Features Extraction

- The full information regarding an image is in its spatial brightness distribution. However, in most cases, comparing objects based on their brightness distribution is not useful
  - Image of a scene strongly depends on pose and lighting
  - Even slight changes in location, size and orientation change the scene
  - Moreover, the comparison of two large arrays of numbers representing the grey-levels of the two scenes to be compared has a high computational complexity.
- For these reasons, detecting similar scenes by comparing grey-level images often fail

Features Extraction

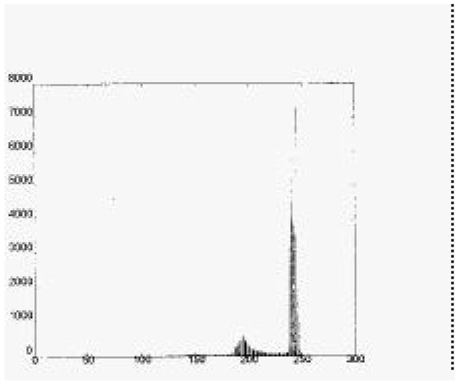
- As far as the shape of an object is concerned, the brightness information is indirect. To draw some conclusions about the identity of an object, it is necessary to extract from the brightness information some features that will describe and represent the object.
- A feature is a unique characteristic or measure of a picture, or some part of it.
- Among the common features are
  - brightness/color features
  - geometrical features
  - transform features

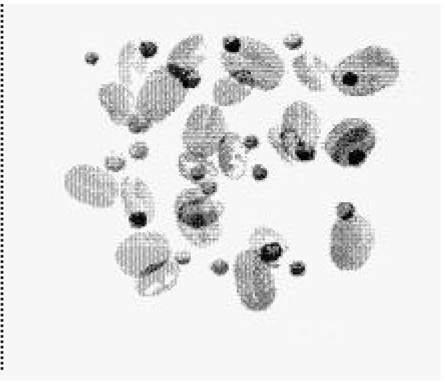
Feature Selection

- For classification of shapes, features are used rather than the set of points constituting the shapes. This is because
  - Features are smaller in number, and enable a smaller computational complexity
  - Features are less sensitive to insignificant changes
  - It is easier to estimate the probability functions of the features for the different shapes, than the estimation of these functions for the image points.
- However, a reliable classification based on features is possible only if no significant information concerning the shapes is lost in deriving the features.

Brightness/Color Features

- When the different objects in the image differ in their brightness/color, brightness/color may be used as a representative feature. For example, when apples are to be sorted based on their color, a search for green parts in the image identifies green apples, while a search for red areas identifies red apples.
- The brightness histogram specifies the brightness distribution in an image or part of it. Other statistical measures can be calculated from the histogram, e.g., the average and variance of the brightness

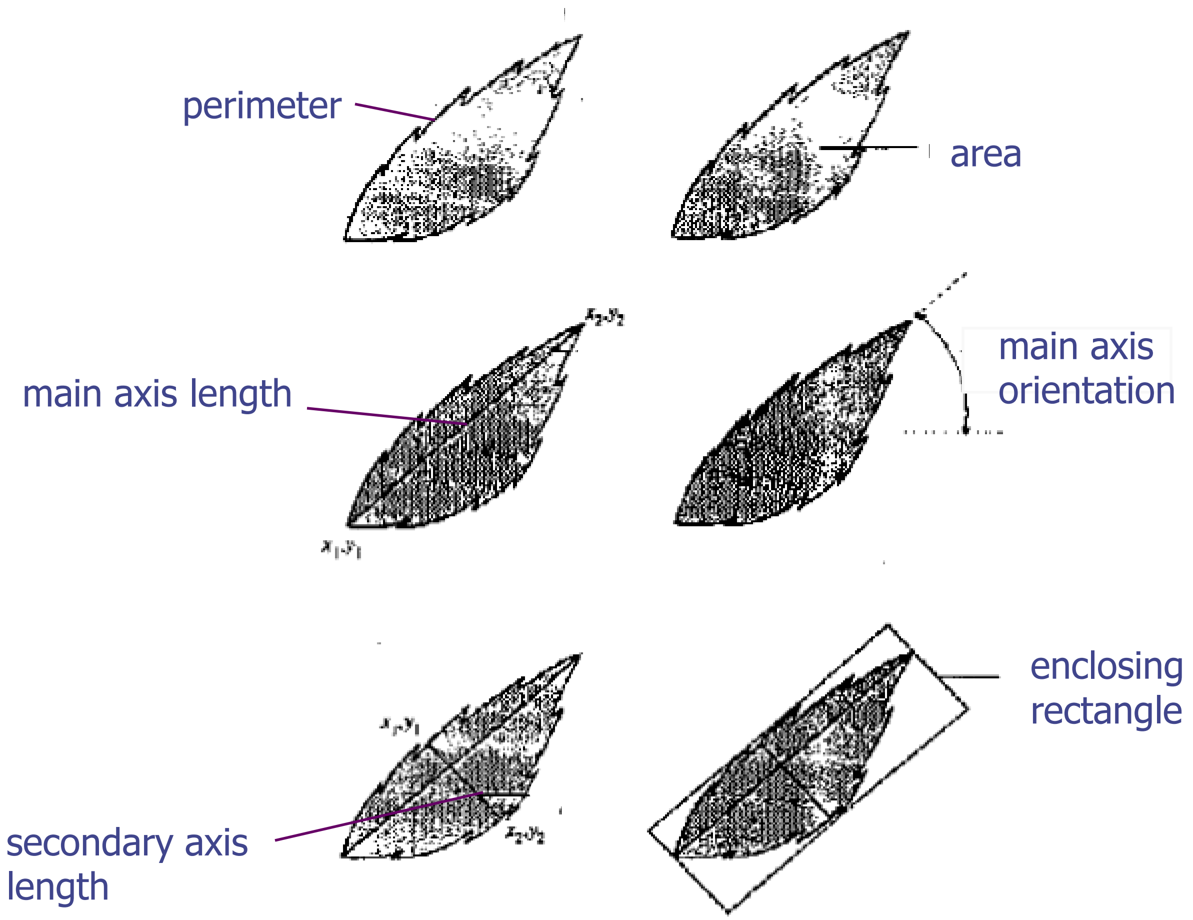




Shape (geometric) Features

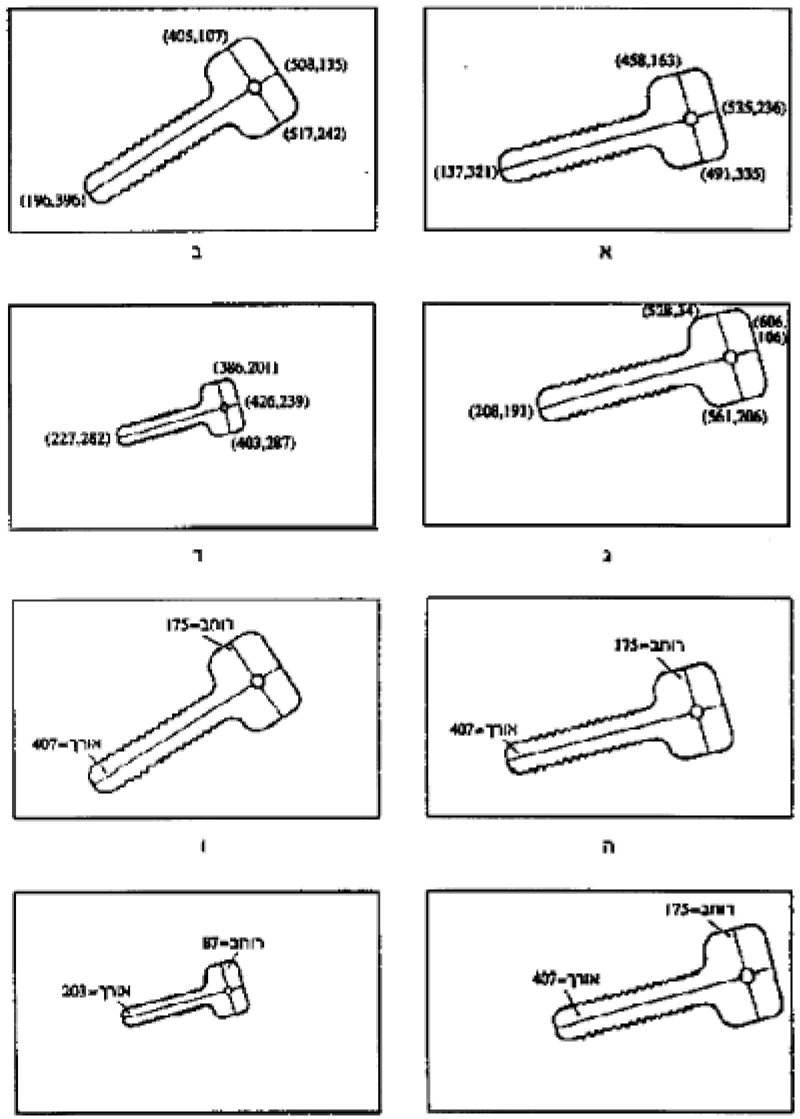
- When the different objects in an image do not differ in their brightness or color, but in their shape, shape features, relating to the geometry of the objects, have to be used. Among them:
  - perimeter
  - area
  - length and orientation of main axis
  - length and orientation of secondary axis
  - the perimeter of the enclosing rectangle

Shape Features



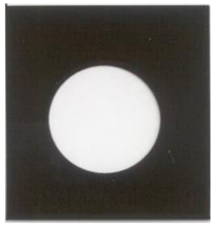
Independence to Insignificant Changes

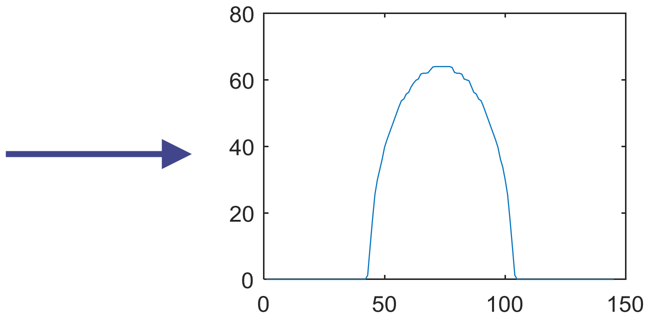
- Shape features should be chosen such that insignificant changes in the shape do not affect them much. Such a feature in the case of the shape of a key is the ratio between the lengths of the two axis, as well as the angle between them. Both remain unchanged under shifting , rotation and scaling



Projection Features

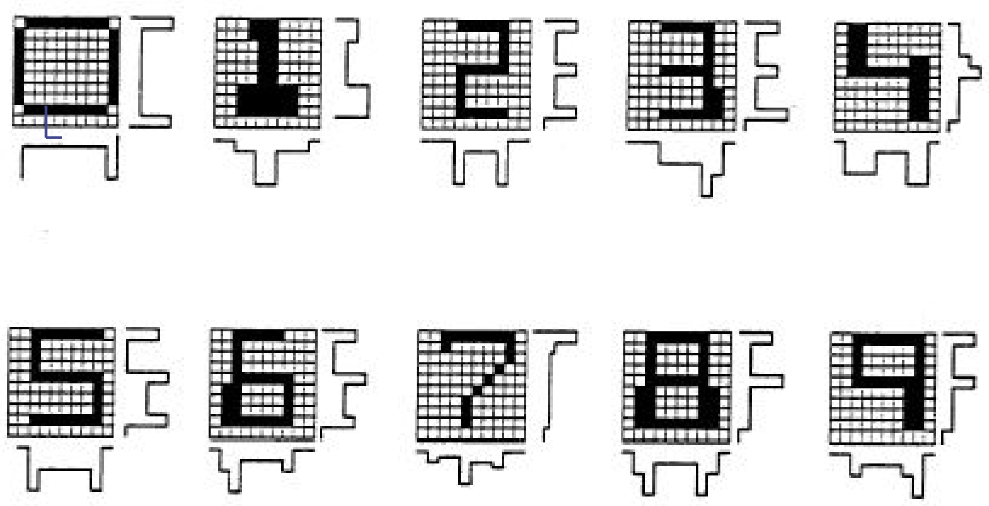
- Objects in an image may be characterized by their projections: horizontal, vertical or at any orientation.
- The calculation of the horizontal and vertical projections is simple, and involves the sum of the pixels values along the lines/columns
- Calculating the projection along any angle θ can be done by generating parallel lines at this angle, and summing the pixels values of the object along this lines. It can also be done using the Radon transform
- When the object is in a binary image, the projection is equivalent to the pixel count
- For example, the projection of a circle in an image is identical for all possible angles





Automatic reading of characters from projections

- The alphabet characters and digits evolve through the centuries. They were not designed to be the ideal characters for the human reader, and definitely were not designed for easy identification by a computer.
- To ease the process of automatic reading by a computer, a special style of characters have been designed. The digits of this set are shown in the figure



Transform Features

- Both shape and brightness/color features are derived from the image itself. To derive the transform features, the image should first undergo a mapping from the image domain to a different one.
- Some of the more common transform features are
  - Parametric features (from the Hough transform)
  - Area features (from the moment transform)
  - Spatial frequency features (from Fourier transform)

Parametric Features - the Hough Transform

- Parametric features can be derived for those shapes in the image, which can be approximated by parametric shapes - shapes with a mathematical formula to specify their boundaries, e.g., a straight line, a circle, an ellipse, etc.
- After computing the parameters of the shape under consideration, they are used as features for the shape. They represent the shape geometry well, since given the parameters, the equation for the shape provides the set of points residing on it

The Hough Transform for Straight Lines Detection

- For image points *(x_i,y_i)*, which reside on a common straight line, only their coordinates are known, and not the parameters of the line they reside on. In the line equation

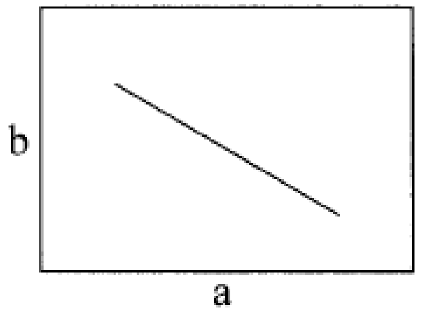
  $$y_i = ax_i + b$$

  the unknowns are the parameters a and b
- If a is considered the independent variable and b the dependent one, then the equation relating them is

  $$b = -x_i a + y_i$$

The Hough Transform

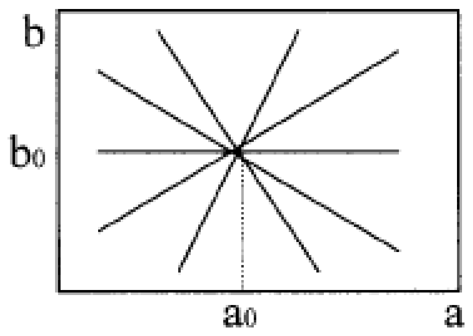
- The line $b=-ax_i+y_i$ obtained by changing a, is the mapping of image point (x_i,y_i). It represents all the possible straight lines which may pass through that point in the image
- The Hough transform then maps each point to a line in the transform plane - the (a, b) plane:



The Hough transform of a point

The Hough Transform

- When a number of points reside on the same line in an image, they are mapped into straight lines in the transform plane, which intersect at a common point
- The values of the parameters at the intersection point, (a_0, b_0), are the parameters of the line on which the points reside.



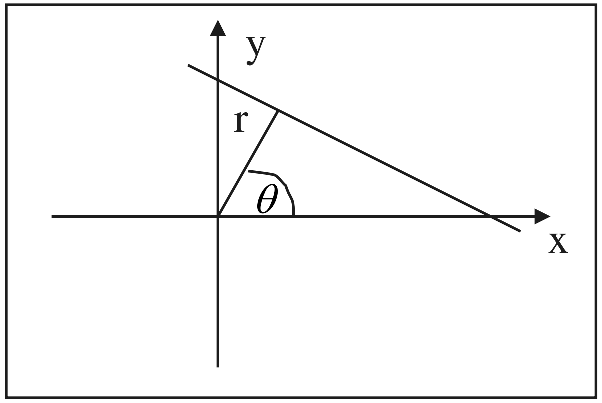
The Hough transform of points on a line

The Hough Transform

- The problem of finding the common line parameters is then replaced by the problem of finding an intersecting point.
- This, in turn, is replaced by the problem of finding a maximum value in the transform (parameter) plane in the following way:
  - a discrete parameter plane is used, in which each entry corresponds to a pair of parameter values (a,b)
  - the transform of each image point into a line in the parameter plane is derived by adding 1 to the array entries along that line
  - the intersecting point at the parameter plane can then be identified by its significant value - each image point (along the image line) adds 1 to its value. An image line of 40 points, generates in the parameter plane an intersection point with a value of 40 - a maximum value

The Hough Transform

- The transformation using the (slope, intercept) parameters is problematic, since in the Cartesian representation of a line the range of the slope is infinite (a vertical line has a slope of infinity).
- An alternative representation of a line, a polar representation, is used instead. The parameters of a straight line are then (r,θ) as in the figure. r is the distance of the line from the origin, and θ is the angle of this distance with the horizontal axis



- The straight line equation is then

$$r = x\cos\theta + y\sin\theta$$

The Hough Transform

- Each image point *(x_i, y_i)* is then transformed into a sinusoidal curve

  $$r = x_i \cos\theta + y_i \sin\theta$$

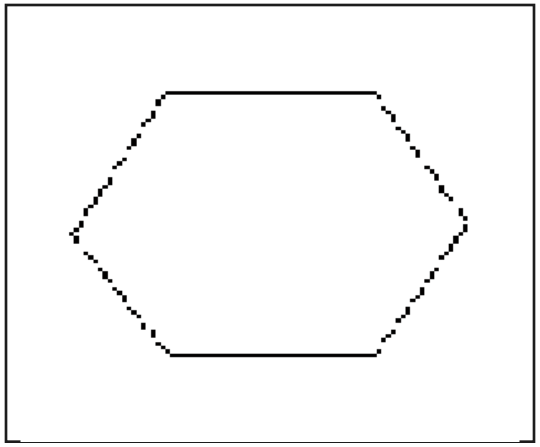
  where θ is changed in the range [0, 180], and for each value of θ the value of r is calculated
- The value at the (r, θ) entry in the transform array is then incremented by 1 by each image point on that line
- Sinusoidal curves corresponding to points on the same image line intersect at a common point (r_0, θ_0) in the transform plane. The parameters r_0 and θ_0 at the intersection point are the parameters of the common line in the image

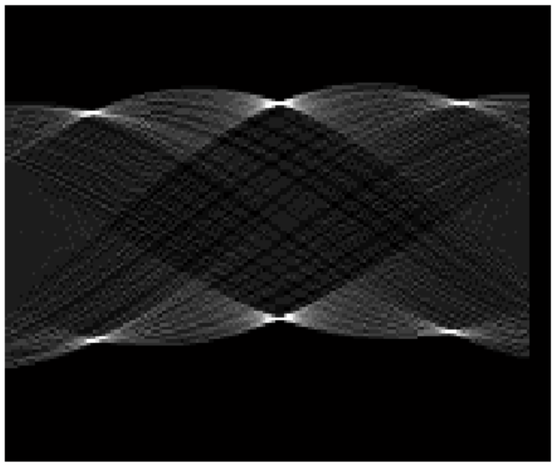
Conclusions from Parameter Plane

- When the shape in the image is composed of a few straight lines, the number of peaks in the parameter plane is equal to the number of straight lines.
- The parameters of the peaks serve as features for the shape
- A few conclusions may be drawn from the parameter plane:
  - parallel line have peaks with the same horizontal location
  - when two lines have an angle of θ between them, their corresponding peaks are θ apart in the horizontal direction
  - the longer the line in the image, the larger is its peak in the parameter plane
  - When the shape in the image is rotated, the horizontal distance between the intersection (maximum) points remain unchanged

The Hough Transform - An Example

- Given is an image of an hexagon and its transform. Six intersection points, with a significant value (bright), can be identified in the parameter plane, corresponding to the six lines of the hexagon.


