Computer Vision

MSc Informatics option GVR
James L. Crowley

Fall Semester 14 November 2013

Lesson 7

Scale Invariant Pyramids, HoG, SIFT and Haar

Lesson Outline:

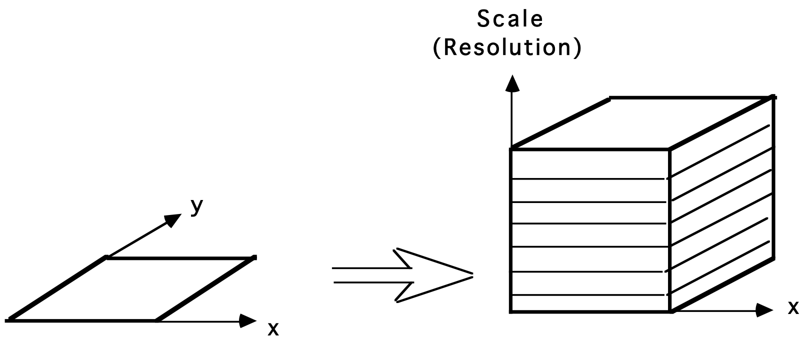
1. Image Scale Space

1.1. Scale Space (Rappel)

Let p(x, y) be an image of size W columns by H rows.
Let G(x, y, σ_s) by a Gaussian function of scale σ_s

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

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



Scale space:
Separates global structure from fine detail.
Provides context for recognition.
Can provide local descriptions (features) of the image that are invariant to position, orientation and scale.

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

$$s = \text{Log}_2(\sigma) = \text{Log}_2(2^s)$$

A logarithmic scale axis is necessary for scale equivariance.
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).

1.2. Discrete Scale Space - Scale invariant impulse response.

In a computer, we need to discretize (sample) the axes x, y, and s.

Let *p(x, y)* is an image array of size WxH pixels, where (x, y) are integers,

We propose to sample scale with a step size of $\Delta\sigma = 2$ so that $\sigma_k=2^k$
For k=0, to K.

$\sigma_0=1$ is the smallest scale that we can represent.

At k=0 : $\sigma_0=2^0=1$.

let M = min(W, H)
K is the largest scale possible: $K=\text{Log}_2(M)$

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

For k > K the scale parameter σ is larger than the image.

1.3. Spatial Resampling and Image Pyramids

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.

Such resampling has the benefit of assuring an invariance of the impulse response of each image. The sample size Δx_k, Δy_k can grow exactly as σ_k.

What sample size is possible? It is possible to show that the sample step must be smaller than σ. For example, let $\Delta x = \sigma$

Thus $\Delta x_k = \Delta y_k = 2^k$ with only minimal aliasing.

Resampling selects every Δx image sample:
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 is $x = i\Delta x_k$ and $y = j\Delta y_k$

Resampling at $\Delta x_k = \sigma_k = 2^k$ results an identical impulse response at each level.
This property is called "scale invariance". (The impulse response is scale invariant).

A resampled scale space (a scale invariant pyramid), with a scale step of one "octave".

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

It is possible to build a scale invariant pyramid with a step size of $\Delta\sigma = 2^{k/2}$

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

Last week we saw that image derivatives can be computed by convolving the image with derivatives of Gaussians

$$p_x(x,y,\sigma) \approx p * G_x(x,y,\sigma)$$

With the Pyramid, derivatives can be obtained directly by sum and difference of the resampled pixels.

Let $i = x/\Delta x_k$ and $j = y/\Delta y_k$

Then

$$p_x(i,j,k) \approx p(i+1,j,k) - p(i-1,j,k)$$
$$p_y(i,j,k) \approx p(i,j+1,k) - p(i,j-1,k)$$
$$p_{xx}(i,j,k) \approx p(i+1,j,k) - 2p(i,j,k) + p(i-1,j,k)$$
$$p_{yy}(i,j,k) \approx p(i,j+1,k) - 2p(i,j,k) + p(i,j-1,k)$$
$$p_{xy}(i,j,k) \approx p(i+1,j+1,k) - p(i-1,j+1,k) - p(i+1,j-1,k) + p(i-1,j-1,k)$$

