Computer Vision

MSc Informatics option GVR

James L. Crowley

Fall Semester 26 November 2015

Lesson 7

Scale Invariant Pyramids, HOG, SIFT and Haar

Lesson Outline:

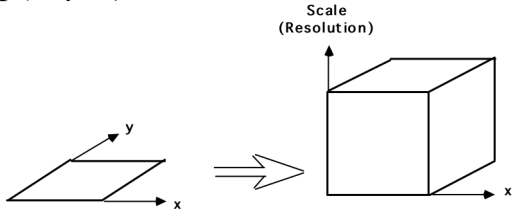
1. Gaussian, Gradient and Laplacian Pyramids

1.1. Scale Space (Rappel)

Let *p(x, y)* be a 2-D signal where *(x, y)* are real values,
Let *G(x, y, σ_s)* by a Normalized Gaussian function of scale σ_s

$$G(x,y,\sigma_s) = \frac{1}{2\pi\sigma_s^2} e^{-\frac{(x^2+y^2)}{2\sigma_s^2}}$$

Scale Space is a continuous 3D space *p(x, y, s)*



$p(x, y, s) = p * G(x, y, \sigma_s)$

Note that the scale axis (s) in scale space is logarithmic

$\sigma_s = \sigma_0^s$ is an exponential axis
$Log(\sigma_s) = s \cdot Log(\sigma_0)$

or $$s = \frac{Log(\sigma_s)}{Log(\sigma_0)}$$

If we use base 2 logarithms and $\sigma_0 = 1$ then $\sigma_s = 2^s$ and $s = Log_2(\sigma_s)$, $s_0 = 0$

A logarithmic scale axis is necessary for scale <u>equivariance</u>.

The appearance of a pattern in the image results in a unique structure in *p(x, y, s)*.
If a shape in an image is made larger by a factor of $D = 2^d$

$p(x,y) \rightarrow p(x2^d, y2^d)$

Then the projection of appearance is translated by d in the scale axis

$p(x, y, s) \rightarrow p(x,y,s+d)$

Scale space is equivariant in position, scale and rotation

Translate a pattern by Δx, Δy and the structure translates by $x+\Delta x$, $y+\Delta y$ in P(x, y, s).
Rotate by θ in x, y and the structure rotates by θ in P(x, y, s).
Scale by a factor of $2^{\Delta s}$, and the structure translates to $s+\Delta s$ in P(x, y, s).
Scale space is an ideal mathematical construct. In a computer we must sample scale space in x, y and s.

1.2. Discrete Scale Space

Let $p(x, y)$ is an image array of size WxH pixels, where (x, y) are integers
A discrete scales space is:

$$p(x, y, k) = p * G(x, y, \sigma_k)$$

For example, we can use a step size of $\Delta\sigma = 2$ so that

$\sigma_k = 2^k$ For k=0, to K.

At k=0 : $\sigma_0 = 2^0 = 1$. where $\sigma_0 = 1$ is the smallest scale that we can represent.

let M = min(W, H) and K=Log_2(M) at k=K, $\sigma_K = 2^K = 2^{log(M)} = M = min(W,H)$

so for k > K the scale parameter σ is larger than the image and the scale space images rapidly tend to constant.

1.3. Spatial Resampling and Image Pyramids

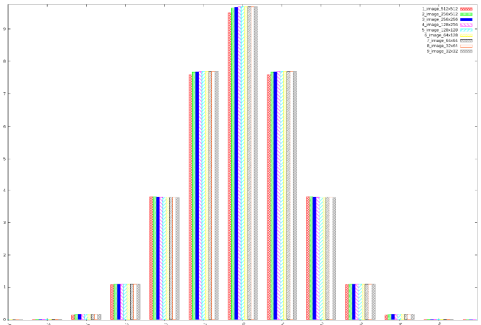
consider $p(x, y, k)$ where $\sigma_k = 2^k$

Because the Gaussian, $G(x, y, \sigma_k)$, is a low pass filter, as σ_k grows it becomes possible to resample the image with a larger step size without loss of information. we can replace (x, y) with $(i \cdot \Delta x, j \cdot \Delta y)$.

$$G(x, y, \sigma_k) \rightarrow G(i \cdot \Delta x_k, j \cdot \Delta y_k, \sigma_k)$$

Sampling theory shows that the sample size at each Δx_k, Δy_k can grow exactly as σ_k. For example, we saw from sample theory that we should use $\sigma \geq \Delta x$.

Thus $\Delta x_k \leq \sigma_k = 2^k$
Resampling $p(x, y, k)$ at $\Delta x_k = \sigma_k = 2^k$ results an identical impulse response at each level. This property is called “scale invariance”.