An hexagon image

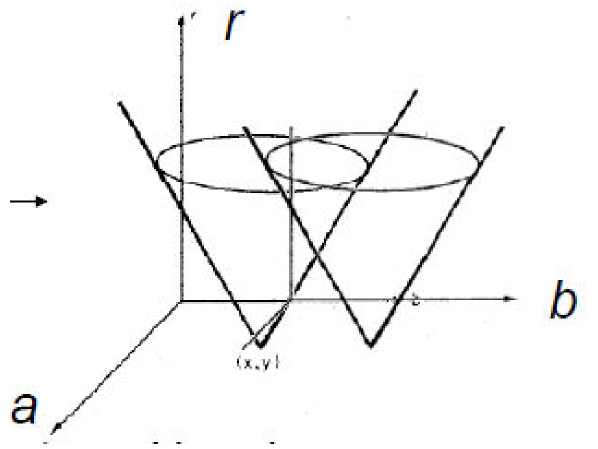

Its hough transform (parameter plane)

The Hough Transform for Circle Detection

- The detection of a circle in an image involves the derivation of its 3 parameters, the radius r and coordinates of its center (a,b)

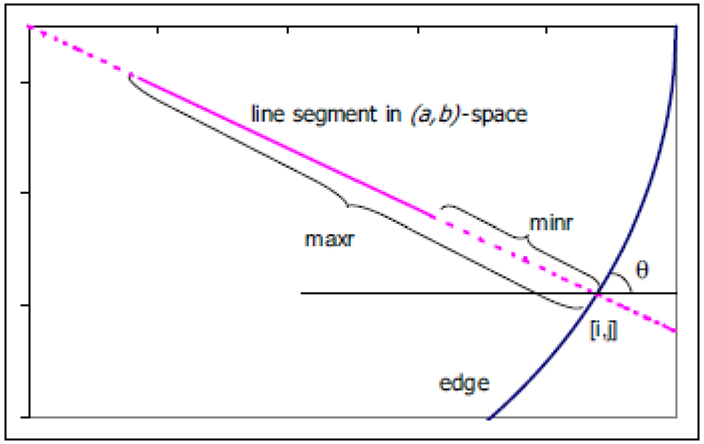
$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- A direct search for the 3 parameters using the Hough transform involves the accumulation of contributions in 3-dimensional array.
- For a fixed radius the point (x_i , y_i) is mapped into a circle in (a, b) plan. Changing the radius, a cone is obtained as the transform of each point
- The 3 circle parameters are obtained from the intersection point of the cones → of the points on the circle



The Hough Transform for Circle Detection

- This solution is too complicated, both in calculations and in storage requirements
- Alternatively, the search for the circle parameters can be split into two parts
  - finding the center of the circle
  - calculating the radius
- The coordinates of the center of the circle are found based on the observation that perpendiculars to edge points on the circle cross in the center of the circle



The Hough Transform for Circle Detection

- The voting in the (a, b) plan is done by

$$\left.\begin{aligned} a &= r\sin\theta \\ b &= r\cos\theta \end{aligned}\right\} \text{ where } r \in (minr, maxr)$$

$$A(i \pm a, j \pm b) \leftarrow A(i \pm a, j \pm b) + E(i, j)$$

  where

  - A is the (a, b) plan

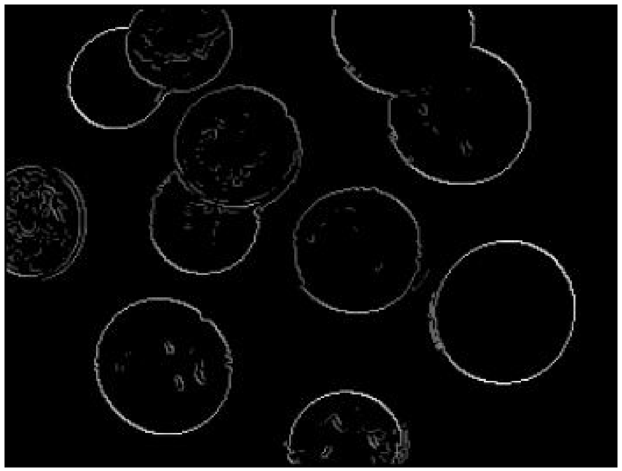
  - (i, j) are the coordinates of the pixel considered

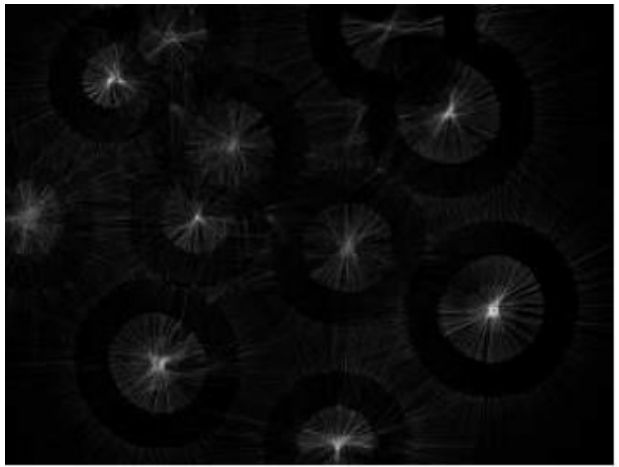
  - E(i,j) is the amplitude of the gradient at the pixel

- The voting of each edge point is along a straight line in the (a,b) plan, as in the figure
- The line segments of different pixels on the circle intersect at (a, b) of the circle they reside on

The Hough Transform for Circle Detection

- The coordinates of the center of the circle are obtained from the intersection point in the (a, b) plan of the perpendiculars at the edge points





The Hough Transform for Circle Detection

- After the center points of the circles in the image are found, the radii of the circles have to be measured
- This is done by accumulating in a one-dimensional r array. All pixels on a certain circle contribute to the circle radius they reside on.
- Multiple peaks are obtained in the r array, one for each circle in the image. The size of the peaks is proportional to the number of pixels residing on each circle

The Features of Moments

- Moments are shape features derived from transforming all the shape points, not only its edge points.
- The moment definition is

$$m_{ij} = \sum_{x}\sum_{y} g(x,y)x^i y^j, \quad i,j = 0,1,2\ldots$$

where *(x,y)* are the coordinates of the shape point, and g(x,y) is its brightness in the image. The summation is over all *x* and *y* values of points within the shape boundaries.

- From the definition above, it is clear that m_{00} (the moment of order zero) is the sum of the brightness values of all the shape points, or in other words, it is proportional to the average brightness of the shape.

The Moments Features

- When the image is binary, *$g(x,y)=1$* for points within the shape, and *$g(x,y)=0$* for points of the background. Then, the moment m_{ij} is the sum of $x^i y^j$ of all the points within the shape boundaries.
- For a binary image, m_{00} is the area of the shape (the number of points belonging to the shape), m_{10} is the sum of the x coordinates of the shape points, and m_{01} is the sum of the y coordinates of the shape points (proportional to the coordinates of the center of gravity of the shape – $(\bar{x}, \bar{y})$)

The Moments Features

- Since the moments are calculated relative to a certain coordinate system, the values of the moments change when the shape is shifted within the image.
- To derive moment features which are invariant to shift in the image (and to the location of the shape), a centralized set of moments is calculated:

$$\mu_{ij} = \sum_x \sum_y g(x,y)(x-\bar{x})^i (y-\bar{y})^j, \quad i,j = 0,1,2\ldots$$

when $(\bar{x}, \bar{y})$ is the center point of the shape:

$$\bar{x} = \frac{m_{10}}{m_{00}}, \qquad \bar{y} = \frac{m_{01}}{m_{00}}$$

- $\bar{x}$ is the average value of the x coordinates of the shape points, and $\bar{y}$ is the average y value.

The Moments Features

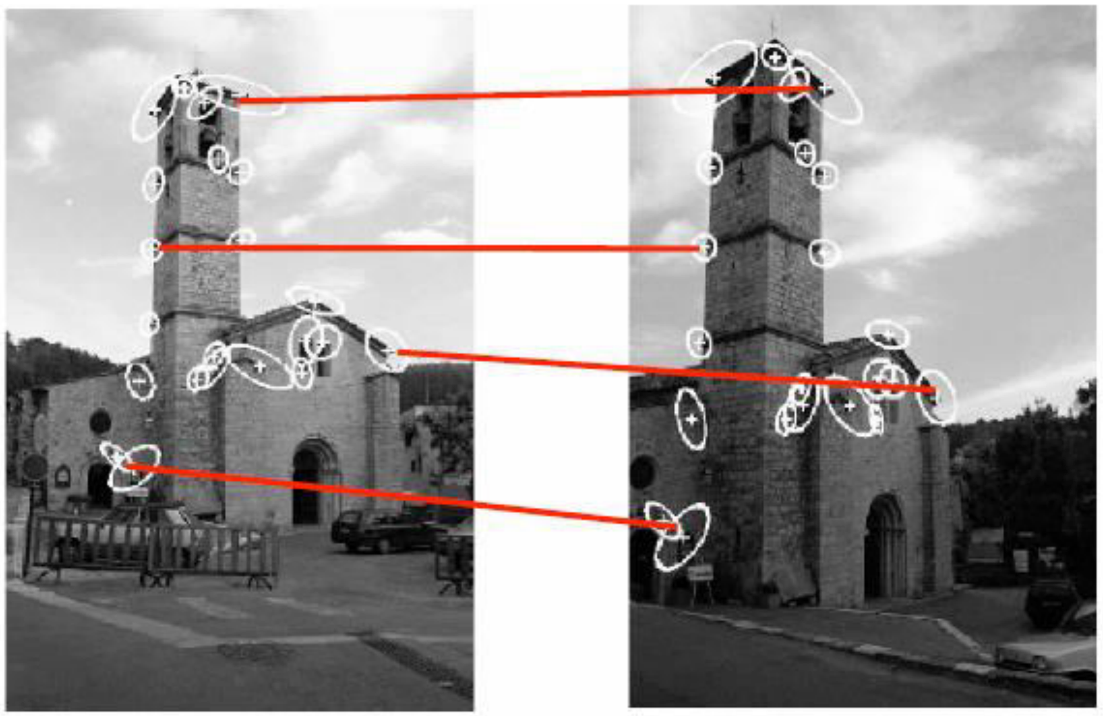
- Since the centralized moments are calculated relative to the centre of the shape, they do not change when the shape is shifted
- The centralized moments are shift invariant, but they change when the shape is rescaled (change of size) or rotated
- A set of moments which is also invariant to scaling is:

$$\eta_{ij} = \frac{\mu_{ij}}{(\mu_{00})^k}, \qquad k = \left\lfloor \frac{(i+j)}{2} \right\rfloor + 1$$

- A certain combination of the η_{ij} moments is invariant to all deformations: shift, scale and rotation
- The representation of shapes by their (infinite) set of moments is accurate, but impractical. Commonly, shapes are represented by a finite subset of their moments

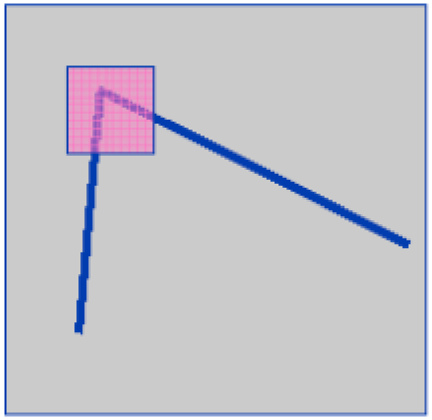
Corner Features

- One good set of features to match is that of corners, which is particularly applicable to images of man-made objects, like buildings



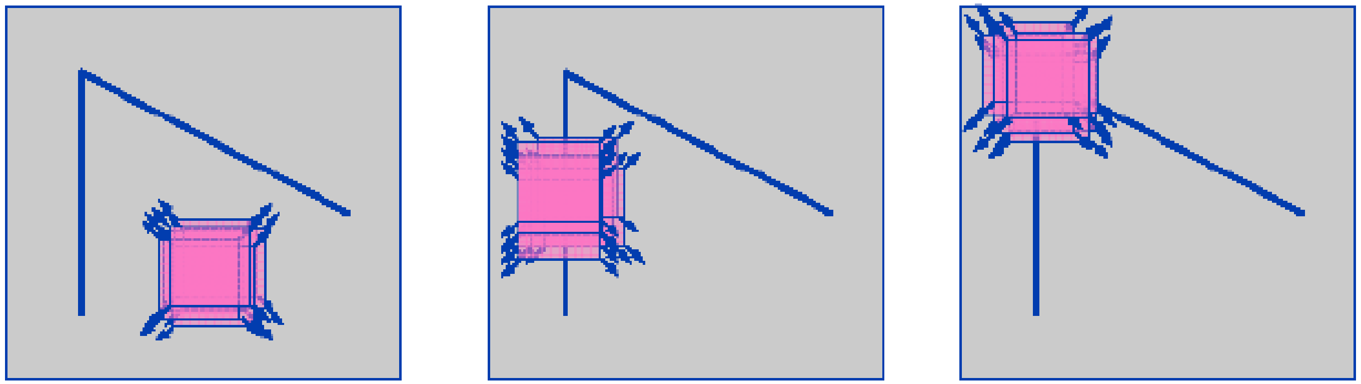
Corner Features

- Intuitively, corners are junctions of contours, presenting large variations of brightness around each corner in all directions
- Corners are generally stable over changes in viewpoint
- They can be easily recognized by looking at intensity values within a small window
- Shifting the window in any direction should yield a large change in appearance



Corner Features

- Shifting the window over an image, three options are possible:
  - The window is over a patch of a constant intensity
  - The window is over an edge
  - The window is over a corner



"flat" region:
no change in
all directions

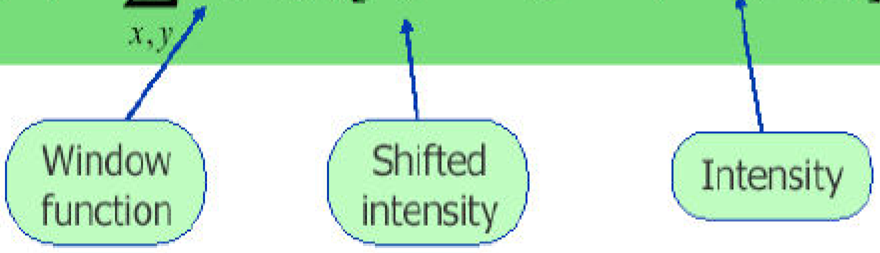
"edge":
no change along
the edge direction

"corner":
significant change
in all directions

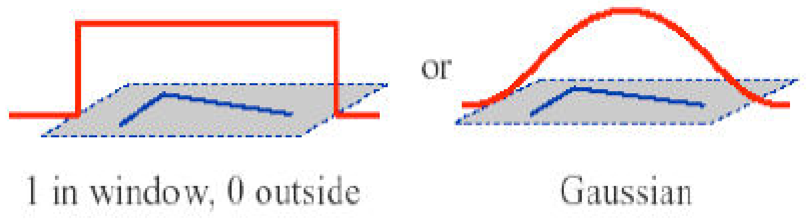
Harris Corner Detection

- Change of intensity for a shift [u, v] :

$$E(u,v)=\sum_{x,y} w(x,y)\left[I(x+u,y+v)-I(x,y)\right]^2$$



Window function $w(x,y) =$



- For a window covering a nearly constant patch, the change of intensity is minor, and E is near 0.
- Looking for corners, large values of E are sought

Harris Corner Detection

- Using Taylor series for a 2D function with small changes [u, v] around point (x, y):

$$f(x+u,y+v) = f(x,y) + uf_x(x,y) + vf_y(x,y) +$$

First partial derivatives

$$\frac{1}{2!}\left[u^2 f_{xx}(x,y) + uvf_{xy}x,y + v^2 f_{yy}(x,y)\right] +$$

Second partial derivatives

$$\frac{1}{3!}\left[u^3 f_{xxx}(x,y) + u^2 v f_{xxy}(x,y) + uv^2 f_{xyy}(x,y) + v^3 f_{yyy}(x,y)\right]$$

Third partial derivatives

$+ \ldots$ (Higher order terms)

First order approx

$$f(x+u,y+v) \approx f(x,y) + uf_x(x,y) + vf_y(x,y)$$

Harris Corner Detection

- And an approximation to E[u, v] around point (x, y) for small values of u and v:

$$\sum [I(x+u, y+v) - I(x,y)]^2$$

$$\approx \sum [I(x,y) + uI_x + vI_y - I(x,y)]^2$$ First order approx

$$= \sum u^2 I_x^2 + 2uvI_xI_y + v^2 I_y^2$$

$$= \sum \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$ Rewrite as matrix equation

$$= \begin{bmatrix} u & v \end{bmatrix} \left(\sum \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}$$

Harris Corner Detection

- And an approximation to E[u, v] around point (x, y):

$$E(u,v) \cong [u,v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

where M is a 2×2 matrix computed from image derivatives:

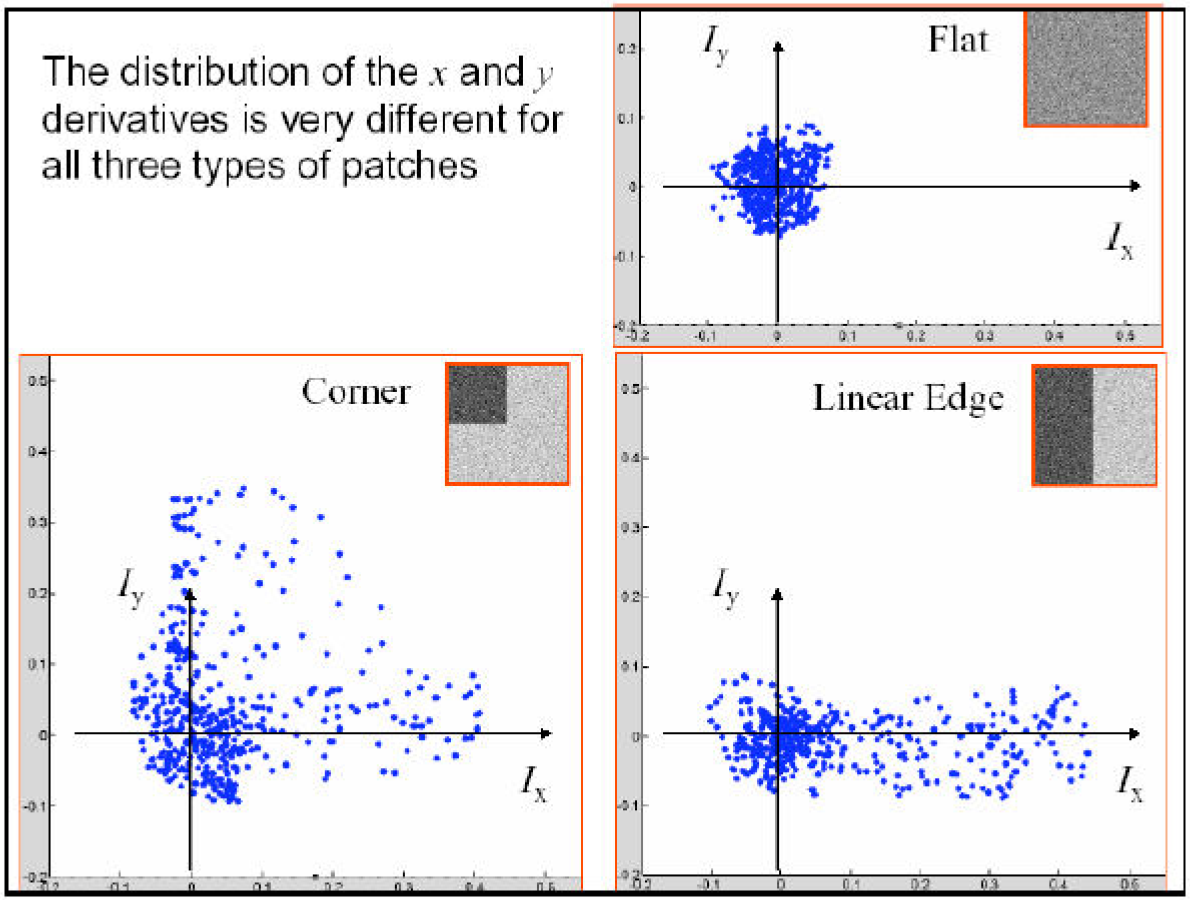
$$M = \underbrace{\sum_{x,y} w(x,y)}_{\text{Windowing function - computing a weighted sum (simplest case, w=1)}} \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}}_{\text{Note: these are just products of components of the gradient, Ix, Iy}}$$

