Pattern Recognition and Machine Learning

James L. Crowley

ENSIMAG 3 - MMIS
Lesson 2

Fall Semester 2020
14 October 2020

Face Detection in Images

Outline

Notation

x_d	A feature. An observed or measured value.
$\vec{X}$	A vector of features. An observation.
D	The number of dimensions for the vector $\vec{X}$
$\{C_k\}$	A set of K classes (or class labels).
K	Number of classes
$\vec{X} \in C_k$	Statement that the observation $\vec{X}$ is a member of class C_k
$\hat{C}_k$	An estimated class label
	For a 2 class detection problem (K=2), $C_k \in \{P,N\}$
$R(\vec{X})$	A recognition function
$\hat{C}_k \leftarrow R(\vec{X})$	A recognition function that predicts $\hat{C}_k$ from $\vec{X}$
$\{\vec{X}_m\}$	Training data for learning.
M	The number of training samples.
$y(\vec{X}_m)$	An annotation (or ground truth) function for $\{\vec{X}_m\}$: $y(\vec{X}_m) \in \{P,N\}$
$g(\vec{X}_m)$	Discriminant function. $0 \le g(\vec{X}_m) \le 1$
$\vec{R}$	Region of Interest Rectangle composed of (t, l, b, r)
$\vec{H}$	Hypothesis for face position and size composed of c_i, c_j, w, h

Programming Teams

This course requires participation in three programming project:

Project 1: Face detection using the Viola Jones cascade detector
(Oral presentations 4 Nov)

Project 2: Face detection using multilayer fully connected neural networks.
(Oral presentations 16 Nov)

Project 3: Face detection using convolutional neural networks.
(Oral presentations 27 Jan)

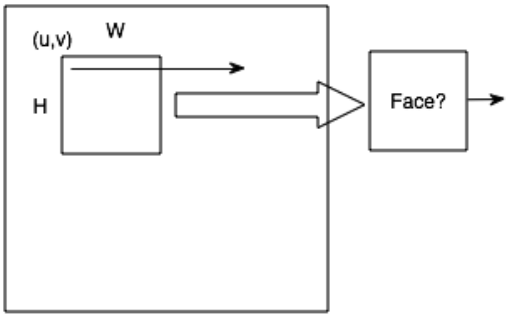
Projects are to be programmed, evaluated and reported by teams of 3 students. Each team will make an oral presentation on one of the 3 projects. With 30 students we can form ten programming teams.

Prenom	Nom	Team	Oral Presentation
ROMAIN	CHAPOULLIÉ		
ANTOINE	CHARBONNEL		
MÉLANIE	DONNEZ	2	
EMILIEN	FACHE	1	
FERGAL	FENEUIL	4	
ANTOINE	FLICHY		
VALENTIN	GAUTIER		
CHARLES	GRANIER		
VINCENT	HACHIN	1	
DORIAN	HOUDELETTE	4	
JONATHAN	JULOU		
MEVEN	KERZREHO		
RÉMI	LEROY	4	
CHRISTOPHE	LUONG		
KILIAN	MAC DONALD	2	
CLÉMENT	MALLERET		
LÉANDRE	PALISSE		
MARIUS	PÉLÉGRIN	2	
MARTIN	PORTALIER		
JEREMY	RAMBOASOLO		
THEO	RECKING		
MATHIS	ROUX		
JULES	SAGOT--GENTIL		
QUENTIN	SARRAZIN		
MARTIN	SCHNEIDER	1	
YOANN	SCHNEIDER	3	
ALEXIS	SONOLET		
LUCAS	SORT	3	
MARGOT	TINTURIER		
Etienne	BECLE		
Leah	RIFI		

1 Face Detection using a Sliding Window Detector

Faces can occur at many different positions, orientations and sizes (scales) in an image. With sufficient computing power, we can test windows of all possible positions, sizes and orientations in parallel for presence of a face. However, lacking a massively parallel computer we can use a sliding window detector.

A sliding window detector is a brute force method to test if a pattern can be found in an image. This approach was made popular by the Viola-Jones face detector, now found in most smart phones.