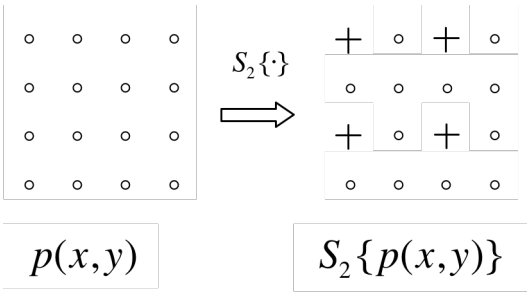
Pyramid samples are at discrete positions ($i\Delta x_k$, $j\Delta y_k$)
for integer values of i, j:

$$p(i, j, k) = p(i\Delta x_k, j\Delta y_k, k) = p(x/\Delta x_k, y/\Delta x_k, k)$$

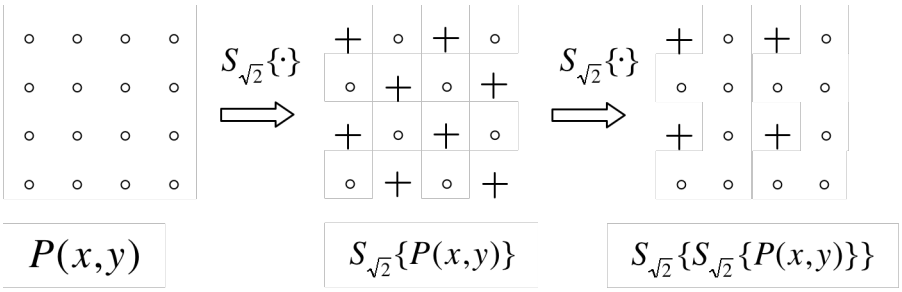
The position in the original image of a sample from level k is $x = i\Delta x_k$ and $y = j\Delta y_k$

A resampled scale space is known as a pyramid.

If we sampling at a scale step of $\sigma_k = 2^k$ this gives a full "octave" pyramid.



It is also possible to build a scale invariant pyramid with a step size of $\Delta\sigma = 2^{k/2}$ using $\sigma_k = 2^{k/2}$ This is known as a “half-octave” pyramid. .



The Half-octave pyramid requires a complicated resampling algorithm that is beyond the scope of this class. For today we will use a full octave pyramid to simplify the explanation.

Let N be the number of pixels in the image.
Normally computing scale space requires Log N convolutions of (N) multiplies.
There are very fast, O(N), recursive algorithms to compute a Gaussian Pyramid.

1.4. Using a Gaussian Pyramid to compute image derivatives at scale

It is possible to use the Gaussian Pyramid to compute image derivatives at scale using sums and differences.

Let *p(x, y)* be an image array of size WxH pixels, where (x, y) are integers,
A full octave Gaussian pyramid of the image is a resampled set of images resulting from the convolution of image with Gaussians at an exponential set of scales.

$$p(x, y, k) = p * G(x, y, \sigma_k)$$

$$p(i, j, k) = p(i\Delta x_k, j\Delta y_k, k) = p(x/\Delta x_k, y/\Delta x_k, k)$$

$$where\ \Delta x_k = \Delta y_k = \sigma_k = 2^k \text{ and } i = x/\Delta x_k \text{ and } j = y/\Delta y_k$$

For any sample *p(i, j, k)* the position in the original image is $x = i\Delta x_k$ and $y = j\Delta y_k$

A pyramid of image derivatives is defined as the convolution of image with derivatives of Gaussians

$$p_x(x,y,k) = p * G_x(x,y,\sigma_k)$$

These are sometimes referred to as "Receptive Fields" because they are similar to the receptive fields observed in the mammalian visual cortex.

With the Gaussian Pyramid, we can obtain a fast approximation to Gaussian derivatives directly by sum and difference of the samples of the Gaussian pyramid.

Then can approximate the Gaussian Derivatives as:

$$p_x(i,j,k) \approx p(i+1,j,k) - p(i-1,j,k) = p(i,j,k) * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

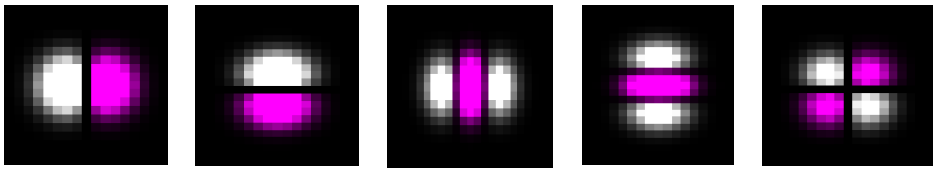
$$p_y(i,j,k) \approx p(i,j+1,k) - p(i,j-1,k) = p(i,j,k) * \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

$$p_{xx}(i,j,k) \approx p(i+1,j,k) - 2p(i,j,k) + p(i-1,j,k) = p(i,j,k) * \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$$

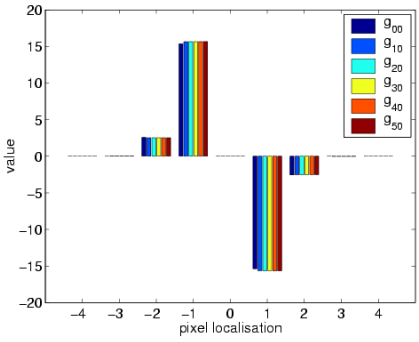
$$p_{yy}(i,j,k) \approx p(i,j+1,k) - 2p(i,j,k) + p(i,j-1,k) = p(i,j,k) * \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

$$p_{xy}(i,j,k) \approx p(i+1,j+1,k) - p(i-1,j+1,k) - p(i+1,j-1,k) + p(i-1,j-1,k) = p(i,j,k) * \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix}$$

