIFT algorithm:
  - 1) Approximate keypoint location
    - Look for intensity changes using Difference of Gaussians at two nearby scales



Texture Descriptors

- SIFT algorithm:
  - 1) Approximate keypoint location
    - Keypoints are maxima or minima in the “scale-space pyramid”



Texture Descriptors

- SIFT algorithm:
  - 2) Refining keypoint location
    - Keypoint location and scale is discrete: interpolation can be used for greater accuracy
    - We express the DoG function in a small 3D neighborhood around a keypoint (x_i ,y_i ,σ_i) by a second-order Taylor-series
    - By this, weak extrema or low contrast points are discarded
    - Keypoints on edges should also be discarded

Texture Descriptors

- SIFT algorithm:
  - 2) Refining keypoint location
    - Hessian matrix is used, and eigenvalues are computed
      - Edge: high maximal curvature, low minimal curvature
      - Corner: high maximal and minimal curvature

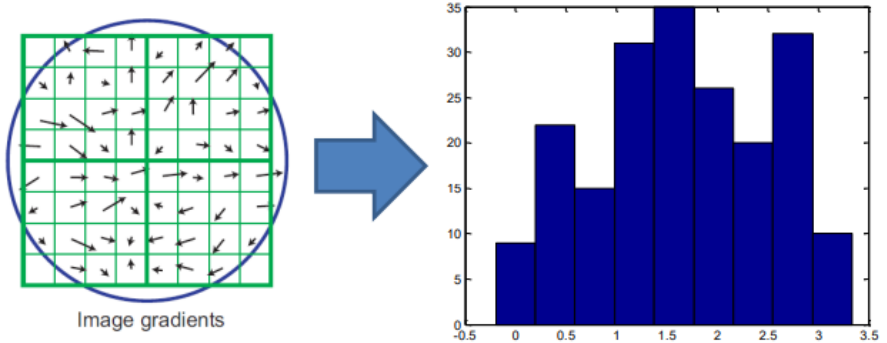
Texture Descriptors

- SIFT algorithm:
  - 3) Assigning orientations
    - Compute the gradient magnitudes and orientations in a small window around the keypoint at its appropriate scale

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

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

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



Histogram of gradient orientation – the bin-counts are weighted by gradient magnitudes and a Gaussian weighting function. Usually, 36 bins are chosen for the orientation.

Texture Descriptors

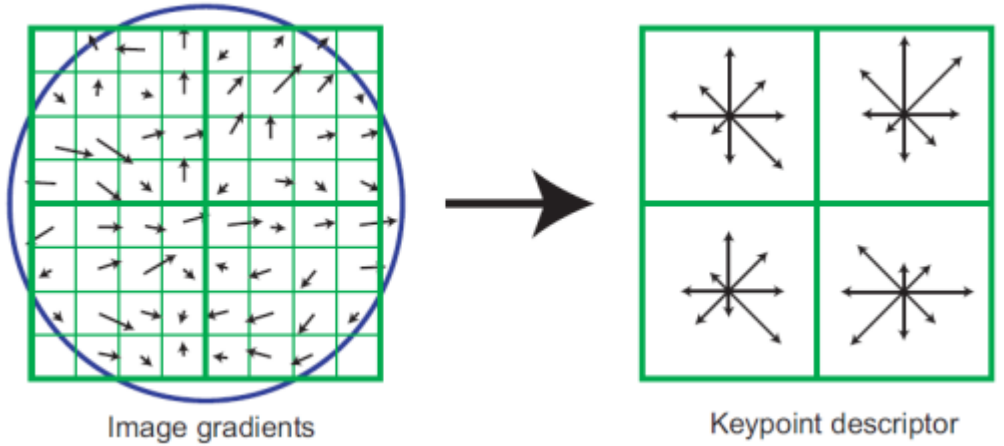
- SIFT algorithm:
  - 3) Assigning orientations
    - Assign the dominant orientation as the orientation of the keypoint
    - If several orientations are higher than 80% of the peak, a separate descriptor for each orientation is used

Texture Descriptors

- SIFT algorithm:
  - 4) Descriptor for each keypoint
    - Consider a small region around the keypoint. Divide it into n x n cells (usually n = 2). Each cell is of size 4 x 4.
    - Build a gradient orientation histogram in each cell. Each histogram entry is weighted by the gradient magnitude and a Gaussian weighting function with $\sigma = 0.5$ times window width.
    - Sort each gradient orientation histogram bearing in mind the dominant orientation of the keypoint (assigned previously).