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

Recall the Gradient $\vec{\nabla} P(i,j) = \begin{pmatrix} p_x(i,j) \\ p_x(i,j) \end{pmatrix}$

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

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

Laplacien: $\nabla^2 p(x,y,k) = p * \nabla^2 G(x,y,\sigma_k) = p_{xx}(x,y,k) + p_{yy}(x,y,k)$

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

This typically requires $\sigma_1 \geq \sqrt{2}\ \sigma_2$
We can use this to show that the Laplacian is approximated by a difference of two pyramid levels:

$$\nabla^2 p(i, j, k) \approx p(i, j, k) - p(i, j, k-1)$$

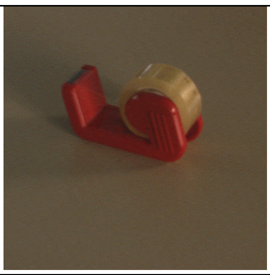
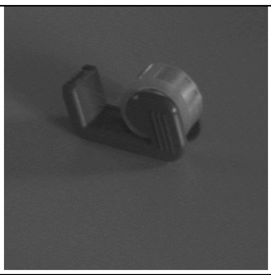
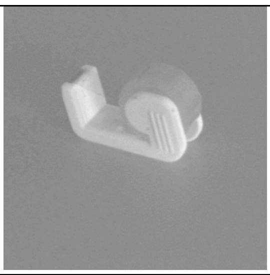
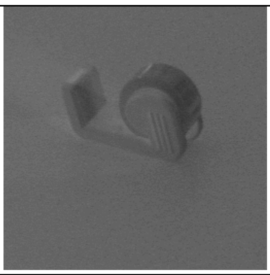
This is called a "Difference of Gaussians" (DoG).
$\nabla^2 p(i, j, k)$ exists for any sample where $p(i, j, k)$ exists.

1.5. Color Opponent Scale Space

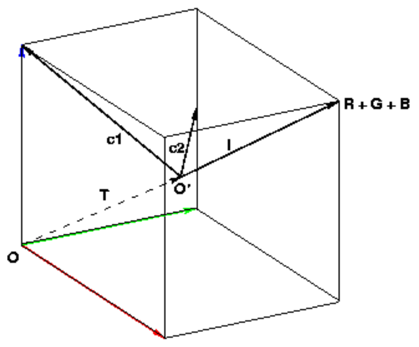
In lesson 3 we saw that a color opponent space was useful for illumination invariance

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

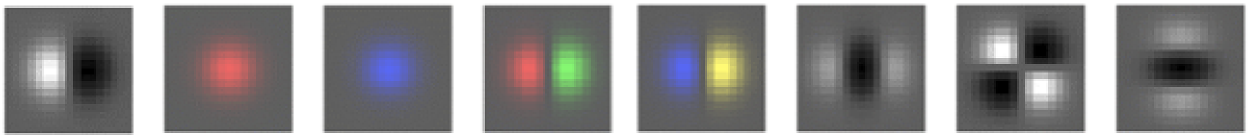
This representation separates luminance and chrominance.

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

Color opponent space can be used to build receptive fields that can be steered in color



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



We then compute 3 pyramids : L(x, y, k), C_1(x, y, k), and C_2(x, y, k),

This gives us a feature vector for appearance:

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

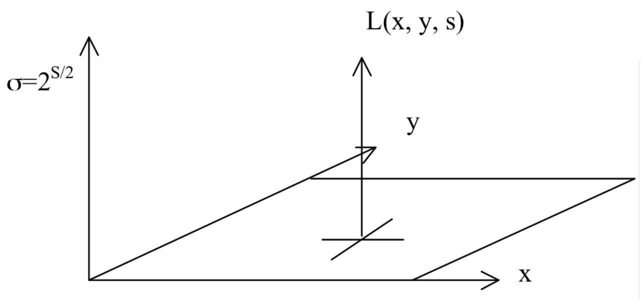
This can be generalized to include multiple scales.