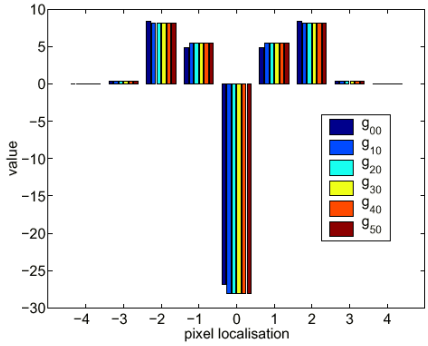
The following are some numerically evaluated examples published in the Scale Space Conference of 2003. [Crowley-Riff 2003]



$G_x(i,j,k)$ $G_y(i,j,k)$ $G_{xx}(i,j,k)$ $G_{yy}(i,j,k)$ $G_{xy}(i,j,k)$



Impulse Response for $G_x(i,j,k)$



Impulse Response for $G_{xx}(i,j,k)$

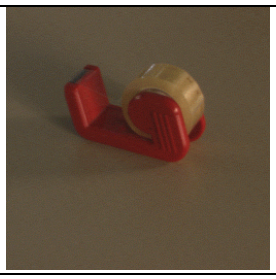
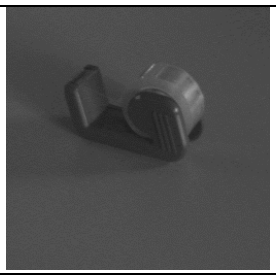
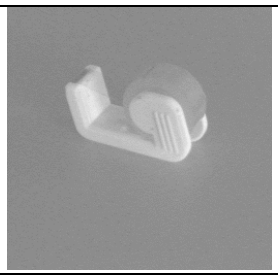
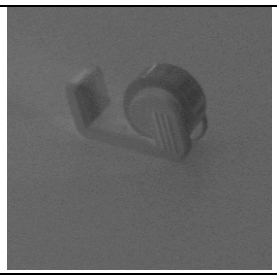
Impulse response for Gaussian derivatives for pyramid levels k=0,1,2,3,4,5.

1.5. Color Opponent Scale Space

In lesson 3 we saw that a color opponent space was useful for illumination invariance

$$(R, G, B) \Rightarrow (L, C_1, C_2) \qquad \begin{pmatrix} L \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} 0.33 & 0.33 & 0.33 \\ -0.5 & -0.5 & 1 \\ 0.5 & -0.5 & 0 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

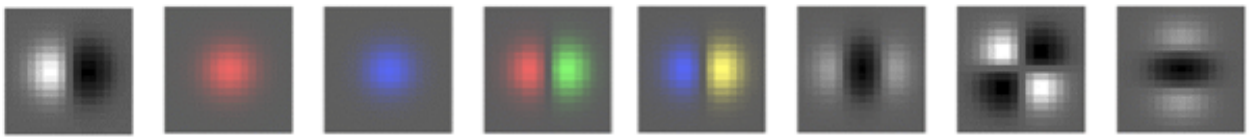
This representation separates luminance and chrominance.

			
RGB		B-W	R-G	R+G-B

Color opponent space can be used to build color opponent receptive fields.

$$\begin{pmatrix} L \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} 0.33 & 0.33 & 0.33 \\ -0.5 & -0.5 & 1 \\ 0.5 & -0.5 & 0 \end{pmatrix} \begin{pmatrix} \alpha_1 R \\ \alpha_2 G \\ \alpha_3 B \end{pmatrix}$$

We then compute 3 pyramids: L(i, j, k), C_1(i, j, k), and C_2(i, j, k),



Examples of color opponent receptive fields.

Color opponent receptive fields can be steered in color to provide color invariance.

This gives us a feature vector for local appearance:

$$\vec{A}(x,y,k) = \begin{bmatrix} G_x^{L\sigma_k} \\ G^{C_1\sigma_k} \\ G^{C_2\sigma_k} \\ G_x^{C_1\sigma_k} \\ G_x^{C_2\sigma_k} \\ G_{xx}^{L\sigma_k} \\ G_{xy}^{L\sigma_k} \\ G_{yy}^{L\sigma_k} \end{bmatrix}$$

This can be generalized to include multiple scales.

1.6. Scale Invariant Interest Points from a Pyramid

Local maximal points in the image derivatives provide landmarks for matching and tracking. Such points can be computed from the local maximum in the Gradient Magnitude or the local maximum in the Laplacian.

Such local maxima are thus called "interest points".

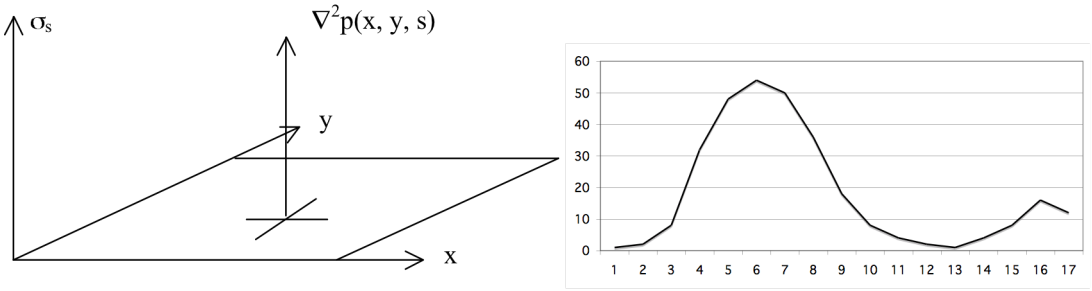
In an image scale space, these points provide landmarks for scale invariant image description.

