Computer Vision

MSc Informatics option GVR
James L. Crowley

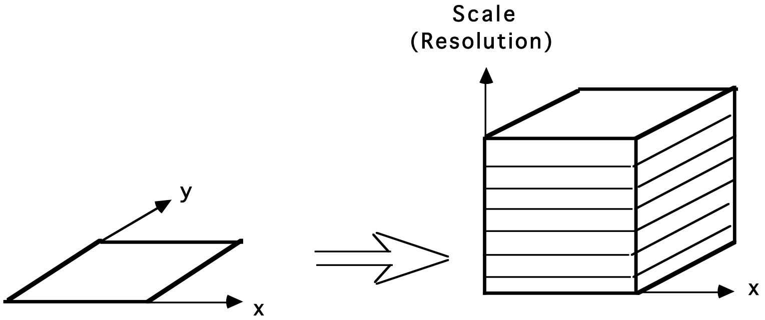
Fall Semester 19 November 2010

Lesson 6

View Invariant Bayesian Recognition

Lesson Outline:

1 Image Scale Space



Our objective is to describe the appearance at each pixel of the image in a manner that does not change (or changes very little) with position, illumination color, distance, or viewing direction.
We will obtain invariance to position by using shift invariant filters
We will obtain invariance to illumination color by estimating the illumination color and correcting the image description in LC_1C_2.
We will obtain invariance to distance by working in a Scale Space

Continuous Scale Case.

Let P(x,y) be the image.
Let G(x, y, σ) by a Gaussian function of scale $\sigma=2^{s/2}$

The image Scale Space is a 3D continuous space P(x,y,s)

$$P(x, y, s) = P(x,y)* G(x, y, 2^{s/2})$$

Note that the scale axis (s) is logarithmic. $s = 2 \text{ Log}_2(\sigma) = \text{Log}_2(2^{s/2})$

Discrete Scale Space

Let P(i,j) be a discrete representation for $P(x,y) = P(i\Delta x, j\Delta x)$
Suppose P(i,j) is an image array of size M x M pixels.

We propose to sample scale with a step size of $\Delta\sigma = 2^{1/2}$ so that $\sigma_k=2^{(k+1)/2}$

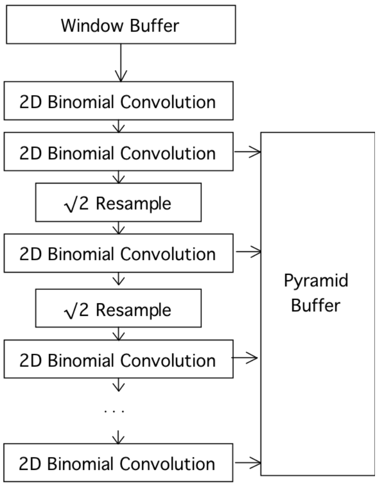
$$P(x, y, s) = p(i\ \Delta x_k, j\ \Delta x_k, k) \quad \text{such that } \Delta x_k = 2^{k/2}$$

For a Gaussian Kernel filter $G(i,j,k) = G(x, y, \sigma_k=2^{k/2})$

The image pyramid becomes :

$$P(i, j, k) = P(x/2^{k/2}, y/2^{k/2}, k) = P(x, y) * G(x, y, 2^{(k+1)/2})$$

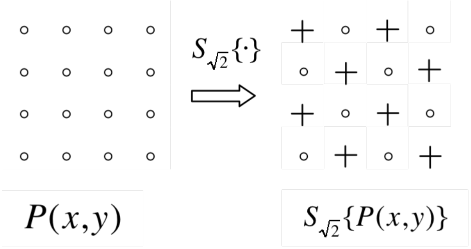
We can compute a discrete Scale Space with a Resampled Pyramid algorithm:



The cumulative Variance, Scale, and sample rates are:

k	$\sigma_k^2 = 2^{k+1}$	$\sigma_k = 2^{\frac{k+1}{2}}$	$\Delta x = 2^{\frac{k}{2}}$
0	2	$\sqrt{2}$	1
1	4	2	$\sqrt{2}$
2	8	$2\sqrt{2}$	2
3	16	4	$2\sqrt{2}$
4	32	$4\sqrt{2}$	4
5	64	8	$4\sqrt{2}$
6	128	$8\sqrt{2}$	8
7	256	16	$8\sqrt{2}$
8	512	$16\sqrt{2}$	16
9	1024	32	$16\sqrt{2}$
10	2048	$32\sqrt{2}$	32
11	4096	64	$32\sqrt{2}$
12	8192	$64\sqrt{2}$	64
13	16384	128	$64\sqrt{2}$
14	32768	$128\sqrt{2}$	128
15	65536	256	$128\sqrt{2}$
15	131072	$256\sqrt{2}$	256

Note that odd levels are resampled on a $\sqrt{2}$ grid:

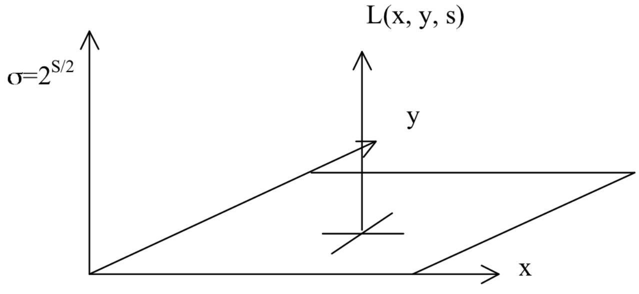


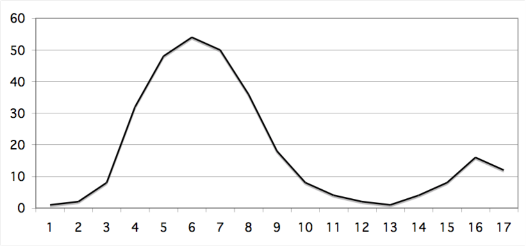
Laplacian Profile

At an every image point, the Laplacian profile the Laplacian of the image computed over a continuous (exponential) range of scales.

$$L(x, y, s) = P(x, y)* \nabla^2 G(x, y, 2^{s/2})$$

The Laplacian profile is invariant to rotation and equivariant to changes in distance.





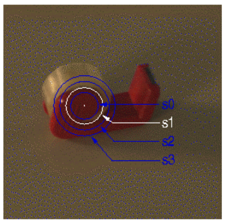
A change in viewing distance at x, y shifts the function L(x,y,s) in s.
The function remains the same. Thus the maximum is a local invariant.

The "intrinsic" scale at a point (x, y) is $\sigma_i = 2^{s_i/2}$

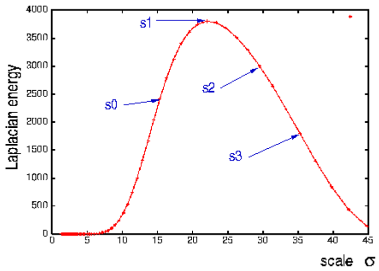
where:

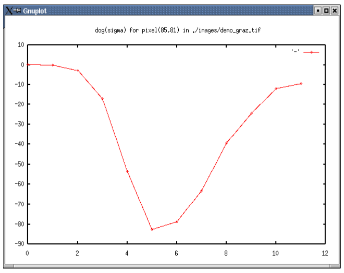
$$s_i = \arg-\max_{s}\{L(x,y,s)\}$$

Examples:



zero crossing of Laplacian at si





dog(sigma) for pixel(85,81) in ./images/demo_graz.tif

The scale of the maximal Laplacian is an invariant at ALL image points.

Scale Invariant Interest Points

Maximal points in the image derivatives provide keypoints.
In an image scale space, these points are scale invariant.

Example: maxima in the Lapacian as invariant "interest points"

Recall the Laplacian of the image :

$$\nabla^2 P(x,y,s) = P * \nabla^2 G(x,y,\sigma) = P * G_{xx}(x,y,\sigma) + P * G_{yy}(x,y,\sigma) \approx P * \nabla^2 G(x,y,\sigma_1) - P * \nabla^2 G(x,y,\sigma_2)$$

We can compute the Laplacian from a Gaussian Pyramid as a difference of samples at adjactent levels.