Windowing function - computing a weighted sum (simplest case, w=1)

Note: these are just products of components of the gradient, Ix, Iy

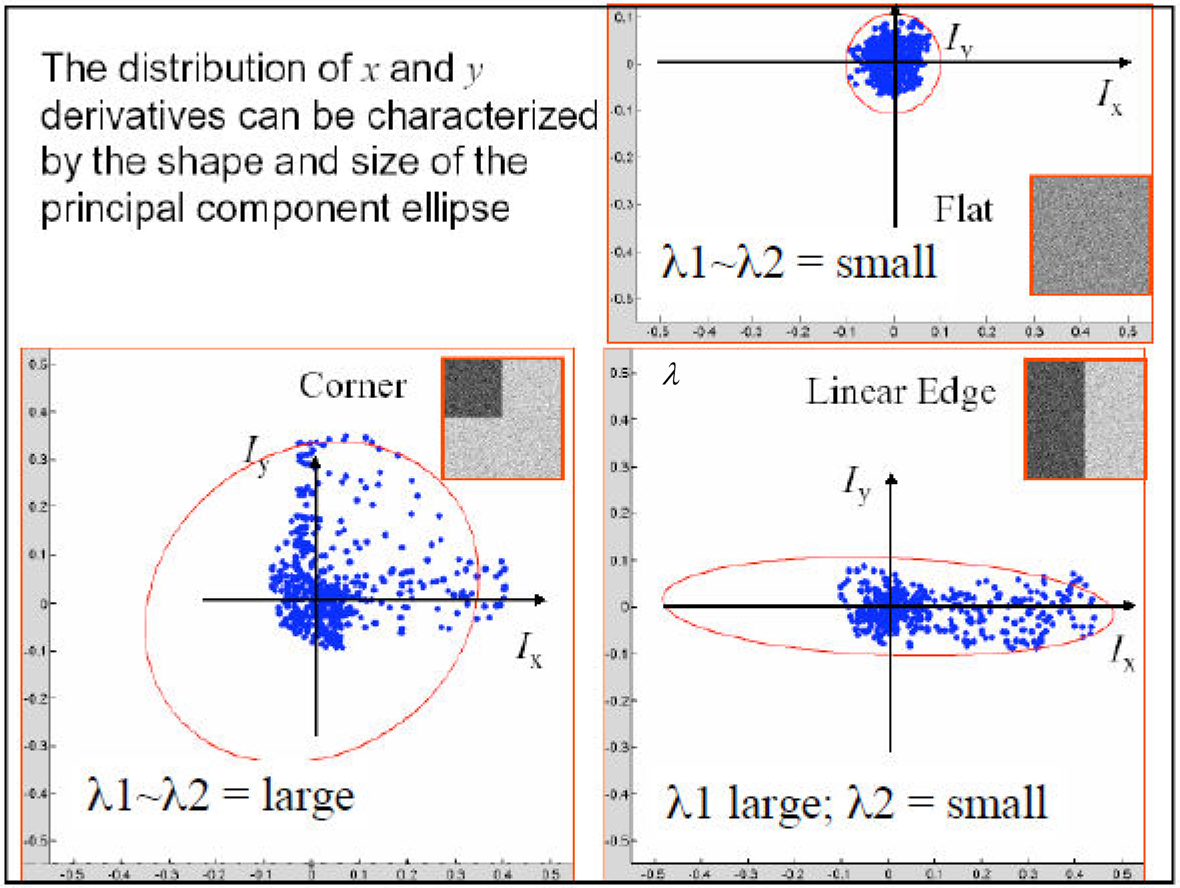
Harris Corner Detection

- Derivatives I_x and I_y as 2D points:



Harris Corner Detection

- Detecting corners by the eigenvalues of matrix M – λ_1 and λ_1

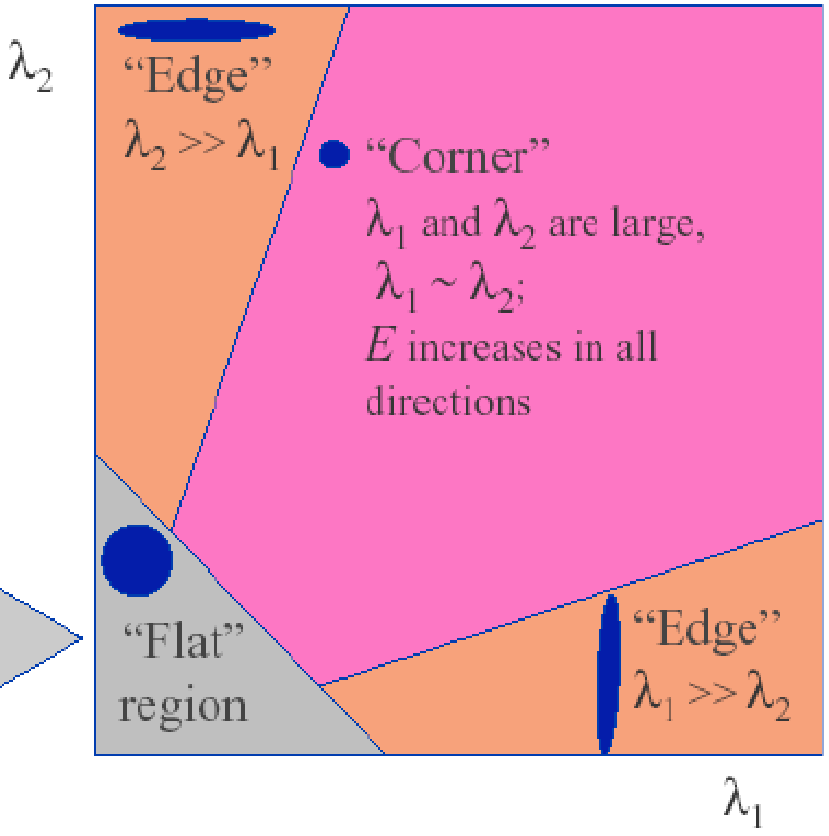


Harris Corner Detection

- Detecting corners by the eigenvalues of matrix M – λ_1 and λ_1

Classification of image points using eigenvalues of M:

λ_1 and λ_2 are small; E is almost constant in all directions



Harris Corner Detection

- A single measure for detecting corners is R :

$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

$$\begin{aligned} \det M &= \lambda_1 \lambda_2 \\ \operatorname{trace} M &= \lambda_1 + \lambda_2 \end{aligned}$$

(k is an empirically determined constant; $k = 0.04 - 0.06$)

- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region

Harris Corner Detection Algorithm

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

2. Compute products of derivatives at every pixel

$$I_{x2} = I_x . I_x \quad I_{y2} = I_y . I_y \quad I_{xy} = I_x . I_y$$

3. Compute the sums of the products of derivatives at each pixel

$$S_{x2} = G_{\sigma'} * I_{x2} \quad S_{y2} = G_{\sigma'} * I_{y2} \quad S_{xy} = G_{\sigma'} * I_{xy}$$

4. Define at each pixel (x, y) the matrix

$$H(x, y) = \begin{bmatrix} S_{x2}(x, y) & S_{xy}(x, y) \\ S_{xy}(x, y) & S_{y2}(x, y) \end{bmatrix}$$

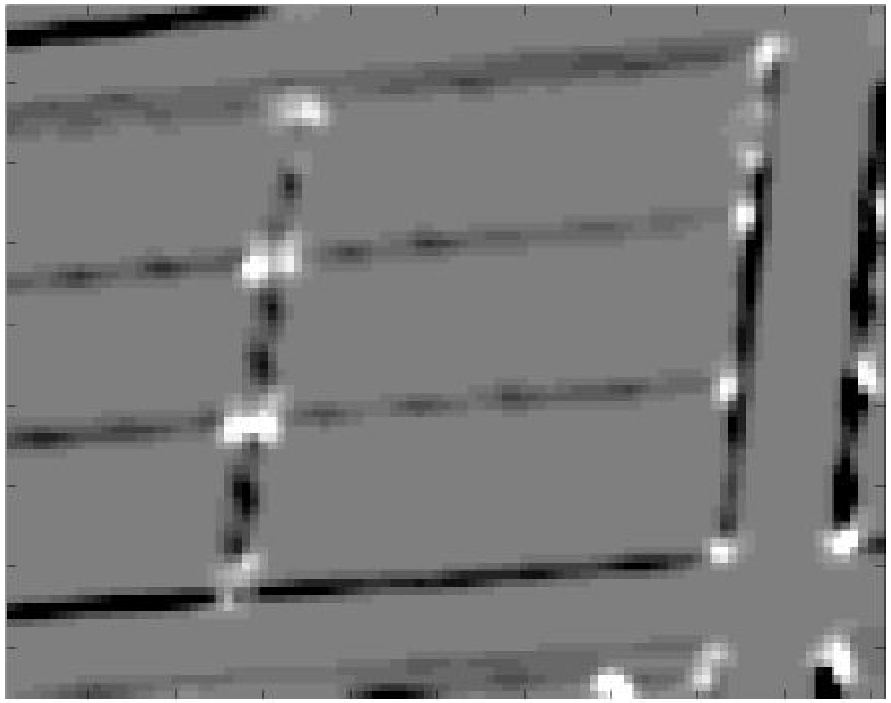
5. Compute the response of the detector at each pixel

$$R = Det(H) - k(Trace(H))^2$$

6. Threshold on value of R. Compute nonmax suppression.

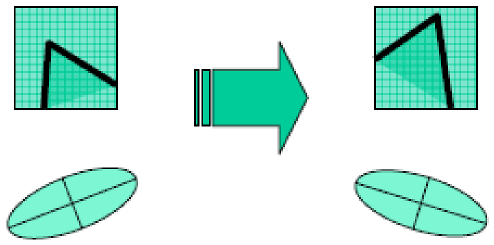
Corner Detection Example

- The following is an example of corner detection. I_x and I_y are computed using Sobel operators. The window is Gaussian with σ=1. A corner is detected for R > 10000



Corner Detection

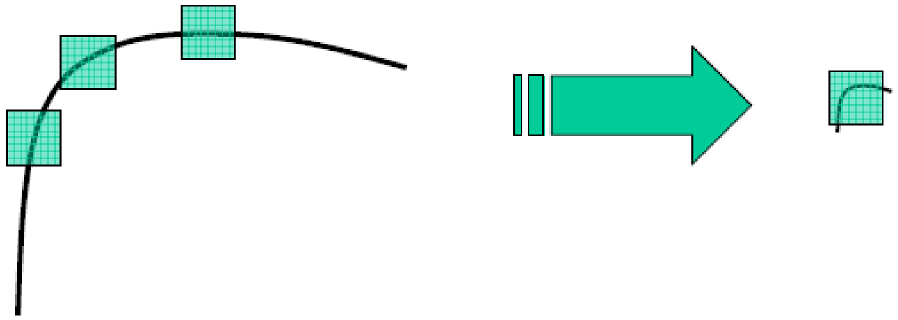
- Harris corner features are invariant to shift and rotation



Ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner response R is invariant to image rotation

but not to scale



All points will be classified as edges

Corner !

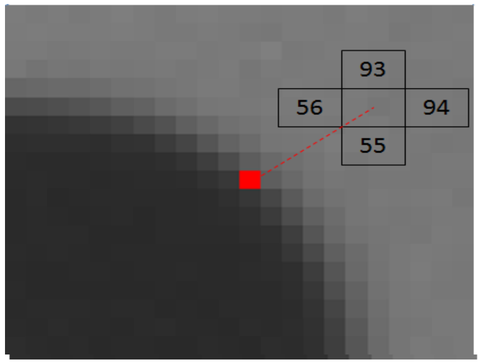
Histogram of Oriented Gradient Features - HOG

- Local shape information is often well described by the distribution of edge directions or gradient intensities
- The histogram of oriented gradients (HOG) is a type of feature descriptor. The intent is that the HOG features of an object remain similar when the object is viewed under different conditions.
- The image is divided into small segments. In each segment a histogram of edge orientations is accumulated
- The combined histogram entries are used as the feature vector describing the object

Histogram of Oriented Gradient Features - HOG

- Consider the image of the penguin, and the marked edge in its head

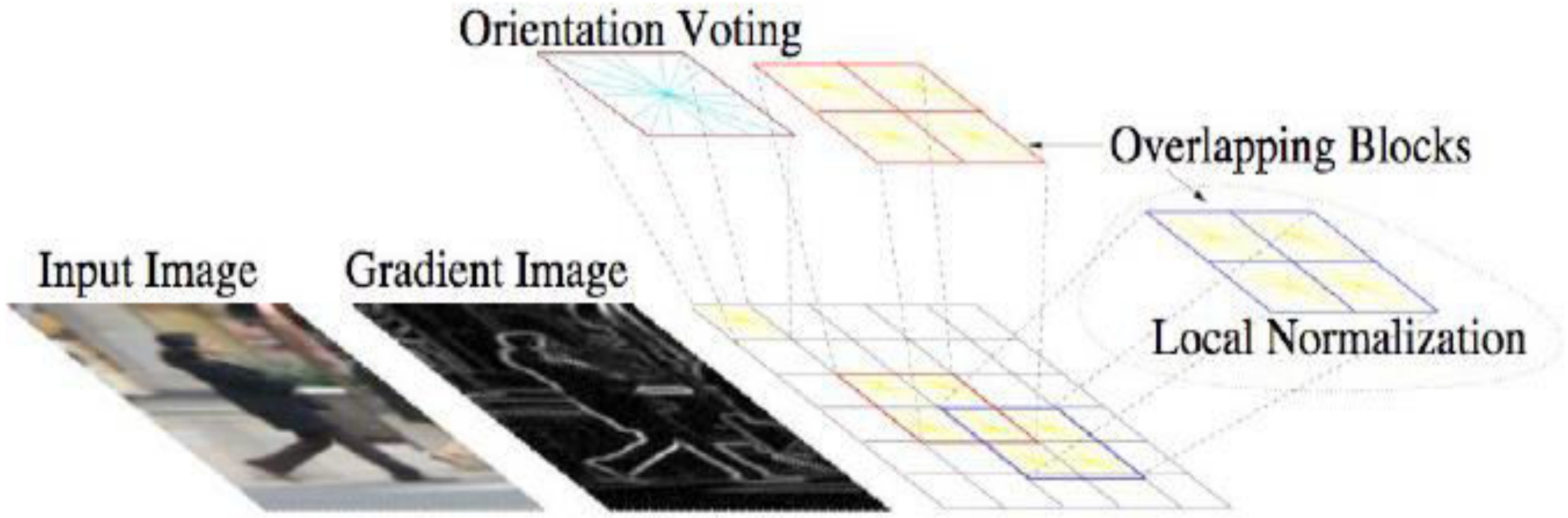
randolph is a
penguin



- Several gradient operators may be used, among them (1, 0, -1). With the pixel values specified, the gradient at the edge point considered is (38, 38), or magnitude of 53.7 and orientation of 45^o
- This gradient remains the same if illumination changes, and the image becomes brighter. Hence, the gradient may be considered illumination invariant (but changes with contrast)

Histogram of Oriented Gradient Features - HOG

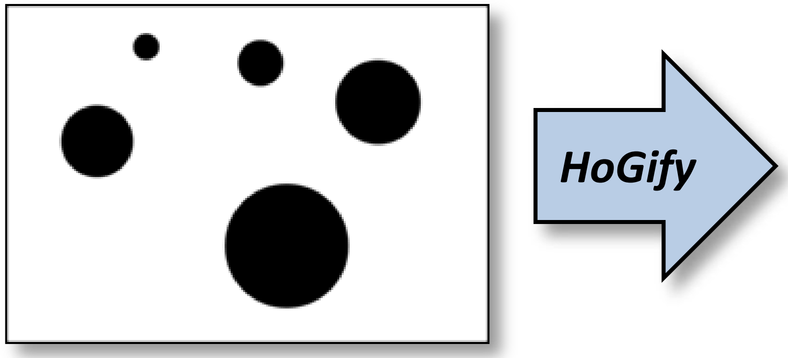
- To compute the HOG, the image is divided into segments, e.g. 8x8 with overlap between adjacent segments.
- The overlap ensures consistency across the image without the loss of local variations

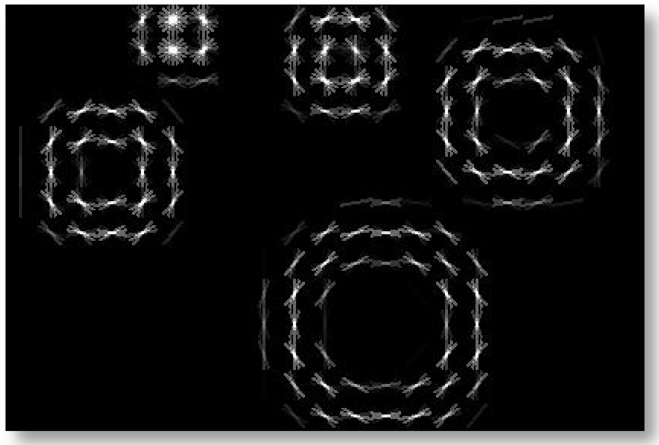


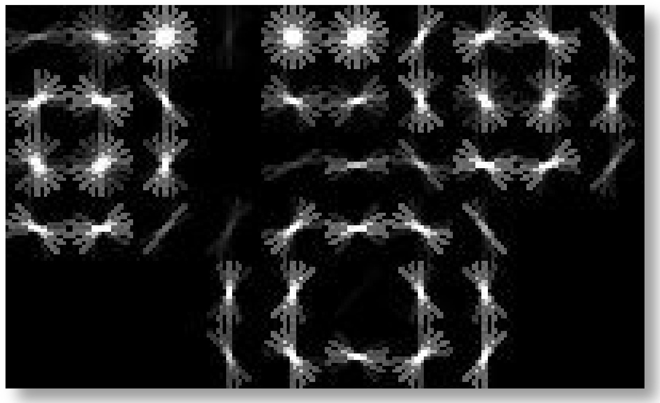
Histogram of Oriented Gradient Features - HOG