Recall that the Laplacian of the image is

$$\nabla^2 p(x,y,s) = p * \nabla^2 G(x,y,\sigma_s) = p_{xx}(x,y,s) + p_{yy}(x,y,s)$$

A Laplacian profile for an image point is the Laplacian of the image computed over a continuous (exponential) range of scales.

The Laplacian profile is invariant to rotation and translation and equivariant to changes in scale. Since scale is proportional to distance, the profile is equivariant to viewing distance.



A change in viewing distance at x, y shifts the function $\nabla^2 p(x,y,s)$ in s.
The function remains the same. Thus the maximum is a local invariant.

A Laplacian interest point is $(x_i, y_i, s_i) = \arg\max_{x,y,s} \{\nabla^2 p(x,y,s)\}$

Such interest points can be computed directly from a difference of levels in the Gaussian pyramid. For a Gaussian Scale Space, we can show that:

$$\nabla^2 G_x(x,y,\sigma) = G_{xx}(x,y,\sigma) + G_{yy}(x,y,\sigma) = \frac{\partial G(x,y,\sigma)}{\partial \sigma}$$

As a consequence: $\nabla^2 G(x, y, \sigma) \approx G(x, y, \sigma_1) - G(x, y, \sigma_2)$

This is called a "Difference of Gaussians" (DoG) and requires $\sigma_1 \geq \sqrt{2}\ \sigma_2$
Thus the Laplacian of the image can be approximated as the difference at adjacent pyramid levels from a Gaussian Pyramid.

$$\nabla^2 p(x, y, k) = p(x, y, k) - p(x, y, k-1)$$

Thus, we can detect scale invariant interest points local maxima in the Laplacian.

$$x_i, y_i, s_i = \arg\max_{x,y,k} \{\nabla^2 p(x,y,k) - \nabla^2 p(x,y,k-1)\}$$

we need to use $x = i\Delta x_k$ and $y = j\Delta y_k$ to assure that the sample are at the same image position.

A difference in scale in the pyramid levels is $\Delta\sigma = 2$ is not precise.
We would like to we need a technique to determine intrinsic scale at finer step sizes.

We can also compute an intrinsic scale for the Gradient.

$$\text{The Gradient } \vec{\nabla} p(x,y,s) = \begin{pmatrix} p_x(x,y,s) \\ p_y(x,y,s) \end{pmatrix} = \begin{pmatrix} p * G_x(x,y,s) \\ p * G_y(x,y,s) \end{pmatrix}$$

For any image point (x,y) the intrinsic scale can be computed from

$$s_i = \arg\max_s \{ \| \vec{\nabla} p(x,y,s) \| \}$$

These are positions in the image that can serve as landmarks for tracking or recognition.

In a scale-invariant pyramid, the gradient is available at any sample in the pyramid as

$$\vec{\nabla} p(i,j,k) = \begin{pmatrix} p_x(i,j,k) \\ p_y(i,j,k) \end{pmatrix} = \begin{pmatrix} p(i+1,j,k) - p(i-1,j,k) \\ p(i,j+1,k) - p(i,j-1,k) \end{pmatrix}$$

For the image gradient, a scale invariant interest point is

$$i_i, j_i, k_i = \arg\max_{i,j,k} \{ \| \vec{\nabla} p(i,j,k) \| \}$$

1.7. Natural Interest points at half octave scales.

To obtain a natural interest point with a scale precision of less than $\Delta\sigma=2$ we can use cascade convolution within each pyramid level.

Consider a pyramid image at level k: $p(i, j, k)$ with $\sigma_k = 2^k$

we compute:

$p_1(i, j, k) = p(i, j, k) * G(i, j, 1)$ $\quad\quad$ $\sigma_{k1} = 2^{k+1/2}$
$p_2(i, j, k) = p_1(i, j, k) * G(i, j, 1)$ $\quad\quad$ $\sigma_{k2} = 2^{k+1}$
$p_3(i, j, k) = p_2(i, j, k) * G(i, j, 2)$ $\quad\quad$ $\sigma_{k2} = 2^{k+2}$
$p_4(i, j, k) = p_3(i, j, k) * G(i, j, 4)$ $\quad\quad$ $\sigma_{k2} = 2^{k+4}$

For each pixel local, we can then calculate 4 Laplacian values. :

$L_{k0} = p_1(i, j, k) - p(i, j, k)$
$L_{k1} = p_2(i, j, k) - p_1(i, j, k)$
$L_{k2} = p_3(i, j, k) - p_2(i, j, k)$
$L_{k3} = p_4(i, j, k) - p_3(i, j, k)$

If $L_{k0} < L_{k1} > L_{k2}$ then the point p(i, j, k) is a natural interest point at with $\sigma = 2^{k+1/2}$
If $L_{k1} < L_{k2} > L_{k3}$ then the point p(i, j, k) is a natural interest point with $\sigma = 2^{k+1}$

The position of the natural interest point is $x = i \cdot 2^k$, $y = j \cdot 2^{k-}$

This is the method used to find natural interest points in the SIFT detector described below.

1.8. Other popular interest point detectors.

Other popular interest point detectors include the

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

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

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 $h(i,j,k) = \text{arg-max}\{\det(H)-\text{Trace}(H)\}$

2. HOG: Histogram of Oriented Gradients

A local histogram of gradient orientation provides a vector of features image appearance 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 (CNRS Labo LJK).