DoG: $L(i, j, k) = \nabla^2 P(i, j, k) = P(i, j, k) - P(i, j, k-1)$

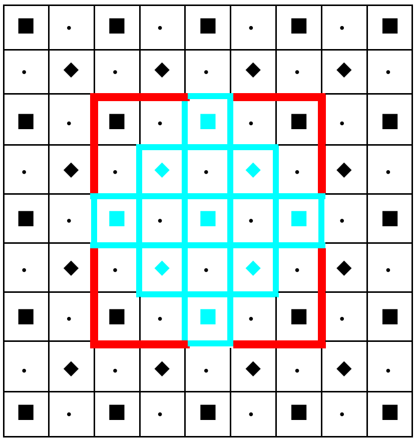
This is referred to as a "Difference of Gaussian" or DoG detector.

We can detect scale invariant interest points as

$$X(i,j,k) = \underset{i,j,k,R}{local - \max}\{L(i,j,k)\}$$

with R=1 at k, and R=2 at k-1, and k+1.

Note that because of resampling, $\Delta x=2^{k/2}$, the neighborhood grows larger as k increases.



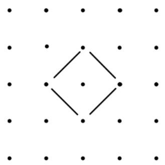
Level k-1 - Red arg-max over i, j for R=2
Level k - Blue (Note neighborhood, $R=\sqrt{2}$ is smaller than level k-1!
level k+1 black. arg-max over i, j for R=2.

Such points are used for tracking, for image registration, and as feature points for recognition.

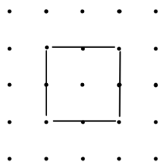
Neighborhoods in a Pyramid

Computing a variable Radius Local-Max operator over a $\sqrt{2}$ image pyramid can be somewhat complex.

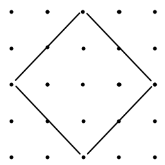
R=1 4 Nearest Neighbors - Odd (Cartesian) levels :



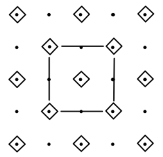
R=$\sqrt{2}$ 8 Nearest Neighbors - Odd (Cartesian) levels :



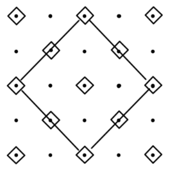
R=2 12 Nearest Neighbors Odd Levels



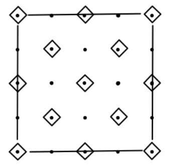
R=1 4 Nearest Neighbors - Even ($\sqrt{2}$)levels



R=$\sqrt{2}$ 8 Nearest Neighbors - Even ($\sqrt{2}$)levels



R=2 12 Nearest Neighbors Even ($\sqrt{2}$)levels



Other popular interest point detectors.

Other popular detectors for scale invariant interest points include:

Gradient Magnitude: $X(i,j,k) = \underset{i,j,k}{Local-}\max\{\| P_x(i,j,k), P_y(i,j,k) \|\}$

and Determinant of the Hessian: $X(i,j,k) = \underset{i,j,k}{Local-}\max\left\{\det\begin{pmatrix} P_{xx}(i,j,k) & P_{xy}(i,j,k) \\ P_{xy}(i,j,k) & P_{yy}(i,j,k) \end{pmatrix}\right\}$

$$X(i,j,k) = \underset{i,j,k}{Local-}\max\left\{P_{xx}(i,j,k)P_{yy}(i,j,k) - P_{xy}(i,j,k)^2\right\}$$

and the Harris-Laplace.

let $b_2(i,j) = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$

$$H_x^2 = b_2 * P_x^2$$
$$H_{xy} = b_2 * P_{xy}$$
$$H_y^2 = b_2 * P_y^2$$

$$H = \begin{pmatrix} {H_x}^2 & H_{xy} \\ H_{xy} & {H_y}^2 \end{pmatrix}$$

Harris interest points h(i,j,k) = arg-max{det(H)-Trace(H)}

2 HoG: Histogram of Gradients

A local histogram of gradient orientation provides a vector of features image appearance that is equivariant that is relatively robust to changes in orientation and illumination.