- The HOG is calculated separately for each segment. Each pixel in a segment votes for an orientation according to the closest bin in the range, e.g. [0, 180] or [0, 360]
- For a color image the largest gradient among the three is used for voting
- The vote is weighted by the gradient magnitude. Namely, the bin of the histogram is not just incremented by one for each pixel with a gradient in that direction, but the increment is weighted by the gradient magnitude
- If the direction histogram is 9-bins long, then the 64 gradient values of the 8x8 segment are mapped into 9 values of the histogram

Histogram of Oriented Gradient Features - HOG




10x10 cells


20x20 cells

[Dalal and Triggs, CVPR 2005]

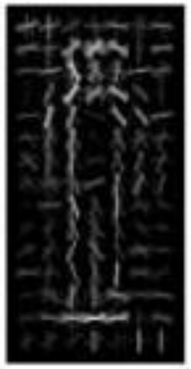
Histogram of Oriented Gradient Features - HOG

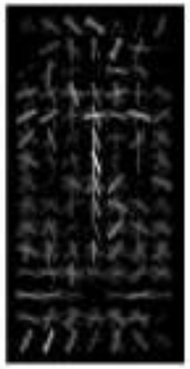
- The feature vector is made up of the multiple HOGs within the image
  - Shape information is encoded by HOGs
  - Spatial location is implicitly encoded by the relative position of the HOG within the image
- An image of 64x128 has 8x16 segments, and with 9-bin histogram for each segment, the feature vector is 128x9 long
- The classification of the feature vector will be discussed later in the classification chapter

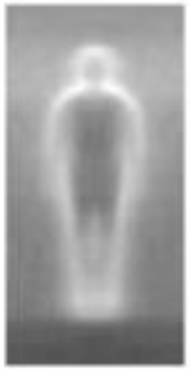
Person Detection using HOG

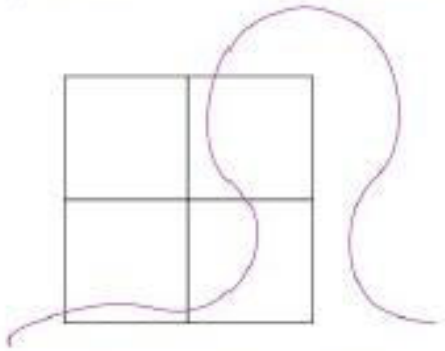
- HOG has successfully been applied to person detection

input image


weighted pos wts


weighted neg wts

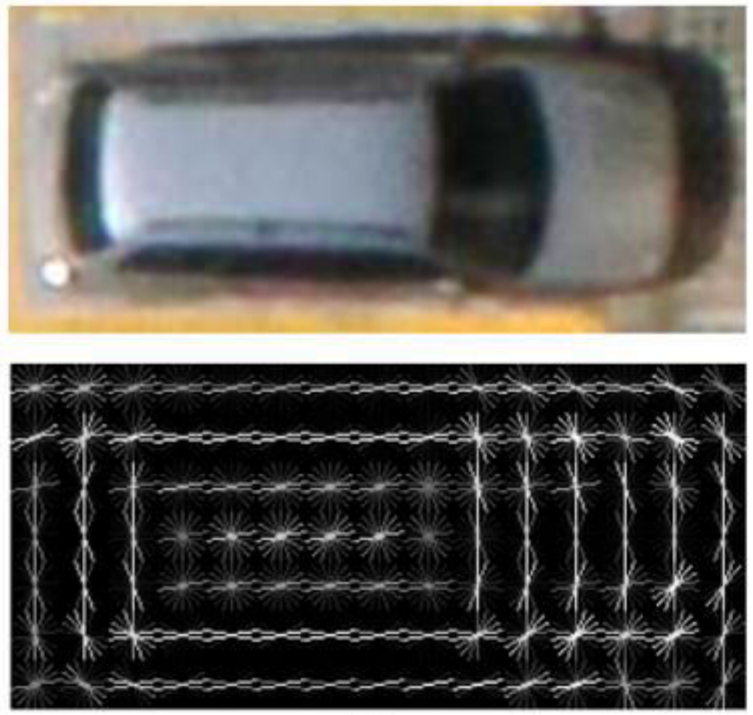

avg. grad


outside in block

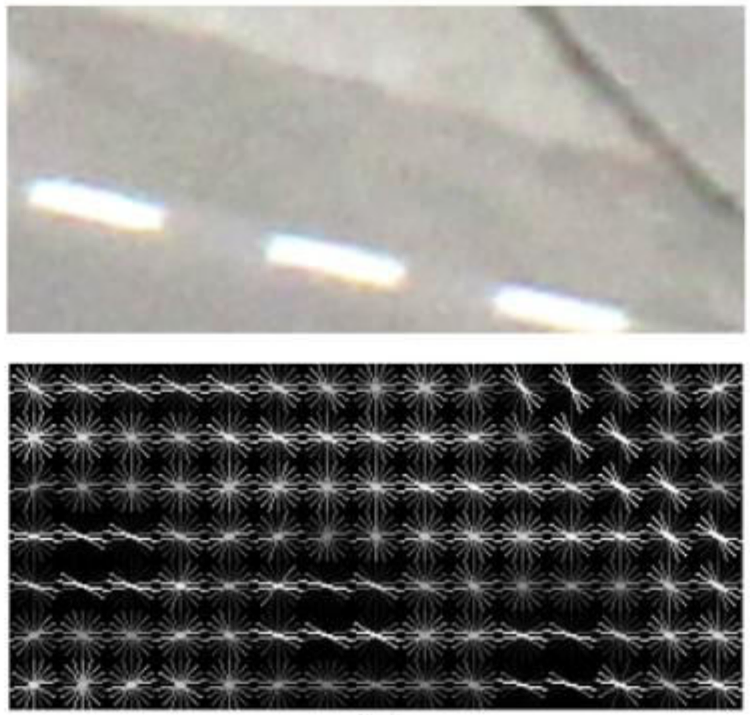
- The most important cues are head, shoulder, leg silhouettes
- Vertical gradients inside the person count as negative
- Overlapping blocks those just outside the contour are the most important

Car Detection using HOG

- HOG has successfully been applied to the detection of cars


(a)

Car and HOG features


(b)

background and HOG features

Scale Invariant Feature Transform - SIFT

- Scale Invariant Feature Transform (SIFT) is an approach for detecting and extracting local feature descriptors that are reasonably invariant to changes in illumination, image noise, rotation, scaling and small changes in viewpoint
- Detection stage for SIFT features:
  - Scale-space extrema detection
  - Key-point localization
  - Orientation assignment
  - Generation of key-point descriptors

Scale Invariant Feature Transform

- Interest points for SIFT features correspond to local extrema of difference of Gaussian filters at different scales
- Given a Gaussian-blurred image

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

  where

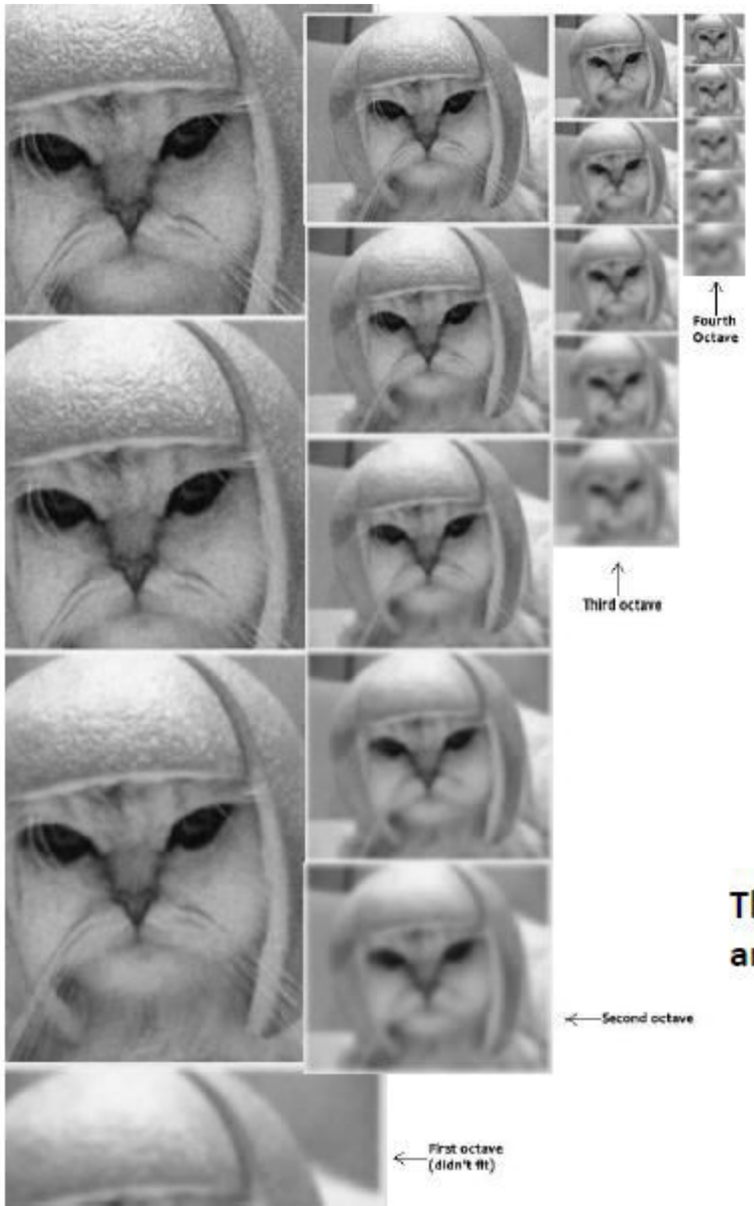
$$G(x, y, \sigma) = 1/(2\pi\sigma^2) \exp^{-(x^2+y^2)/\sigma^2}$$

  is a variable scale (σ) Gaussian
- The result of convolving an image with a difference-of-Gaussian filter $G(x, y, k\sigma) - G(x, y, \sigma)$ is given by

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

- The difference-of-Gaussian filter provides an approximation to the Laplacian of Gaussian $\sigma^2\nabla^2 G.$ and it is in effect a tunable bandpass filter

Scale Invariant Feature Transform - SIFT



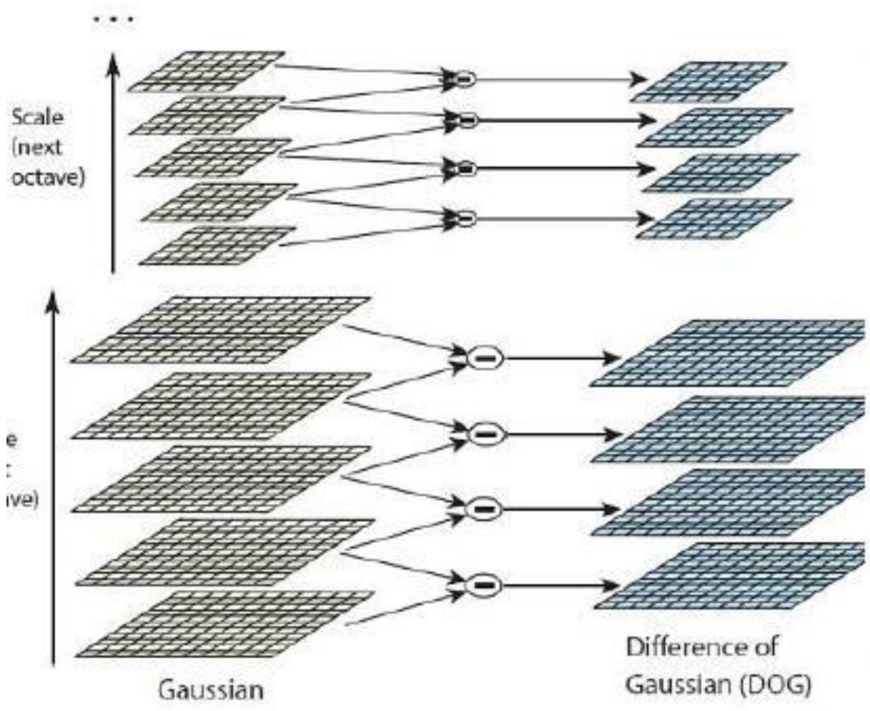
Construction of a scale space

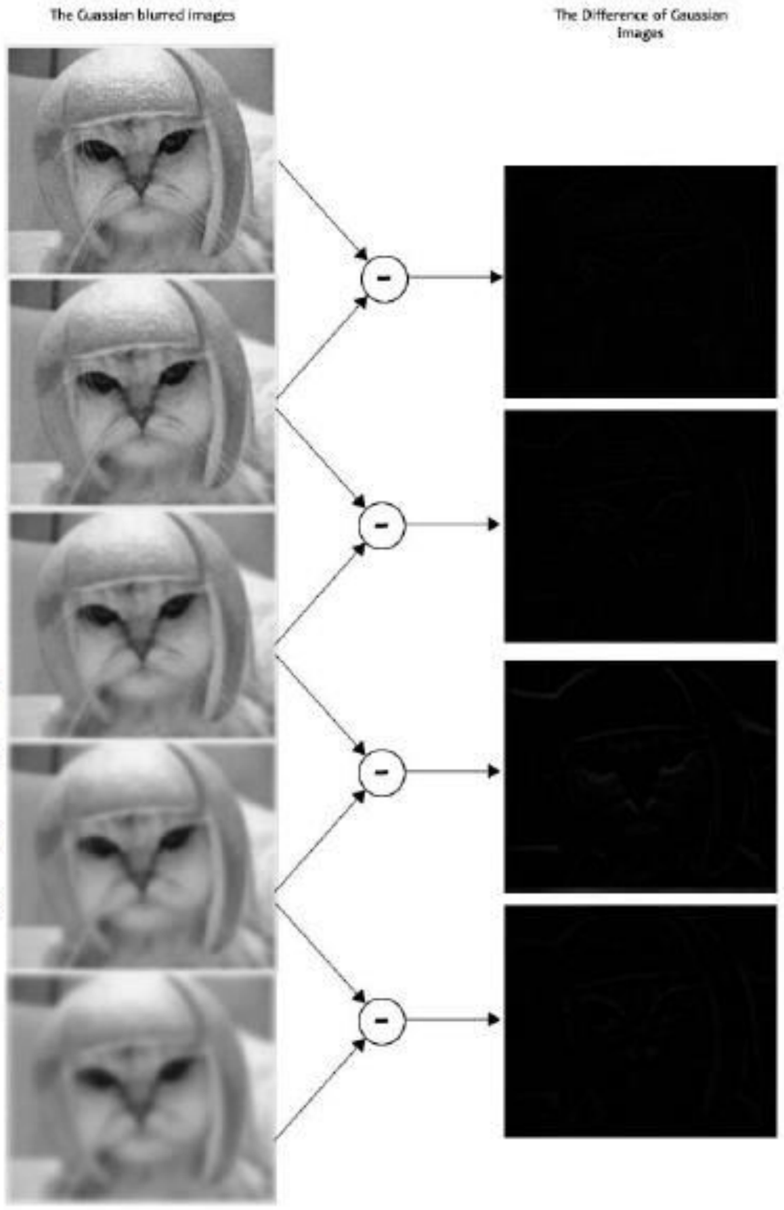
SIFT takes scale spaces to the next level. You take the original image, and generate progressively blurred out images. Then, you resize the original image to half size. And you generate blurred out images again. And you keep repeating.

The creator of SIFT suggests that 4 octaves and 5 blur levels are ideal for the algorithm

Scale Invariant Feature Transform -

LoG approximation



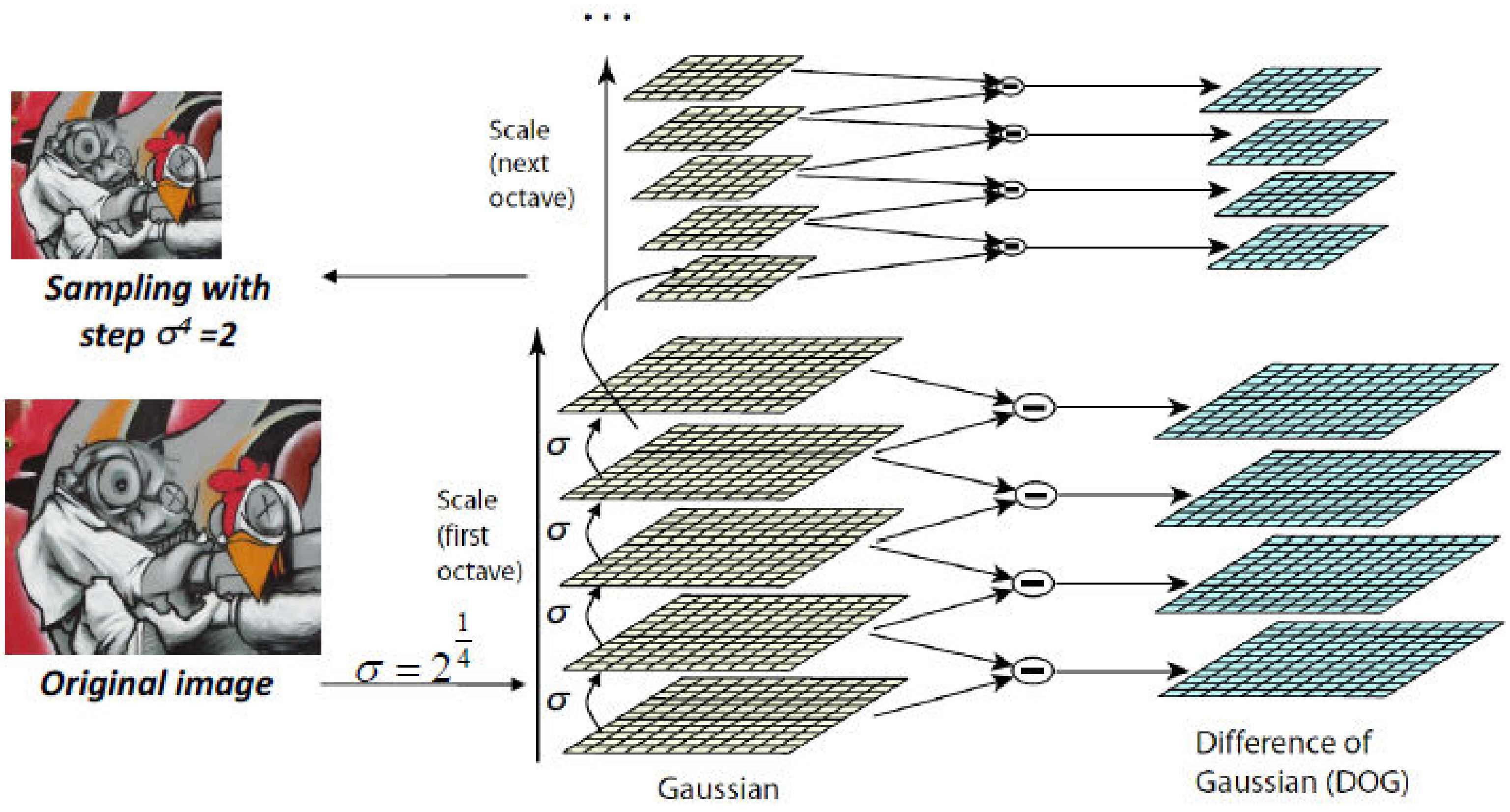


The Guassian blurred images
The Difference of Gaussian Images

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2\nabla^2 G.$$

Scale Invariant Feature Transform - SIFT

- $D(x, y, \sigma)$: is just the difference between the Gaussian blurred images at scales σ and $k\sigma$