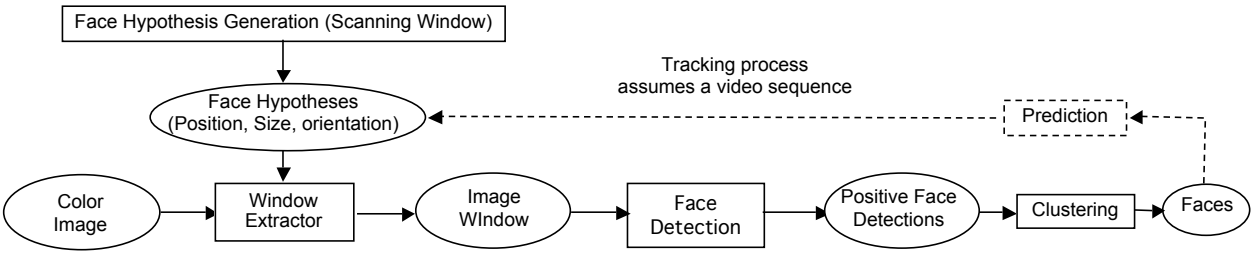
Assume an RGB color image, *P(i,j)*. Note that the origin is commonly in the upper left corner. A Region of Interest or ROI is a rectangular region of the image, sometimes called a window or an imagette. A ROI can be defined by the top-left and bottom-right corners, represented by a vector (*t, l, b, r*).

t - "top" - first row of the ROI.
l - "left" - first column of the ROI.
b - "bottom" - last row of the ROI
r - "right" - last column of the ROI.

Note that in image coordinates, when the origin is in the upper left corner, b will be larger than t.

Generally a target pattern can occur at many different positions, orientations and sizes (scales) in an image. A common approach is to train a detection function with a standard size and orientation for the pattern. A sliding window process is then designed to extract (copy) the contents from every possible ROI for a range of positions, orientations and sizes (scales). ROIs are texture mapped to a standard size window to be processed by the recognizer. This transformation is easily performed by the "texture mapping" function found in OpenCV.

A typical architecture for a sliding window detector looks like this:



ROIs over a range of positions and sizes (and possibly orientations) are extracted and projected (flattened) into a standard size feature vector. This feature vector is run through a trained detection function resulting in possible face detections. Adjacent detections may be grouped into clusters and used to provide a precise estimate for the position and size (and orientation) for each face.

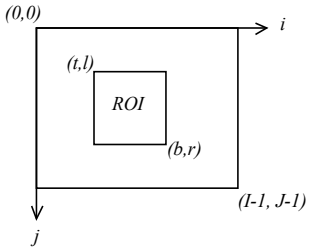
This process can be used to build an efficient tracking system for video sequence. In this case, face detection is limited to a small range of positions and sizes near where faces have been found in the previous image.

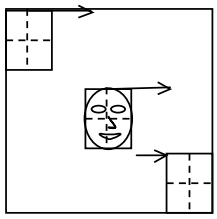
Detection is commonly performed with a pattern detector that has been constructed using Machine Learning. A large number of possible techniques are available for building pattern detectors. Almost all detection functions require training at a specific window size (resolution or number of pixels). ROIs of other sizes must be transformed (normalized). This can be done, for example, with the texture-mapping function found in OpenCV.

Scale vs Resolution for Face Imagettes

The original size of the face ROI is called the "scale". The number of coefficients of the feature vector is called the "resolution". For faces, it is well known that faces require a resolution of at least 8 x 8 pixels to detect and that a resolution larger than 32 x 32 provide very little gain in detection rate. However, modern images that can range to 4K pixels by 4K pixels (or larger) and faces can and do occur at all scales (sizes) depending on the optics and distance from the camera. Thus it is necessary to transform region of interests to a standard size vector. Note, also, that faces tend to be oval in shape. It is often possible to improve detection by using a rectangular ROI with a larger vertical dimension, such as a 4 to 3 or similar ratio.

Notation for Windows and ROIs





A common representation for the Region of Interest is a rectangle represented by four coordinates: (top, left, bottom, right) or (*t, l, b, r*)

t - "top" - first row of the ROI.
l - "left" - first column of the ROI.
b - "bottom" - last row of the ROI
r - "right" - last column of the ROI.

The vector *(t,l,b,r)* can be seen as a bounding box, expressed by opposite corners (*l*,*t*), (*r*,*b*).

In some cases it is easier to identify the ROI as a center point (c_i, c_j) width, *w*, and height, *h*. Assuming an image with the origin in the upper left hand corner, the center position, width and height are simply

$$c_i = \frac{l+r}{2}, \quad c_j = \frac{t+b}{2}, \quad w = l - r + 1, \qquad h = b - t + 1$$

In some cases it is more convenient to fix c_i, c_j, *w*, *h* and calculate (*l, t, r, b)*. In this case the bounding box ROI is:

$$t = c_j - \frac{h-1}{2}, \qquad b = c_j + \frac{h-1}{2}, \qquad l = c_i - \frac{w-1}{2}, \qquad r = c_i + \frac{w-1}{2}$$

It is often convenient to use odd numbers for *w* and *h* are so that the center pixels falls on integer positions. In the following, we will use the notation:

Region of Interest Rectangle $\vec{R} = \begin{pmatrix} t \\ l \\ b \\ r \end{pmatrix}$

