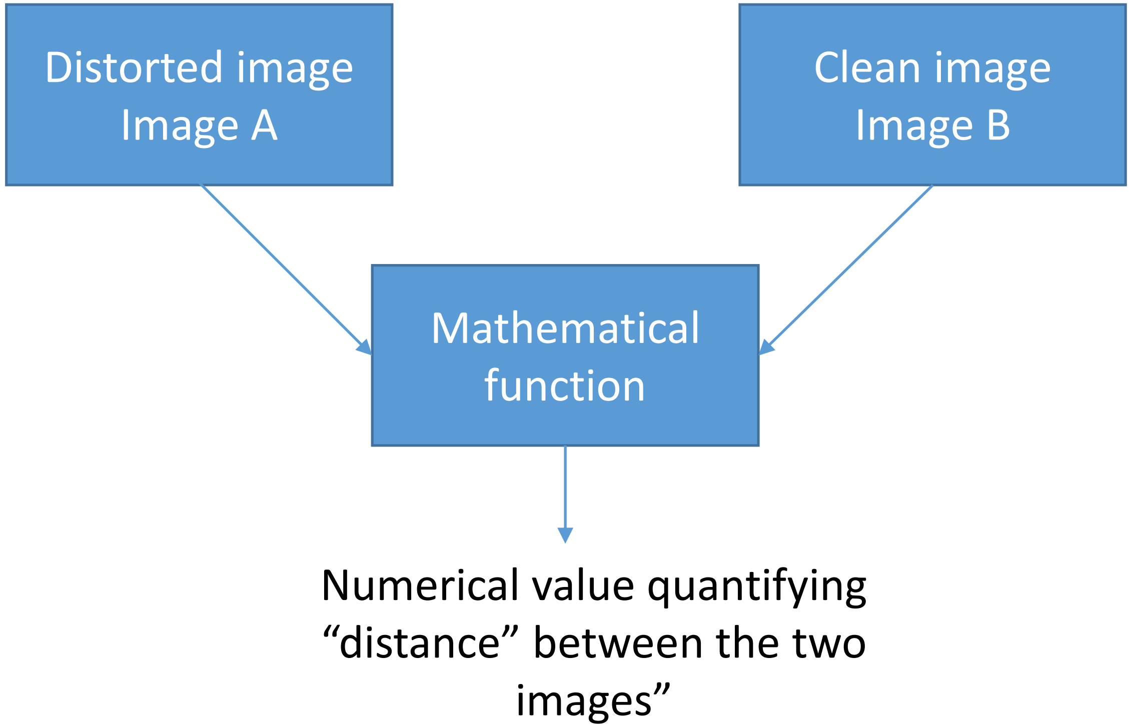
Image Descriptors

Outline

1. Introduction
2. Image enhancement
3. Frequency domain operations
4. **Image descriptors**
5. Machine learning
6. Neural networks
7. Segmentation and object detection
8. Morphological processing
9. Geometric transformations
10. Motion analysis and optical flow
11. Compression
12. Other topics

Image Quality Assessment

- Full reference: Given two images how to measure the difference/similarity between them?
  - Important for denoising and image restoration problems



Quality metrics

- Mean Squared Error (MSE)

$$MSE(A,B) = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}(A[m,n] - B[m,n])^2$$

- Root Mean Squared Error (RMSE)

$$RMSE(A,B) = \sqrt{MSE(A,B)}$$

$$RMSE(A,B) = \sqrt{\frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}(A[m,n] - B[m,n])^2}$$

Quality metrics

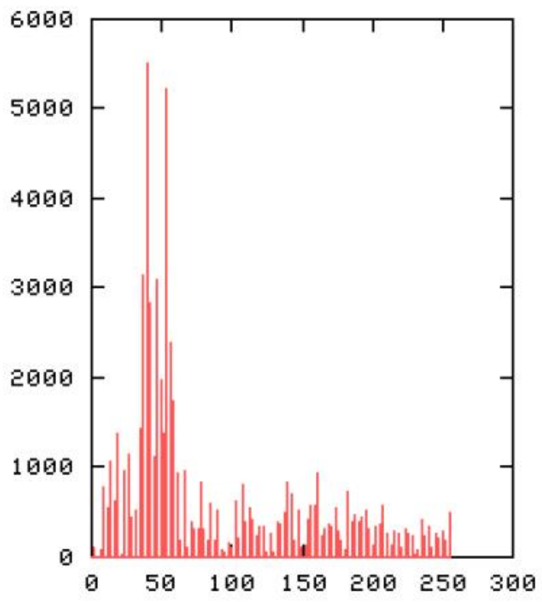
- Peak Signal to Noise Ratio (PSNR)

$$PSNR(A,B) = 10\ \log_{10}\frac{MaxI^2}{MSE}$$

- Where *MaxI* is the maximum possible intensity value (e.g. 255 for 8-bit images)

Image retrieval

- For image retrieval, we need a more general descriptor
- Histograms are frequently used in image retrieval systems



Computing distance between histograms

Euclidean distance (or L[1] distance)

$$\mathrm{dist}(H_i, H_j) = \sqrt{\sum_{m=1}^{M} \left(H_i(m) - H_j(m)\right)^2} \quad \text{or} \quad \sum_{m=1}^{M} \left|H_i(m) - H_j(m)\right|$$

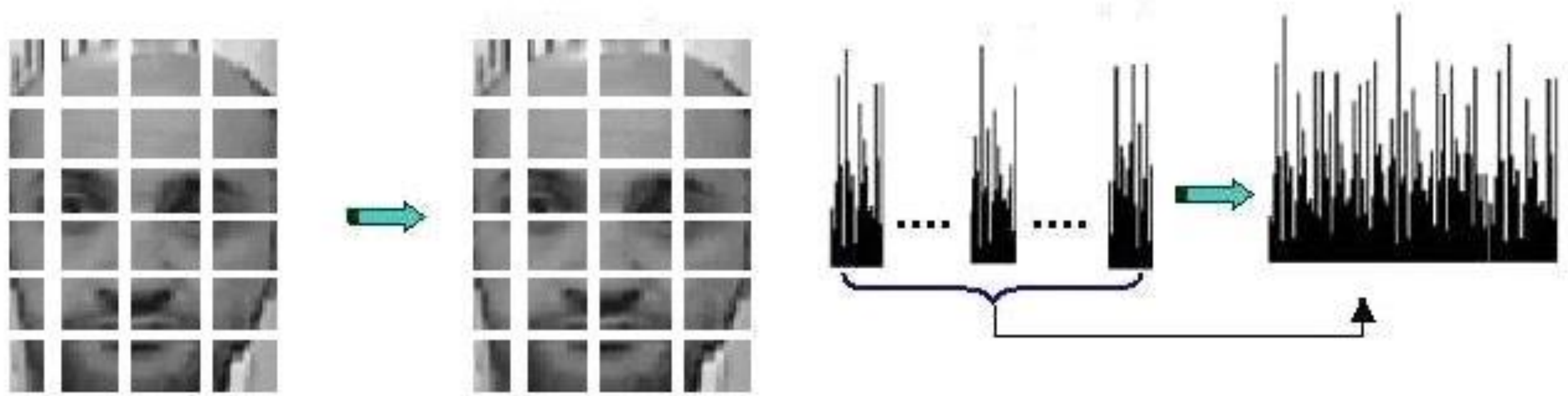
Chi-squared histogram matching distance

$$\chi^2(H_i, H_j) = \frac{1}{2} \sum_{m=1}^{M} \frac{\left(H_i(m) - H_j(m)\right)^2}{H_i(m) + H_j(m)}$$

Cars found by color histogram matching using chi-squared

Image retrieval

with histograms is that they provide no spatial information, so often the image is divided into sub-regions before taking the histogram.



Histogram concatenation

Histogram Normalization

- To be able to compare these histograms, we should normalize them.
- Histograms can be normalized by dividing the value in each bin by the total number of histogram count data.

Choosing the number of bin in histograms

- Quantization

Few Bins
Need less data
Coarser representation

Many Bins
Need more data
Finer representation

- Matching
  - L_1 or L_2 distance measures.

$$D(I,J) = \sum_i |H_I(i) - H_J| \qquad D(I,J) = \sqrt{\sum_i |H_I(i) - H_J(i)|^2}$$

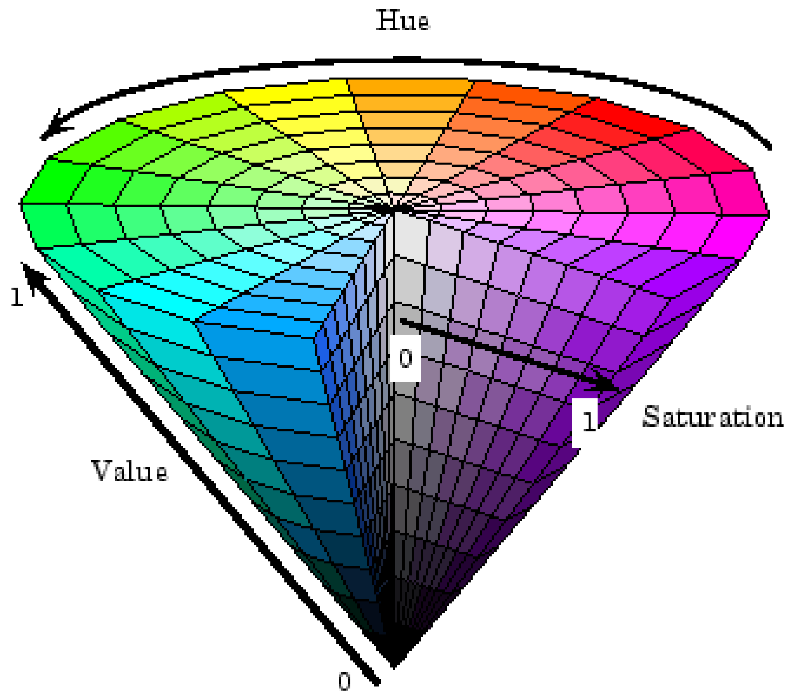
  - Chi-squared distance

$$D(I,J) = \sum_i \frac{2(H_I(i) - H_J(i))^2}{H_I(i) + H_J(i)}$$

  - Euclidian distance is faster, Chi-squared often works better

What kind of things do we compute histograms of in machine vision?

- Color (RGB or HSV)/intensity



- “Does this image has a red car?” Getting the color right is relatively easy.

Histogram in classification

- In several applications, variations in brightness or colors are not considered relevant.
- The difference between a pixel and its surrounding is critical in image retrieval.
- We will review different methods for computing these descriptors
  - Local Binary Patterns (LBP)
  - Histogram of Oriented Gradients (HOG)
  - SIFT features

Local Binary Patterns (LBP)

- A well-known feature detector
- It uses a feature vector that summarizes the spatial relationship between a pixel and its neighbors over the image.
- They work well with repeated patterns (i.e. textures)
- Image retrieval: identifying different materials: fabric, granite.

Local Binary Patterns (Threshold at the center pixel intensity)

- For each pixel p, create an 8-bit number $b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7\ b_8$, where $b_i = 1$ if neighbor i has value larger than or equal p's value and 0 otherwise.

- Represent the texture in the image (or a region) by the histogram of these numbers.

	1	2	3	
	100	101	103	
8	40	50	80	4
	50	60	90	5
	7	6		

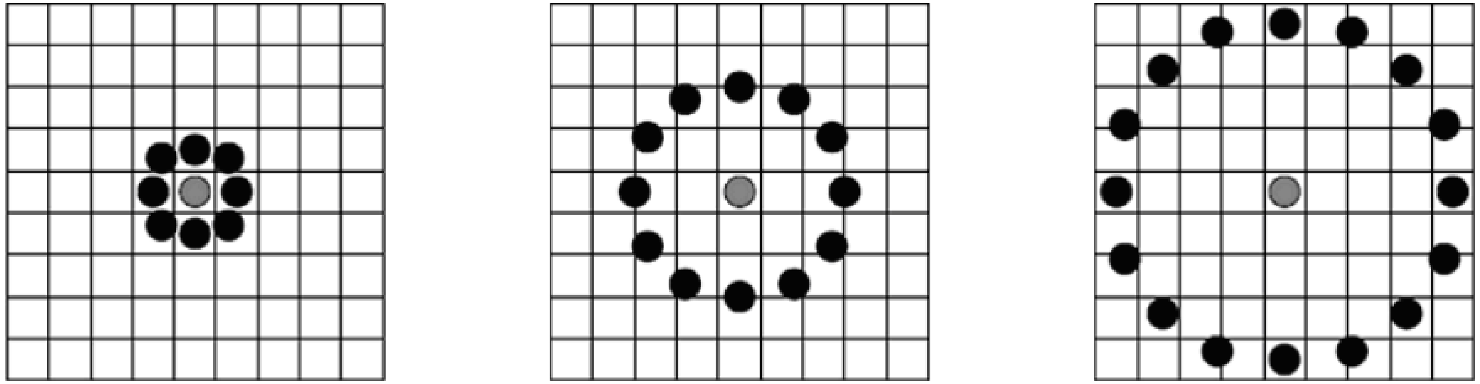
1 1 1 1 1 1 1 0

Decimal 254

Local Binary Patterns (Threshold at the center pixel intensity)