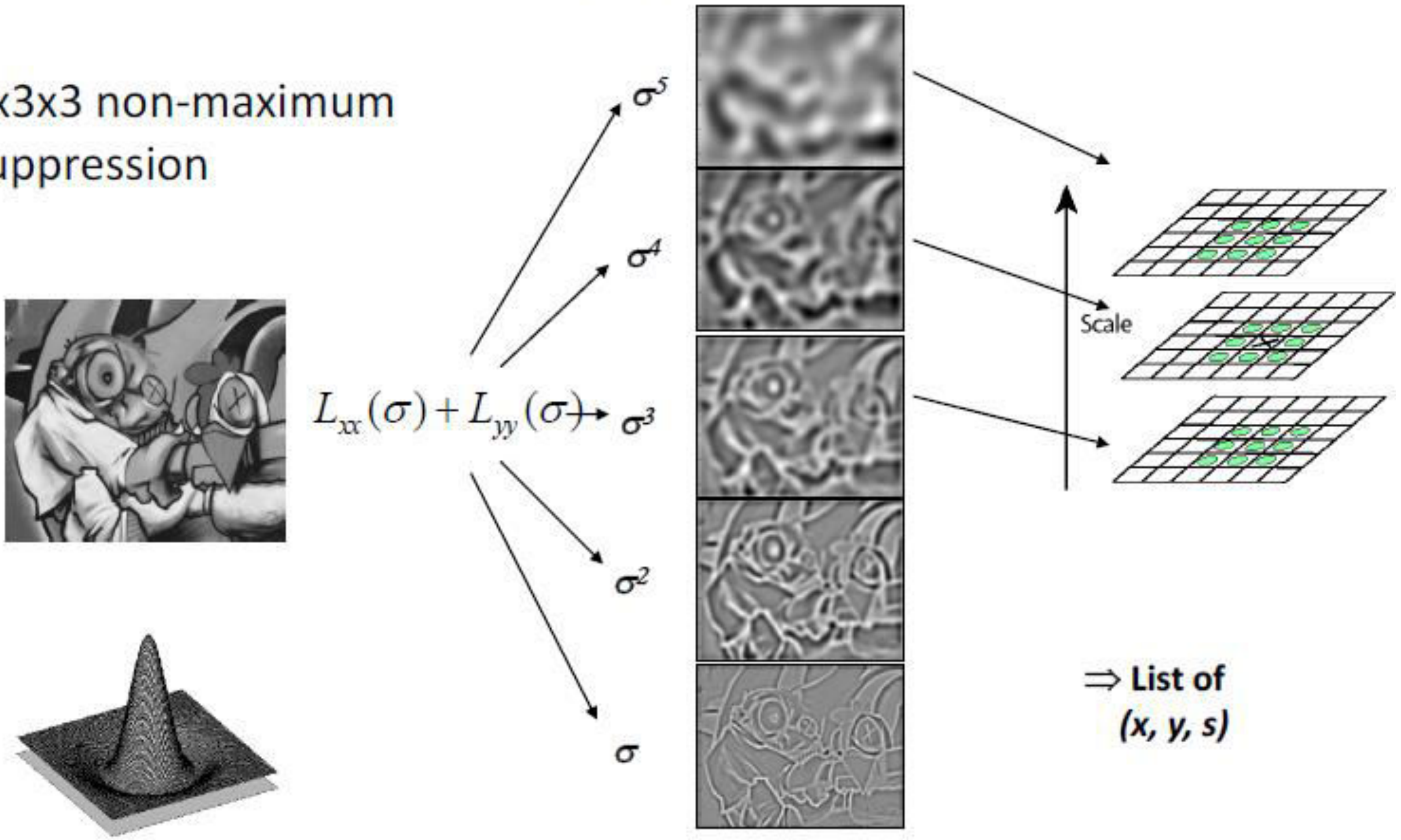
Scale Invariant Feature Transform - SIFT

- The convolved images are grouped by octave, where an octave corresponds to doubling the value of σ , and the value of *k* is selected so that a fixed number of blurred images per octave is obtained
- It also ensures that the same number of difference-of-Gaussian images per octave is obtained

Scale Invariant Feature Transform - SIFT

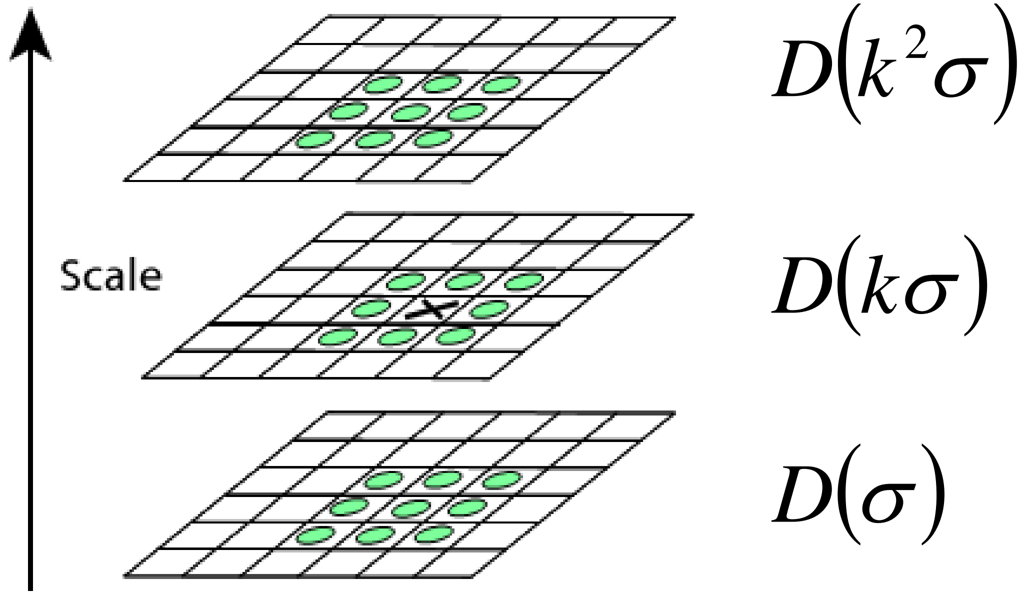
Find local maxima in position-scale space of Difference-of-Gaussian

3x3x3 non-maximum suppression



$\Rightarrow$ **List of *(x, y, s)***

Scale Invariant Feature Transform - Scale-Space extrema Detection



- In the local extrema detection the pixel marked x is compared against its 26 neighbors in a 3x3x3 neighborhood from adjacent DoG images

Scale Invariant Feature Transform - Scale-Space extrema Detection

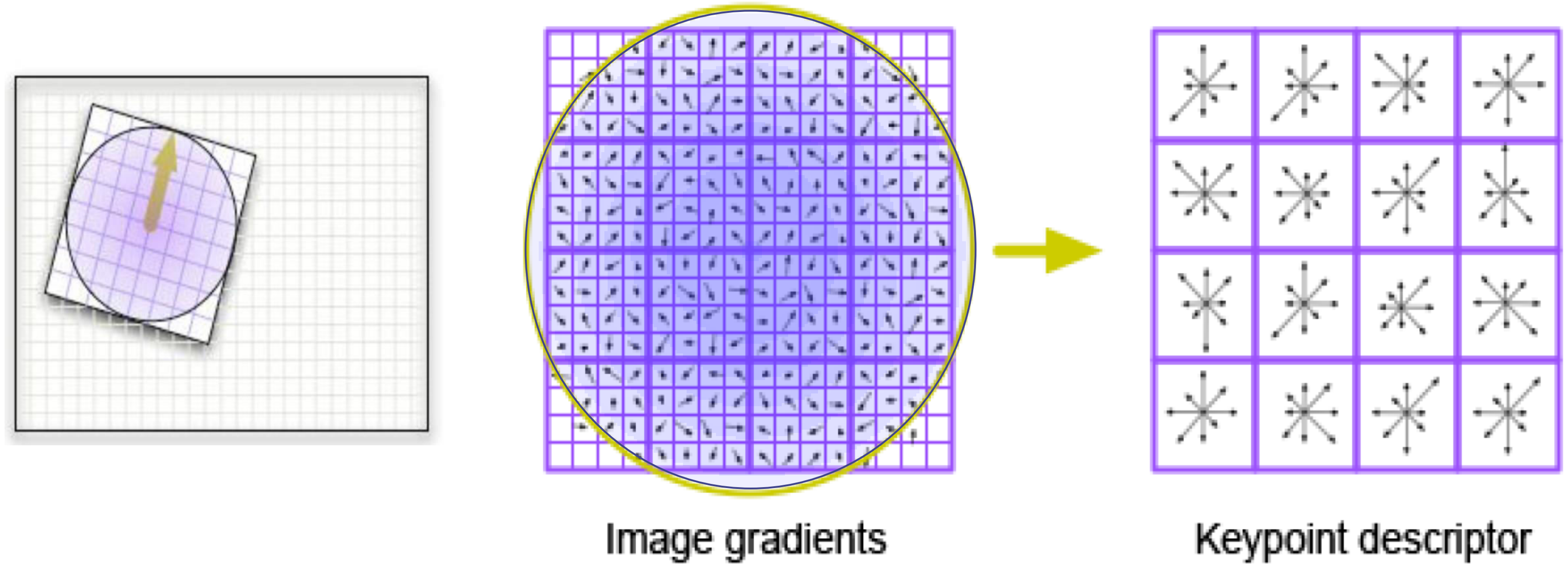
- Interest points, (called key-points), are identified as local maxima or minima of the DoG images across scales
- Each pixel in the DoG images is compared to its 8 neighbors at the same scale and the 9 corresponding neighbors at adjacent scales
- If a pixel is a local maximum or minimum, it is selected as a candidate interest point

Scale Invariant Feature Transform - Scale-Space extrema Detection

- Each interest points is assigned an orientation
- To determine the interest point orientation, a gradient orientation histogram is computed in the neighborhood of the interest point (using edge direction calculation)
- The contribution of each pixel is weighted by the magnitude of the gradient
- Peaks in the histogram correspond to dominant orientations. A separate interest point is created for the direction corresponding to the histogram maximum
- All the properties of the interest point are measured relative to the interest point orientation, providing invariance to rotation

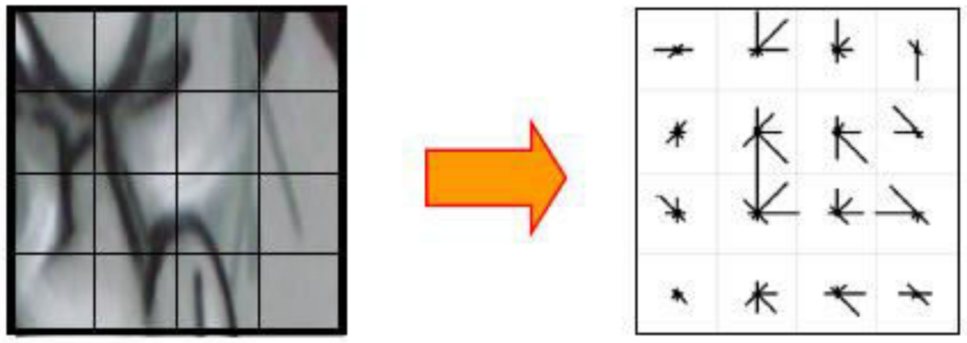
SIFT Feature Representation

- Once an interest point orientation has been selected, the feature descriptor is computed as a set of orientation histograms on 4x4 pixel neighborhood
- The orientation histograms are relative to the interest point orientation. The contribution of each pixel to the histogram is weighted by the gradient magnitude

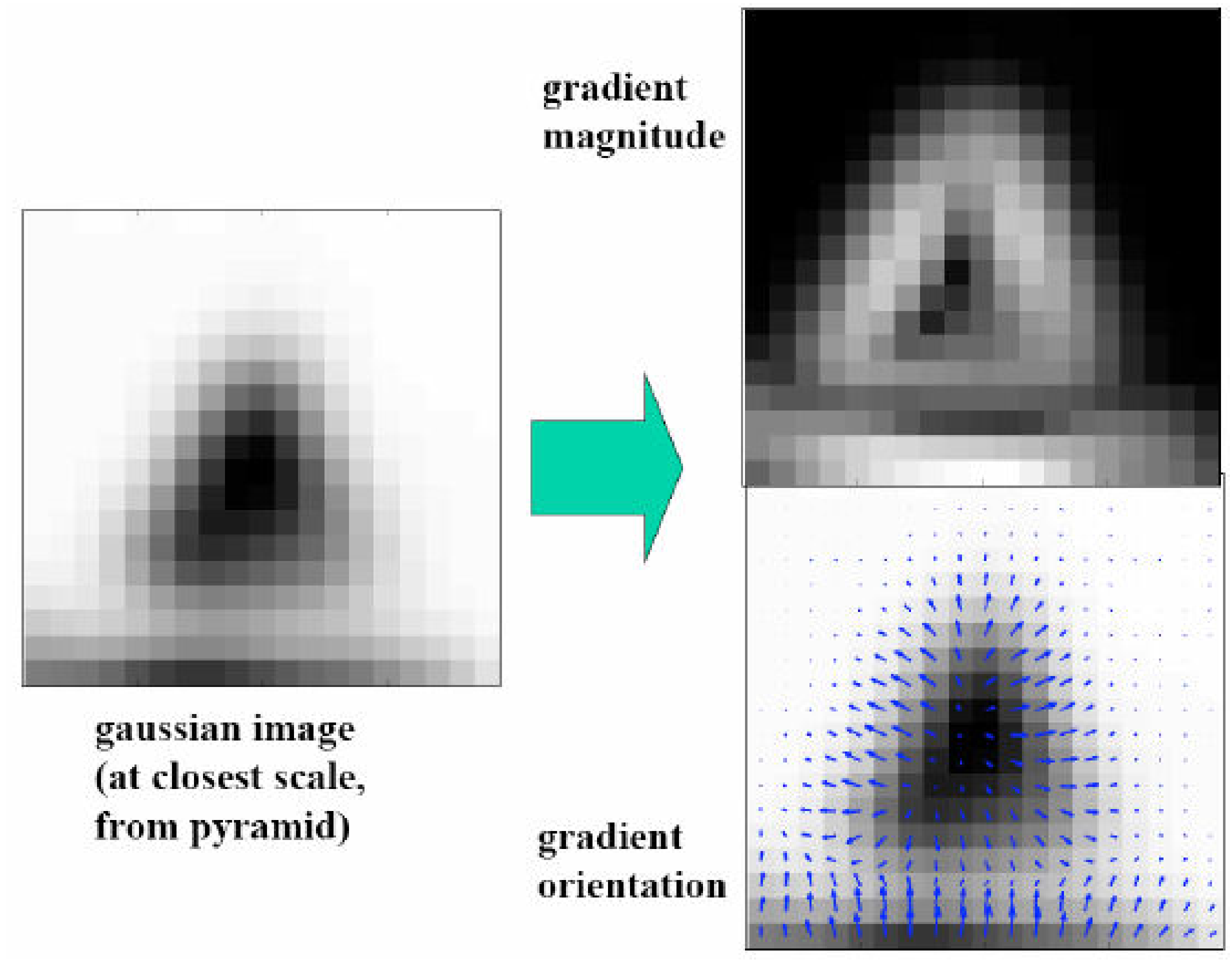


SIFT Feature Representation

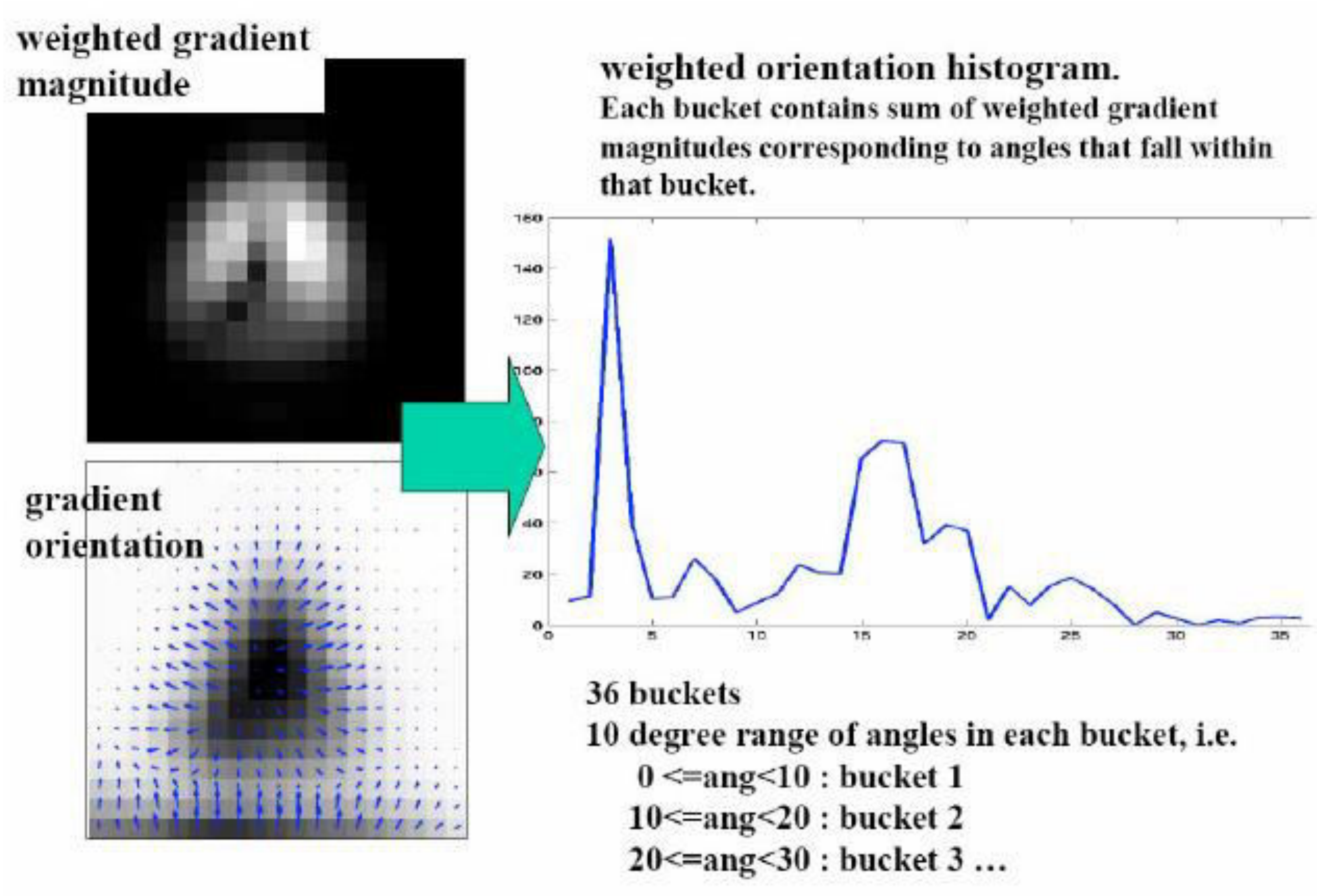
- Around each interest point a patch of 16x16 pixels is created. It is divided into 16 segments of 4x4 pixels
- An orientation histogram with 8 bins is created for each 4x4 pixels segment
- Thus, a descriptor contains an array of 16 histograms around the interest point. This leads to a SIFT feature vector with 16x8=128 elements for each interest point
- For each 4x4 pixels segment the orientation marked is that of highest value of the histogram
- Besides the gradient information, the descriptor of each key point includes its coordinates, orientation and scale