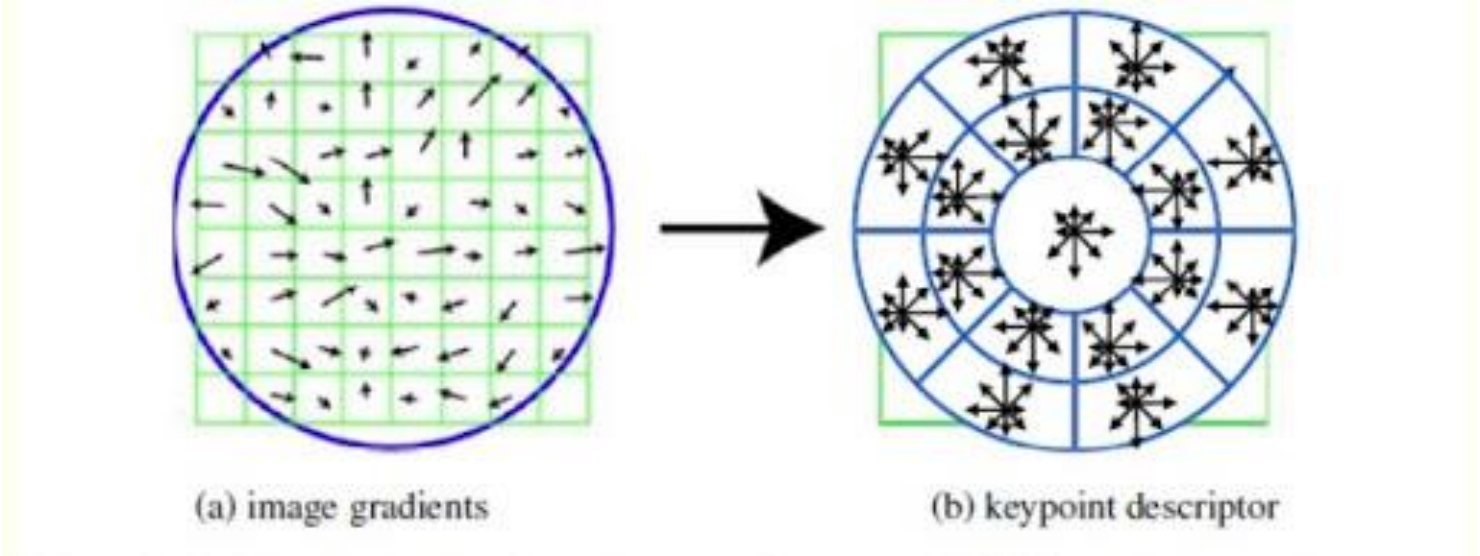
Texture Descriptors

- SIFT algorithm:
  - 4) Descriptor for each keypoint
    - We have now a descriptor of size rn^2
    - Typical values: r=8, n=4 (size 128)
    - Invariant to rotations due to sorting
    - How to get scale invariance: adjust the size of the window according to the scale of the keypoint



Texture Descriptors

- ORB (Oriented Fast and Rotated Brief):
  - Similar to SIFT but it uses circular regions with 17 spatial bins and 16 orientation bins (278 dimension)



Texture Descriptors

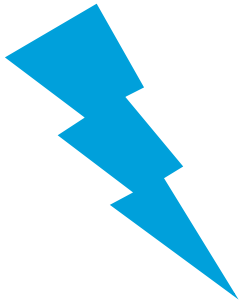
- GLOH (Gradient Location and Orientation Histograms):
  - Evolution of SIFT, free for commercial use and faster
  - Two parts:
    - Feature detector→ FAST (Feature from Accelerated Segment Test)
    - Feature descriptor → BRIEF (Binary Robust Independent Elementary Features)