HoG gained popularity because of its use in the SIFT feature point detector (described next). It was subsequently explored and made popular by Navneet Dalal (M2R GVR 2003) and Bill Triggs.

Recall: The orientation of a gradient at pyramid sample (i,j,k) is:

$$\theta(i,j,k) = Tan^{-1}\left\{\frac{P_y(i,j,k)}{P_x(i,j,k)}\right\}$$

This is a number between 0 and π. We can quantize it to a value between 1 and N value by

$$a(i,j,k) = N \cdot Trunc\left\{\frac{\theta(i,j,k)}{\pi}\right\}+1$$

We can then build a local histogram for a window of size WxH, with upper left corner at i_o, j_o, k. We allocate a table of N cells: h(a). Then for each pixel i,j in our window:

$$\overset{W}{\underset{i=1}{\forall}}\ \overset{H}{\underset{j=1}{\forall}}\ h(a(i+i_o, j+j_o, k)) = h(a(i+i_o, j+j_o, k))+1$$

The result is a local feature composed of N values.
Recall that with histograms, we need around 8 samples per bin to have a low RMS error. Thus a good practice is to have N=W=H. For example N=4, W=4 and H=4. Many authors ignore this and use values such as N=8, W=4, H=4, resulting in a sparse histogram.

Remark: A fast version when N=4 replaces the inverse tangent by computing the diagonal derivatives with differences:

$$P_{\frac{\pi}{4}}(i,j,k) = P(i+1,j+1,k) - P(i-1,j-1,k)$$
$$P_{\frac{\pi}{2}}(i,j,k) = P(i,j+1,k) - P(i,j-1,k)$$
$$P_{\frac{3\pi}{4}}(i,j,k) = P(i+1,j-1,k) - P(i-1,j+1,k)$$
$$P_{\pi}(i,j,k) = P(i+1,j,k) - P(i-1,j,k)$$

To determine a(i,j,k) simply choose the maximum.

3 Scale Invariant Feature Transform (SIFT)

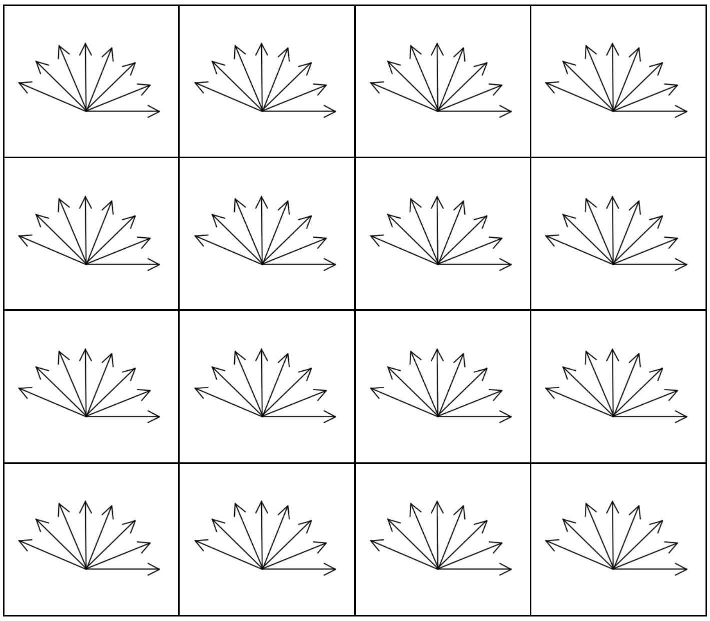
SIFT uses a scale invariant pyramid to compute a scale invariant DoG interest point detector to detect local scale-invariant interest points.

$$X(i,j,k) = \underset{i,j,k,R=2}{Local-\max}\{P(i,j,k) - P(i,j,k-1)\}$$

It then computes a U x V grid of HoG detectors with N=8, W=4, H=4 at the level k Typically U=V=4.