SIFT Orientation Assignment



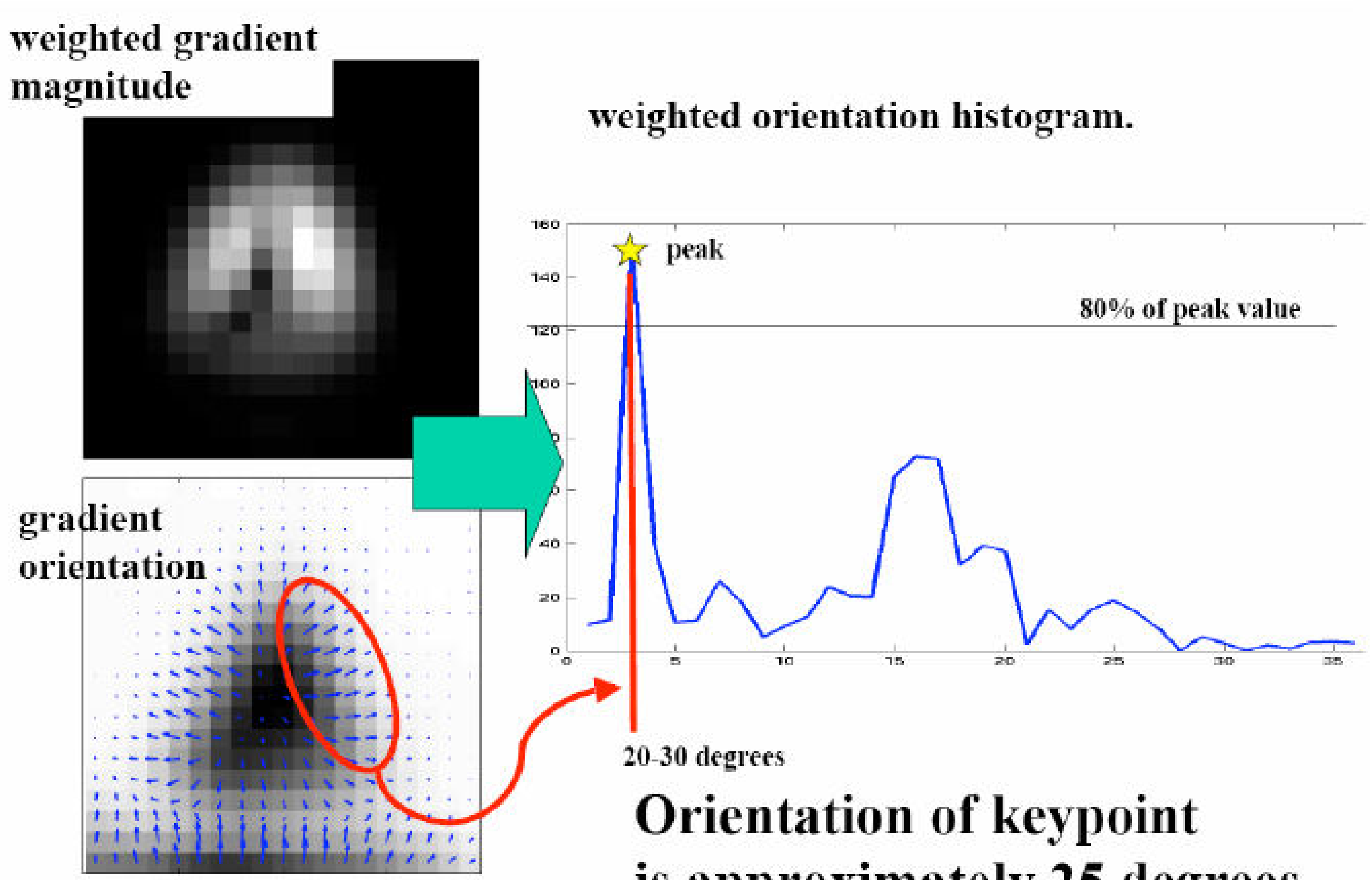
SIFT Orientation Assignment



weighted orientation histogram.
Each bucket contains sum of weighted gradient magnitudes corresponding to angles that fall within that bucket.

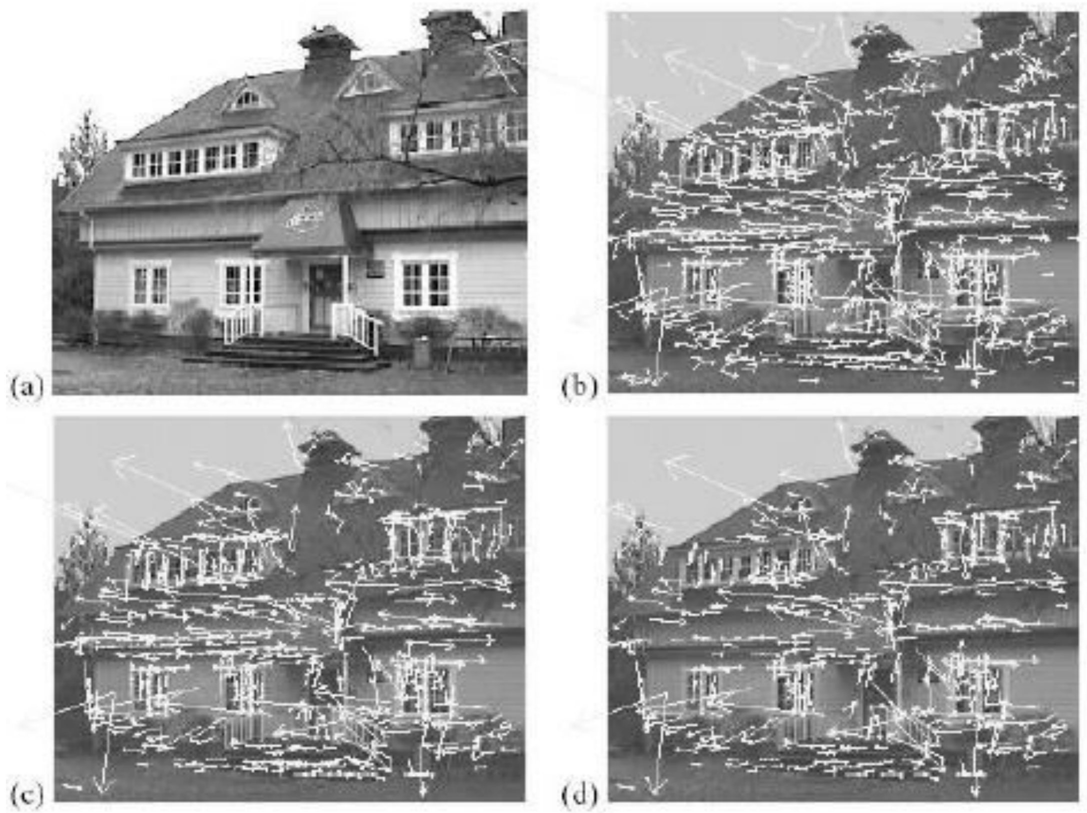
36 buckets
10 degree range of angles in each bucket, i.e.
0 <=ang<10 : bucket 1
10<=ang<20 : bucket 2
20<=ang<30 : bucket 3 …

SIFT Orientation Assignment



SIFT Feature Representation

- The following is SIFT results for a given image



(a) 233x189 image
(b) 832 DOG extrema
(c) 729 left after peak value threshold
(d) 536 left after testing ratio of principle curvatures (removing edge responses)

SIFT Advantages

- SIFT features are invariant to scale, position and rotation changes
- SIFT can handle changes of view point and pose
- It is insensitive to changes of illumination
- It is robust in the presence of noise
- Matching of SIFT features vectors from different images, and classification based on SIFT features will be discussed later in the classification chapter

SIFT Advantages

1. Keypoint selection
   - **Enforce invariance to scale**: Compute Gaussian difference over many different scales; non-maximum suppression; find local maxima as keypoint candidates.
   - **Localizable corner:** For each maximum fit quadratic function - compute center with sub-pixel accuracy by setting first derivative to zero.
   - **Eliminate edges:** Compute ratio of eigenvalues, drop keypoints for which this ratio is larger than a threshold.

2. Point description
   - **Enforce invariance to orientation:** Compute orientation, to achieve rotation invariance, by finding the strongest second derivative direction in the smoothed image; rotate patch so that orientation points up.
   - **Compute feature signature:** Compute a "gradient histogram" of the local image region in a 4x4 pixel region; do this for 4x4 regions of that size; orient so that largest gradient points up. Result: feature vector with 128 values (15 fields, 8 gradients).
   - **Enforce invariance to illumination change and camera saturation:** Normalize to unit length to increase invariance to illumination; threshold all gradients, to become invariant to camera saturation.

SIFT Advantages

- The process to calculate the SIFT features is compose of four steps, each leads to the features being invariant to another aspect
- The first is devoted to the identification of possible locations which are invariant to scale changes. This objective is carried out by searching for stable points across various possible scales created by convolving the image with a variable scale Gaussian filter
- The detection of stable locations is done by identifying scale-space extrema in the Difference-of-Gaussian function convolving the image
- Features ought to be invariant to the rotation point of view. This is achieved by assigning to each key point a consistent local orientation.
- In the last step each key point is assigned with a gradient vector to provide a further invariance, especially invariance of viewpoint and illumination
- The invariance to illumination changes is provided by normalizing the descriptor vector to a unit length
- Robustness to noise is achieved by disregarding key points where the gradient value is too low (poor contrast)

Haar-like Features

- Haar-like features have been demonstrated to be successful in object detection tasks.
- Haar-like features are an over complete set of two-dimensional Haar functions, which can be used to encode local appearance of objects
- The feature value of a Haar-like feature which has k rectangles is obtained from

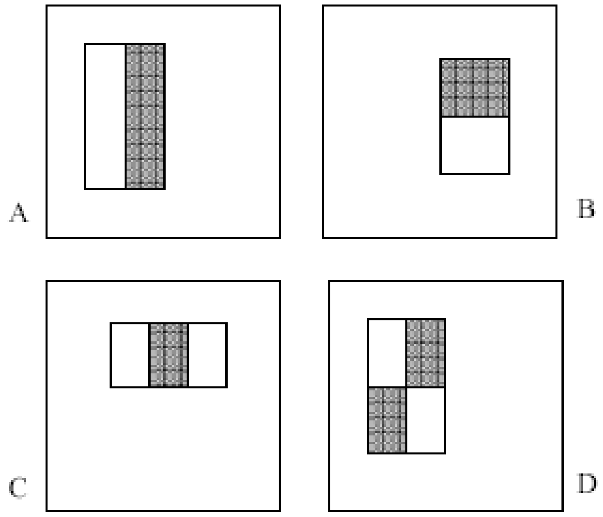
$$f = \sum_{i=1}^{k} w^{(i)} \cdot \mu^{(i)}$$

where $\cdot\ \mu^{(i)}$ is the average (sum) of the image pixels enclosed by the i-th rectangle and $w^{(i)}$ is the weight assigned to the i-th rectangle.

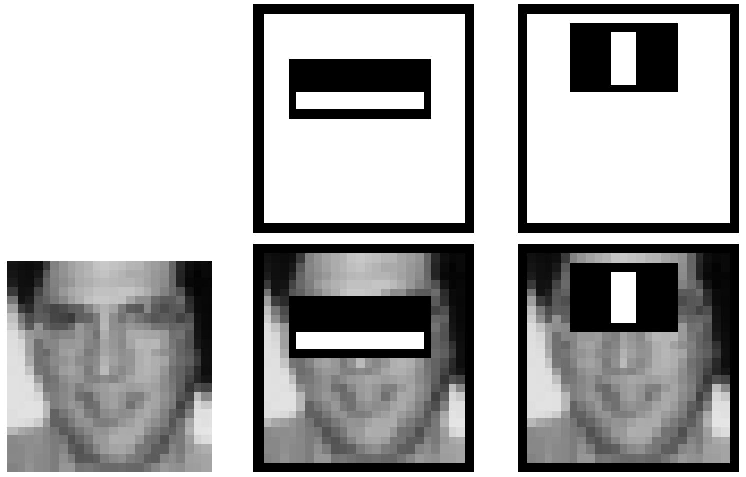
The sum of the rectangles weights are commonly set to 0

Haar-like Features

- Multiple set of simple features are obtained by applying simple rectangular filters to an image segment



Rectangular filters

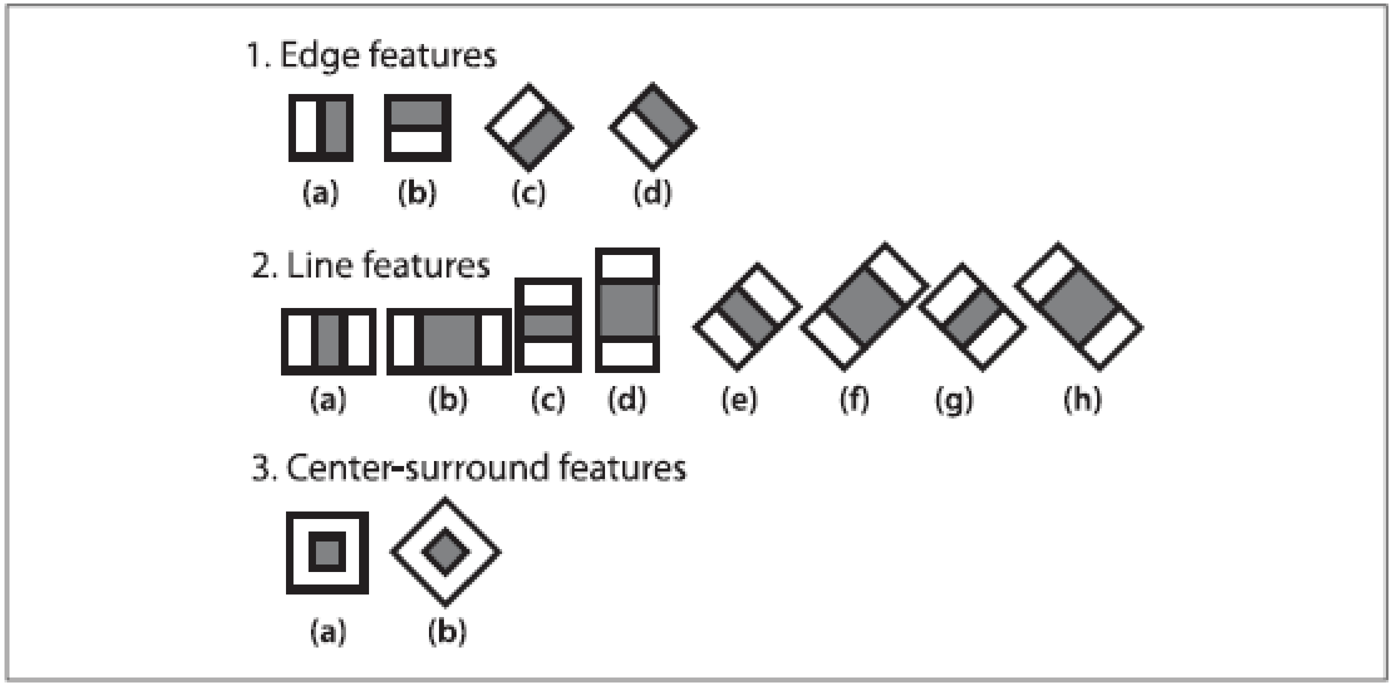


Forehead, eye features can be captured

- The figure shows 3 types of filters: 2-rectangle features (A and B), 3-rectangle features (C) and 4-rectangle features (D)

Haar-like Features

- Other possible Haar-like features:

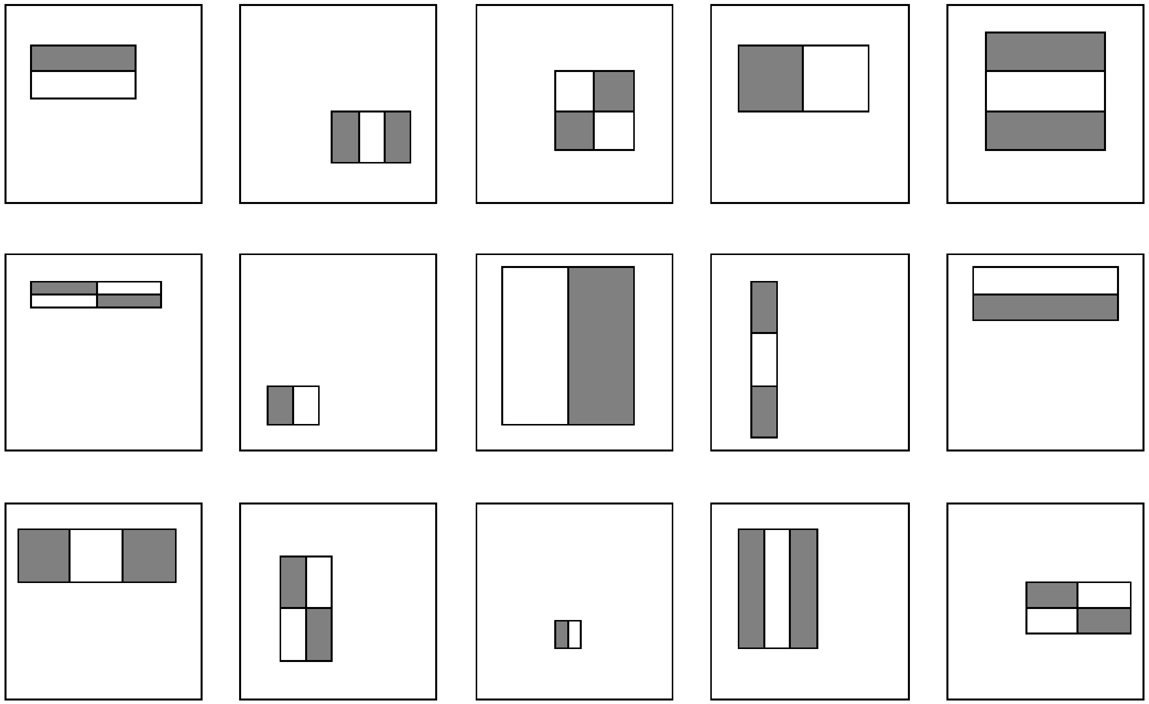


- The filters subtracts the sum of image pixels covered by the black area from the image pixels covered by the white area

$$f_i = Pixel\_Sum(\,Rect\_W) - Pixel\_Sum(\,Rect\_B) \qquad h_i(x) = \begin{cases} 1 & if\ \ f_i > threshold \\ -1 & if\ \ f_i < threshold \end{cases}$$

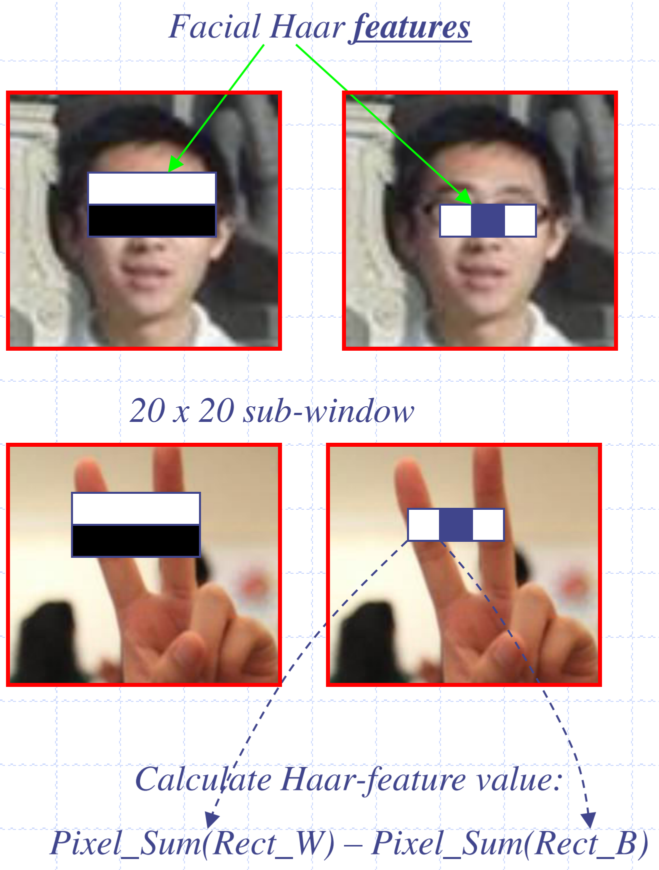
Haar-like Features

- Each feature (filter) can be of different width and height as in the figure



- Thus, even for a small size image, the total number of feature calculated is very large

