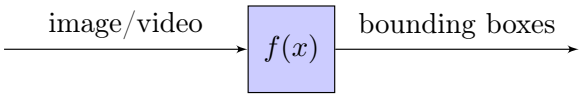
Boosted Multi-block LBP Face Detection

James Batten batten_j@epita.fr
Valentin Iovene toogy@lrde.epita.fr

October 6, 2016

Face detection is the task of localizing faces in images. One must distinguish between face *detection* and face *recognition*. The latter is the task of recognizing someone from an image of her face.

Face detection can be seen as a function f taking an image as input and returning *bounding boxes* around the faces contained in this image.



Face detection

— is used by *Facebook* to detect faces in images you share and make it easier for you to tag your friends,

— can be used to **track** faces in videos of people moving around (e.g. *Snapchat*'s face swap feature requires face detection),

— and is a prerequisite for any real-life applications of face recognition *(e.g. Law Enforcement localizing a suspect in a big city using video surveillance systems)*.

The detection process needs to be fast enough to keep up with the demand (millions of users or streamed images).

There has been extensive research in this area but one of the major breakthrough happened in 2001 when *Paul Viola* and *Michael Jones* published their object detection framework [3] (which was named the "Viola-Jones framework").
Even though it was invented more than 15 years ago *(time of writing)*, *OpenCV*'s face detection implementation (source) is still based on it.

This article explains how the *Multi-Block Local Binary Patterns (MB-LBP)* visual descriptors (which will soon be introduced) can be used in lieu of the *Haar-like features* used by the original *Viola-Jones* framework for face detection. This approach was demonstrated by Zhang et al. in 2007. [1]

Contents

1 Principle

A face detection algorithm looks at *features* of an image to determine whether it is a face or not. In the Image Processing domain, they are called *visual descriptors*. Multi-block Local Binary Patterns (MB-LBP) is one type of visual descriptor that can be used for detecting faces (and more generally, objects) in images.

It is important to note that the detection is developed for a very specific window size, say 20×20 pixels, and this window is then shifted and scaled in the detection procedure in order to detect faces of different sizes and at different locations in the image. We will refer to this unshifted and unscaled window as the *window of reference*.

1.1 Multi-block Local Binary Patterns

Traditional Haar-like rectangle features measure the difference between the average intensities of rectangular regions. For example, the value of a two-rectangle filter is the difference between the sums of the pixels within two rectangular regions. If we change the position, size, shape and arrangement of rectangular regions, the Haar-like features can capture the intensity gradient at different locations, spatial frequencies and directions. Viola an Jones [3] applied three kinds of such features for detecting frontal faces. By using the integral image, any rectangle filter type, at any scale or location, can be evaluated in constant time. However, the Haar-like features seem too simple and show some limits.

MB-LBP is an extension of LBP [2] that can be computed on multiple scales in constant time using the integral image. 9 equally-sized rectangles are used to compute a feature. For each rectangle, the sum of the pixel intensities is computed. Comparisons of these sums to that of the central rectangle determine the feature, similarly to LBP.

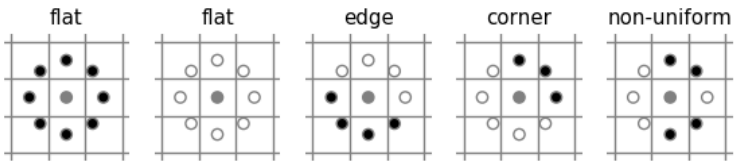


Figure 1 – Local Binary Patterns (3×3 MB-LBP)

The output of the MB-LBP operator can be obtained as follows:

$$\text{MB-LBP} = \sum_{i=1}^{8} \text{sign}(s_i - s_c) 2^i$$

where s_i is the sum of the pixel intensities of the ith neighborhood rectangle, s_c is the sum of the pixel intensities of the central rectangle,

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

MB-LBP of different sizes and at different locations (inside the reference window) are considered. The following function constructs a `std::vector` containing all MB-LBP features inside a considered window:

```
std::vector<mblbp_feature> mblbp_all_features()
{
  std::vector<mblbp_feature> features;

  for(int block_w = min_block_size; block_w <= max_block_size; block_w += 3)
    for(int block_h = min_block_size; block_h <= max_block_size; block_h += 3)
      for(int x = 0; x <= initial_window_w - block_w; ++x)
        for(int y = 0; y <= initial_window_h - block_h; ++y)
          features.push_back(mblbp_feature(x, y, block_w, block_h));

  return features;
}
```

1.2 Feature selection using Gentle Adaboost

2 Detection procedure

This section describes the different aspects of the detection process, used to apply the trained classifier.

2.1 Sliding window

The sliding window model is conceptually simple: **independently** classify all image patches as being object or non-object. Sliding window classification is the dominant paradigm in object detection and for one object category in particular – faces – it is one of the most noticeable successes of computer vision.

The following function constructs a `std::vector` containing all potential windows given the size of the image to be analyzed:

```
std::vector<window> get_potential_windows(const int img_w, const int img_h)
{
  std::vector<window> potential_windows;

  double w = initial_window_w; // current window width
  double h = initial_window_h; // current window height
  double scale = 1.0; // current scale
  double shift = scale * shift_delta; // current window shift (in pixels)
```

```
  while(w <= max_window_w && h <= max_window_h)
  {
    for(double x = 0.0; x < static_cast<double>(img_w) - w; x += shift)
      for(double y = 0.0; y < static_cast<double>(img_h) - h; y += shift)
        potential_windows.push_back(window(x, y, w, h, scale));

    // update values for next loop
    w = w * scaling_factor;
    h = h * scaling_factor;
    scale = scale * scaling_factor;
    shift = scale * shift_delta;
  }

  return potential_windows;
}
```

`scaling_factor` and `shift_delta`, as described in [3], are used to parametrize the sliding window process.

2.2 Cascade of classifiers

3 Classifier training process

3.1 Data

3.1.1 Dataset

3.1.2 Preprocessing

3.2 Gentle Adaboost

References

[1] S. Liao, X. Zhu, Z. Lei, L. Zhang, and S. Z. Li. *Learning Multi-scale Block Local Binary Patterns for Face Recognition*, pages 828–837. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[2] T. Ojala, M. Pietikainen, and D. Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision amp; Image Processing., Proceedings of the 12th IAPR International Conference on*, volume 1, pages 582–585 vol.1, Oct 1994.

[3] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001.