At level k, $\Delta i=\Delta j=2^{k/2}$

This gives 16 x 16 = 128 features at each interest point.
This feature vector is invariant to changes in position and scale and very robust with changes in image plane rotation and illumination intensity.



Various authors experiment with other grid sizes.

For example, let the grid size be G.

G=4, W=4, H=4, N=4

Gives 64 features.

4 Integral Images

An integral image, *ii(x,y)* of a window *p(x,y)* is the sum of all pixels from the upper left corner (1,1) to the current pixel (x,y).

$$ii(x,y) = \sum_{u=1,v=1}^{x,y} p(u,v)$$

A recurrence formula may be used to compute the integral image in 2 operations (memory access and additions) per pixel. This operation starts by computing an intermediate sum for each row:

$$s(x,y) = s(x,y-1) + i(x,y)$$

This intermediate sum is then used to compute the sum of rectangles.

$$ii(x,y) = ii(x-1,y)+s(x,y)$$

Thus the total cost of computing ii(x,y) for a window of size W·H is 2WH adds.

Any sum of the image within any rectangle from (x_1,y_1) to (x_2,y_2) can be computed as the sum or difference of 4 values:

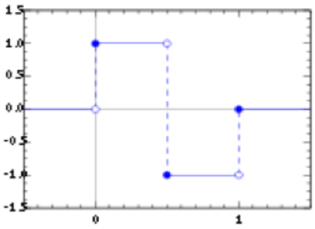
$$r(x_1, y_1, x_2, y_2) = ii(x,y) – ii(x_1,y_2) – ii(x_2,y_1) + ii(x_1,y_1)$$

Thus integral images can be used to provide VERY fast computation of “box” features. These can be used to compute Haar wavelets.

Haar Wavelets

Haar A. Zur Theorie der orthogonalen Funktionensysteme, Mathematische Annalen, 69, pp 331–371, 1910.

The Haar wavelet is a difference of rectangular Windows.



$$h(t) = \begin{cases} 1 & \text{for } 0 \le t < 0.5 \\ -1 & \text{for } 0.5 \le t < 1 \\ 0 & \text{for } t < 0 \text{ and } t \ge 1 \end{cases}$$

The Haar wavelet may be shifted by d and scaled by s

$$h(t;s,d) = h(t/s - d)$$

Note that the Haar Wavelet is zero gain (zero sum).

$$G = \int_{-\infty}^{\infty} h(t)dt = 0$$

The Digital (discrete sampled) form of Haar wavelet is

$$h(n;d,k) = \begin{cases} 1 & \text{for d} \le n < d + k/2 \\ -1 & \text{for d + k/2} \le n < d + k \\ 0 & \text{for } n < d \text{ and } n \ge d + k \end{cases}$$

Haar wavelets can be used to define an orthogonal transform analogous to the Fourier basis. This can be used to define an orthogonal transform (the Walsh-Hadamard Transform). The basis is

$$H_0 = +1$$

$$H_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H_2 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

…

$$H_m = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{bmatrix}$$

Haar Functions, and the Walsh-Hadamard transform have been used in Functional Analysis and signal processing for nearly a century.

In the 1980s the Wavelet community re-baptized the Haar functions as "wavelets" and demonstrated that the Walsh-Hadamard transform is the simplest form of wavelet transform.
A 2-D form of Walsh-Hadamard transform may be defined using DoB features.

5 Bayesian Recognition

Recognition is a fundamental ability for intelligence, and indeed for all life.
To survive, any creature must be able to recognize food, enemies and friends.

Recognition: The fact to recognize, to identify an object as itself.
Identify: To recognize an entity as an individual
Classify: The recognize an individual as a member of a class.

A class is defined by a membership test.

Classification is a process of associated an entity (or an event) as a member of a class. The entity is described by a vector of features, provided by an observation.
The assignment of an entity to a class provided by a test made on the feature vector.