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:

$$\forall_{i=1}^{W} \forall_{j=1}^{H} h(a(i+i_o, j+j_o, k)) = h(a(i+i_o, j+j_o, k)) + 1$$

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 scale invariant interest points as shown above.

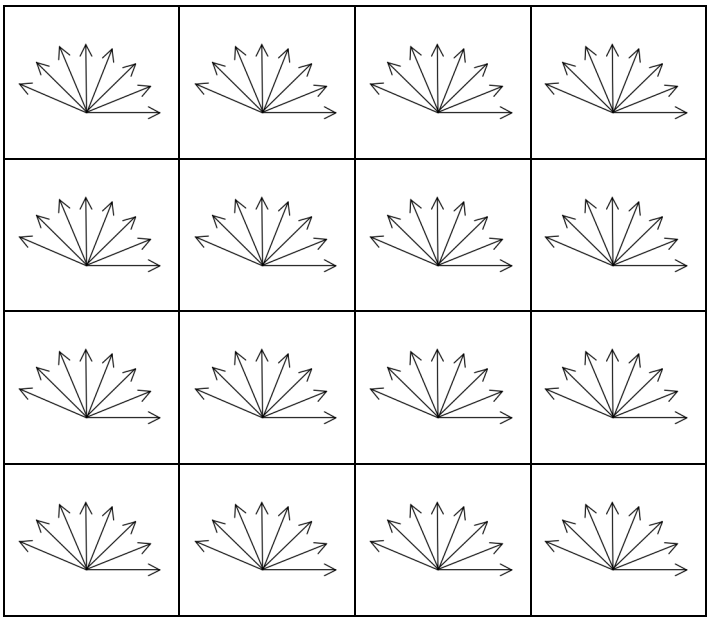
$$L_{k0} = p_1(i, j, k) - p(i, j, k)$$
$$L_{k1} = p_2(i, j, k) - p_1(i, j, k)$$
$$L_{k2} = p_3(i, j, k) - p_2(i, j, k)$$
$$L_{k3} = p_4(i, j, k) - p_3(i, j, k)$$

If $L_{k0} < L_{k1} > L_{k2}$ then the point p(i, j, k) is a natural interest point at with $\sigma = 2^{k+1/2}$
If $L_{k1} < L_{k2} > L_{k3}$ then the point p(i, j, k) is a natural interest point with $\sigma = 2^{k+1}$

For each interest point, 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. Fast 2D Haar-like features using Integral Image

In 2001, Paul Viola and Mike Jones at MERL (Misubishi Research Labs) showed that Haar wavelets could be used for real time face detection using a cascade of linear classifiers.

They computed the Haar Wavelets (difference of adjacent boxes) for a window from integral images.

4.1. Integral Images

Given a window of an image p(i,j), of size W, H, an integral image is a sum of the pixels from the upper left corner:

$$ii(u,v) = \sum_{i=1}^{H}\sum_{j=1}^{W} p(i,j)$$

An integral image provides a structure for very fast computation of 2D Haar wavelets.

A fast recursive algorithm for computing the integral images

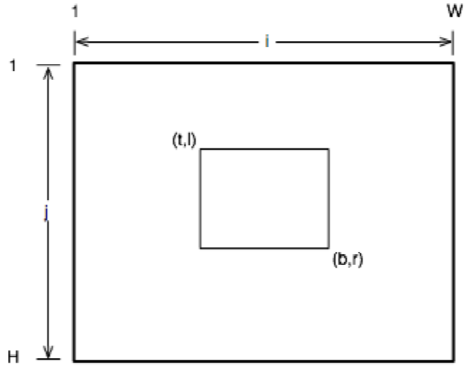
For j = 1 to H
For i = 1 to W
ii(i,j) = p(i) + ii(i-1,j) + ii(i,j-1)

Cost = 2WH ops.

4.2. Difference of Boxes

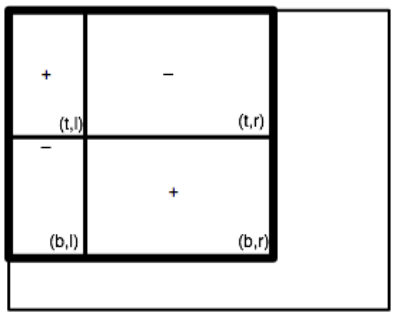
A box feature is a sum of pixel from (t, l) to (b, r)
With the constraints : t < b and r > l.

$$b(t,l,b,r) = \sum_{x=l}^{r}\sum_{y=t}^{b} p(x,y)$$



A box feature can be computed from an intergral image with 4 operations

box(t,l,b,r)=ii(b,r)–ii(t,r)–ii(b,l)+ii(t,l)



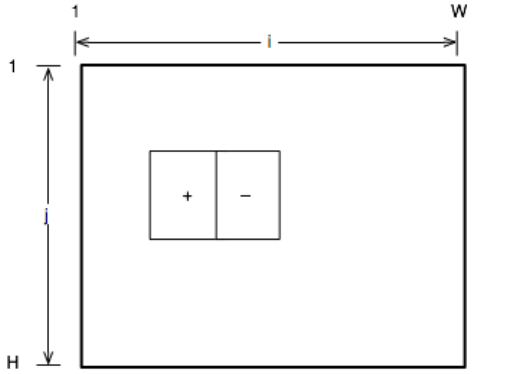
Notice that this is equivalent to a product of the image with a rectangular window with constant coefficients of size M, N: $W_{m,n}(i,j)$.