- Alternate implementations: P Number of neighboring pixels R radius

P = 8, R=1



No interpolation

Interpolation is needed

Interpolation: finding pixel values at locations not on the grid

Local Binary Patterns (Alternate implementation)

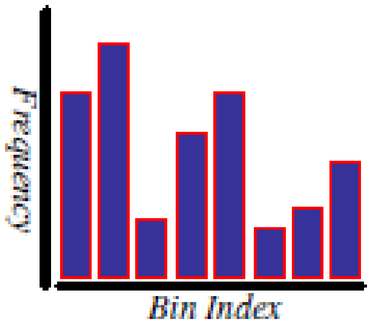
- Threshold value = center pixel intensity + 30

	1	2	3	
	100	101	103	
8	40	50	80	4
	50	60	90	5
	7	6		

→

1 1 1 1 1 0 0 0

Decimal 248



Bin index 0,1,2,…N

N is the number of binary codes generated

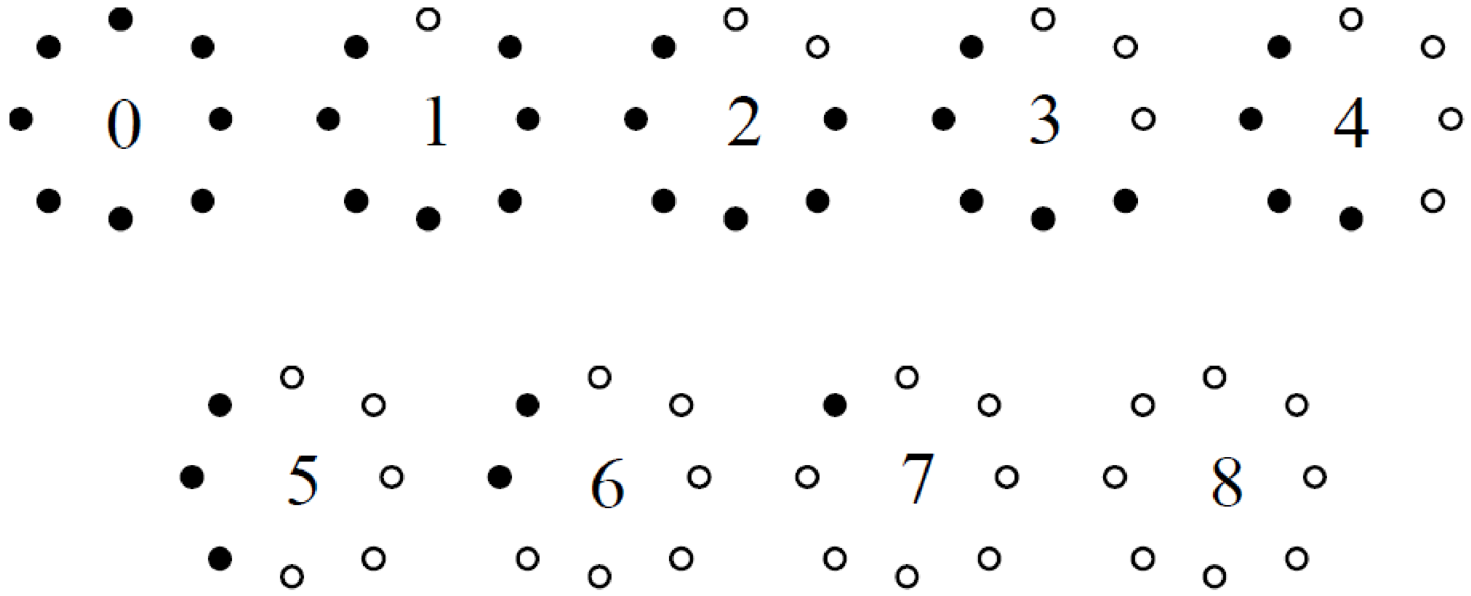
An alternate implementation using a threshold of center pixel intensity +30.

Uniform LBP

- Computing the original LBP is time consuming. A better approach is to include uniform patterns only.
- An LBP is called uniform if the circular binary pattern contains maximal 2 transitions from 0 to 1 and vice versa.

Uniform LBP

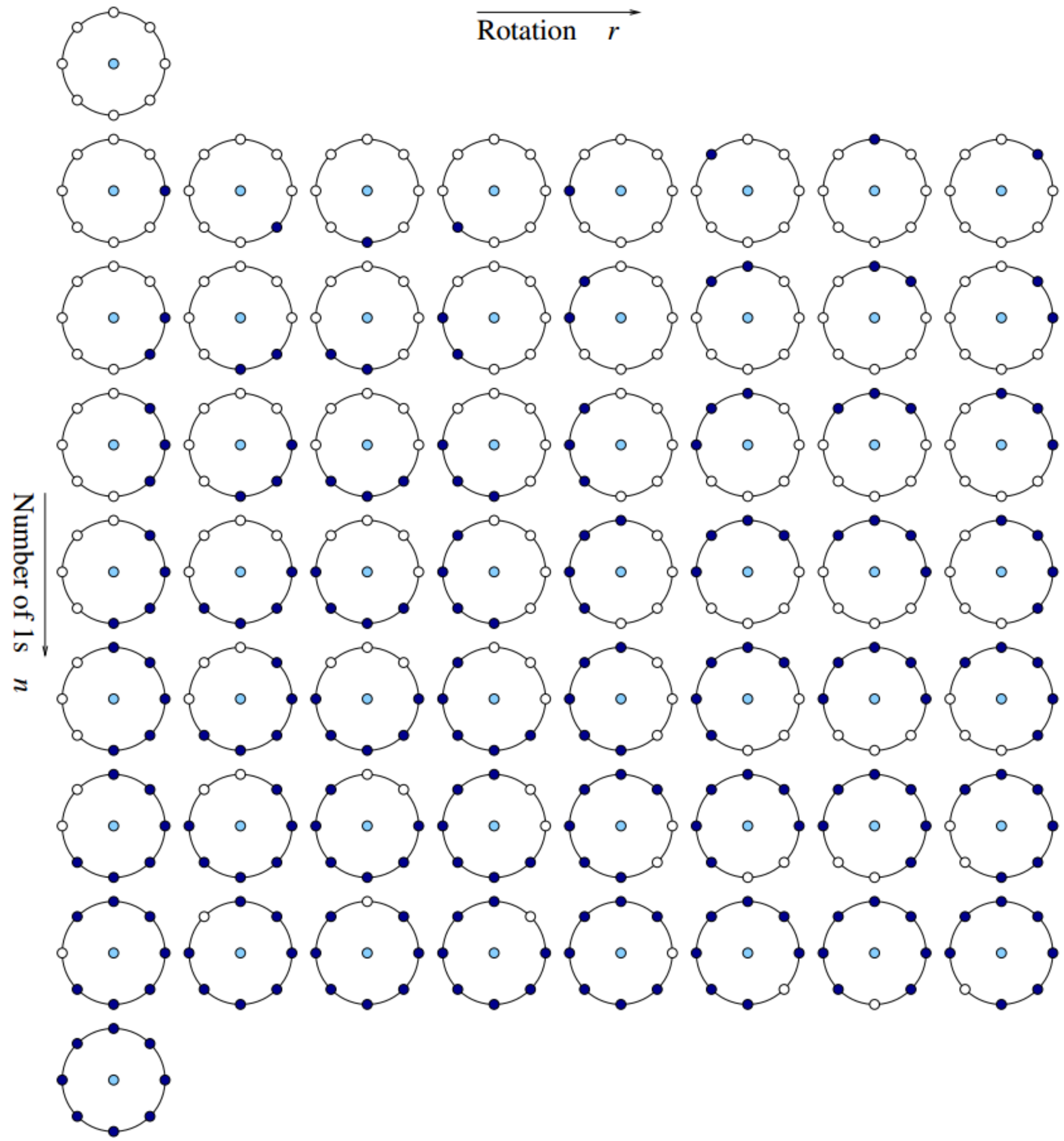
- It was observed that 80%-90% of patterns in normal images are uniform.
- The impact of non-uniform patterns is marginal.