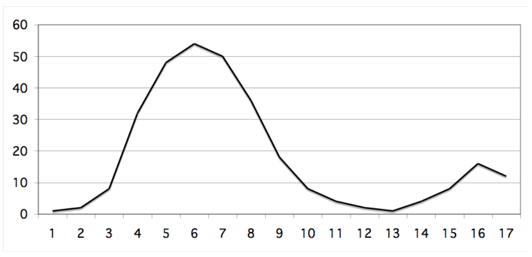
1.6. Intrinsic scale at a point in an image

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

$$L(x, y, s) = p(x, y) * \nabla^2 G(x, y, 2^s)$$

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 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 $\sigma_i = 2^{s_i}$

such that $s_i = \underset{s}{\arg-\max}\{L(x,y,s)\}$

We can do the same with a Gradient.

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

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

1.7. Scale Invariant Interest Points

Maximal points in the image derivatives provide landmarks.
These can serve to focus processing, and are thus called "interest points".
In an image scale space, these points are scale invariant. They provide landmarks for scale invariant image description.

Maxima in the Lapacian Scale Space provide Scale invariant interest points

Recall that using an image Pyramid, the Laplacian is simply the difference at adjacent levels.

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

We can detect scale invariant interest points local maxima in the Laplacian.

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

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

1.8. Other popular interest point detectors.

Other popular detectors for scale invariant interest points include:

Gradient Magnitude: $A(i,j,k) = \underset{i,j,k}{Local-\max}\{\|\vec{\nabla} p(x,y,s)\|\}$

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

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

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 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.

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)

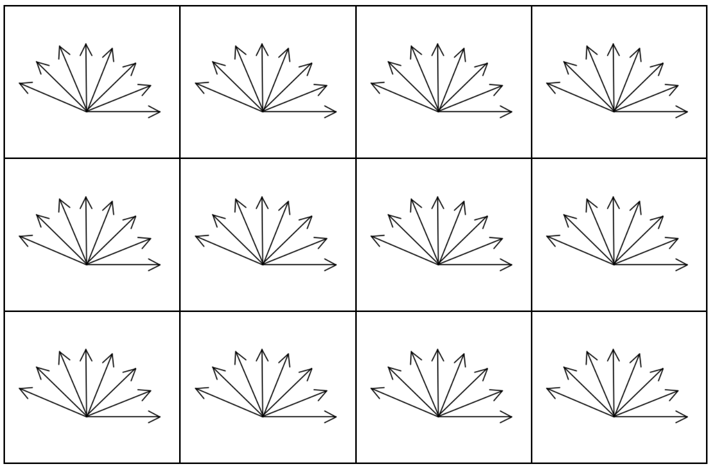
SIFT uses a scale invariant pyramid to compute scale invariant interest points

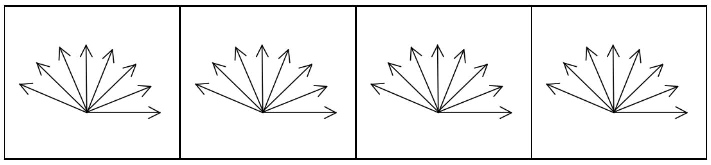
$$X(i,j,k) = \underset{i,j,k,R=2}{Local-\max}\{P(i,j,k) - P(i,j,k-1)\}$$

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 Wavelets using Integral Image

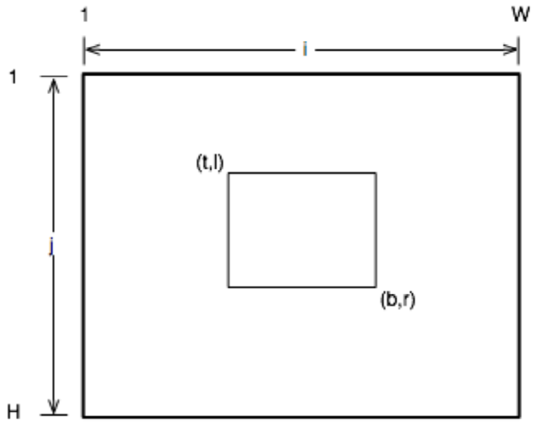
In 2001, Paul Viola and Mike Jones at MERL (Misubishi Research Labs) showed that Haar wavelets could be used for real time face detection using a cascade of linear classifiers.