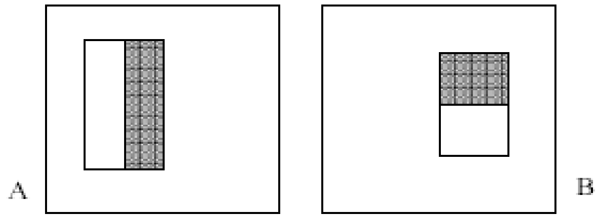
Haar-like Features Calculation


Facial Haar features
20 x 20 sub-window
Calculate Haar-feature value:
Pixel_Sum(Rect_W) – Pixel_Sum(Rect_B)

- The image is segmented into 20x20 pixels segments, and in each an exhaustive search for the features is conducted
- For example, for human face detection the two features shown can be useful. The feature on the left can differentiate between the forehead and the area around the eyes since the later is commonly darker.
- Similarly, the feature on the right can differentiate between the eyes and nose

Haar-like Features

- Each Haar-like feature is in use in all its possible sizes
- For example, suppose that for a certain detection task features A and B are used



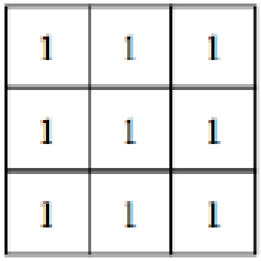
- If the image is segmented into 20x20 pixels segments, and the features are calculated for each segment separately, then for each segment a total of (18+16+14+…+2)(20+19+18+…+1)*2=37800 features are extracted (assuming the same dimensions for the white and black parts)
- For example, feature A may have 20x1 white and black rectangles or 2x3 white and black rectangles

Calculating the Haar-like Features

- Haar-like features can be efficiently calculated using the integral image
- The integral image at location *(x,y)* is the sum of pixel values in the original image above and to the left of *(x,y)*:

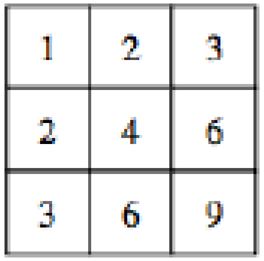
$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x',y'),$$

where *i(x,y)* is the pixel value of the original image at (x,y), and *ii(x,y)* is the value of the integral image at that point



1	1	1
1	1	1
1	1	1

Input image



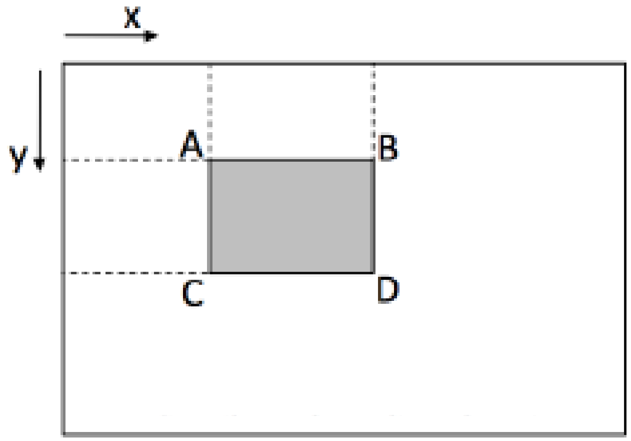
1	2	3
2	4	6
3	6	9

Integral image

. Example of integral image |

Calculating the Haar-like Features

- Using the integral image to compute the sum of pixels values of any rectangular segment is simple
- The sum of pixels values in rectangle ABCD in the image



can be calculated with only four values from the integral image

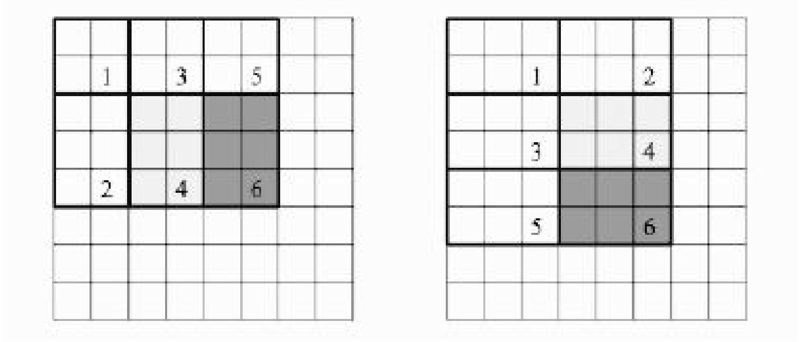
$$\sum_{(x,y)\in ABCD} i(x,y) = ii(D) + ii(A) - ii(B) - ii(C).$$

Calculating the Haar-like Features

- The integral image is used to compute efficiently the Haar-like features
- It has been demonstrated how simple it is to calculate the sum of pixels values of a rectangle using the integral image.
- And since the Haar-like features involve the calculation the sum of pixels values in the black and white rectangles of the corresponding filter, the integral image is used in calculating the Haar-like features

Calculating the Haar-like Features

- Consider the two features



Computing two-rectangle Haar-like features

- The Haar-like feature is equal to the sum of pixels in the darker rectangle minus the sum of pixels in the lighter rectangle
- Then, each feature can be calculated as the sum of 6 values from the integral image I (subtracting the two rectangle sums above)

$$\begin{aligned} f = &-I(x_1, y_1) + I(x_2, y_2) + 2I(x_3, y_3) \\ &- 2I(x_4, y_4) - I(x_5, y_5) + I(x_6, y_6). \end{aligned}$$

Classification of the Haar-like Features

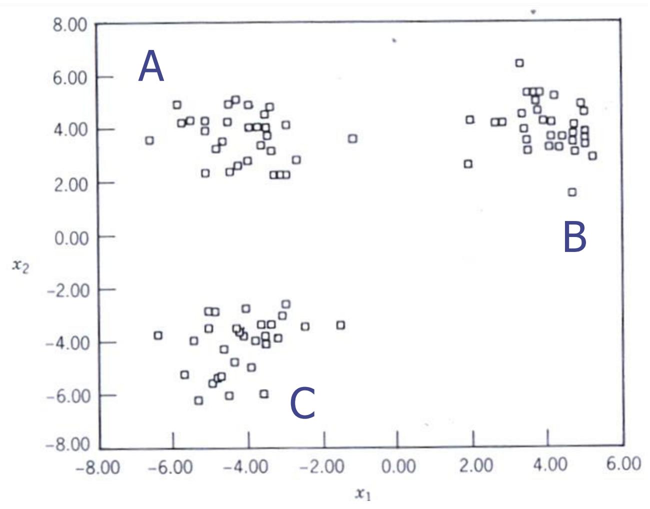
- As mentioned, multitude of small and simple features are created for each image segment.
- A way is required to choose the meaningful features, and to classify the image segment based on them
- Classification of the multiple Haar-like features will be discussed later in the chapter on classification

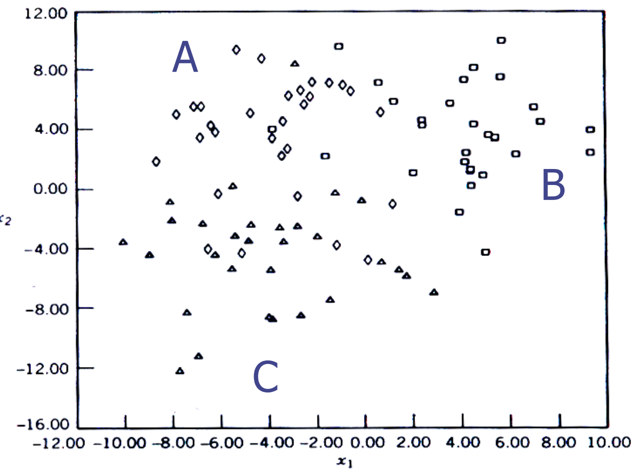
Criteria for Features Selection

- The features should be distinctly different for the different shapes to be recognized. Only then can the shapes be recognized based on their features values
- It is desired that the features are insensitive to insignificant change in the image like scaling, rotation or a shift (a shape does not change when these changes occur)
- The features can have a physical meaning, or they can be a result of some mathematical manipulation of the shape
- A recognition task can be initiated with a small number of features, and if they are found to be insufficient (many errors in recognition), more features may be added

Criteria for Features Selection

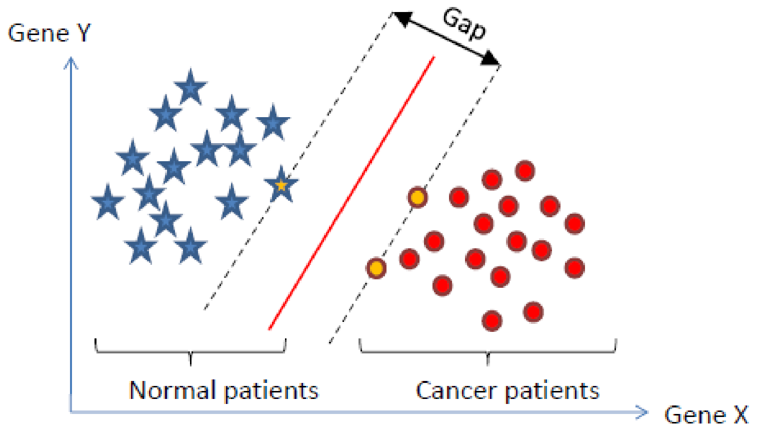
- The following two figures show adequate and improper choices of features. In both cases the 3 shapes under consideration are represented by a vector of two features each. In the upper figure, the features separate the 3 shapes into 3 distinct clusters. In the bottom figure, the choice of the features does not allow a proper separation among the 3 clusters, namely, the features are not distinctly different for the 3 shapes.

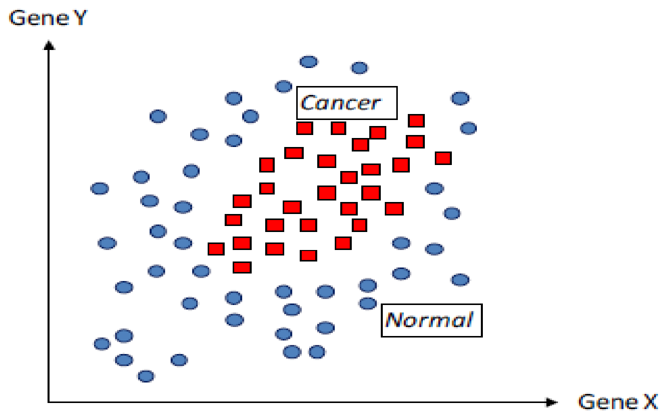




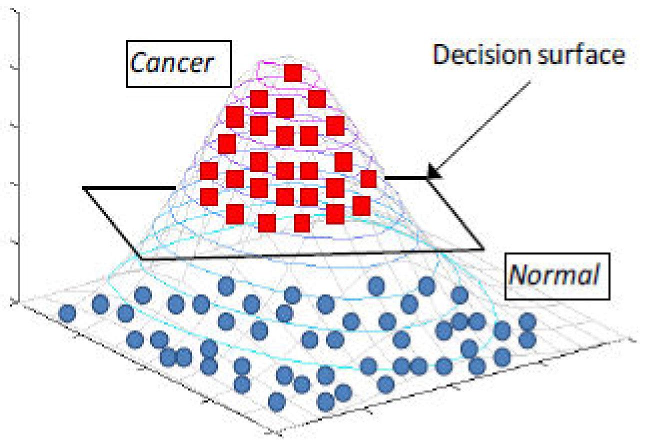
Class Separation using Extra Features

- The two cases of linearly separable data and non-linearly separable data are shown in the figure





- The non-linearly separable data may be separable if more features are added:



Features Reduction

- It is essential to reduce the number of features
  - To reduce the computational complexity
  - To make the estimation of the distribution functions in the reference classes easier
  - To eliminate correlation among features
  - To make classification more robust, and less sensitive to insignificant changes in the shapes
- In the process of reducing the number of features, part of the data needed for the separation of the reference shapes is lost
- The intention is to keep those features that contribute the most to the separation of the reference classes, and thus to reduce the classification error

Features Mapping

- Feature mapping may be applied to improve the properties of the given features. The mapping of features may be linear or non-linear.
- A liner mapping may be represented by a matrix relation between the original feature vector $\underline{x}$ and the new feature vector $\underline{y}$

$$\underline{y}=A\underline{x}$$

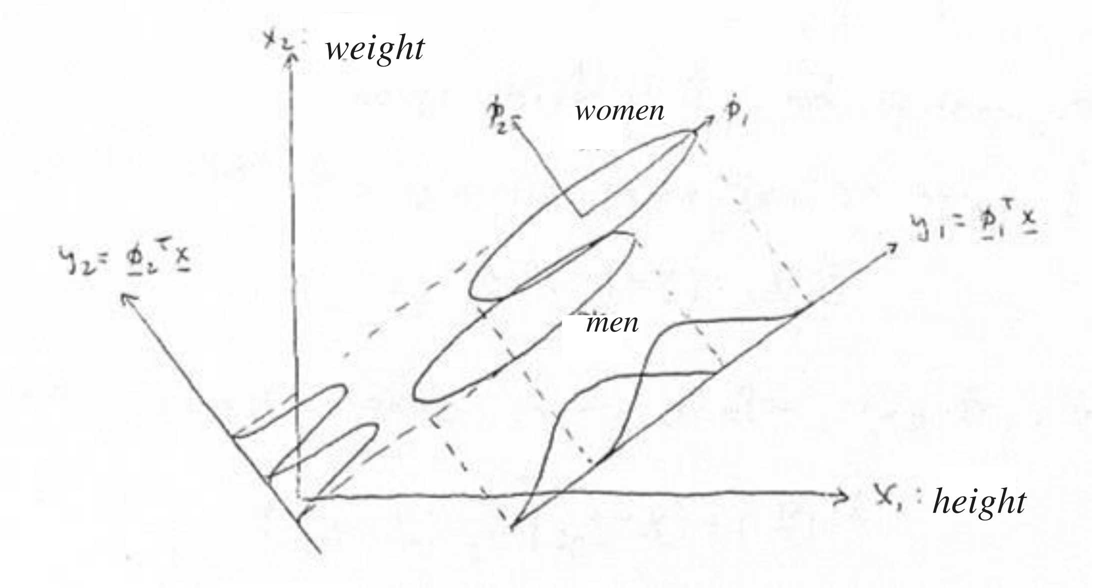
- A non-linear mapping of features may be specified by a functional relation between the original and new features

Features Mapping

Example – linear mapping

Man and women are to be classified based on their weight and height. It is assumed that the features vector (weight, height) is normally distributed in both classes. In the following figure, slices from the distributions are shown (the two ellipses).

Linear mappings from the two features to a single feature are to be tested.


x_2 weight
ϕ_2
women
ϕ_1
$y_1 = \underline{\phi}_1^T \underline{x}$
$y_2 = \underline{\phi}_2^T \underline{x}$
men
x_1 : height

Features Mapping

Example – continue

In the figure two linear mappings from two features to one are presented. It may be expected that a single feature is capable of separation between the classes, since there is a some correlation between height and weight.

The mapping $y_1 = \underline{\phi}_1^T \underline{x}$ causes a deterioration in the separation between men and women, as can be seen from the overlap between their one-dimensional distribution functions.

The mapping $y_2 = \underline{\phi}_2^T \underline{x}$ maintains the separation between men and women, as can be seen from the overlap between their one-dimensional distribution functions.

Hence, y_2 is a better feature than y_1 for separating and classifying men and women

Features Reduction

- A measure is required for the separation of the reference classes, and it should be measured while reducing the number of features
- A possible measure is the distance between the centers of the classes, namely, between the average features vectors of the classes. It is obvious that a larger value to this measure, implies a better separation between classes
- Evaluating this measure can be done for each feature independently – a feature is included, if its average value in the different classes is significantly different

Features Reduction

- However, the requirement of a significant distance between the centers of classes is not sufficient.
- It is possible that the average values of a feature are significantly different in the various classes, but its large variance values reduce its contribution to the separation of the classes
- A significantly different average value, and a small variance guarantee the separability of the classes by a feature

Features Reduction

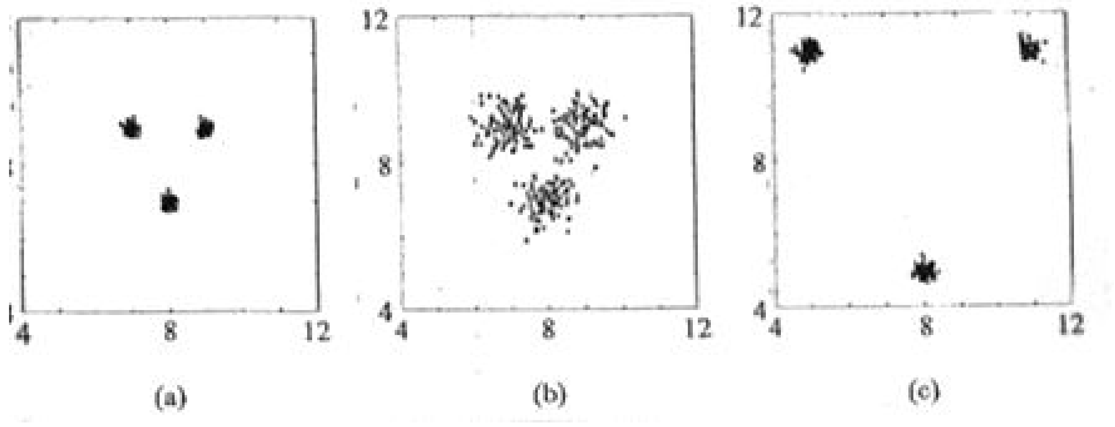
- For a single feature and two classes a reliable separation measure is

$$J = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$$

  where μ_1 and μ_2 are the average values of the feature in the two classes, and σ_1^2 and σ_2^2 are its variance values in the two classes
- If there are more than two classes, the measure *J* is calculated for the feature for each pair of classes

Features Reduction

- The separation measure *J* gets high values when the features vectors of the samples are clustered close to the center of their class, and the clusters of the different classes are further apart
- In the distributions of samples in the following figure, the value of *J* is significantly higher for the clustering in (c)



Selecting a Subset of Features

- A measure has been defined for testing the efficiency of a feature (or features) for the task of classification (separation)
- The best measure is the classification error, but this cannot commonly be estimated from the set of samples
- Once the measure has been chosen, the task of reducing the number of features may be approached - that of choosing *m* out of the *n* given features