Nine ‘uniform’ patterns in LBP_8

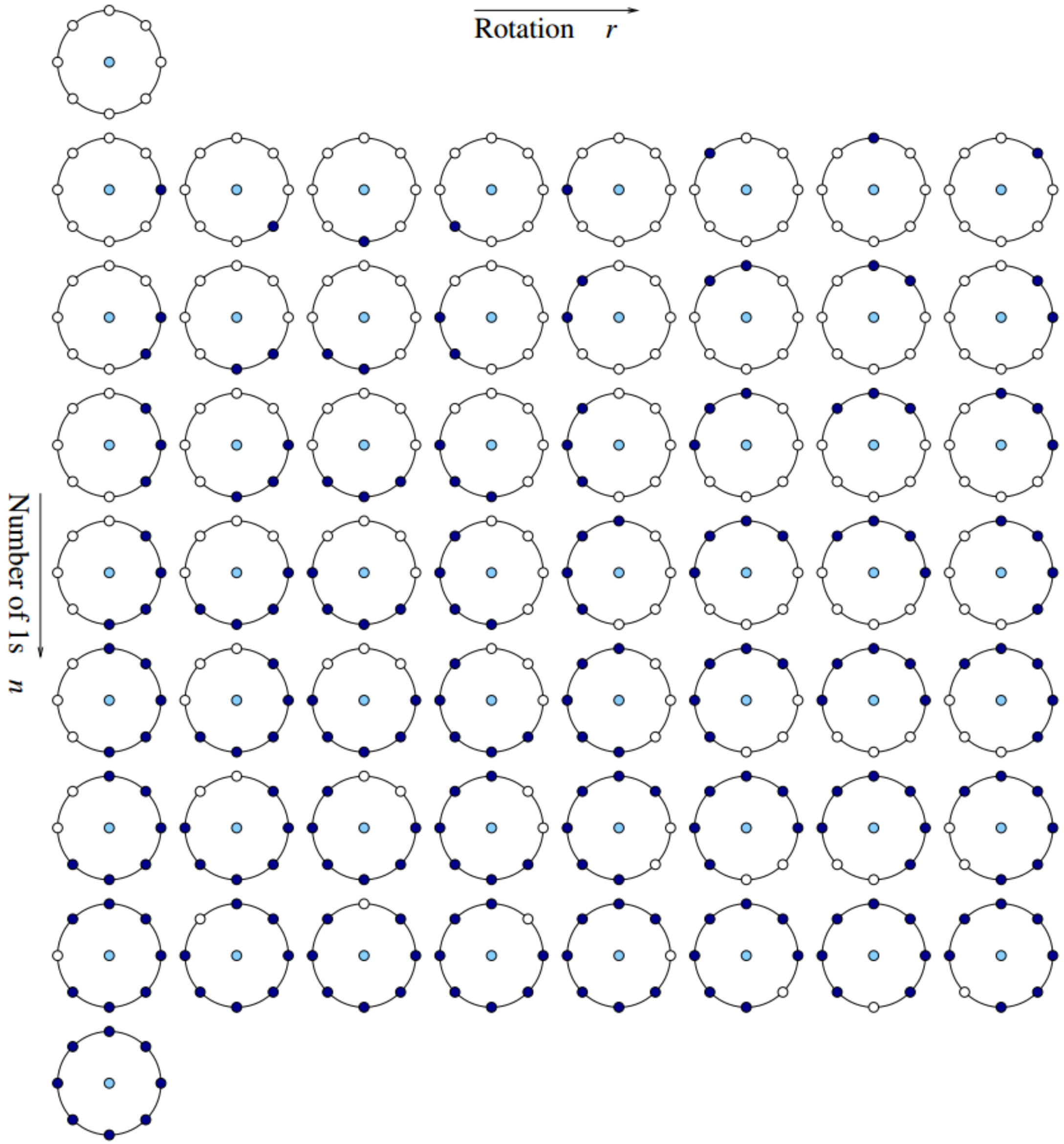
Uniform LBP Patterns

- In an (8, R) neighborhood there are 58 uniform patterns



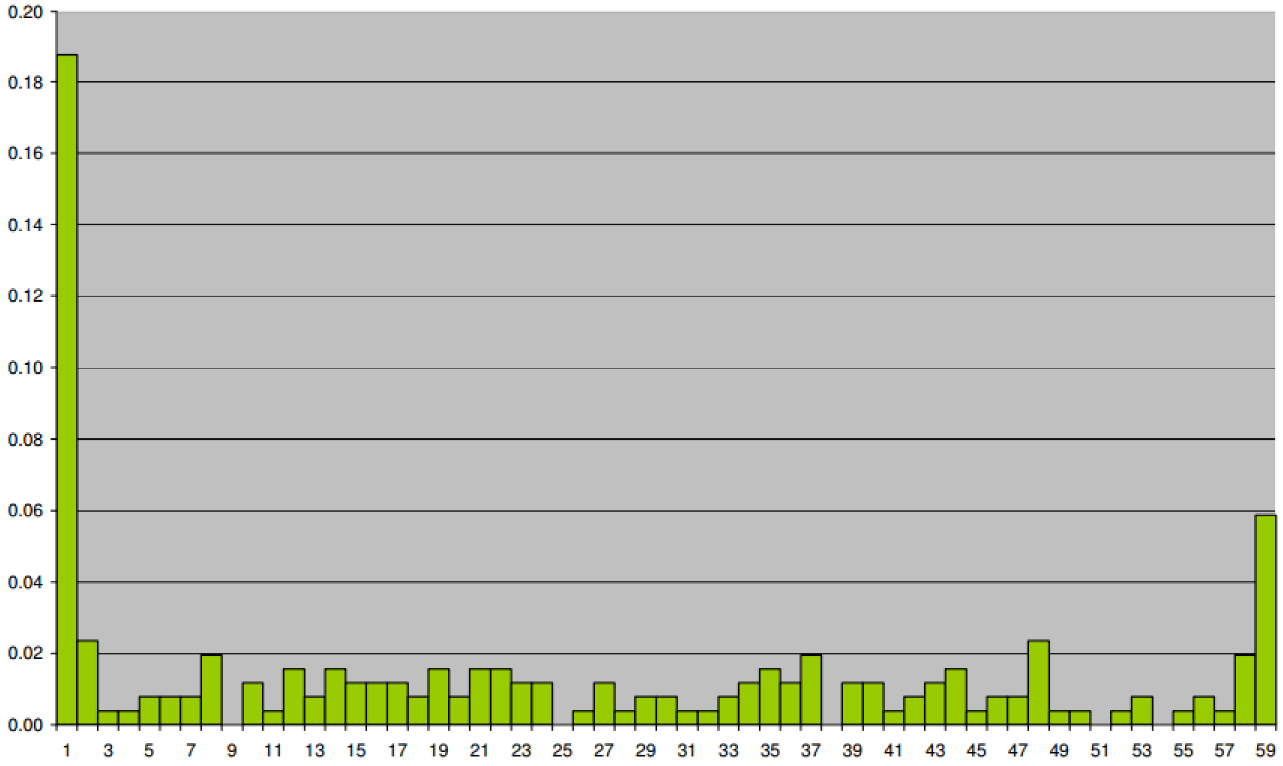
Uniform LBP

- An improved variant of LBP.
- Counting the number of different patterns. No conversion to decimal.



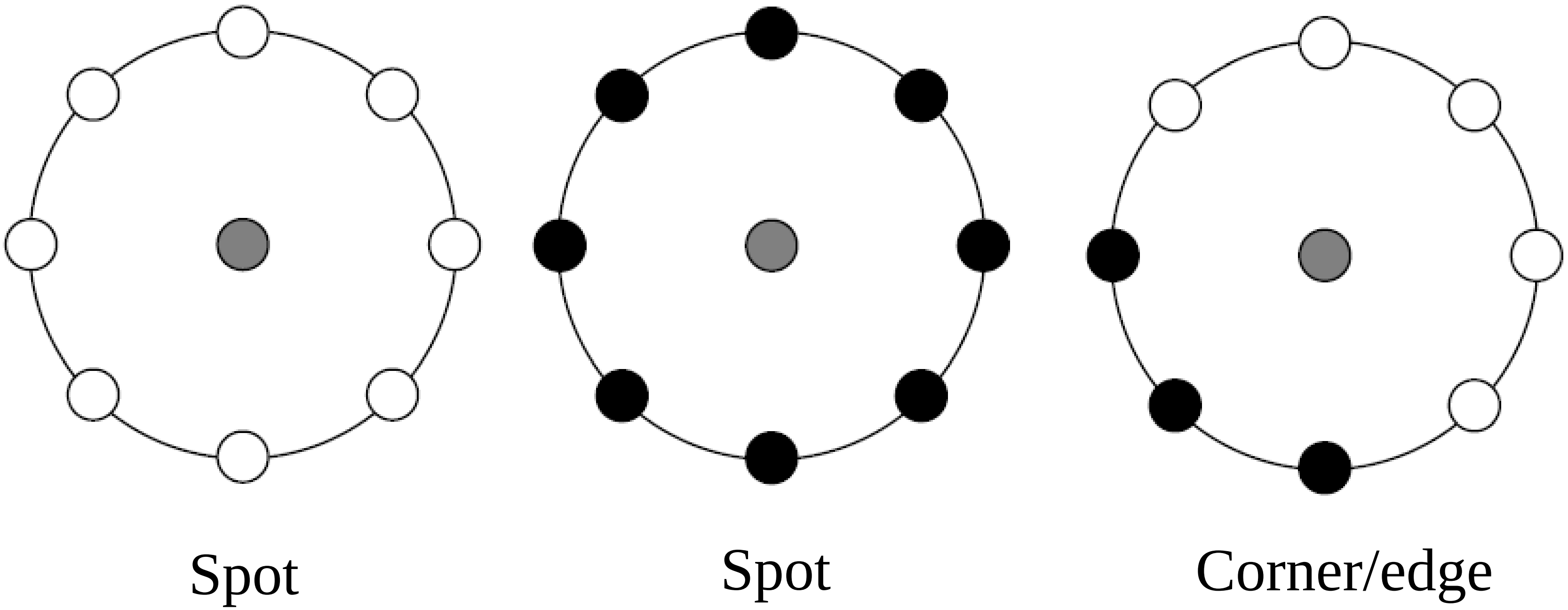
Uniform LBP

- Examples LBP histogram of a sample image. All other non-uniform patterns are counted in histogram bin 59.

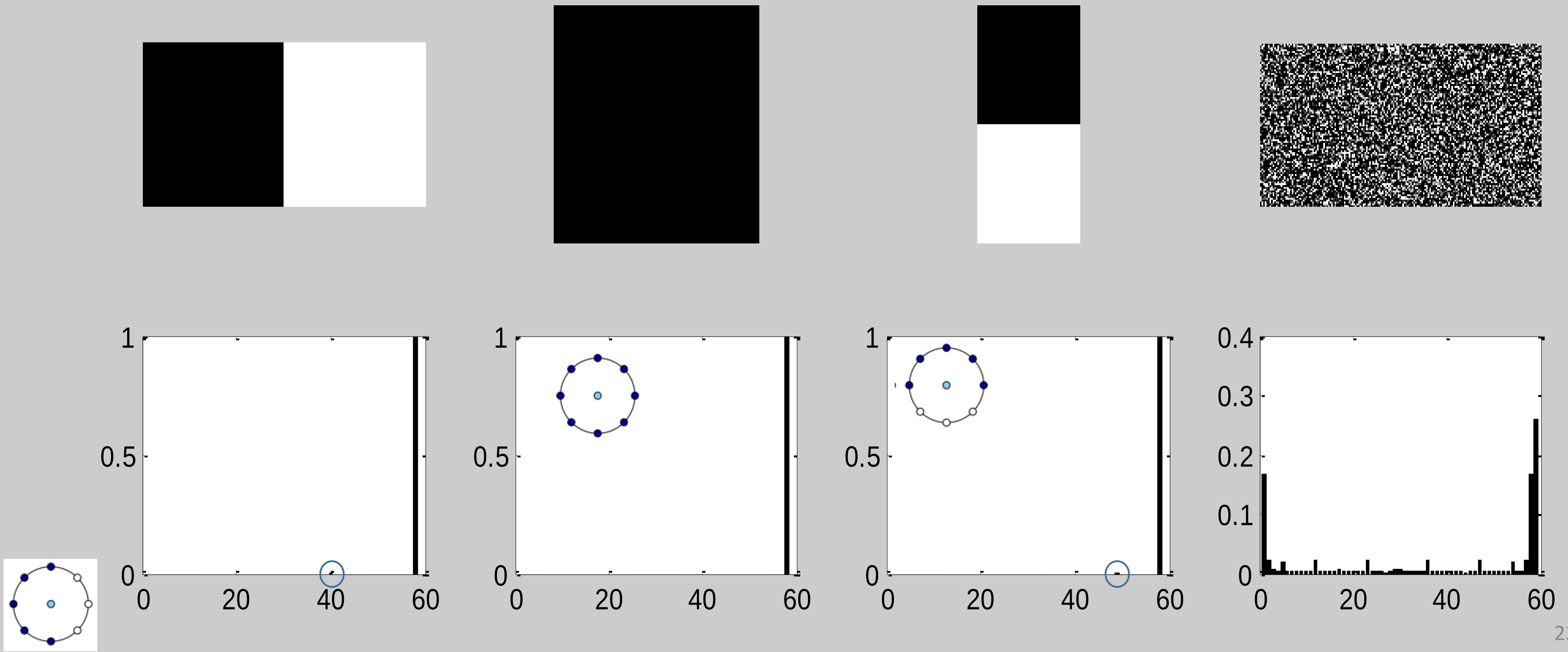


The 59 histogram bins acquired from a sample face image

Primitive shapes detected by LBP

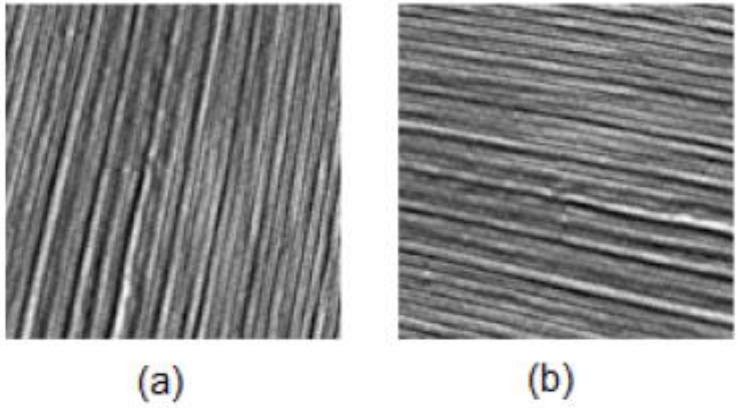


Sample images and their uniform LBPs



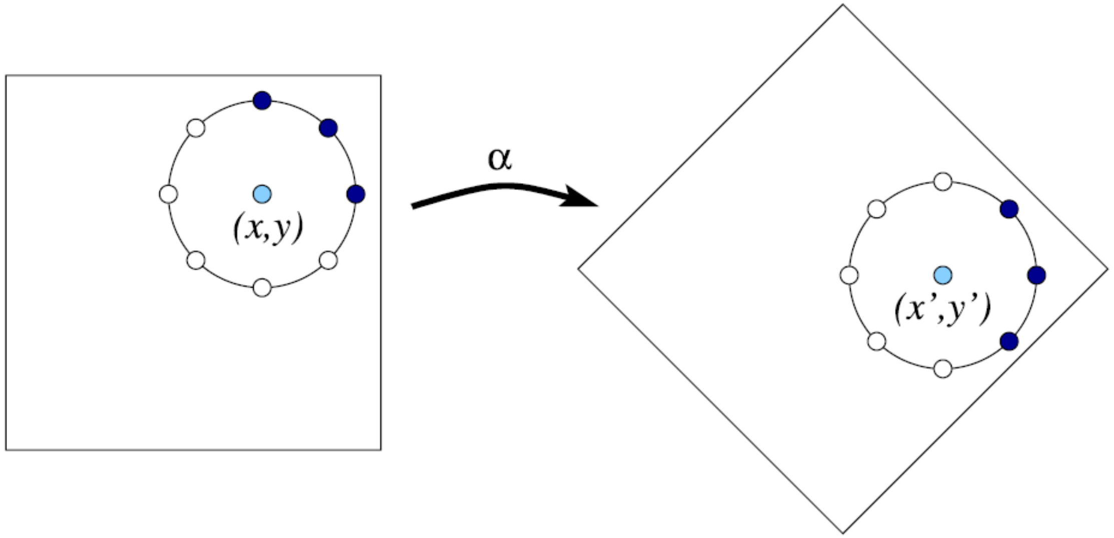
Rotation invariance

- Goal: Develop an LBP descriptor that is invariant to rotation.



- We like the LBP descriptor for images (a) and (b) to be equal or at least close to each other.