$$W_{MN}(i,j) = \begin{cases} 1 & \text{for } 0 \le i \le M \text{ and } 0 \le j \le N \\ 0 & \text{elsewhere} \end{cases}$$

$$box(t,l,b,r) = \langle W_{NM}(i,j), p(i+t, j+l) \rangle$$

A Difference of Boxes (DoB) feature is a difference of two boxes $box(t1,l1,b1,r1)$.

$$DoB(t1,l1,b1,r1,t2,l2,b2,r2) = box(t1,l1,b1,r1) - box(t2,l2,b2,r2)$$



An arbitrary 1st order difference of boxes costs 8 ops.

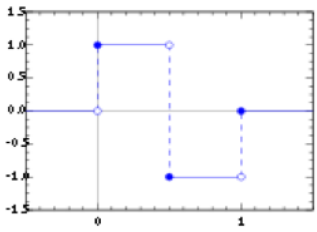
$DoB(t_1,l_1,b_1,r_1,t_2,l_2,b_2,r_2) = box(t_1,l_1,b_1,r_1) - box(t_2,l_2,b_2,r_2)$
$= ii(b_1,r_1) - ii(t_1,r_1) - ii(b_1,l_1) + ii(t_1,l_1) - (ii(b_2,r_2) - ii(t_2,r_2) - ii(b_2,l_2) + ii(t_2,l_2))$

An interesting subclass of DoB are Difference of Adjacent Boxes where the sum of pixels is 0. These are Haar wavelets. They can be computed for an image, or for an extracted window of an image (an "imagette").

4.3. 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.



The Digital (discrete sampled) form of Haar wavelet is

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

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

$$H_0 = +1 \qquad H_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad H_2 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \qquad \ldots$$

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

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.

4.4. Haar-like Features from Difference of Adjacent Boxes

A 2-D form of Haar features transform may be computed using difference of adjacent boxes. These can be calculated VERY fast using Integral Images. They give a VERY large number of possible image features.

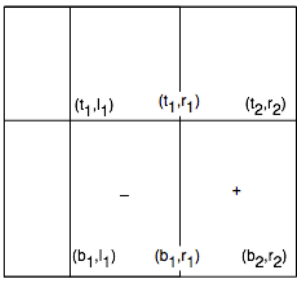
Using integral images, an arbitrary 1st order difference of boxes costs 8 ops.

$$DoB(t_1,l_1,b_1,r_1,t_2,l_2,b_2,r_2) = box(t_1,l_1,b_1,r_1) - box(t_2,l_2,b_2,r_2)$$

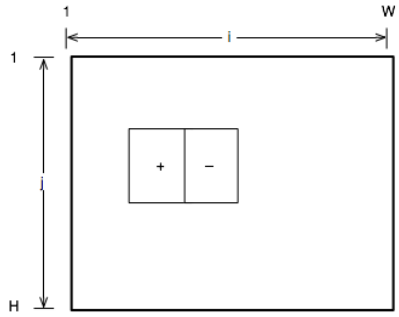
$$= ii(b_1,r_1) - ii(t_1,r_1) - ii(b_1,l_1) + ii(t_1,l_1) - (ii(b_2,r_2) - ii(t_2,r_2) - ii(b_2,l_2) + ii(t_2,l_2))$$

However, a 1st order difference of adjacent boxes costs only 6 ops.
This is because $r_1 = l_2$ and thus

$$ii(t_1,r_1) = ii(t_2,l_2) \text{ and } ii(b_1,r_1) = ii(b_2,l_2)$$



$$Haar(t_1,l_1,b_1,r_1,b_2,r_2) = ii(b_2,r_2) - 2ii(b_1,r_1) + ii(b_1,l_1) - ii(t_2,r_2) + 2ii(t_1,r_1) - ii(t_1,l_1)$$



Assume a window is extracted from an image and mapped to the WxH imagette.
Label the window coordinates (x, y) from [1,W] and [1,H]

The possible parameters for the Haar Wavelet/Difference of Boxes are:

1) The "polarity" of the difference ([1 -1] or [-1 1])
1) order (number of adjacent boxes): 2nd or 3rd
2) orientation: vertical or horizontal
3) center position - (c_x, c_y) WxH possible positions
4) box size (d_x, d_y) (W/2)x(H/2) possible sizes

These can provide N image features. Label these with an integer index, n, $H_n(x,y)$ Note that each Haar wavelet corresponds to a specific position, size, and orientation in the imagette.

The product of each Haar wavelet $H_n(x,y)$ with the imagette $W(x,y)$ gives a number: X_n. This number is an image "feature" that describes the imagette.

$$X_n = \sum_{x=1}^{W} \sum_{y=1}^{H} W(x,y) H_n(x,y)$$

Given a WxH imagette of a face we can obtain N Feature numbers, X_n. Not all features are useful. We will use "machine learning to determine the subset of useful features for detecting faces.

Do not be confused by the reuse of W and H. W and H are the size of the imagette, W(x,y) is the imagette and $H_n(x,y)$ are the

5. Linear Classifiers for Face Detection

The innovation in the Viola-Jones face detector resulted from

1) A very large number of very simple features (Haar wavelets).
2) The use of the Adaboost learning algorithm to learn an arbitrariy good detector.