Features Selection and Mapping

- The intention in features selection is to find an appropriate mapping between the set of given samples, given as raw data, and a set of features.
- If a good mapping is chosen, a few features represent most of the information in the samples required for correct classification
- Mostly, linear mappings are considered: they involve simpler calculations, and it is easier to analyze their effect. A linear mapping may be considered as a projection of the original raw data on a smaller in dimension features space

Vectorial Selecting of Features

- There are two approaches to finding efficient linear mappings:
  - Principal Component Analysis - PCA
  - Multiple Discriminant Analysis – MDA
- PCA searches the projection which provides the best approximation (with the minimal error) to the original data
- MDA searches the projection which best separates the original data samples into their classes

Principal Component Analysis

- The intention is to reduce the dimensionality of the data space from the original N into a smaller value m
- In case of shapes in images, the original data vector $\underline{x}$ is obtained by rearranging the two-dimensional data into a vector, line by line.
- If the size of the original shapes is $n_1 x n_2$, then the original data vector is of size $N = n_1 n_2$
- The new feature vector $\underline{y}$ is of m uncorrelated features, with $m<N$. The features have to be those which minimized the error between the original data, and its new representation (by m-features vectors)

Principal Component Analysis

- Initially, a mapping is searched, which does not reduce the dimensionality of data, but only generates uncorrelated *N* features (of the original *N*-dimensional data)
- Let this mapping be A. The new data vector $\underline{y}$ is obtained by the linear mapping $\underline{y}=A\underline{x}$
- The autocorrelation matrix of the new vector $\underline{y}$ is

$$R_y = E\{\underline{y}\,\underline{y}^T) = E(A^T \underline{x}\,\underline{x}^T A) = A^T R_x A$$

Principal Component Analysis

- R_x is symmetric, and thus its eigen-vectors are orthogonal
- Hence, if the mapping *A* is chosen such that its columns are the eigen-vectors of R_x, then R_y is diagonal:

$$R_y = A^T R_x A = \Lambda$$

- Λ is a diagonal matrix, and its elements are the eigen-values of R_x, λ_i, *i=0,1,..N-1*
- Since the autocorrelation matrix of the new vector $\underline{y}$ is diagonal, its elements, the new features, are uncorrelated
- The mapping above is also called Karhunen-Loeve transform (KLT), and its result is a new feature vector with uncorrelated features

Principal Component Analysis

- The columns of the mapping matrix *A* constitute a basis for the set of data samples vectors $\underline{x}$, thus:

$$\underline{x} = \sum_{i=0}^{N-1} y(i)\underline{a}_i, \qquad y(i) = \underline{a}_i^T \underline{x}$$

  Namely, the original data vector may be expressed as a linear combination of the *N* columns vectors of the mapping matrix *A*

- A new vector $\underline{\hat{x}}$ is defined, which is a linear combination of only m<N basis vectors :

$$\underline{\hat{x}} = \sum_{i=0}^{m-1} y(i)\underline{a}_i,$$

Principal Component Analysis

- If the vector $\underline{\hat{x}}$ is used as an estimate to the data vector $\underline{x}$, then the average square error is : $E\{(\underline{x}-\underline{\hat{x}})^2\} = E\{[\sum_{i=m}^{N-1} y(i)\underline{a}_i]^2\}$
- It is desired to choose for the representation of $\underline{\hat{x}}$ the eigen-vectors $\underline{a}_i$, that produce the minimal error
- From the orthogonality of the eigen-vectors:

$$E\{[\sum_{i=m}^{N-1} y(i)\underline{a}_i]^2\} = E\{\sum_i \sum_j (y(i)\underline{a}_i^T)(y(j)\underline{a}_j)\} =$$

$$= \sum_{i=m}^{N-1} E\{y^2(i)\} = \sum_{i=m}^{N-1} \underline{a}_i^T E\{\underline{x}\underline{x}^T\}\underline{a}_i$$

Principal Component Analysis

- a_i and λ_i are the eigen-vectors and eigen-values of the correlation matrix R_x, hence,

$$R_x \underline{a}_i = \lambda_i \underline{a}_i$$

  and it follows that

$$E\{(\underline{x} - \underline{\hat{x}})^2\} = E\{[\sum_{i=m}^{N-1} \underline{a}_i^T \lambda_i \underline{a}_i]\} = \sum_{i=m}^{N-1} \lambda_i$$

- Therefore, if it is desired to represent the N-features vector $\underline{x}$ with an m-features vector $\underline{\hat{x}}$, the *m* features with the largest eigen-values are chosen. Then, the representation error, the sum of the other eigen-values, is minimized.
- It can be shown that this error is the minimal possible in representing an N-features vector by an m-features one.

Principal Component Analysis

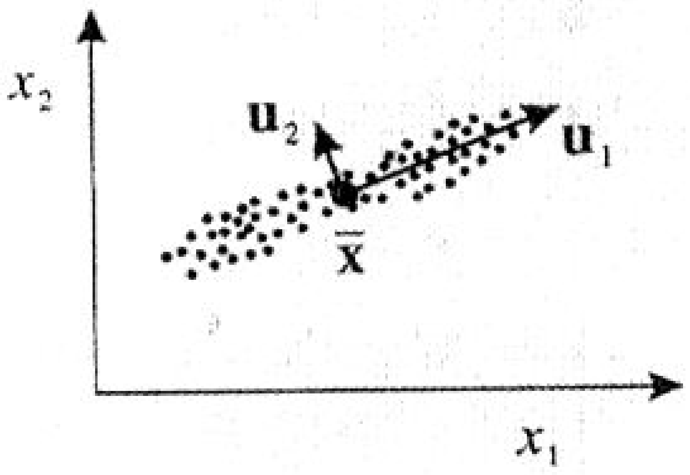
- A different version of the PCA is based on the covariance matrix Σ_x, rather than the correlation matrix R_x. The algorithm then is:
  - Given are *L* vectors $\underline{x}_1, \underline{x}_2, \ldots \underline{x}_L$ (*N*-dimensional)
  - The covariance matrix is $\Sigma(i,j) = E\{(x_i - \mu_i)(x_j - \mu_j)\}$
  - Its estimate for the *L* given vectors is

$$\hat{\Sigma} = \frac{1}{L}\sum_{i=1}^{L}(\underline{x}_i - \underline{\mu}_i)(\underline{x}_j - \underline{\mu}_j)^T$$

  - The eigen-vectors $\underline{a}_1, \underline{a}_2, \ldots \underline{a}_N$ and the eigen-values $\lambda_1 > \lambda_2, \ldots > \lambda_N$ of the covariance matrix are calculated
  - To reduce the number of features, the *m* eigen-vectors with the largest eigen-values are selected as features
  - These m vectors are called the principal components, hence the name of the method PCA
  - These features are a linear combination of the data vectors, hence, in general they lack physical meaning

Principal Component Analysis

- The PCA algorithm maps the given data vectors on the directions (basis vectors) where the variance of the data vectors is maximal.
- The size of the eigen-value λ_i corresponds to the variance of the data vectors in the direction of the eigen-vector $\underline{a}_i$



- It can be shown that the mean square error using *m* out of *N* features after PCA mapping is

$$\bar{e} = \frac{1}{2} \sum_{i=m+1}^{N} \lambda_i$$

Principal Component Analysis

- The PCA algorithm maintains, after the reduction in the number of features, the maximal variance : out of all sets of m (out of N) features, the one of PCA has the largest sum of variances.
- PCA algorithm is optimal regarding the mean-square error, and a good data compression is achievable
- However, the reduction in the number of features using PCA does not lead necessarily to the best separation among classes in the reduced features space

Principal Component Analysis

- PCA algorithm example:

Given is the covariance matrix of the features vector $\underline{x}$,

$$\Sigma_x = \begin{pmatrix} 6 & 2 & 0 \\ 2 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}$$

$\underline{x}$ is of 3 correlated features.

The eigen-vectors of the matrix are :

$\underline{a}_1=(0.918, 0.392, -0.067)^T$, $\underline{a}_2=(0.333, -0.667, 0.667)^T$ $\underline{a}_3=(-0.217, 0.634, 0.742)^T$

The eigen-values of the matrix are $\underline{\lambda} = (6.854, 2, 0.146)^T$

The linear mapping which maps the vector $\underline{x}$ to a new vector with uncorrelated features is

$$\underline{y} = A\underline{x} = (\underline{a}_1 \;\; \underline{a}_2 \;\; \underline{a}_3)^T = \begin{pmatrix} 0.918 & 0.333 & -0.217 \\ 0.392 & -0.667 & 0.634 \\ -0.067 & 0.667 & 0.742 \end{pmatrix}^T \underline{x}$$

The covariance matrix of the new features vector $\underline{y}$ is then diagonal

$$\Sigma_y = A\Sigma_x A^T = \begin{pmatrix} 6.854 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0.146 \end{pmatrix}$$

The elements on the diagonal are the variances of the new features

Principal Component Analysis

- PCA algorithm example continued:

The mapping is inverse : $\underline{x} = A^{-1}\underline{y} = A^{T}\underline{y}$

The reduction in the number of features is obtained by neglecting one or more of the eigen-vectors, those which correspond to the lower eigen-values.

A two-dimensional features vector is obtained by neglecting the third eigen-vector, the one corresponding to $\lambda_i=0.146$

Then,

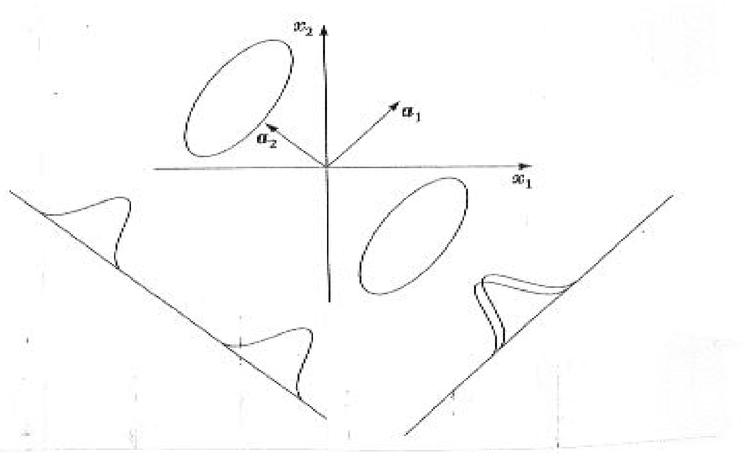
$$\underline{y} = B\underline{x} = \begin{pmatrix} 0.918 & 0.333 \\ 0.392 & -0.667 \\ -0.067 & 0.667 \end{pmatrix}^{T} \underline{x}$$

Now, the original features vector can only be restored approximately

$$\underline{\hat{x}} = B^{T}\underline{y} = \begin{pmatrix} 0.918 & 0.333 \\ 0.392 & -0.667 \\ -0.067 & 0.667 \end{pmatrix} \underline{y}$$

Principal Component Analysis

- Reducing the number of features using PCA algorithm does not lead necessarily to a better separation of the clusters in the reduced features space
- This is because the optimization using PCA is of the mean-square error and not of groups separation (which is measured by the separability measure)
- This result is demonstrated by the following figure:



- The features using PCA are $\underline{a}_1$ and $\underline{a}_2$, and $\underline{a}_1$ is chosen as a single feature since the variance is larger along this direction.

Principal Component Analysis

- However, it can be seen that projecting the data on the $\underline{a}_1$ axis, both groups almost coincide – the separation is bad. Using $\underline{a}_2$ as a single feature (projecting the data on this direction) keeps the groups separated.
- For groups separation the method of Multiple Discriminant Analysis (MDA) is preferred.

Multiple Discriminant Analysis

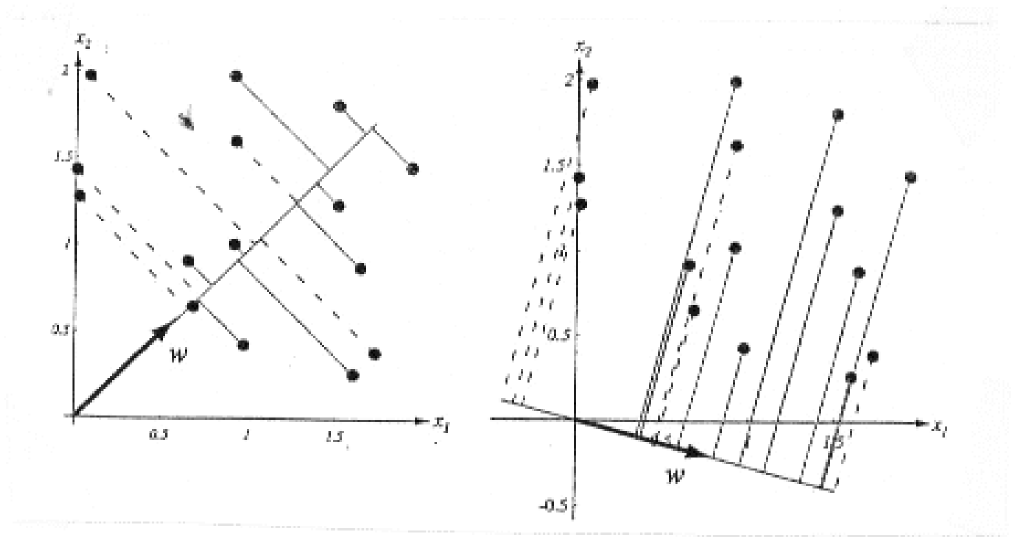
- In this approach a linear mapping of the features is sought that will keep the separation of the classes
- Initially, a mapping of the n-dimensional features vector to a single feature is sought. It is obvious that even if the samples are well separated in the features space, the mapping into a single feature may lead to indistinguishable clusters.
- Still, there may be a single feature along which the classes are reasonably separated

Multiple Discriminant Analysis

- *L* samples are given, each n-dimensional. L_1 belong to class C_1, and L_2 belong to class C_2. A linear mapping $\underline{w}$ is performed on them, which produces a single feature:

$$y = \underline{w}^T \underline{x}$$

- Two possible mappings to a single feature are shown



Multiple Discriminant Analysis

- It is clear that if the set of samples is not well separated in the n-dimensional features space, there is no mapping into a single feature that can make the separation better
- The original average vectors (class centers) $\underline{\mu}_i = \frac{1}{L_i} \sum_{\underline{x} \in C_i} \underline{x}$ are mapped to the centers of the projection

$$m_i = \frac{1}{L_i} \sum_{y \in C_i} y = \frac{1}{L_i} \sum_{\underline{x} \in C_i} \underline{w}^T \underline{x} = \underline{w}^T \underline{\mu}_i$$

- The distance between the centers of the mapped classes is

$$|m_1 - m_2| = \left| \underline{w}^T (\underline{\mu}_1 - \underline{\mu}_2) \right|$$

Multiple Discriminant Analysis

- A large distance between the centers (averages) is not sufficient for class separation. A small within-class variance is also required
- The variance of the mapped samples in both classes is

$$\sigma_i^2 = \frac{1}{L_i} \sum_{y \in C_i} (y - m_i)^2$$

- A common separation measure is $J(\underline{w}) = \frac{|m_1 - m_2|^2}{\sigma_1^2 + \sigma_2^2}$
- To obtain the separation measure as a function of $\underline{w}$

$$\sigma_i^2 = \frac{1}{L_i} \sum_{\underline{x} \in C_i} (\underline{w}^T \underline{x} - \underline{w}^T \underline{\mu}_i)^2 =$$

$$= \frac{1}{L_i} \sum_{\underline{x} \in C_i} \underline{w}^T (\underline{x} - \underline{\mu}_i)(\underline{x} - \underline{\mu}_i)^T \underline{w} = \underline{w}^T \hat{\Sigma}_i \underline{w}$$

Multiple Discriminant Analysis

- The denominator of *$J(\underline{w})$* is then

$$\sigma_1^2 + \sigma_2^2 = \underline{w}^T \hat{\Sigma}_1 \underline{w} + \underline{w}^T \hat{\Sigma}_2 \underline{w} = \underline{w}^T S_w \underline{w}$$

- Similarly, the nominator is:

$$(m_1 - m_2)^2 = [\underline{w}^T (\underline{\mu}_1 - \underline{\mu}_2)]^2 = \underline{w}^T (\underline{\mu}_1 - \underline{\mu}_2)(\underline{\mu}_1 - \underline{\mu}_2)^T \underline{w} = \underline{w}^T S_B \underline{w}$$

- Then

$$J(\underline{w}) = \frac{|m_1 - m_2|^2}{\sigma_1^2 + \sigma_2^2} = \frac{\underline{w}^T S_B \underline{w}}{\underline{w}^T S_w \underline{w}}$$

  where S_W is the intra-class dispersion matrix and S_B is the inter-class dispersion matrix

- It can be shown that the mapping vector $\underline{w}$ which produces a maximal value to *$J(\underline{w})$* is such that

$$S_B \underline{w} = \lambda S_w \underline{w} \qquad \text{for some constant } \lambda$$

Multiple Discriminant Analysis

- If S_w is non-singular, then

$$S_w^{-1} S_B \underline{w} = \lambda \underline{w}$$

and $\underline{w}$ is an eigen-vector of the matrix $S_w^{-1} S_B$, which also satisfies

$$\underline{w} = S_w^{-1}(\underline{\mu}_1 - \underline{\mu}_2)$$

- After performing the mapping, the classification task is of a single feature. In most cases, reducing the number of features to one entails an increase of the classification error. Occasionally, such an increase is acceptable, for the ease of working with a single feature
- It was shown that when the features vector $\underline{x}$ is normally distributed, with an equal covariance matrix Σ in all classes, the separation line between classes using maximum likelihood classification is

$$\underline{w}^T \underline{x} + w_0 = 0 \,, \quad \underline{w} = \Sigma^{-1}(\underline{\mu}_1 - \underline{\mu}_2)$$

which is the same $\underline{w}$ for which the separation measure $J(\underline{w})$ is maximal (separation and classification are then the same)

Multiple Discriminant Analysis

- When the classification task includes C classes, and it is desired to map the n-dimensional features vector $\underline{x}$ into a d-dimensional features vector $\underline{y}$, the previous result has to be extended.
- The intra-class dispersion matrix is then $S_w = \sum_{i=1}^{C} p(C_i)\hat{\Sigma}_i$ and the inter-class dispersion matrix is

$$S_B = \sum_{i=1}^{C} p(C_i)(\underline{\mu}_i - \underline{\mu}_0)(\underline{\mu}_i - \underline{\mu}_0)^T , \quad \underline{\mu}_0 = \sum_{i=1}^{C} p(C_i)\underline{\mu}_i$$

- mappings from n features to one:

$$y_i = \underline{w}_i^T \underline{x} , \qquad i = 1,2,....,d$$

or $$\underline{y} = W\underline{x}$$

when the columns of the matrix W are the mappings $\underline{w}_i$

Multiple Discriminant Analysis

- The *d* mappings $\underline{w}_i$ are the *d* eigen-vectors with the largest eigen-values, solving the equation

$$S_w^{-1} S_B \underline{w}_i = \lambda_i \underline{w}_i$$

  provided S_w is non-singular
- The eigen-values λ_i are found from the characteristic equation

$$\left| S_B - \lambda_i S_w \right| = 0$$

- Once found, λ_i may be substituted into the equation

$$(S_B - \lambda_i S_w) \underline{w}_i = 0$$

  to find the *d* mappings $\underline{w}_i$
- Using the MDA approach a mapping is searched which reduces the number of features while preserving the best separation of classes

Multiple Discriminant Analysis

- The following figure illustrates two possible mapping from 3-dimensional features vector into 2-dimensional. The mapping $\underline{w}_1$ (on a plane perpendicular to $\underline{w}_1$) generates a better class separation