Rotation invariant uniform LBP



Effect of rotation on an LBP neighborhoods

Rotation invariant uniform LBP

- The 58 different uniform patterns in (8,R) neighborhood
- To achieve rotation-invariance:

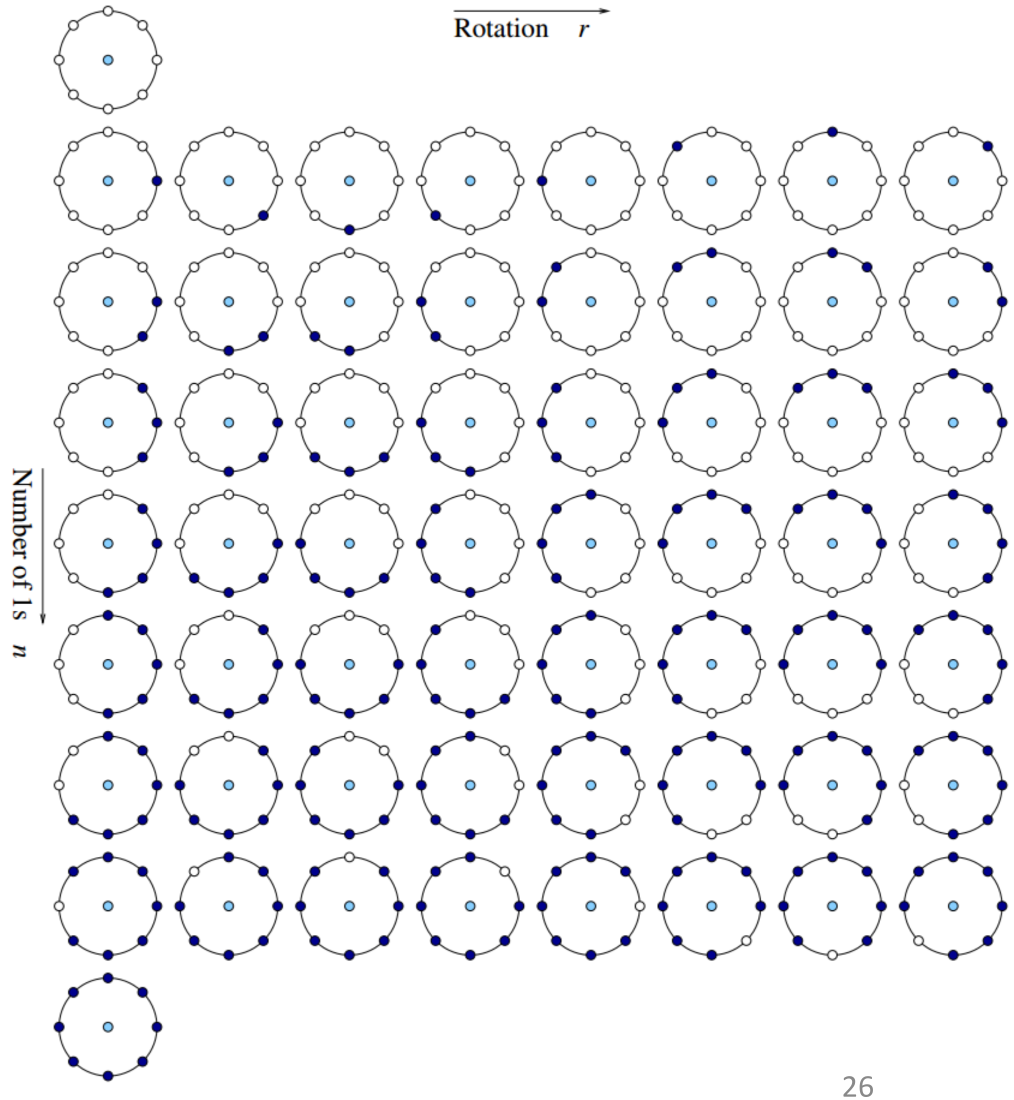
$$LBP_8^{riu2} = \mathrm{sum}\{cirrightshift(LBP_8^{u2}, i)\}$$

where $0 \leq i \leq 7$

- $cirrightshift$ performs a circular bit-wise right shift on the 8-bit number x i times.
- This does not apply to patterns $(00000000)_2$ and $(11111111)_2$

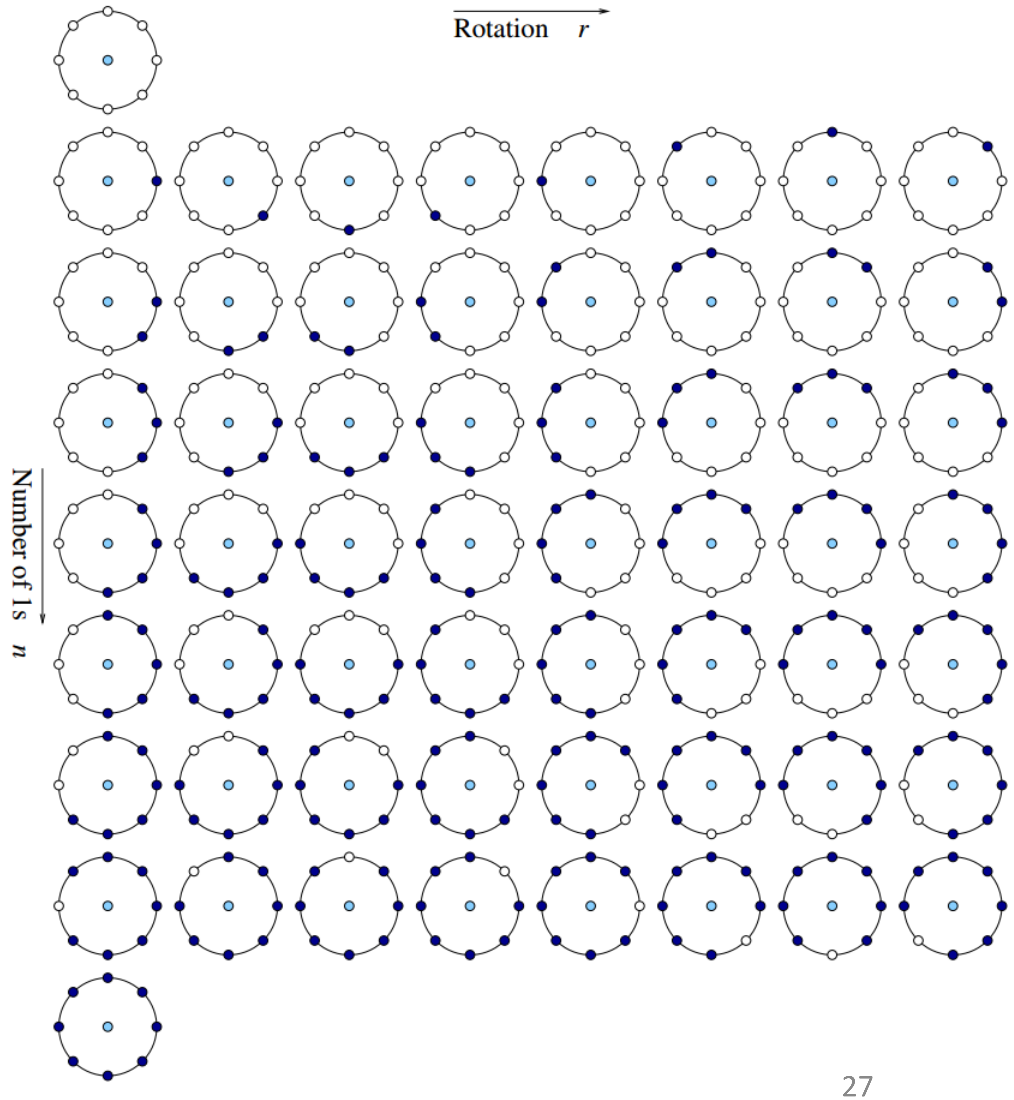
  which remain constant at all rotation

  angles



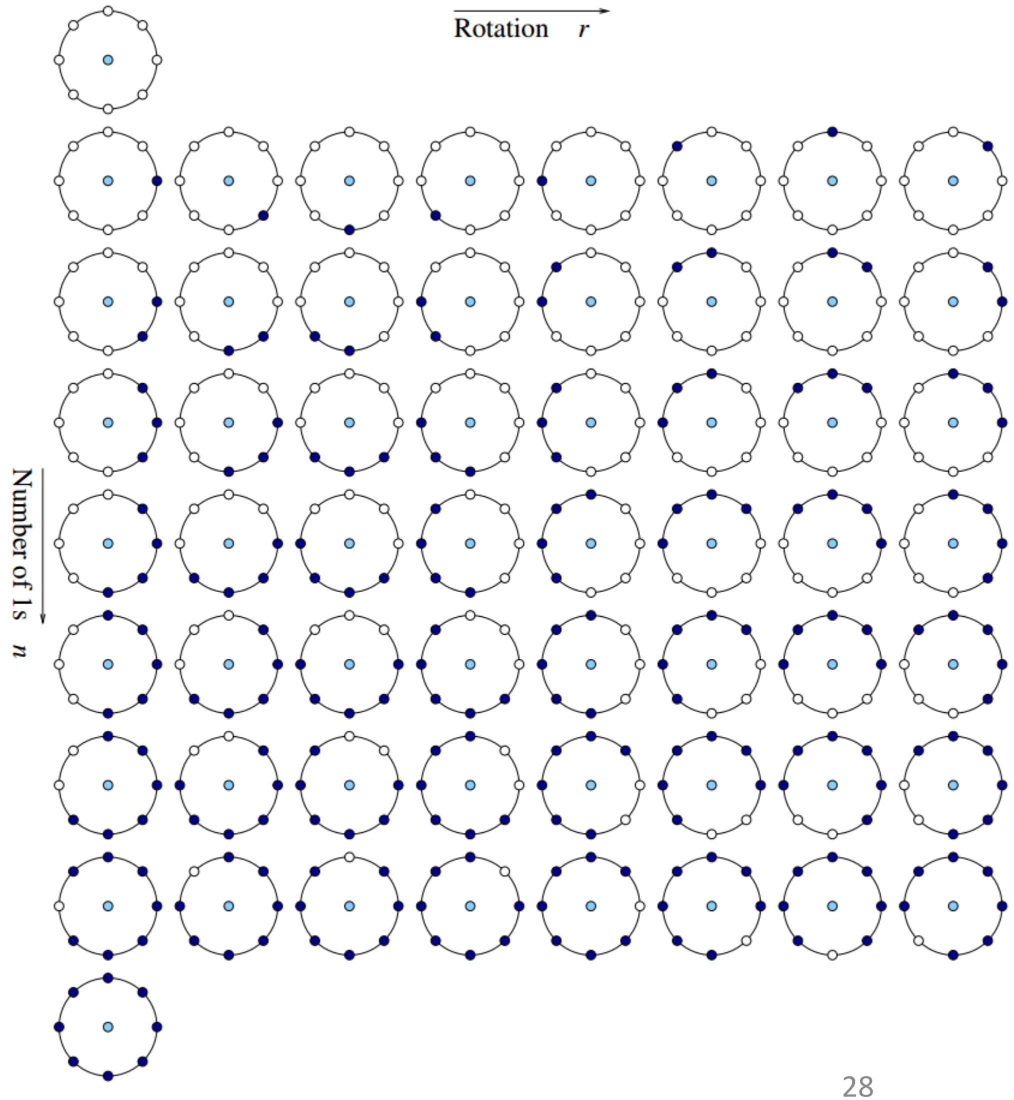
Rotation invariant uniform LBP

- Achieves rotation invariance for angles given by $\omega = a\,\frac{360}{P}$, a = *0,1,2,…,* P-1

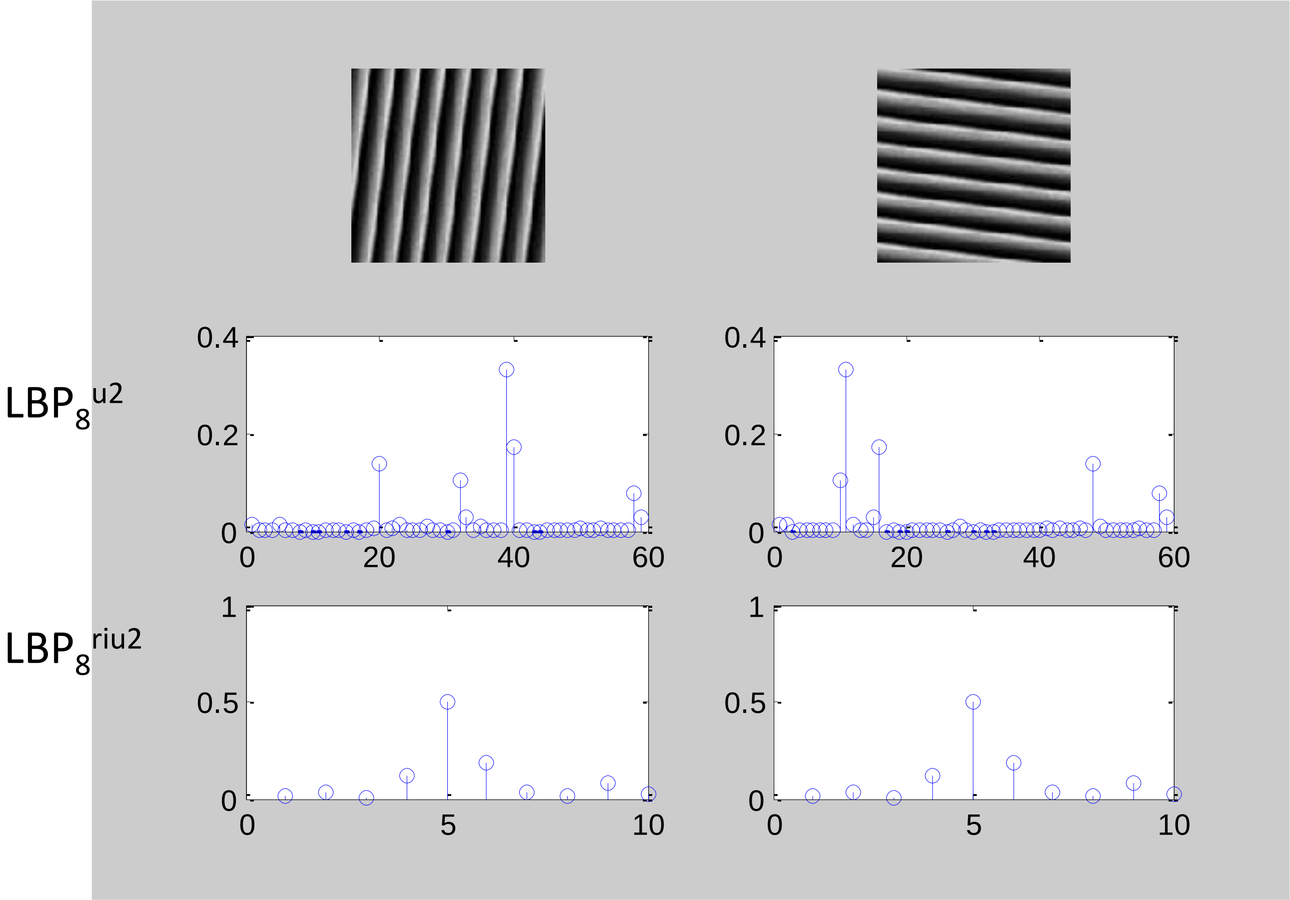


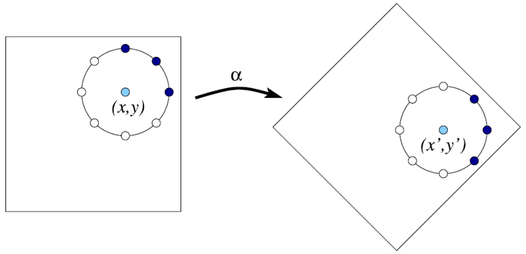
Rotation invariant uniform LBP

- In LBP_8^{riu2} the total number of histogram bins is 10
- One bin for $(00000000)_2$
- One bin for $(11111111)_2$
- Seven bins for the other uniform LBP's
- One bin for all non-uniform LBP's



Rotations in LBP





How does LBP achieve invariance to illumination changes?