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

4.1. Difference of Boxes

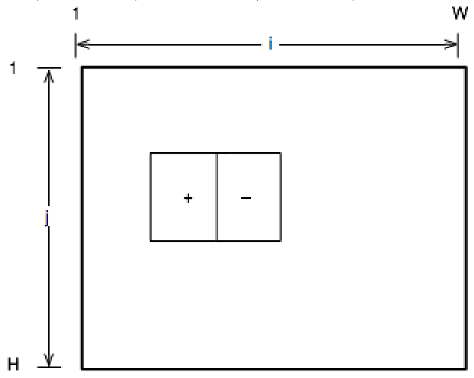
A box feature is a sum of pixel from (t, l) to (b, r)
With the constraints : t < b and r > l.

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



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

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

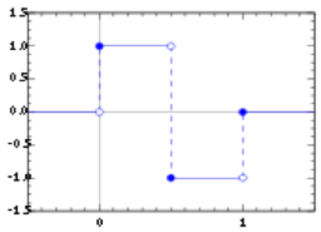


An interesting subclass are Difference of Adjacent Boxes where the sum of pixels is 0. These are Haar wavelets. They can be computed for an image, or for an extracted window of an image (an "imagette").

4.2. 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

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

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

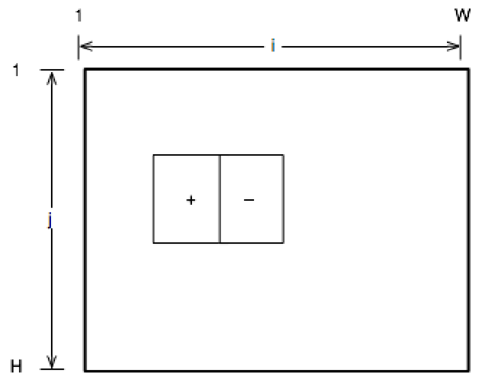
$$H_0 = +1 \qquad H_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad H_2 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \qquad \cdots$$

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

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 using adjacent boxes. These can be calculated VERY fast using an algorithm known as Integral Images. They give a VERY large number of possible image features.



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]

Parameters:
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

4.3. Integral Images

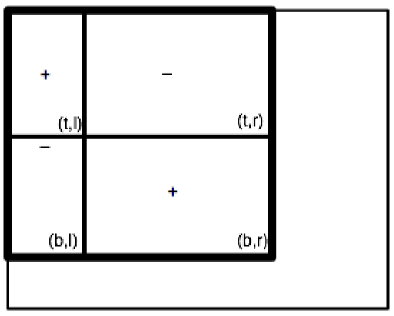
An integral image is an image where each pixel contains the sum from the upper left corner:

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

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

Any box feature can be computed with 4 operations (additions/subtractions).

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

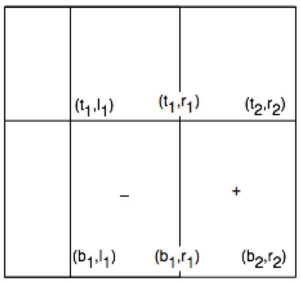


An arbitrary 1st order difference of boxes costs 8 ops.

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

However, a 1st order Haar wavelet costs only 6 ops because $r_1=l_2$ and thus

$ii(t_1,r_1) = ii(t_2,l_2)$ and $ii(b_1,r_1)= ii(b_2,l_2)$



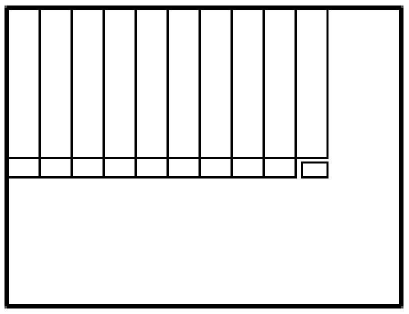
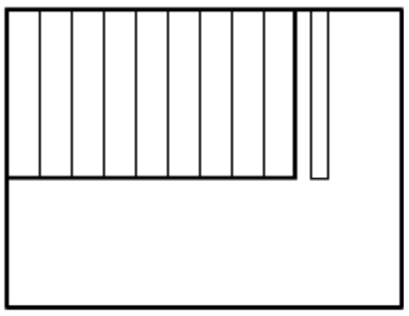
$Haar(t_1,l_1,b_1,r_1,b_2,r_2) = ii(b_2,r_2)–2ii(b_1,r_1)+ii(b_1,l_1)–ii(t_2,r_2)+2ii(t_1,r_1)–ii(t_1,l_1)$