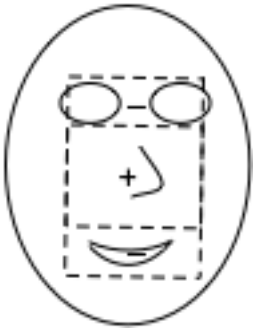
HAAR wavelets are computed using difference of Boxes, with Integral Images.

A WxH imagette contains $W^2H^2/4$ possible 1st order Haar wavelets H_n
(difference of adjacent boxes of same size).

Similarly, any 2nd order Haar wavelet can be computed with 8 ops.

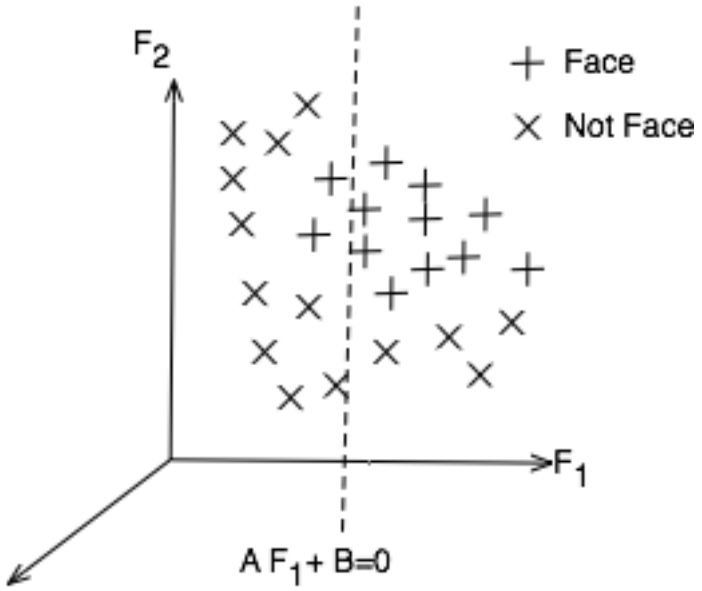
Each feature, X_n is defined as the product of a Haar wavelet with the image window.

$$X_n = \sum_{x=1}^{W} \sum_{y=1}^{H} W(x,y) H_n(x,y)$$



Some features respond to the appearance of a face. These can be used to determine if the imagette contains a face or not.

Given an image of a Face (F), and a set of Haar wavelets H_n



Each feature can be used to define a hyper-plane $<W, H_n> + b = 0$.

where $\langle W, H_n \rangle = \sum_{x=1}^{W} \sum_{y=1}^{H} W(x,y), H_n(x,y)$

and b is a "bias" that shifts the plane along the H_n axis.
b determines the tradeoff between False Positives and False Negatives.

this can be noted as $<WH_n> + b > 0$ or simply $WH_n + b > 0$

The problem is to choose the best H_n so that most non-face windows are on one side of the hyperplane and most face windows are on the other.

This can be obtained from a "training" set of M imagette, $\{W_m\}$, some of which contain faces. The imagettes in the training set are labeled with an "indicator variable" y_m.

For imagettes that contain faces, $y_m = 1$. Imagettes that do not contain faces, $y_m = -1$.

5.1. Training a committee of classifiers

Assume a very large set of M face windows $\{W_m\}$ that have been labeled by a set of labels $\{y_m\}$ such that y=+1 if face and y=–1 if not face,

Then for an imagette, W_m, each feature "votes" for a face (P for positive) or not a face (N for negative).

if $W_m H_n + B > 0$ then P else N.

Whether this vote is true (T) of false (F) can be determined by the indicator variable.

if $(W_m H_n + B) \cdot y_m > 0$ then T else F.

For the training set $\{W_m\}$, the error rate for the feature H_n is

$$E_n = \text{Card}\{(W_m H_n + B) \cdot y_m < 0\}$$

(Card is the cardinality operator - it counts the number of times something happens)

The error rate is composed of two parts: False Positives and False Negative.

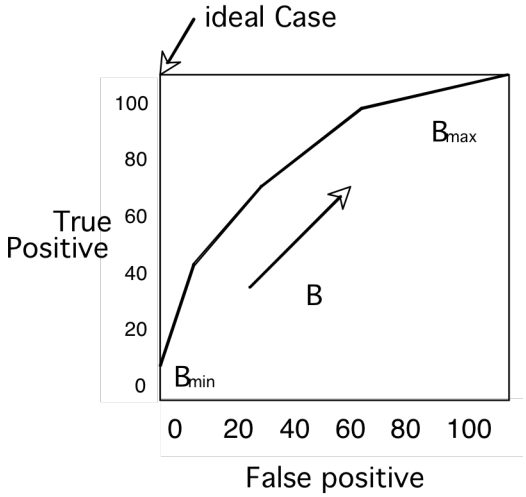
$$FP_n = \text{Card}\{(W_m H_n + B) > 0) \text{ and } (y = -1)\}$$
$$FN_n = \text{Card}\{(W_m H_n + B) < 0) \text{ and } (y = +1)\}$$
$$E_n = FP_n + FN_n$$

note that the number of true positives (TP) is TP = 1 – FP

We can trade FPs for FNs by adding to the global Bias B,

For a feature H_n

$$FP = \text{Card}\{ (W_m H_n + B) > 0 \text{ and } y_m = -1\}$$
$$FN = \text{Card}\{ (W_m H_n + B) < 0 \text{ and } y_m = +1\}$$



the ROC plots the True Positive Rate (TPR) against False Positive Rate (FPR) for a classifier as a function of the global bias B.