Distance between histograms

- Log-likelihood distance

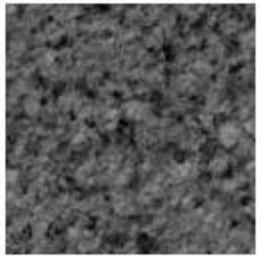
$$\sum_i h_1(i) \times \log(h_2(i))$$

- LBP authors recommend the following log-likelihood approach to measure the similarity between images

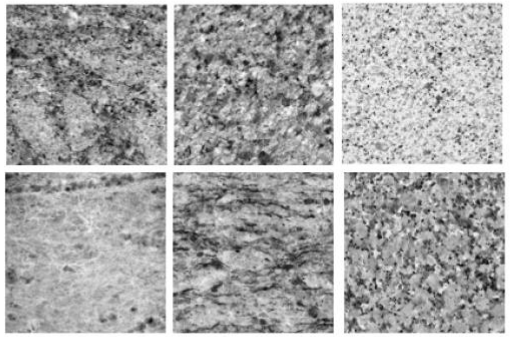
$$D(A,B) = \sum_i h_{A\,LBP_{(1,8)}}(i) \times \log\left(h_{B\,LBP_{(1,8)}}(i)\right) + \sum_j h_{A\,LBP_{(3,24)}}(j) \times \log\left(h_{B\,LBP_{(3,24)}}(j)\right)$$

Using LBP in image retrieval

- We have a database of images and a query image



Query image



Database

- Goal is to find the correct match for the query image in the database.

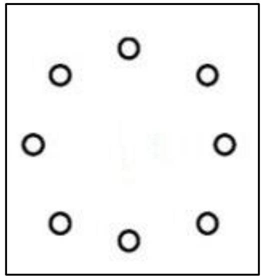
Using LBP in image retrieval

1. Compute the LBP histogram for each image in the database.
2. Compute the LBP histogram for the query image.
3. The image with the closest histogram in the database is considered the first candidate for the correct match. The 2^{nd} closest histogram is the second match and so on.

Some Implementation details

- Coordinates for points x_p and y_p around a center pixel (x,y) are computed according to the following equation:

$$x_p = x + R\cos(2\pi p/P),$$
$$y_p = y - R\sin(2\pi p/P).$$

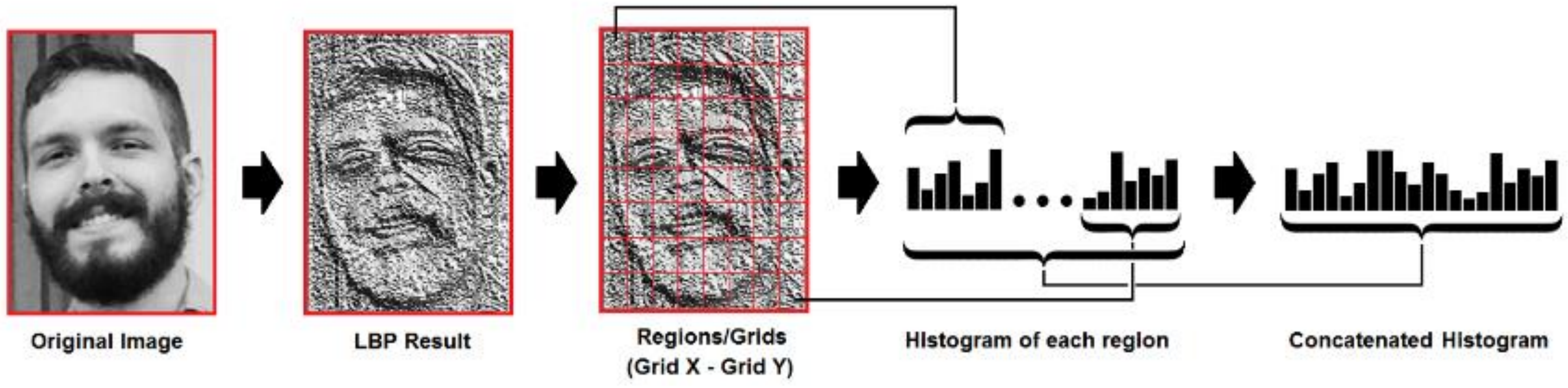


These equations find the x and y coordinates of points in LBP patterns

- Where *p = 0,1,2,…,P*

LBP in face recognition

- LBP can be used for face recognition: finding the correct match for a given face image in a database.



- The concatenated histogram is computed for every image in the database. Histograms of query images are computed and matched with the image in the database with the most similar histogram

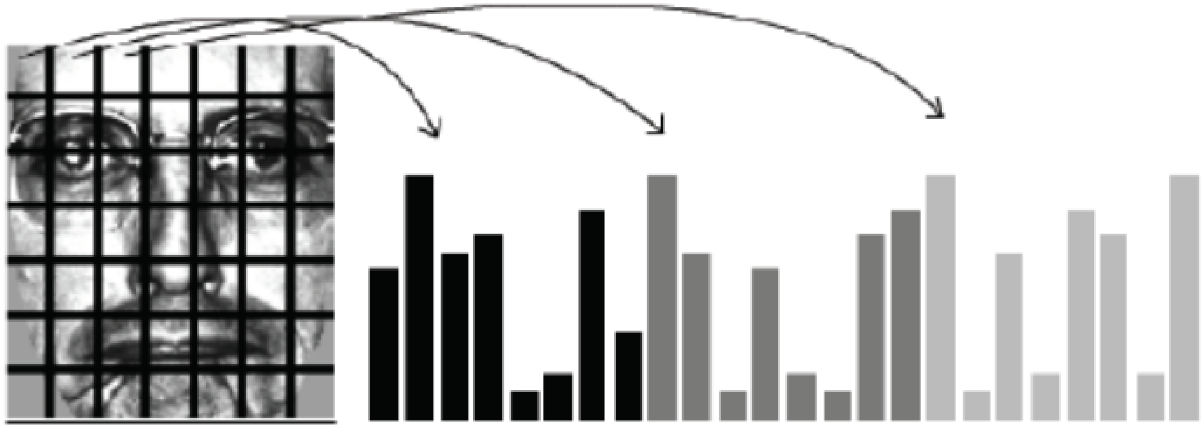
Local Binary Patterns with weights per subregion

- In face recognition, our retrieval performance improves if more emphases was placed on certain regions.

Computing of face object histogram

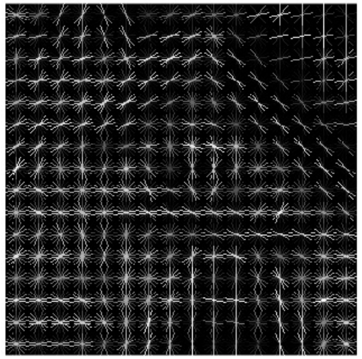
Weight matrix

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



Histogram of oriented gradients (HOG)

- Rationale: Objects within images can be often represented by directional distributions of intensity gradients (or edges)
- Thus, histograms of gradients are considered valuable descriptors of image fragments (e.g. key points)
- Histogram of oriented gradients (HOG) method computes a feature descriptor vector by computing histograms of gradient orientations



Histogram of Oriented Gradients (HOG)

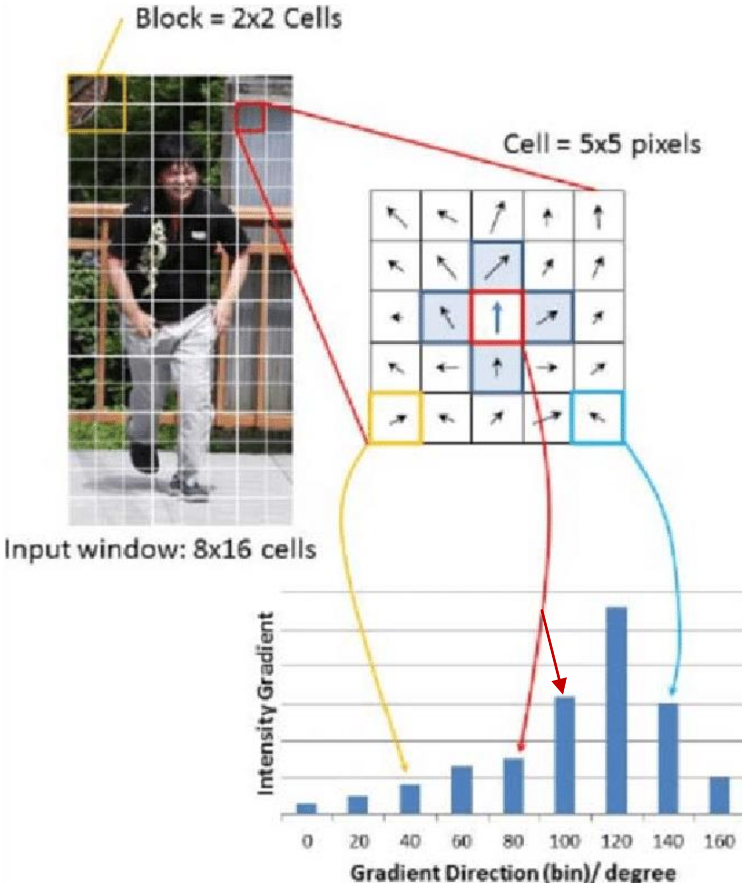
- Computing HOG histograms

1. Divide the image into cells and blocks. Each block should be composed of 2x2 cells.
2. Compute the gradient magnitude and direction in each cell

$$G_x = \frac{f(x+1,y)-f(x-1,y)}{2},\ G_y = \frac{f(x,y+1)-f(x,y-1)}{2}$$

$$|G| = \sqrt{G_x^2 + {G_y}^2},\ \theta = \tan^{-1}\frac{G_y}{G_x}$$

3. Compute the histogram of gradient directions and let the magnitude be the y-axis. Gradient directions (0°-180°) fall into one of 9 bins. If the value falls between the center of two bins you should interpolate (ex: 70° falls between 60 and 80 so the gradient magnitude should be divided equally between 70° and 80°.
4. Normalize computed histograms by the total value in each <u>*block*</u>
5. Like LBP, your feature vector that summarizes image content is the vector concatenating computed histograms.



SIFT Features

- Scale-Invariant Feature Transform (SIFT) was one of the most successful method for feature extraction from images.
- It improves over HOG by having a multiscale approach for “keypoint” detection
- Keypoints in SIFT image locations where significant activity occurs (Laplacian keypoints)
- The algorithm finds keypoints in a multiscale feature ignoring keypoints with significant edge components
- Next, histograms for edge directions in image patches around keypoints are created
- These concatenated histograms represent SIFT features
- In what follows, a general description of SIFT will be given

Smoothing prior to edge detection (review)

- Smoothing an image can improve the quality of edge detection.

a b
c d

FIGURE 10.16 (a) Original image of size 834×1114 pixels, with intensity values scaled to the range [0, 1]. (b) $|g_x|$, the component of the gradient in the x-direction, obtained using the Sobel mask in Fig. 10.14(f) to filter the image. (c) $|g_y|$, obtained using the mask in Fig. 10.14(g). (d) The gradient image, $|g_x| + |g_y|$.

Smoothing prior to edge detection (review)

FIGURE 10.18 Same sequence as in Fig. 10.16, but with the original image smoothed using a 5 × 5 averaging filter prior to edge detection.

