Face detection based on skin color

Henning Arthur

# Introduction

Face detection can be regarded as a specific case of object-class detection. In object-class detection, the task is to find the locations and sizes of all objects in an image that belong to a given class. Examples include upper torsos, pedestrians, and cars.

Face detection can be regarded as a more general case of face localization. In face localization, the task is to find the locations and sizes of a known number of faces (usually one). In face detection, one does not have this additional information.

Early face-detection algorithms focused on the detection of frontal human faces, whereas newer algorithms attempt to solve the more general and difficult problem of multi-view face detection. That is, the detection of faces that are either rotated along the axis from the face to the observer (in-plane rotation), or rotated along the vertical or left-right axis (out-of-plane rotation), or both. The newer algorithms take into account variations in the image or video by factors such as face appearance, lighting, and pose.

Eigenfaces is the name given to a set of eigenvectors when they are used in the computer vision problem of human face recognition. The approach of using eigenfaces for recognition was developed by Sirovich and Kirby (1987) and used by Matthew Turk and Alex Pentland in face classification. The eigenvectors are derived from the covariance matrix of the probability distribution over the high-dimensional vector space of face images. The eigenfaces themselves form a basis set of all images used to construct the covariance matrix. This produces dimension reduction by allowing the smaller set of basis images to represent the original training images. Classification can be achieved by comparing how faces are represented by the basis set.

The Viola–Jones object detection framework is the first object detection framework to provide competitive object detection rates in real-time proposed in 2001 by Paul Viola and Michael Jones. Although it can be trained to detect a variety of object classes, it was motivated primarily by the problem of face detection. This algorithm is implemented in OpenCV as cvHaarDetectObjects().

# Project Specification

Using OpenCV framework, implement a color based face detection application. The application should work on real-time data acquired from a webcam. Face detection should occur on every frame of the video capture.

The color of the human skin should be a variable for the application, in this way the user can change the object which he wants to detect. The application should identify pixels which match the color range for detection, and interconnect regions of the same color.

\*The application should examine the regions found, and output only those which match the characteristics of the human face. A box shape should be drawn over the face in the current frame.

# Flowchart with Modules

1. Input real-time video capture from web camera
2. Get frame from capture in RGB color space
3. Convert from RGB to HSI color space for skin color thresholding
4. Pixels in between threshold values should become black, the rest of the pixels white
   * H value is used for thresholding
5. Apply morphological operations on binary image (dilation, erosion, median smoothing)
6. Draw rectangle around remaining binary objects
7. Convert from HSI to RGB for output

## Module Description

1. Reads web camera input.
2. Gets a frame from the video (capures). This frame will be a bitmap image in RGB color code.
3. Convert the image from RGB to HSI. We use HSI in order to determine if the pixel matches the default value of the human skin. This value has to be located in an interval delimited by min\_threshol and max\_threshold, both expressed as Hue values. Comparative studies show that HSI color space offers the best results in this kind of thresholding. Alternatively, RGB or YCbCr can be used, or a combination of these.

From the “Skin segmentation using multiple thresholding” paper by Francesca Gasparini and Raimondo Schettini, we can read about optimal threshold values for different color spaces:

* YCbCr:
  + 77 <= Cb <= 127 AND 133 <= Cr <= 173
* RGB:
  + R > 95, G > 40, B > 20
  + Max{R, G, B} – Min{R, G, B} < 15
  + |R – G| > 15, R > G, R > B
  + \*Uniform daylight illumination
* HSI:
  + I > 40
  + If 13 < S < 110
    - 0 < H < 28 AND 332 < H < 360
  + If 13 < S < 75
    - 309 < H < 331
  + \*This offers the best results during experiments

These values can further be improved by personal experiments.

1. Using a 5x5 kernel, we apply Dilation and Erosion. Using a 3x3 kernel, we apply Median Smoothing. We apply the above operations once on every binary image. \*references
2. We label the objects in the image (grouped by black pixels).
3. For eyes-mouth triangle, we search the face region for non-face regions. Real-time processing is hard to achieve, as some images may take up to 28 seconds to process.

Alternative would be to verify only the eyes.

Other alternative is simply to see if we can find 3 non-regions in our face region. To remove eyebrows, we can give a condition that these regions have to be larger than the area of our face region. These 3 regions will form our triangle. If such a triangle is found, face detection is validated.

# Pseudocode

Open(videoCapture);

Frame = GetFrame(videoCapture);

Convert(frame, RGB, HSI);

For (pixel in frame) {

If minThreshold < pixel.H < maxThreshold

Pixel = black;

Else

Pixel = white;

}

Dilation(frame);

Erosion(frame);

MedianSmooth(frame);

Objects = Label(frame);

\*Detect facial features(Objects);

DrawRectangleAround(Objects);

Convert(frame, HSI, RGB);

Show(frame);

\*//facial feature detection

//search for triangle

For (every pixel in object)

If (pixel is not object)

Label(non\_region\_object)

For (every non\_region\_object)

If (non\_region\_object.area() > object.area() / 100)

Label\_object\_as\_feature(non\_region\_object)

If (feature\_objects.count == 3)

Validate face detection

\*axa simetrie