The probability of error for a committee detectors can be computed as

$$P(Error) = \frac{F}{N} = \frac{\#FP + \#FN}{N}$$

Where N is the number of training samples, and F is the number of False detections within the N training samples.

5.2. Boosted Learning

To boost the learning, after selection of each "best" classifier, (F_n, B_n)
we re-weight the incorrectly classified training samples with a weight, a_m to increase the weight for incorrectly classed imagettes:

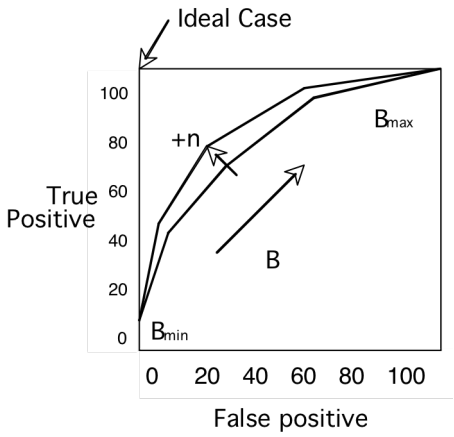
For all m = 1 to M if $(W_m H_n + b) \cdot y_m^{(i-1)} < 0$ then $a_m^{(i)} = a_m^{(i-1)} + 1$

We then learn the i^{th} classifier from the re-weighted set

$E_{min} = M$
For n=1 to N do
$E_n = Card\{a_m^{(i)}(W_m, H_n) \cdot y_m < 0\}$
if $E_n < E_{min}$ then $E_{min} := E_n$

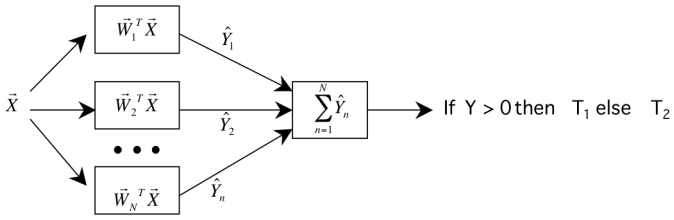
The Boosting theorem states that adding a new boosted detectors to a committee always improves the committee's ROC curve. We can continue adding classifiers until we obtain a desired rate of false positives and false negatives.

However, in general, the improvement provided for each new classifier becomes progressively smaller. We can end up with a very very large number of classifiers.



5.3. Learning a Committee of Classifiers with Boosting

We can improve classification by learning a committee of the best I classifiers.



The decision is made by voting. An imagettes W is determined to be a Face if the majority of classifiers (features) vote > 0.

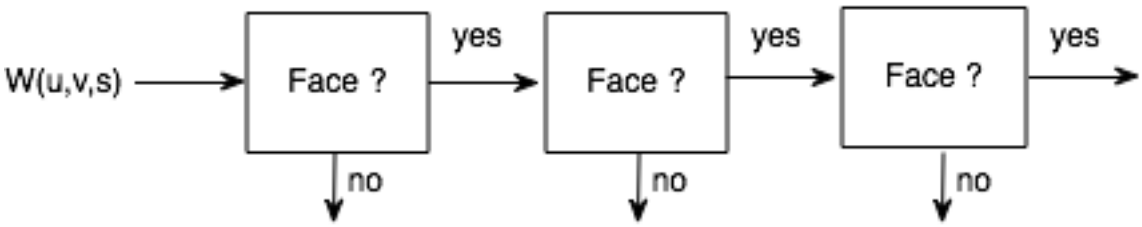
If $\sum_{i=1}^{I} W_m H_n + B > 0$ then Face else Not-Face.

5.4. Learning a Multi-Stage Cascade of Classifiers

We can optimize the computation time by using a multistage cascade.

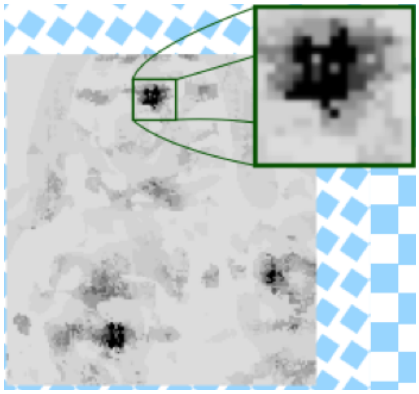
Algorithm:

1) Set a desired error rate for each stage j : (FP_j, FN_j).
2) For j = 1 to J
For all windows labeled as Face by j-1 stage, learn a boosted committee of classifiers that meets (FP_j, FN_j).



Each stage acts as a filter, rejecting a grand number of easy cases, and passing the hard cases to the next stage.

This is called a "cascade classifier"
Note that applying this to every position gives an "image" of cascade depths.



Faces can be detected as the center of gravity of "deep" detections.
Faces can be tracked using the Bayesian tracking described in the previous session.

This algorithm is part of the OpenCV tool box. It is widely used in digital cameras and cell phones for face detection and tracking.