4.4. Fast Integral Image algorithm.

Integral images have been used for decades to compute local energy for normalization of images. A fast recursive algorithm for computing the integral image makes use of a buffer, c(i). The buffer keeps a running sum of each column.

```
For j = 1 to H
For i = 1 to W
{       c(i) = c(i) +  p(i,j)
        ii(i,j) = ii(i-1,j) + c(i) }
```

Cost = 2WH ops.

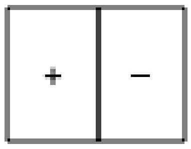
5. Linear Classifiers for Face Detection

The innovation in the Viola-Jones face detector resulted from

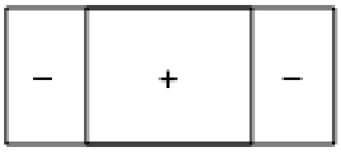
1) A very large number of very simple features (Haar wavelets).
2) The use of the Adaboost learning algorithm to learn an arbitrariy good detector.

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).

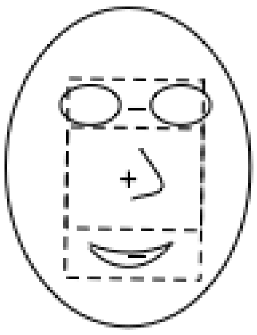


Similarly, any 2nd order Haar wavelet can be computed with 8 ops.



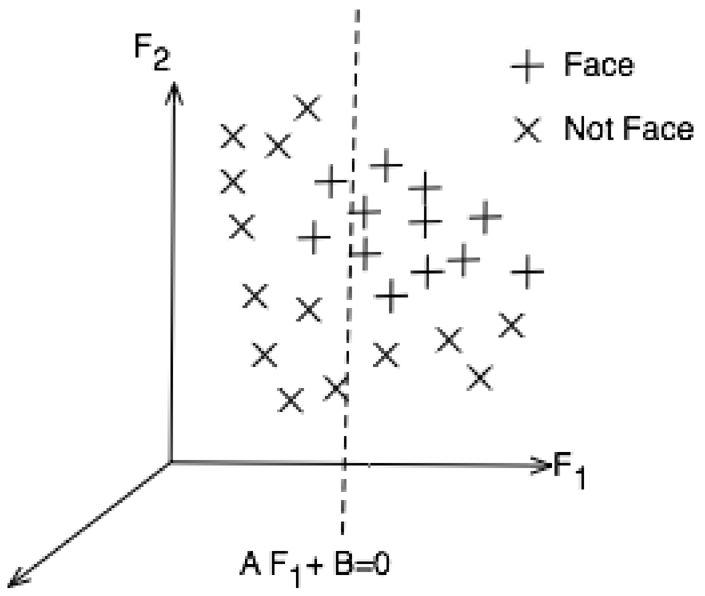
Each feature, X_n is defined as the product of a Haar wavelet with the image window.

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



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 global "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.

To do this we will use a "training" set of M imagette, {W_m}. some of which contain faces. We will note whether the imagette contains a face 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.

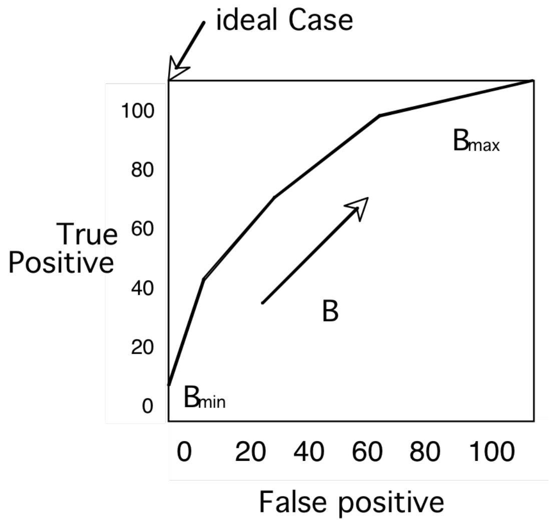
$$FP_n = \text{Card}\{(W_m H_n + B) > 0) \text{ and } (y = -1)\}$$
$$FN_n = \text{Card}\{(W_m H_n + B) < 0) \text{ and } (y = +1)\}$$
$$E_n = FP_n + FN_n$$