Texture Descriptors

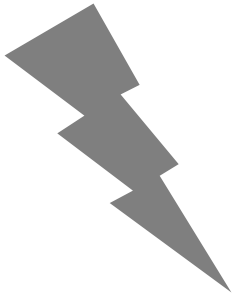
- GLOH (Gradient Location and Orientation Histograms):
  - FAST Feature Detector algorithm
    - Convert to gray level image
    - For each pixel we study a neighborhood of 16 pixels around it
    - The pixel is a corner if there is a chain of consecutive pixels much brighter surrounding the center point

Texture Descriptors

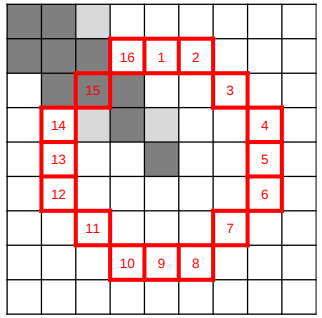
- GLOH (Gradient Location and Orientation Histograms):
  - FAST Feature Detector algorithm



Original image



Grey level image



Around pixel study

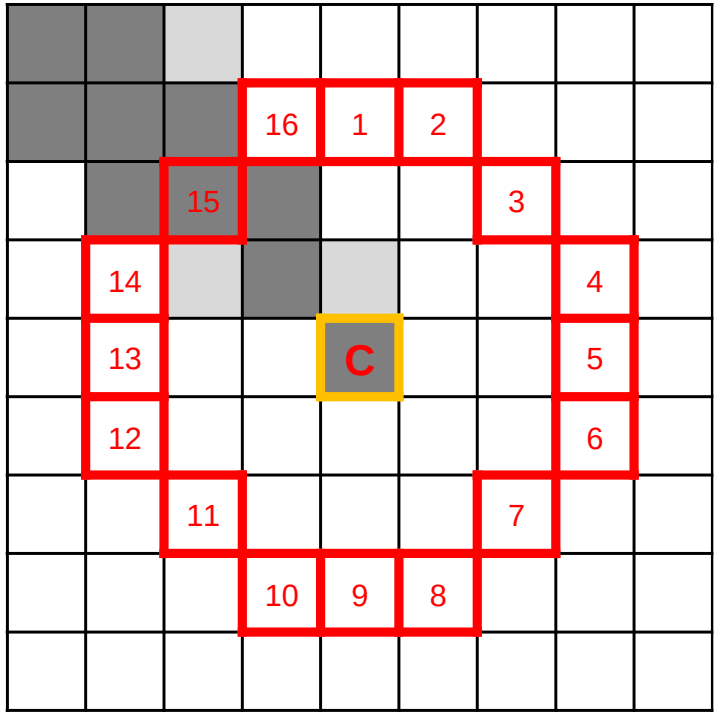
Case: dark pixel over a light background

Texture Descriptors

- GLOH (Gradient Location and Orientation Histograms):
  - FAST Feature Detector algorithm

1. Consider pixel C
2. Let J_C the intensity at C
3. Consider 16 pixels around C with intensities $J_1, \ldots, J_{16}$
4. Select a threshold T
5. C is a corner if there is a serie of n consecutive pixels where in the case of:

- Dark Corner: $J_C < J_n - T$
- Bright Corner: $J_C > J_n + T$

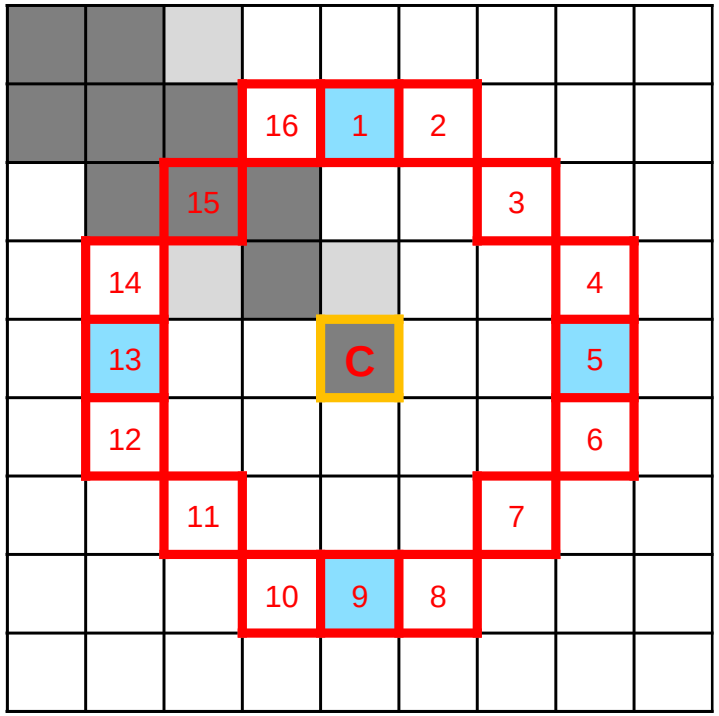


Texture Descriptors

- GLOH (Gradient Location and Orientation Histograms):
  - FAST Feature Detector algorithm

1. Consider pixel C
2. Let J_C the intensity at C
3. Consider 16 pixels around C with intensities $J_1, \ldots, J_{16}$
4. Select a threshold T
5. C is a corner if there is a serie of n consecutive pixels where in the case of:

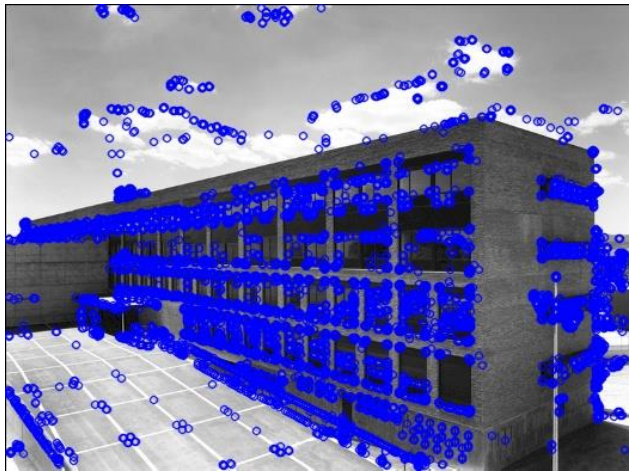
- Dark Corner: $J_C < J_n - T$
- Bright Corner: $J_C > J_n + T$



Texture Descriptors

- GLOH (Gradient Location and Orientation Histograms):
  - FAST Feature Detector algorithm corner detection

```
import numpy as np
import cv2 as cv

img = cv.imread('ETSE.png',0)

fast = cv.FastFeatureDetector_create(threshold =
25,nonmaxSuppression=0,
type = cv.FAST_FEATURE_DETECTOR_TYPE_9_16))

kp = fast.detect(img,None)

out = cv.drawKeypoints(img, kp, None,
color=(255,0,0), flags=None)

cv.imshow('corner image', out)
cv.waitKey(0)
cv.destroyAllWindows()
```



threshold = 25

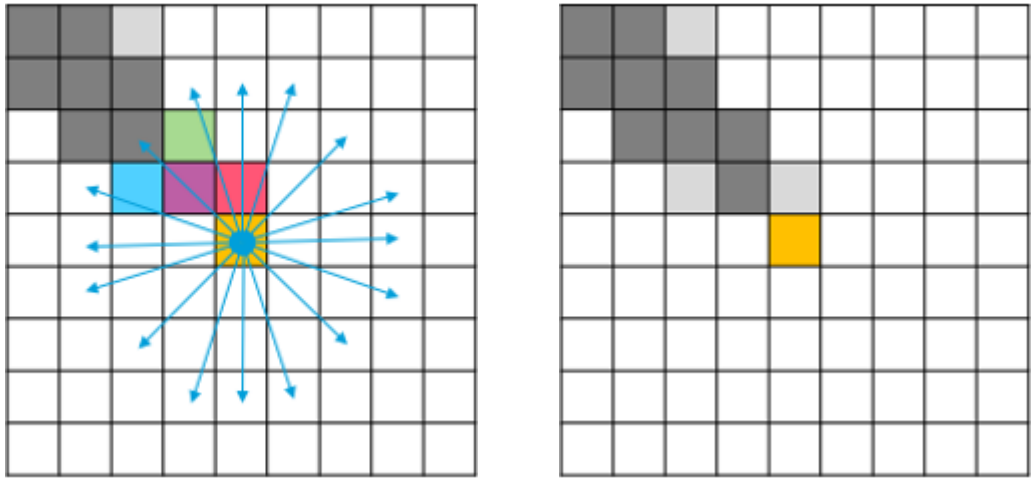
n = 9

contiguous pixels in the circle

Texture Descriptors

- GLOH (Gradient Location and Orientation Histograms):
  - FAST Feature Detector algorithm corner detection
    - Problem with detections in a similar position
    - Solution: computation of score function (non-maximum suppression)