SIFT Features – Keypoint detection

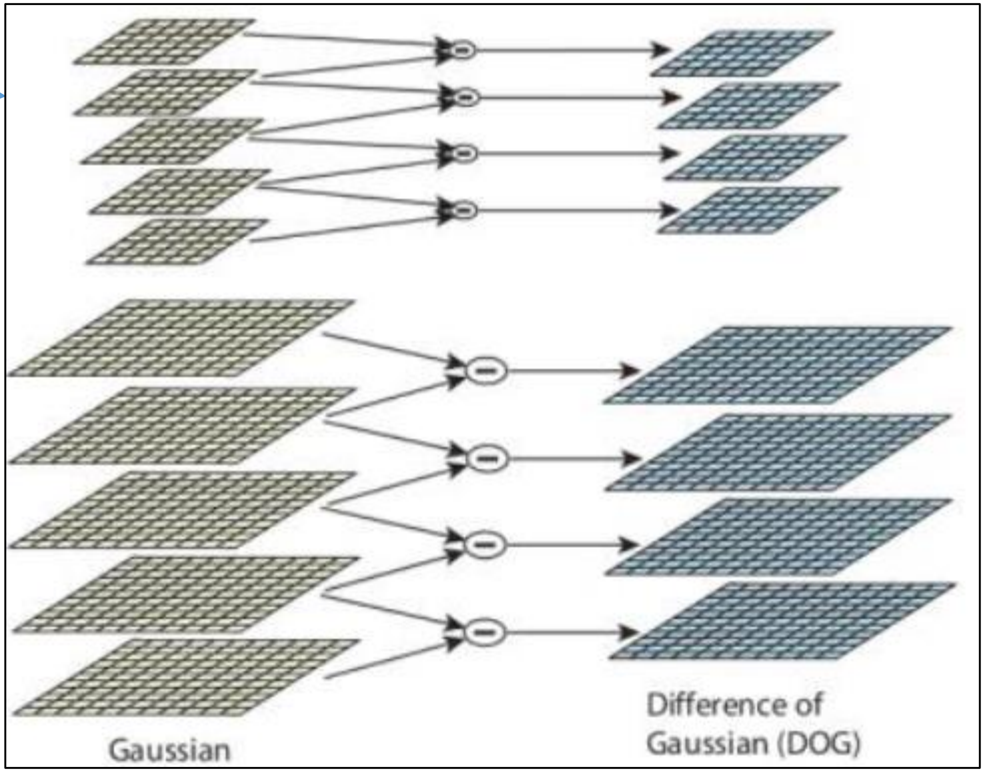
- The difference of Gaussian operator (DOG) is used to detect keypoints.
- DOG approximates the Laplacian of Gaussian (LOG) Gaussian smoothing followed by Laplacian.

Downsampling (reducing image size) the image by ½. Default is 4 octave levels.

difference of Gaussian blurring of an image with two different σ, let it be σ and kσ. Default is 5 levels (1σ, 2σ, 3σ, 4σ, 5σ)

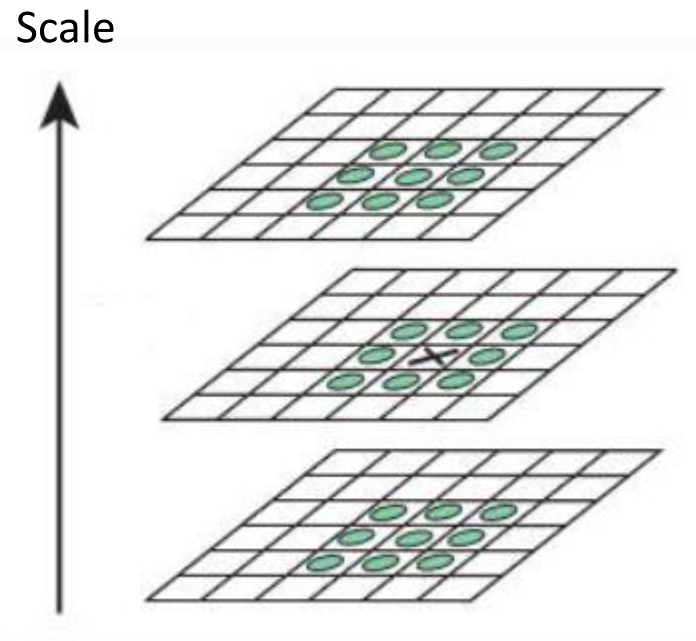
Octave 2

Octave 1



SIFT Features – Keypoint detection

- Next, in the DOG images, find points that are either higher or lower than all surrounding points in the same scale and across scales.
- Number of points to be compared with is 8+9+9=26



SIFT Features – Keypoint detection

- Similar to Laplacian, DOG will detect many edges. In SIFT we remove the keypoints that represent edges by using the Hessian matrix.

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

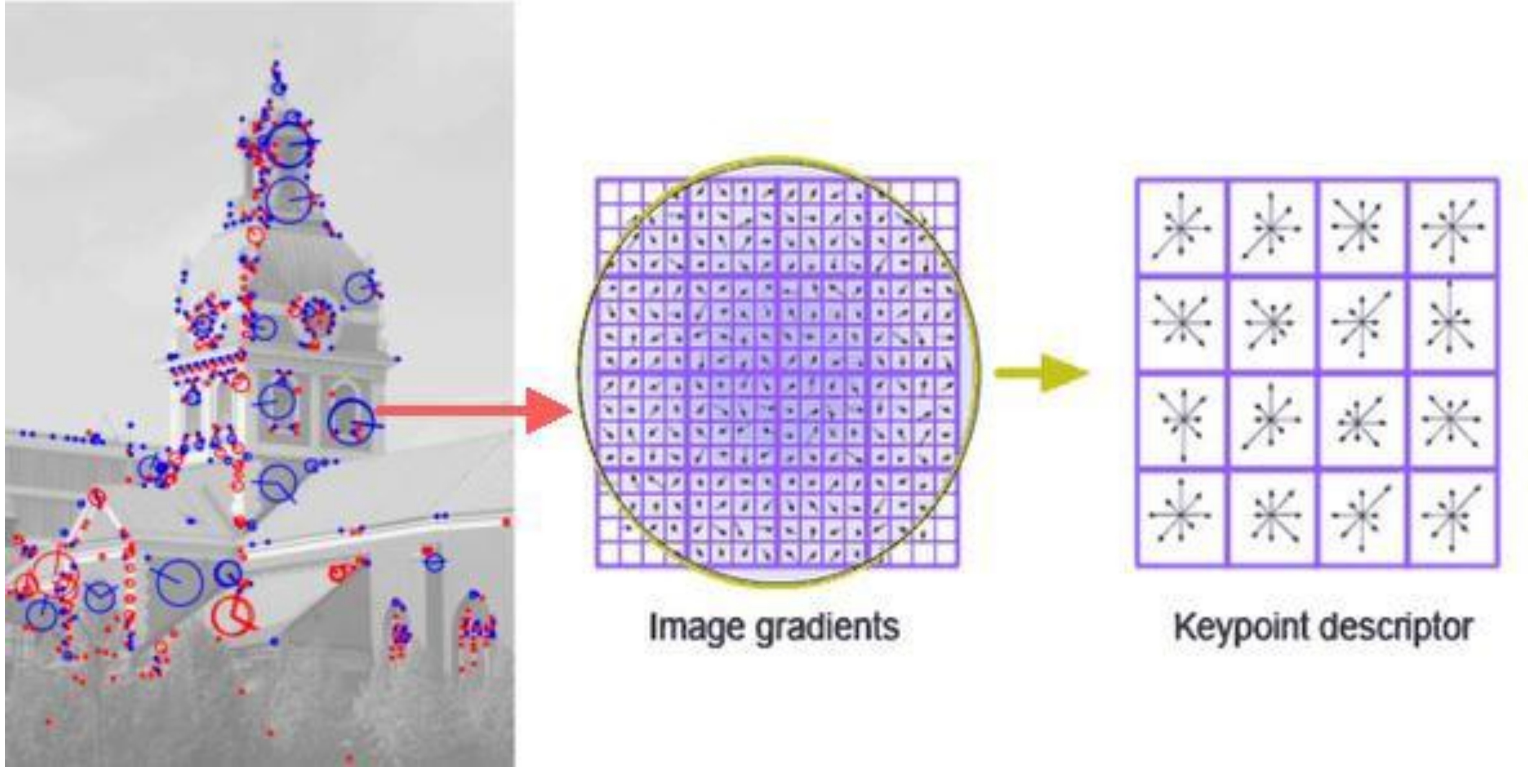
- Compute the eigenvalues of the Hessian matrix. Edges occur when $\lambda_1 > \lambda_2$
- In SIFT, Keypoints are removed when $\frac{\lambda_1}{\lambda_2} > 10$

SIFT Features

- Scale-Invariant Feature Transform (SIFT) was one of the most successful method for feature extraction from images.
- It improves over HOG by having a multiscale approach for “keypoint” detection
- Keypoints in SIFT image locations where significant activity occurs (Laplacian keypoints)
- The algorithm finds keypoints in a multiscale feature ignoring keypoints with significant edge components
- **Next, histograms for edge directions in image patches around keypoints are created**
- **These concatenated histograms represent SIFT features**

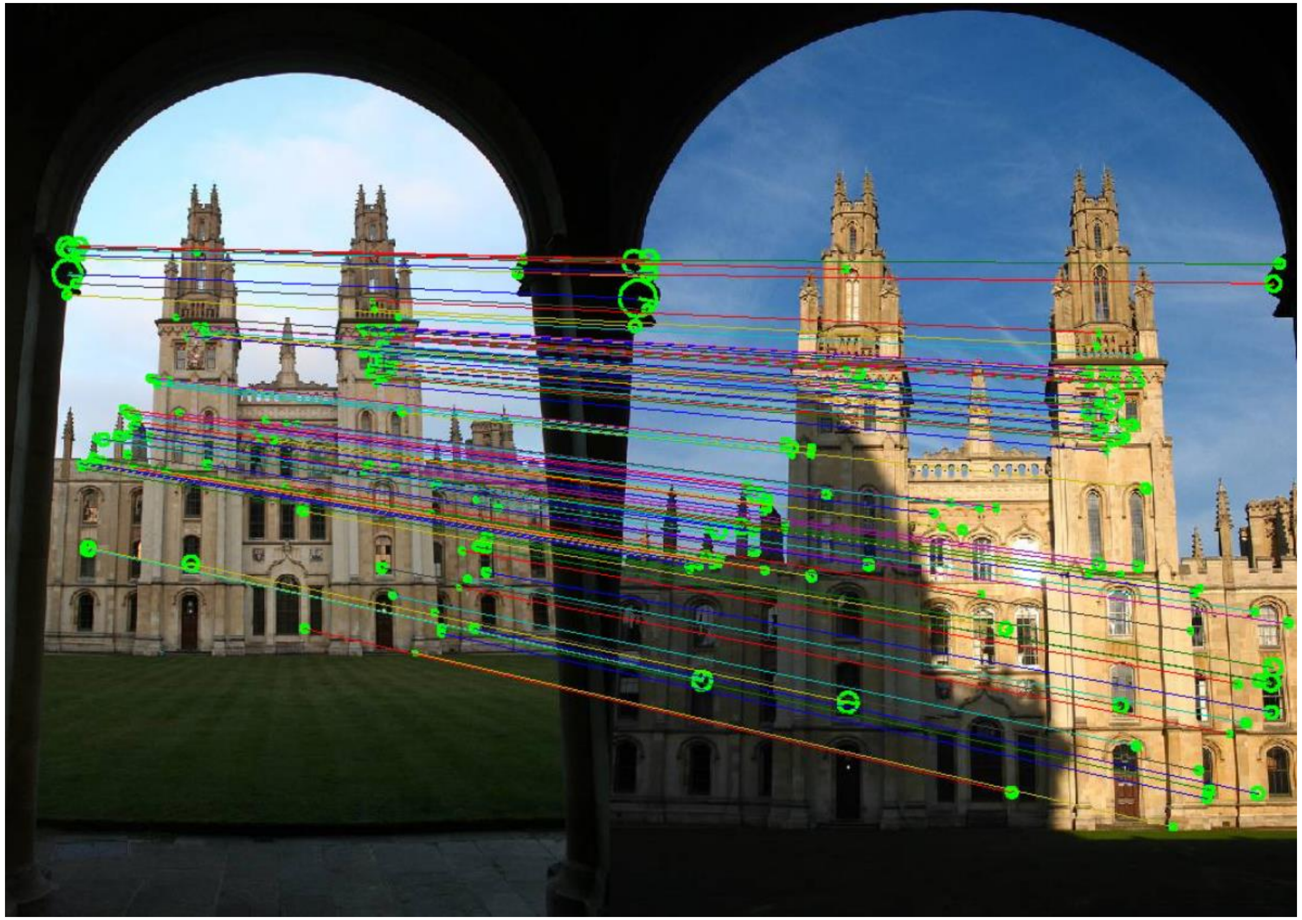
SIFT Features – Feature Vector

- A 16x16 neighborhood around the keypoint is taken (in Octave 1, smaller regions for smaller scales). 16 8-bin histograms of gradients are created. The length of the feature vector is $16 \times 8 = 128$

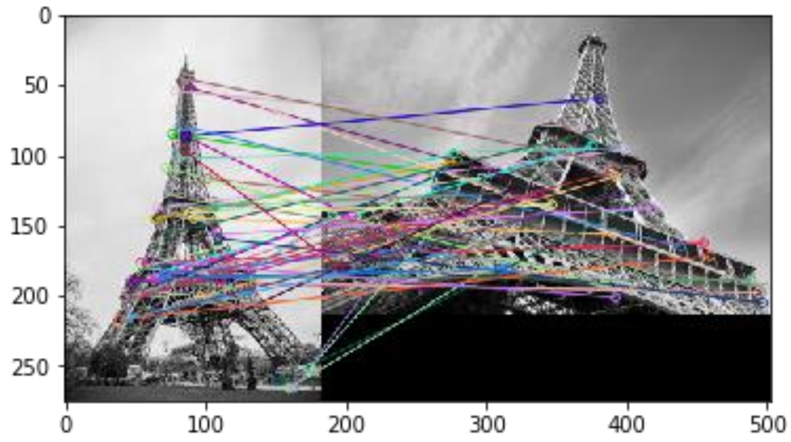


Lowe, David G. "Distinctive image features from scale-invariant keypoints." *International journal of computer vision* 60, no. 2 (2004): 91-110.