Features: observable properties that permit discrimination between classes.
A set of D features, x_d, are assembled into a feature vector $\vec{X}$

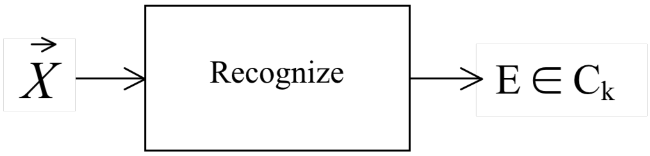
$$\vec{X} = \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_D \end{pmatrix}$$

The Pattern Recognition Problem

Assume that we have a sensor that produces discrete observations of the world. Each observation is an event, E. Assume that for each observation, the sensor provides a vector of D features, $\vec{X}$

Observation: $(E, \vec{X})$

Our problem is to build a box that assigns each observation to one of K classes $\{C_k\}$ labeled k=1 to K.



This problem is known as "Decision Theory". $\hat{\omega}_k = decide(E \in C_k)$

We can decompose this into two component functions $d()$ and $y(\vec{X})$:

$$\hat{\omega}_k \leftarrow d(y(\vec{X}))$$

Where $y(\vec{X})$ is a discriminant function that maps $R^D \rightarrow R^K$
d() is a decision function d(): $R^K \rightarrow \{\hat{\omega}_k\}$

Generally we choose d() to make as few mistakes as possible.
We can express this mathematically using probability theory as:

$$\hat{\omega}_k = \underset{\omega_k}{\arg-\max}\{p(\omega_k \mid \vec{X})\}$$

In this case, our primary tool is Bayes Rule, that tells us:

$$p(\omega_k \mid \vec{X}) = \frac{p(\vec{X} \mid \omega_k)}{p(\vec{X})} p(\omega_k)$$

In general, $p(\vec{X})$, $p(\vec{X} \mid \omega_k)$ and $p(\omega_k)$ are estimated from a set of training data composed of M sample observations $\{\vec{X}_m\}$ labeled with an "indicator" variable $\{t_m\}$ telling the class k for each observation.

Equivalently, we can partition the training set $\{\vec{X}_m\}$ into K subsets $\{X_m^k\}$ each of which contains M_k samples.

Typically $p(\omega_k)$ is estimated as $p(\omega_k) = \frac{M_k}{M}$ although this can also be obtained from other sources.

The Gaussian density that allows us to estimate

$$p(\vec{X} \mid \omega_k) = \mathcal{N}(\vec{X} \mid \vec{\mu}_k, \Sigma_k) = \frac{1}{(2\pi)^{\frac{D}{2}} \det(\Sigma_k)^{\frac{1}{2}}} e^{-\frac{1}{2}(\vec{X}-\vec{\mu}_k)^T \Sigma_k^{-1} (\vec{X}-\vec{\mu}_k)}$$

$$p(\vec{X}) = \sum_{k=1}^{K} p(\vec{X} \mid \omega_k) = \sum_{k=1}^{K} \mathcal{N}(\vec{X} \mid \vec{\mu}_k, \Sigma_k)$$

Where the parameters $\vec{\mu}_k$ (mean) and Σ_k (covariance) for $p(\vec{X} \mid \omega_k)$, as well as $p(\omega_k)$ are estimated from the training data $\{\vec{X}_m\}$ and $\{t_m\}$.

Generative vs Discriminative Approaches

Generally, there are three approaches to building a decision function:

1) Generative Approach: Construct an explicit estimate of the probabilities $p(\vec{X} | \omega_k)$, $p(\vec{X})$ and $p(\omega_k)$ and use Bayes rule to compute the most likely decision.

2) Discriminative Approach: Use Bayesian theory to construct a function that partitions the feature space $\vec{X}$ into discrete regions for each class K.

3) Ad-hoc approach: "invent" an arbitrary theory of tests. This generally gives unreliable results.

Generative approach is more general, but imposes restrictions. In many cases a discriminative approach can provide a more practical solution. We will examine both in this course.