Position and size for a Face Hypothesis: $\vec{H} = \begin{pmatrix} c_i \\ c_j \\ w \\ h \end{pmatrix}$

Region of Interest pixels for Face Hypothesis $R_{\vec{H}}(i,j) = \text{crop}(image, t, l, b, r)$

Fixed Resolution window for Face Hypothesis: $W_{\vec{H}} = texture\text{-}map(R_{\vec{H}}, N)$

Flattened 1-D vector from window: $\vec{X}_{\vec{H}} = \text{Flatten}(W_{\vec{H}})$

Multiple Scales and Step Sizes

Generally both the size and the position of the ROI are scanned. ROI size is equivalent to image "Scale" and can be sampled with linear scale steps or alogarithmic scale steps. There are good reasons to use a logarithmic scale for ROI size (e.g. Fractional powers of 2), for covering large ranges of scales, but this is beyond the scope of this lecture.

In a scanning window system, the scanning step size (S) may also be varied. This may be the same size for row and column direction, or different step sizes may be used (say S_i and S_j). Step size should vary with ROI size. When the discriminant function is highly correlated at adjacent pixels (as is the case with color skin detection), then the resulting detection function will be very smooth. In this case a larger step size may be sufficient.

The largest reasonable value for S is half the width of a target. Beyond this the detection degrades rapidly because of aliasing effects, as can be demonstrated using signal processing techniques that are beyond the scope of this class.

When computing time is an issue, the detection algorithm can be optimized using hierarchical search, starting with a large step size (typically a power of 2) and recursively searching at higher step sizes as the discriminant function returns a higher detection likelihood.

2. Evaluating a Sliding Window Face Detector.

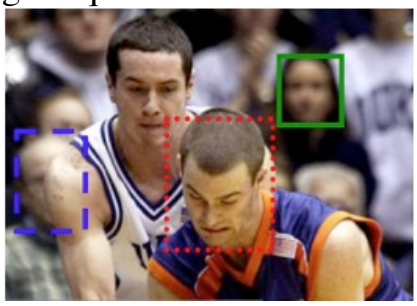
There are several problems that must be solved to evaluate a sliding window detector.
1) Building a balanced test data set for evaluation of the detection function.
2) Avoiding multiple detections of the same face in the sliding window detector.
3) Evaluating the accuracy of position and size estimation for the target.

Building a balanced data set for evaluation

Most of the ROIs processed by a sliding window detector do not contain a face. In order to perform a proper evaluation, we need to test a similar number of P and N hypothesis. Thus we need to build a "balanced" data set for evaluation. Using a Balanced Data set will also be very important for successful learning of a detection function using any machine learning technique. The data set will be composed of known P and N examples.

Building the set of P examples.

Consider the FDDB data set. The FDDB data set contains 2845 images with a total of 5171 faces extracted from news articles selected using an automatic face detector. Faces with a height or width less than 20 pixels, as well as faces that are looking away from the camera were rejected. The remaining 5171 faces have been noted in a ground-truth data set and labeled with a bounding box. The images in this data set exhibit large variations in pose, lighting, background and appearance due to factors such as motion, occlusions, and facial expressions, which are characteristic of the unconstrained setting for image acquisition.



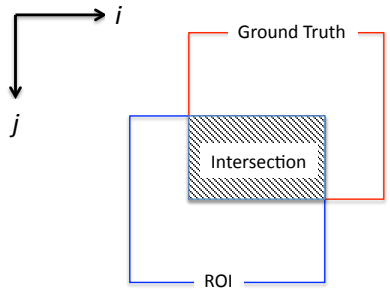
Face Boxes

The easiest way to build a set of Positive face examples from FDDB is to simply extract an example of a Positive ROI for each face listed in the ground truth. The problem is that faces exist over a range of sizes, and our detector only works for a single size window (for example, 24 x 24 for Viola Jones). We can we use the pixels in each face bounding box as a ROI and transform (rectify) this ROI to the standard size used for use with the detection function. When this set is to be used for training, using centered examples concentrates detection to the center of the face.

The number of positive images is M_P (or #P) for FDDB will be 5171. For a balanced data set, we need a similar number of Negative examples, M_n (#N). We can do this by using a random number generator to choose the upper left corner for M_n candidates for Negative ROIs. Normally, most ROIs are Negative. However, occasionally a random ROI will overlap a Face. Thus we need to set a criteria for when ROI can be considered as a candidate ROI is a False Negative.

Intersection over Union (IOU)

A Rectangular ROI is typically considered to be true detection when it has sufficient overlap with a Ground Truth Bounding Box. Overlap is measured as the ratio of Intersection over Union (IOU).