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

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



These are plotted in a graph called an "ROC" or Receiver Operating Characteristics Graph.

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:

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

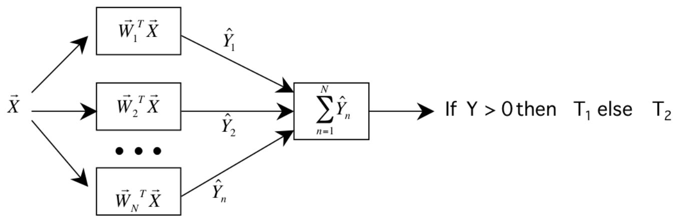
We then learn the ith classifier from the re-weighted set

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

Haar features are removed from the set after being used.

6. 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.

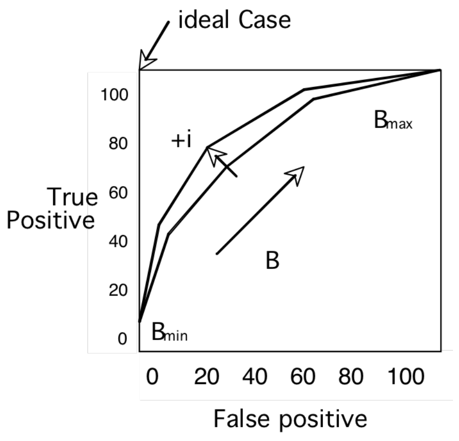
6.1. ROC Curve

We can describe a committee of classifiers with an ROC curve, but defining a global bias, B. The ROC describes the number of False Positives (FP) and False Negatives (FN) for a set of classifier as a function of the global bias B.

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

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

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

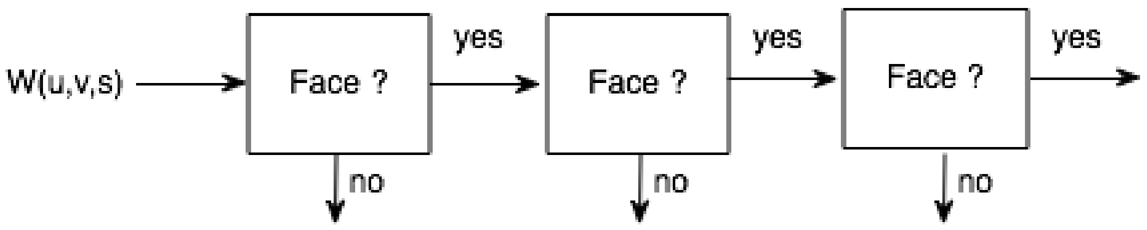


7. Learning a Multi-Stage Cascade of Classifiers

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

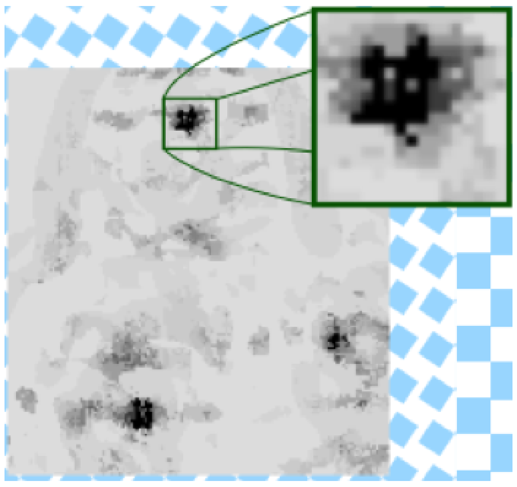
Algorithm:

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.