1. Compute a score function, *V* for all the detected feature points
2. *V* is the sum of absolute difference between *C* and 16 surrounding pixels values
3. Consider two adjacent keypoints and compute their *V* values
4. Discard the one with lower *V* value

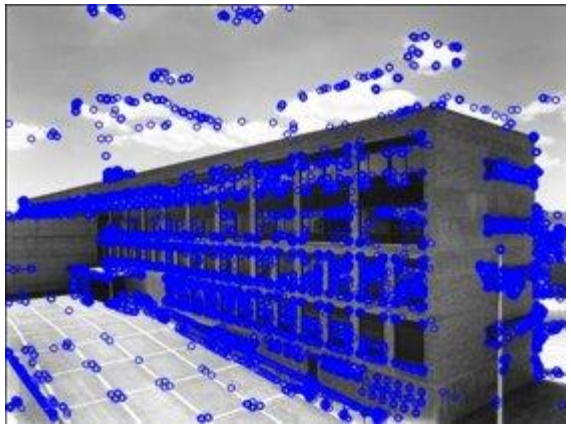


Non-maximum Suppression

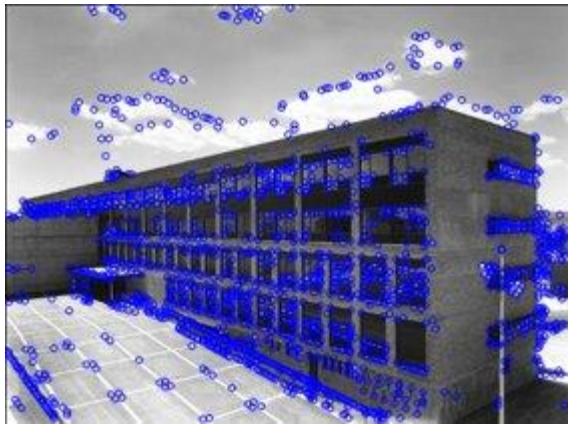
Texture Descriptors

- GLOH (Gradient Location and Orientation Histograms):
  - FAST Feature Detector algorithm corner detection

```
import numpy as np
import cv2 as cv

img = cv.imread('ETSE.png',0)

fast = cv.FastFeatureDetector_create(threshold =
25,nonmaxSuppression=1,
type = cv.FAST_FEATURE_DETECTOR_TYPE_9_16))

kp = fast.detect(img,None)

out = cv.drawKeypoints(img, kp, None,
color=(255,0,0), flags=None)

cv.imshow('corner image', out)
cv.waitKey(0)
cv.destroyAllWindows()
```



Total Keypoints: 5209



Total Keypoints: 1655

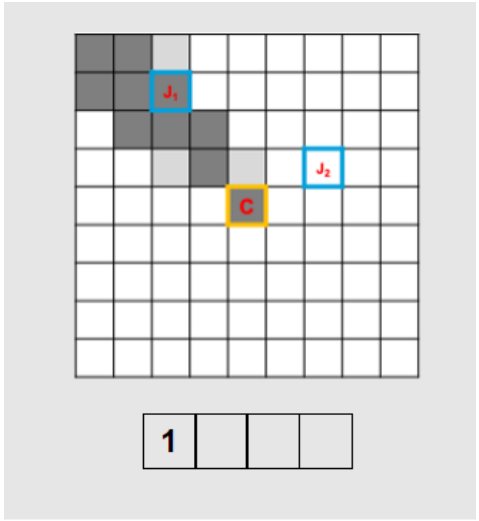
Texture Descriptors

- GLOH (Gradient Location and Orientation Histograms):
  - BRIEF Feature Descriptor
    - We define a neighborhood around each keypoint (smoothing works)

1. Randomly select ***n*** pair of points inside the patch $P_1=(x_1,y_1)$ and $P_2=(x_2,y_2)$

   NOTE: ***n*** is the length of the binary feature vector. It could be 128, 256, and 512