The Gaussian Assumption

The Gaussian assumption underlies many learning techniques.
If we assume that the

1) the feature vector $\vec{X}$ for each class k is generated by a stationary random process, and

2) The process for generating $\vec{X}$ is composed of a sequence of independent random events, then, then probability theory tells us that

$$p(\vec{X} | \omega_k) = \mathcal{N}(\vec{X} | \vec{\mu}_k, \Sigma_k) = \frac{1}{(2\pi)^{\frac{D}{2}} \det(\Sigma_k)^{\frac{1}{2}}} e^{-\frac{1}{2}(\vec{X}-\vec{\mu}_k)^T \Sigma_k^{-1} (\vec{X}-\vec{\mu}_k)}$$

where $\vec{\mu}_k$ is the average (center of gravity) of the features for class k.
and Σ_k is the co-variance of the features for class k.

We can use a training set to "learn" a discriminant function, $y_k(x)$, for each class as:

$$y_k(\vec{X}) = p(\omega_k | \vec{X}) = \frac{p(\vec{X} | \omega_k)}{p(\vec{X})} p(\omega_k)$$

using a labeled set of training samples $\{\vec{X}_m\}$ for which we know the class labels $\{t_m\}$.

The class labels partition the training set $\{\vec{X}_m\}$ into K subsets $\{X_m^k\}$ each of which contains M$_k$ samples, so that $\{X_m\} = \bigcup_{k=1,K} \{X_m^k\}$ and $M = \sum_{k=1}^{K} M_k$

In this case we can use the training data to estimate the parameters $\vec{\mu}_k, \Sigma_k$ for

$$p(\vec{X} \mid \omega_k) = \mathcal{N}(\vec{X} \mid \vec{\mu}_k, \Sigma_k)$$

We can estimate that $p(\omega_k) = \frac{M_k}{M}$

We can note that for the training data

$$p(\vec{X}) = \sum_{k=1}^{K} p(\vec{X} \mid \omega_k) \text{ and thus } p(\vec{X}) = \sum_{k=1}^{K} \mathcal{N}(\vec{X} \mid \vec{\mu}_k, \Sigma_k)$$

This gives

$$p(\omega_k \mid \vec{X}) = \frac{\mathcal{N}(\vec{X} \mid \vec{\mu}_k, \Sigma_k)}{\sum_{k=1}^{K} \mathcal{N}(\vec{X} \mid \vec{\mu}_k, \Sigma_k)} \cdot \frac{M_k}{M}$$

6 Classification by Ratio of Histograms of pixel values

Histograms provide an alternate view of Bayes Rule.

Histograms

As we saw, for integer x from a bounded set of values, such that $x \in [x_{min}, x_{max}]$,

the probability that a random observation X takes on x is

$$P(X=x) = \frac{1}{M} h(x)$$

The validity of this depends on the ratio of the number of sample observations M and the number of cells in the histogram Q=N

This is true for vectors as well as values.

For a vector of D values $\vec{x}$ the table has D dimensions. $h(x_1, x_2, \ldots, x_D) = h(\vec{x})$

The average error depends on the ration $Q=N^D$ and M. : $E_{ms} \sim O(\frac{Q}{M})$

We need to assure that $M >> Q = N^d$

As a general rule : $M = 10N^d$

Example: Object detection by pigment color

We can use Bayes rule to detect objects based on their pigment.

The observed chrominance $C = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$ is a signature for an object.

$$c_1 = r = \frac{R}{R+V+B} \qquad c_2 = v = \frac{V}{R+V+B}$$

Suppose that these are coded with N values between 0 and $N-1$

$$c_1 = \text{Round}\ ((N-1) \cdot \frac{R}{R+G+B})\ c_2 = \text{Round}\ ((N-1) \cdot \frac{G}{R+G+B})$$

Allocate a 2D table $h(c_1, c_2)$., of size N x N.