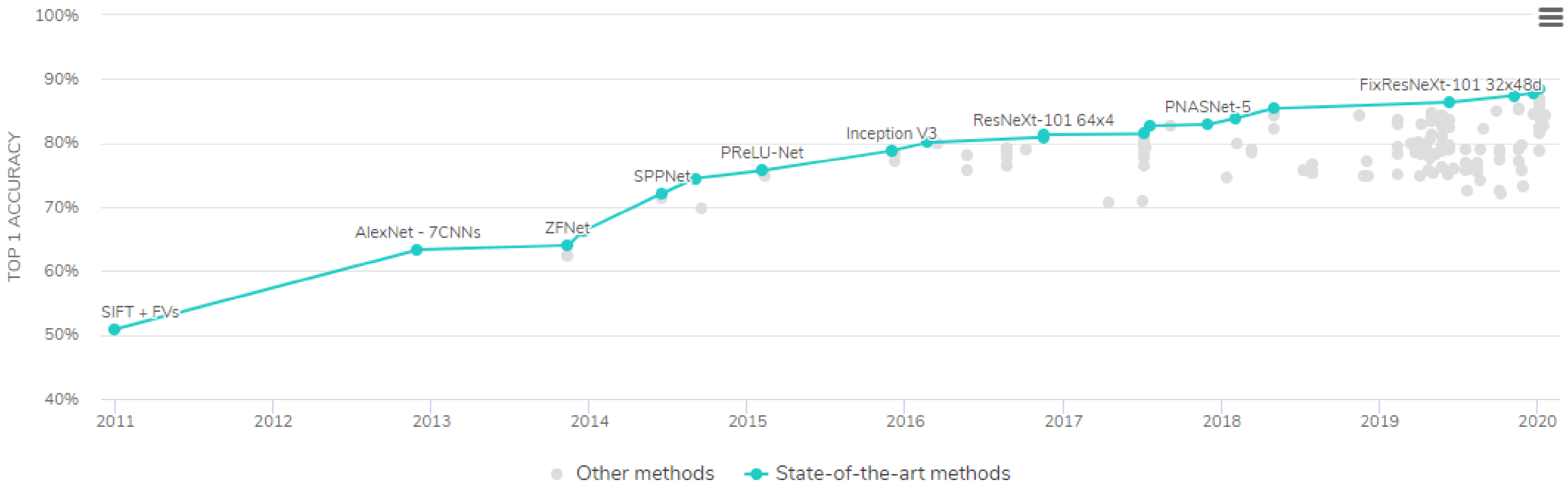
SIFT Feature Matching



SIFT Feature Matching



SIFT in Image Classification



Classification accuracy on ImageNet dataset. Largest dataset of annotated images. Now it has tens of millions of cleanly sorted images.

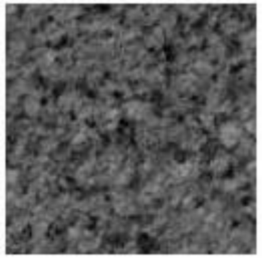
Feature Engineering

- LBP, HOG, and SIFT are examples of algorithms based on direct feature engineering (i.e., hand crafted features)
- The information extracted from images is a result of algorithms specified in detail by the developer
- Pros: we know exactly what is going on, can see why they succeed and why they fair → Interpretability
- Cons: Modest performance levels, difficult design process

Image retrieval algorithms

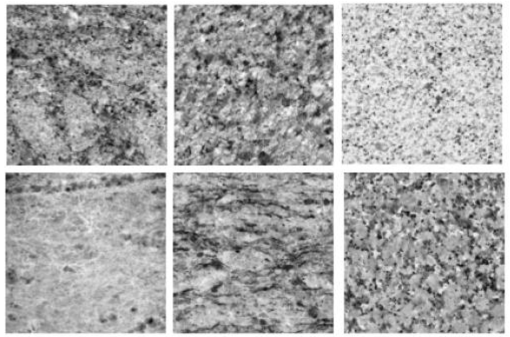
- We have described three approaches for feature extraction from images
  - Local Binary Patterns (LBP)
  - Histogram of Oriented Gradients (HOG)
  - Scale Invariant Feature Transform (SIFT)
- These methods rely on concatenating multiple image histograms that summarize the change in image intensities within the spatial region they represent
- Retrieval occurs by comparing the distance between a given image and database images.

Image retrieval algorithms



Query image

1 2 3



4 5 6

Database

Assume that the distance between query and other images is

D = [0.96 0.28 0.72 0.56 0.86 0.09]

The query image matches image 6. The second-best match is image 2

How can we measure the quality of retrieval?

- Retrieval uses the same measure used for classification
- Total accuracy is the most widely used measure:

$$\frac{\text{number of correct matches retrieved}}{\text{the total number of correct matches}}$$

- These measures don’t consider the order where the correct match is.
- Definition assumes one match per query. It is not good for use when we have multiple matches per query.

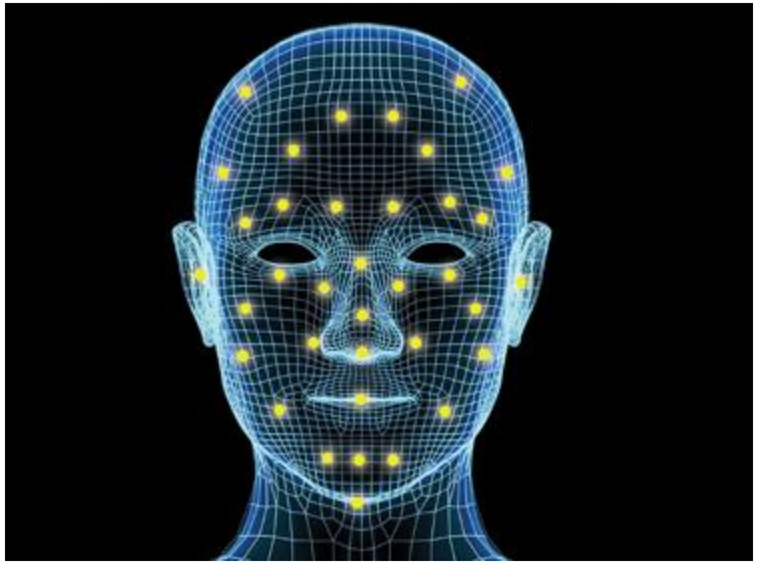
How can we measure the quality of retrieval?

- Example: Assume we have a database of 10,000 face images where each face image matches a single image in the database. Using the first retrieved image (i.e. minimum distance) we get the correct match in 9,000 cases.

  Accuracy is 90%

- Assume that the system retrieves the closest 10 images and again gets the correct match in 9,000 cases

  Accuracy is still 90%



How can we measure the quality of retrieval?

- **Precision**

# of correct matches retrieved / # of total images retrieved

- Assume that a system retrieves the closest two images to a given face image. Only one of these images is correct
- Precision = $\frac{1}{2} = 0.5$

How can we measure the quality of retrieval?

- **Recall**

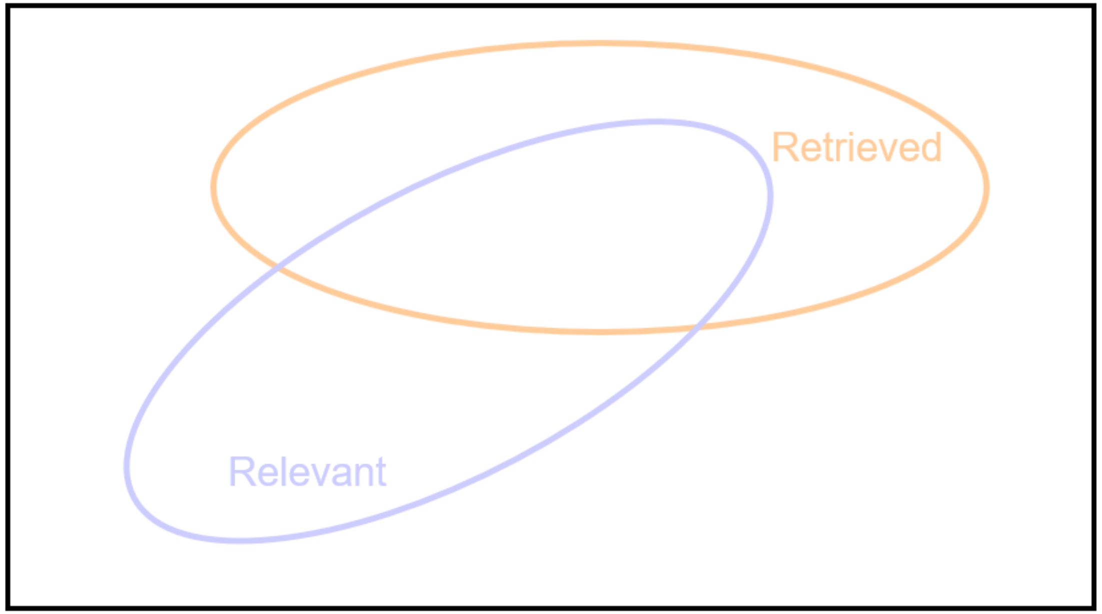
# of correct matches retrieved / # of total correct matches in the dataset

- Example: Assume that a system retrieves 1 image that correctly match a given face image. The database has two correct matches for each query image. What is recall?

Recall = $\frac{1}{2}$ = 0.5

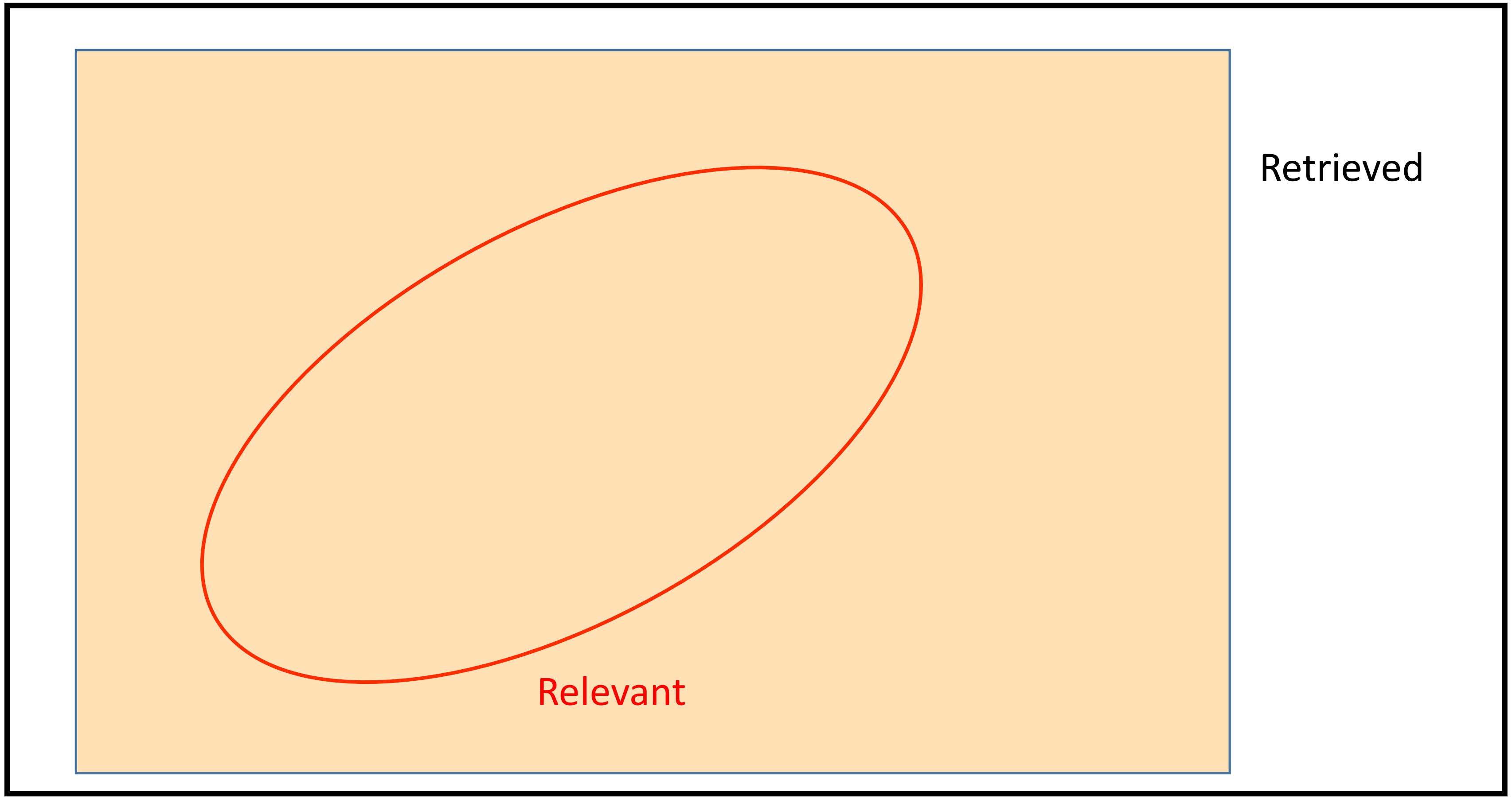
How can we measure the quality of retrieval?