2. Let J_1 and J_2 the intensity at P_1 *and* P_2
3. if ($J_1 < J_2$)
4. output = 1
5. else
6. output = 0



n = 4

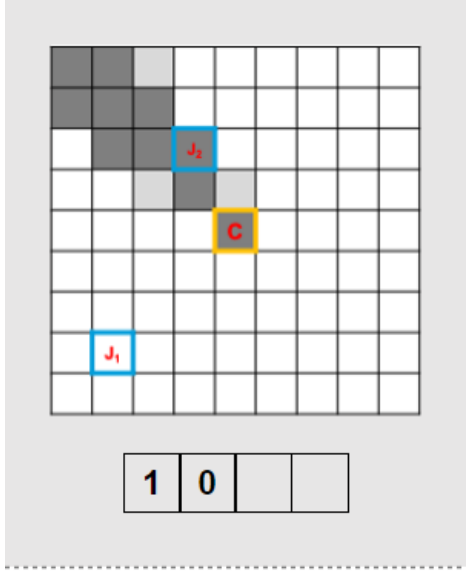
Texture Descriptors

- GLOH (Gradient Location and Orientation Histograms):
  - BRIEF Feature Descriptor
    - We define a neighborhood around each keypoint (smoothing works)

1. Randomly select ***n*** pair of points inside the patch $P_1=(x_1,y_1)$ and $P_2=(x_2,y_2)$

   NOTE: ***n*** is the length of the binary feature vector. It could be 128, 256, and 512

2. Let J_1 and J_2 the intensity at P_1 *and* P_2
3. if ($J_1 < J_2$)
4. output = 1
5. else
6. output = 0



n = 4

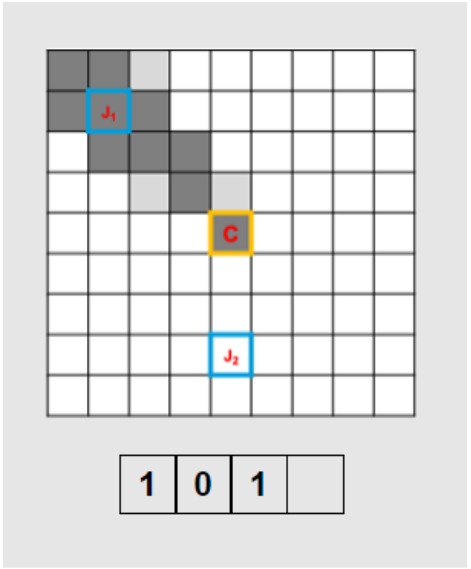
Texture Descriptors

- GLOH (Gradient Location and Orientation Histograms):
  - BRIEF Feature Descriptor
    - We define a neighborhood around each keypoint (smoothing works)

1. Randomly select ***n*** pair of points inside the patch $P_1=(x_1,y_1)$ and $P_2=(x_2,y_2)$

   NOTE: ***n*** is the length of the binary feature vector. It could be 128, 256, and 512

2. Let J_1 and J_2 the intensity at P_1 *and* P_2
3. if ($J_1 < J_2$)
4. output = 1
5. else
6. output = 0



n = 4

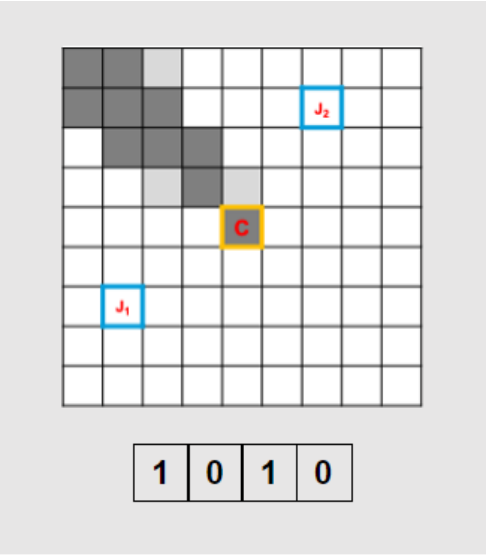
Texture Descriptors

- GLOH (Gradient Location and Orientation Histograms):
  - BRIEF Feature Descriptor
    - We define a neighborhood around each keypoint (smoothing works)