(for example, for 32 x 32 Q = 32 x 32 = 1024 cellules)

For each pixel in the image $\vec{C} = C(i, j)$

$h(\vec{C}) := h(\vec{C}) +1$

That is $h(c_1, c_2) := h(c_1, c_2) +1$

After M pixels, the chrominance histogram $h(\vec{C})$, gives :

$$P(\vec{C}) \approx \frac{1}{M} h(\vec{C})$$

Consider a region W of M_o pixels of a known object class O.

$$\forall(i,j) \in W \ : \ h_o(\vec{C}(i,j)) := h_o(\vec{C}(i,j)) + 1$$

Then $\vec{C}(i, j) = \begin{pmatrix} r \\ v \end{pmatrix}(i, j)$: $p(\vec{C} \mid \text{objet}) \approx \frac{1}{M_o} h_o(\vec{C})$

Because W is part of the image, the probability of observing a pixel from W is

$$P(W) = \frac{M_o}{M}$$

From Bayes rule, for any pixel $\vec{C}(i, j)$ the probability that it belongs to O is

$$p(\text{objet} \mid \vec{C})$$

for S images de IxJ pixels we have M=S·I·J pixels.

Suppose that each contains a known region of the object, W_s. so that
we have M_o total pixels of the object in the S images.

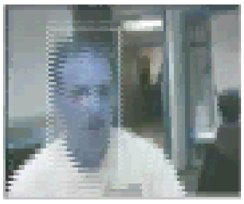
$$p(\text{objet}) = \frac{M_o}{M}$$

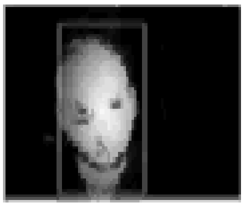
$$p(\vec{C}) = \frac{1}{M} h(\vec{C})$$

$$p(\vec{C} \mid \text{objet}) = \frac{1}{M_o} \; h_o(\vec{C})$$

thus

$$p(\text{objet} \mid \vec{C}) = p(\vec{C} \mid \text{objet}) \frac{p(\text{objet})}{p((\overset{\varnothing}{C}))} = \frac{1}{M_o} \; h_o(\vec{C}) \frac{\frac{M_o}{M}}{\frac{1}{M} h(\vec{C})} = \frac{h_o(\vec{c})}{h(\vec{c})}$$





Histograms of Receptive Field Values

This method can be generalised to ANY vector of feautures. For example, the appearance of a neighborhood give by the receptive field vector.

$\vec{V}(i,j;\sigma_i,\theta_i)$ = P(i,j) * (G_x, G_{xx}, G_{xy}, G_{yy}) at σ_i and θ_i.

ATTENTION. The histogram must have sufficient samples M.

$M \geq 10\ Q \geq 10\ N^{D.}$

For the above exemple: D = 4.

Here is a table of numbers of cells in a histogram of D dimensions of N values.

N \ d	1	2	3	4	5	6
2	2^1	2^2	2^3	2^4	2^5	2^6
4	2^2	2^4	2^6	2^8	2^{10} =1 Kilo	2^{12} =2 Kilo
8	2^3	2^6	2^9	2^{12}	2^{15}	2^{18}
16	2^4	2^8	2^{12}	2^{16}	2^{20} = 1 Meg	2^{24} = 4 Meg
32	2^5	2^{10} =1 Kilo	2^{15}	2^{20} = 1 Meg	2^{25}	2^{30} = 1 Gig
64	2^6	2^{12}	2^{18}	2^{24}	2^{30} = 1 Gig	2^{36}
128	2^7	2^{14}	2^{21} = 2 Meg	2^{28}	2^{35}	2^{42} =2 Tera
256	2^8	2^{16}	2^{24}	2^{32} = 2 Gig	2^{40} = 1 Tera	2^{48}

Consider the chromatic receptive fields normalized in scale and orientation σ_i and θ_i.

$$\vec{P}_{\sigma,\theta} = (P_x^L,\ P_x^{C_1}, P_x^{C_2}, P_{xx}^L, P_{xy}^L,\ P_{xx}^{C_1}, P_{xx}^{C_2}\ )$$

D= 7.

$$p(\text{objet}(i,j)\ |\ \vec{V}(i,j)\ ) = \frac{p(\vec{V}(i,j) \mid object(i,j)}{p(object(i,j))} p(\vec{V}(i,j) \approx \frac{h_o(\vec{V}(i,j))}{h(\vec{V}(i,j))}$$