- We like high precision and high recalls
- Get as much correct matches, while at the same time minimize wrong retrievals.



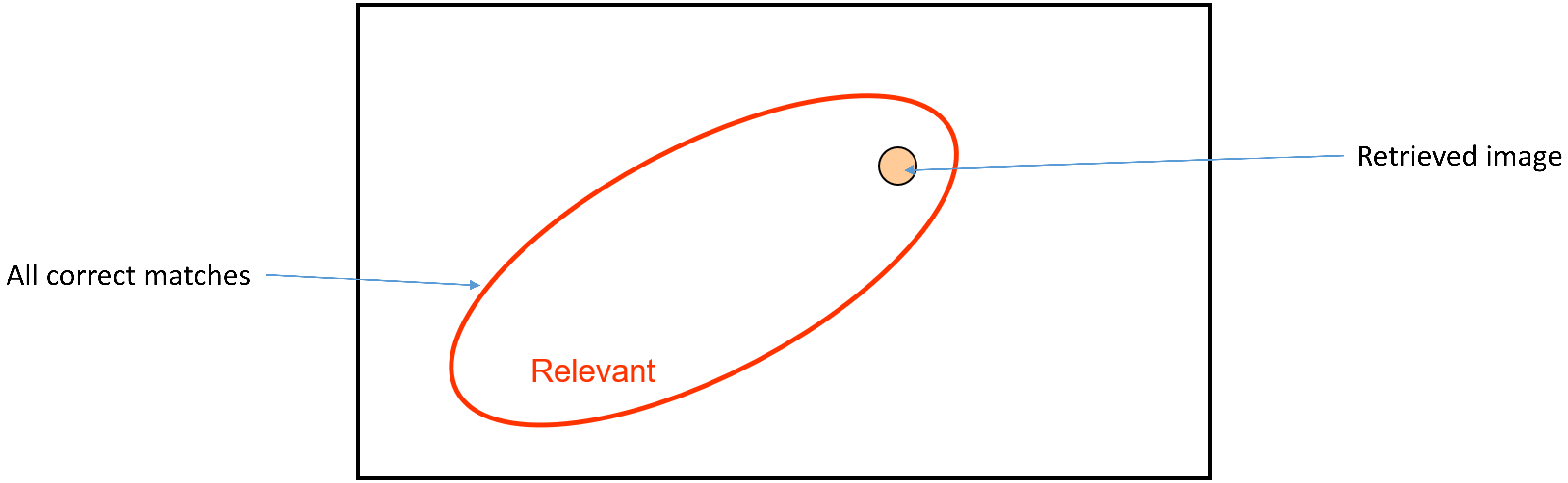
How can we measure the quality of retrieval?

High recall (recall=1), but low precision



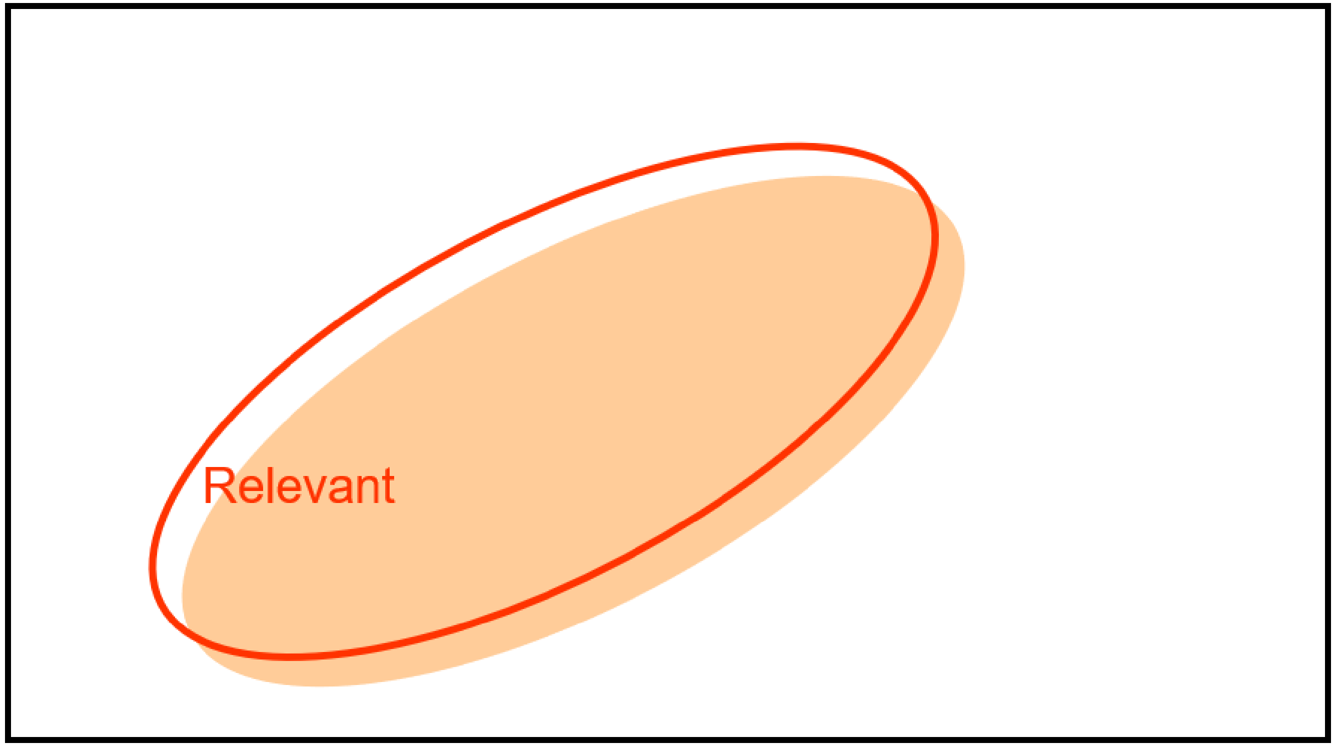
How can we measure the quality of retrieval?

Very high precision, very low recall



How can we measure the quality of retrieval?

High precision, high recall (at last!)

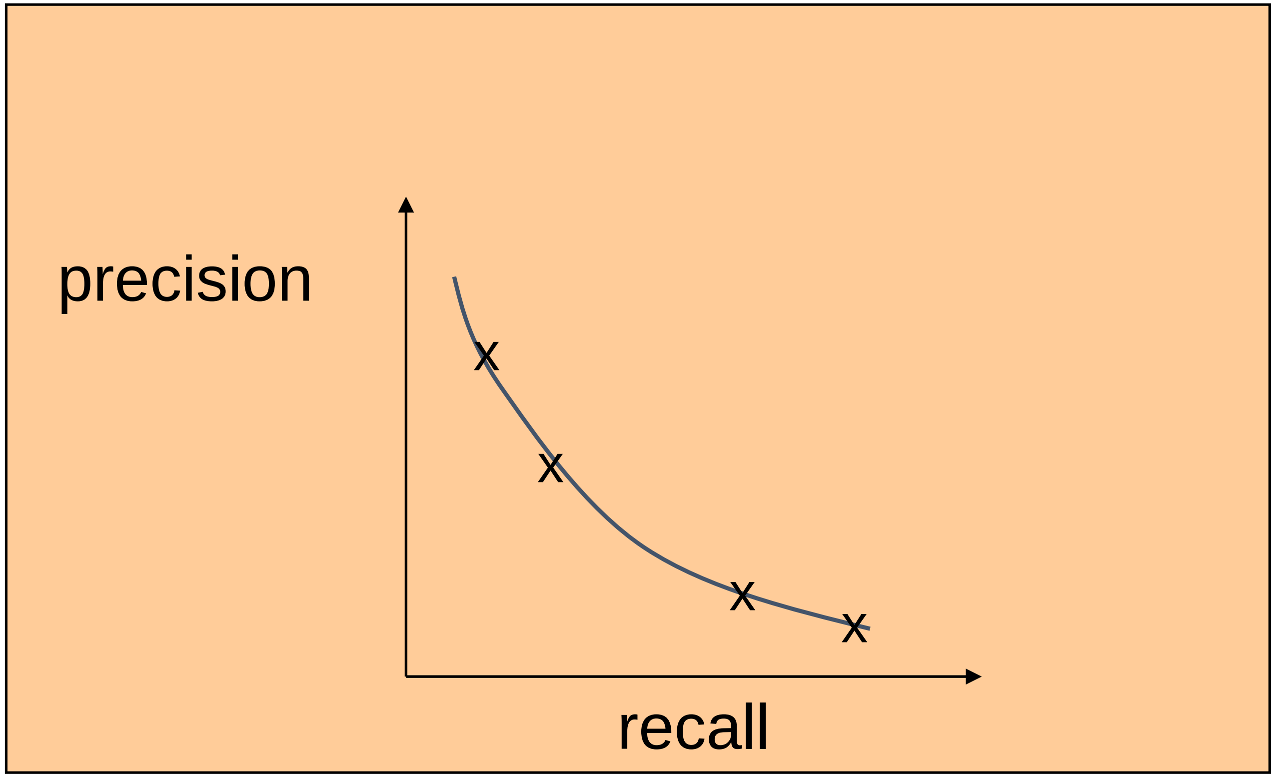


How can we measure the quality of retrieval?

- A precise classifier is selective
- A classifier with high recall is inclusive
- Precision is computed as a function of the number of retrieved images: Compute precision when the system is retrieving the closest match p(1), precision when the system is retrieving the 10 closest matches p(10)...

Precision/Recall Curves

- There is a tradeoff between precision and recall
- So measure precision at different levels of recall



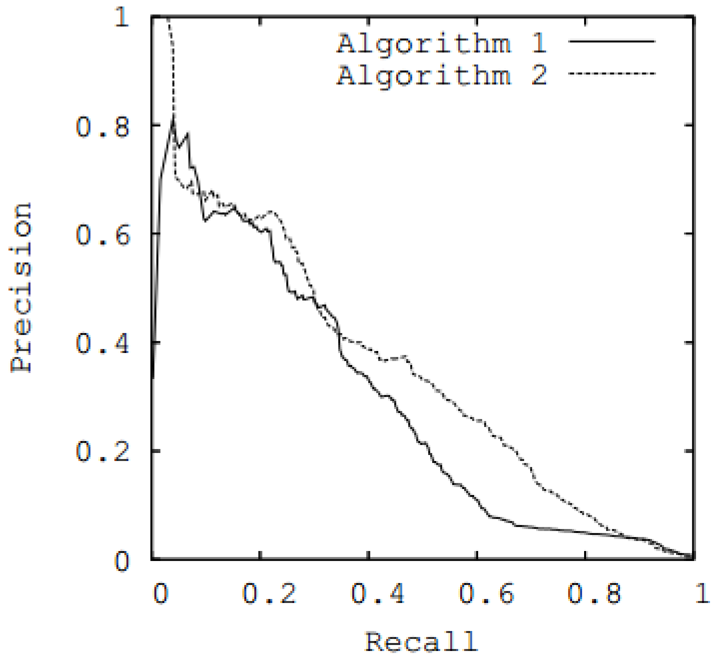
How can we measure the quality of retrieval?

- Average Precision (AP) $= \frac{\sum_{k=1}^{N} p(k)}{N}$

 where k is the number of retrieved images, N is the number of recall values tested and is equal to the number of retrieved images.

- To consider the performance of retrieval systems, we test them with multiple queries and compute the mean of average precision values.
- Mean Average Precision (MAP) $= \frac{\sum_{k=1}^{Q} AP}{Q}$, where Q is the number of queries

Precision/Recall Curves



- Algorithm 1 works better because it has a higher area under its curve
- Mean average Precision (MAP) estimates the area under precision/recall curve.

The F-1 Score

- Since we like to get both precision and recall high, why not combine those two measures?
- The F_1 score is given by

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Precision and recall in binary classification

- Its is useful to define these terms in terms of true positive, true negative, false positive, and false negative

$$\textbf{Precision} = \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$