1. Randomly select ***n*** pair of points inside the patch $P_1=(x_1,y_1)$ and $P_2=(x_2,y_2)$

   NOTE: ***n*** is the length of the binary feature vector. It could be 128, 256, and 512

2. Let J_1 and J_2 the intensity at P_1 *and* P_2
3. if ($J_1 < J_2$)
4. output = 1
5. else
6. output = 0



n = 4

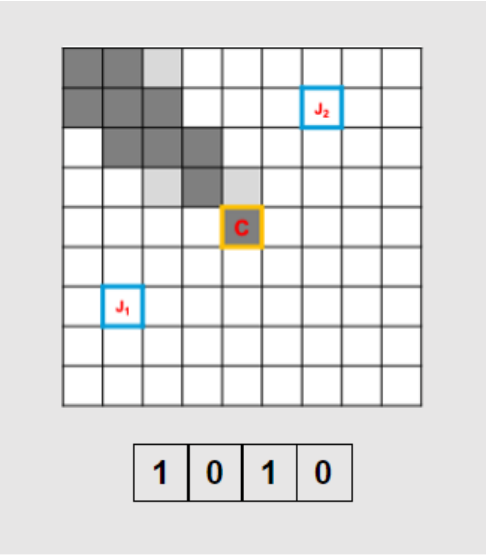
Texture Descriptors

- GLOH (Gradient Location and Orientation Histograms):
  - BRIEF Feature Descriptor
    - We define a neighborhood around each keypoint (smoothing works)

1. Randomly select ***n*** pair of points inside the patch $P_1=(x_1,y_1)$ and $P_2=(x_2,y_2)$

   NOTE: ***n*** is the length of the binary feature vector. It could be 128, 256, and 512

2. Let J_1 and J_2 the intensity at P_1 *and* P_2
3. if ($J_1 < J_2$)
4. output = 1
5. else
6. output = 0



n = 4

Texture Descriptors

- GLOH (Gradient Location and Orientation Histograms):
  - BRIEF Feature Descriptor

```
import numpy as np
import cv2 as cv


img = cv.imread('ETSE.png',0)

img = cv.GaussianBlur(img,(3,3),2)

# Initiate FAST detector
fast = cv.FastFeatureDetector_create(threshold =
25,nonmaxSuppression=1,
type = cv.FAST_FEATURE_DETECTOR_TYPE_9_16))
# Initiate BRIEF extractor
brief = cv.xfeatures2d.BriefDescriptorExtractor_create()
# find the keypoints
kp = fast.detect(img,None)
# compute the descriptors with BRIEF
kp, des = brief.compute(img, kp)


out = cv.drawKeypoints(img, kp, None,
color=(255,0,0), flags=None)


cv.imshow('corner image', out)
cv.waitKey(0)
cv.destroyAllWindows()
```

Smoothed image

Total Keypoints: 589

Texture Descriptors

- GLOH (Gradient Location and Orientation Histograms):
  - BRIEF Feature Descriptor

```
# continue from the previous slide

print( "Total Keypoints: {}".format(len(kp)) )

n_kp = 8 # select one point of the list of keypoints

x,y = kp[n_kp].pt
print("(x,y) ",x,y) # print coordinates

out = cv.drawKeypoints(img, [ kp[n_kp] ], None,
color=(255,0,0), flags=None)

print(”Byte descriptor: ",descriptor[n_kp]) # print byte
descriptor

b = [bin(n)[2:] for n in descriptor[n_kp]]
b="".join(b)
print("Bianry descriptor: ",b) # print binary descriptor
```


```
Byte descriptor:
[24 68 76 189 45 51 204 201 50 135 187 176
184 206 160 8 102 248 87 181 10 136 205 42
205 216 164 122 250 72 66 119]

Byte descriptor:
11000100010010011001011110110110111001111 0
01100110010011100101000011110111011100 0
01011100011001110101000010001100110111110
00101011110110101101010001000110011011010 1
01100110111011000101001001111010111110101 0
0100010000101110111
```

Vector dimension 256

Fundamentals of Computer Vision

Unit 7: Feature Description

Jorge Bernal