IOU is area of intersection divide by the area of Union of the rectangles: $IOU = \frac{A_I}{A_U}$

Assume image coordinates with the origin at the top left corner, the area of a rectangle, $\vec{R}$ is: $A = w \cdot h = (l - r + 1) \cdot (b - t + 1)$

For two rectangles: $\vec{R}_1$ and $\vec{R}_2$, the area of the intersection of two rectangles is

(t_i, l_i, b_i, r_i)

where $l_i = \max(l_1, l_2)$ $r_i = \min(r_1, r_2)$ $t_i = \max(t_1, t_2)$ $b_i = \min(b_1, b_2)$

The area of intersection is $A_I = (l_i - r_i + 1) \cdot (b_i - t_i + 1)$

Area of the Union of two rectangles is: $A_U = A_1 + A_2 - A_I$

A typical threshold for a True Positive is IOU > 0.5. A True Negative requires an IOU $\leq$ 0.5. We can then evaluate the detection function using this test set. We can also use such the set as a training set.

Avoiding multiple detections

In evaluating a sliding window detector, we typically want to compare the number of detected faces to the number of faces in the ground truth. However, a sliding window detector will return positive detections over a range of positions and sizes. All of these are valid detections. However, there is only one face, located somewhere in the cloud. This requires grouping adjacent detections into a single face hypothesis.

There are several methods that we can use to avoid multiple detections. The most direct is to perform "connectivity analysis" on Positive Detections. Typically, with connectivity defined as any of the adjacent 8 neighbors, this will create a detected "blob" for each face. We count each blob as a single positive detection.

Detection vs Localization

Detection states that a ROI represents a Face. There will typically be blob of detections for each face.

Localization is a form of parameter estimation that specifies precisely where, and what size (and possible at orientation) the face may be found. There should be only ONE face detection for each True face. The face should be located at a position and size as close to the True face as possible. When searching at multiple positions, scales and orientations, multiple adjacent ROIs will be detected for each face.

Adjacent detections can be grouped together to form a "blob" using a connected components algorithm. Given a blob composed of adjacent detections, there are several techniques to estimate the position, size and orientation of the face.

1) Center of the Bounding Box.
A connected components algorithm groups together all detections that share one of the 8 immediate neighbors into a single "blob". Each blob may be described by a "bounding box". This is the smallest rectangle that encloses all the detections of the blob. The center of this rectangle can be considered as the face position.

2) Center of Gravity of the Detections.
Let *B(i,j)* be a binary window representing the blob with *B(i,j) =1* when $R(\vec{X}_{\vec{H}}) = P$ and *B(i,j) =0* when $R(\vec{X}_{\vec{H}}) = N$

The face can be estimated to be at center of gravity of the blob

$$c_i = \frac{1}{A_B}\sum_{j=t}^{b}\sum_{i=l}^{r} B(i,j)\cdot i \qquad c_j = \frac{1}{A_B}\sum_{j=t}^{b}\sum_{i=l}^{r} B(i,j)\cdot j$$

where A_b is the area of the blob. Note that these equations are shift invariant. The location of the blob with respect to the origin does not matter.

3) Center of Gravity of the Discriminant Detections.

Recall that the detection function is based a decision function applied to a discriminant function. For example:

$$R(\vec{X}_{\vec{H}}) = d(g(\vec{X}_{\vec{H}})) = \begin{cases} P & \text{if } g(\vec{X}_{\vec{H}}) + B > 0 \\ N & \text{if } g(\vec{X}_{\vec{H}}) + B \leq 0 \end{cases}$$

The Discriminant Function returns a likelihood, $g(\vec{X}_{\vec{H}}) = g(flatten(W_{\vec{H}}))$ for a target at the position and size of the face hypothesis $\vec{H}$. Rather than apply a threshold to this discriminant function, we can use it to directly estimate the face position. In this case the detection "blob" is constructed as:

$$B(i,j) = g(\vec{X}_{\vec{H}})$$

The face can be estimated to be at center of gravity of the blob

$$c_i = \frac{1}{A_B}\sum_{j=t}^{b}\sum_{i=l}^{r} B(i,j)\cdot i \qquad c_j = \frac{1}{A_B}\sum_{j=t}^{b}\sum_{i=l}^{r} B(i,j)\cdot j$$

We can also estimate the size and orientation from the second moments:

$$\sigma_i^2 = \frac{1}{A_b}\sum_{j=t}^{b}\sum_{i=l}^{r} B(i,j)\cdot (i-c_i)^2$$

$$\sigma_j^2 = \frac{1}{A_b}\sum_{j=t}^{b}\sum_{i=l}^{r} B(i,j)\cdot (j-c_j)^2$$

Then width is $w = 2\sigma_i + 1$ and height is $h = 2\sigma_j + 1$.