Zagazig University
Faculty of Engineering
ECE Department
Second Year (2024-2025)

Face Detection Project

Report

1. **Project Definition** is identifying and locating human faces in a digital Face detection image. Unlike face recognition, the goal here is not to identify who the person is, but simply to determine if a face exists and where it is located.
2. **Project Objective** is to develop a MATLAB program that performs face detection without relying on any high-level built-in functions.

Why not use built-in functions?

To gain a deeper understanding of:

- How image processing works internally.
- How face detection algorithms work conceptually.
- How to control data manually using low-level operations.

Algorithm and its mathematical foundation:

1. Convert RGB to YCbCr for better skin tone filtering
2. Threshold Cb and Cr to create a skin mask
3. Limit region to upper image (likely face area)
4. Clean mask using morphological operations
5. Extract regions and filter by shape (eccentricity)
6. Inside each region: check for 2 eyes and 1 nose using intensity filters
7. If found, draw a bounding box (Face Detected)

Mathematical implementation in MATLAB:

1. **Image Normalizing:**
   Normalize the input image to a fixed size 300*300 for faster processing using the resize function:
   ```
   img = imread("Path");
   img = imresize(img, [300 300]);
   ```
2. **YCbCr Transformation:**
   transform the image to YCbCr color space and extracting the chroma channels (Cb and Cr) to detect skin-colored pixels.
   ```
   imginycbcr = rgb2ycbcr(img);
   Cb = imginycbcr(:,:,2);
   Cr = imginycbcr(:,:,3);
   ```
3. **Making a skin color mask:**
   Thresholds the chroma channels to identify skin colored pixels
   ```
   skinrangemask = (Cb >= 77 & Cb <= 130) & (Cr >= 140 & Cr <= 160);
   ```

4. **Focusing on the upper 0.75 of the image and combining both filters:**

```
[rows, cols] = size(skinrangemask);
upperpartmask = false(size(skinrangemask));
upperpartmask(1:round(rows*0.75), :) = true;
skininupperpart = skinrangemask & upperpartmask;
```

5. **Cleaning the image:**

   Filling holes in the upper part of the image:

```
skininupperpart = imfill(skininupperpart, 'holes');
```

   and removing small regions (noise) with less than than 200 pixels:

```
skininupperpart = bwareaopen(skininupperpart, 200);
```

6. **Region Extraction**

   Which analyzes each connected component and measure their shape using eccentricity (how oval or circular the shape is) and then gets bounding boxes for drawing rectangles later:

```
result = regionprops(skininupperpart, 'BoundingBox', 'Eccentricity');
```

7. **Face filtering test :**

   Keeps regions with eccentricity < 0.9 (nearly oval) this filters out the long or skinny regions:

```
for i = 1:length(result)
    e = result(i).Eccentricity;
    if e < 0.9
```

8. **Eyes and nose tests**

   Crops and gray scales the candidates first then tests for eyes (dark regions <60) and tests for nose (mid gray regions (80-120)) then cleans up noise by bwareaopen function:

```
facepart = imcrop(img, rect);
```

```
gfacepart = rgb2gray(facepart);
gfacepart = double(gfacepart);
gfacepart = (gfacepart - min(gfacepart(:))) / (max(gfacepart(:)) -
min(gfacepart(:))) * 255;
gfacepart = uint8(gfacepart);

eyespart = gfacepart < 60;
eyespart = bwareaopen(eyespart, 8);
eyesresult = regionprops(eyespart, 'BoundingBox');

nosepart = (gfacepart > 80) & (gfacepart < 120);
nosepart = bwareaopen(nosepart, 20);
noseresult = regionprops(nosepart, 'BoundingBox');
```

9. **Face confirmation and rectangle drawing :**

   Only accepts a region if at least there is 2 eyes and 1 nose and then draws a rectangle on the image if it passes through the tests:

```
  if length(eyesresult) >= 2 && ~isempty(noseresult)
            rectangle(app.UIAxes, 'Position', rect, 'EdgeColor', 'cyan',
'LineWidth', 3);
            face = 1;
    end
```

10. **Display the results in the title :**

```
if face==1
title('Face Detected');
else
title('No Face Detected');
end
```

Problem faced and solutions:

1. **Lighting Variation**
   Skin color or facial shades can vary significantly under different lighting conditions, leading to inaccurate color detection.
   Solution is to convert to a more stable color space like ”YCbCr” or “HSV”.
2. **Background Interference with Skin Color**
   If the background contains skin-like colors, many false positives may occur.
   The solution is to limit the detection to the upper region and eccentricity filtering and reduce noise.
3. **Variations in Skin Colors and Types**
   The thresholds used (Cb and Cr) Uses a wider range of thresholds to cover diverse skin tones.
4. **False Positives Due to Shape Similarity**
   Some regions may resemble eyes or a mouth but are not actual facial features.
   Check the “Bounding Box” aspect ratio (width-to-height) and eccentricity for logical consistency.
5. **Performance and Speed**
   Processing large images or multiple images may slow down the system.
   The solution is to reduce image size (resizing to 300*300) and to use grayscale operations instead of RGB.

Final MATAB Code

```
img = imread("Path");
img = imresize(img, [300 300]);
imginycbcr = rgb2ycbcr(img);
Cb = imginycbcr(:,:,2);
Cr = imginycbcr(:,:,3);
skinrangemask = (Cb >= 77 & Cb <= 130) & (Cr >= 140 & Cr <= 160);
[rows, cols] = size(skinrangemask);
upperpartmask = false(size(skinrangemask));
upperpartmask(1:round(rows*0.75), :) = true;
skininupperpart = skinrangemask & upperpartmask;
skininupperpart = imfill(skininupperpart, 'holes');
skininupperpart = bwareaopen(skininupperpart, 200);
result = regionprops(skininupperpart, 'BoundingBox', 'Eccentricity');
imshow(img); hold on;
face = 0;
for i = 1:length(result)
e = result(i).Eccentricity;
if e < 0.9
rect = result(i).BoundingBox;
facepart = imcrop(img, rect);
gfacepart = rgb2gray(facepart);
gfacepart = double(gfacepart);
gfacepart = (gfacepart - min(gfacepart(:))) / (max(gfacepart(:)) - min(gfacepart(:))) * 255;
gfacepart = uint8(gfacepart);
eyespart = gfacepart < 60;
eyespart = bwareaopen(eyespart, 8);
eyesresult = regionprops(eyespart, 'BoundingBox');
nosepart = (gfacepart > 80) & (gfacepart <120);
nosepart = bwareaopen(nosepart, 20);
noseresult = regionprops(nosepart, 'BoundingBox');
if length(eyesresult) >= 2 && ~isempty(noseresult)
rectangle('Position', rect, 'EdgeColor', 'cyan', 'LineWidth',3);
face = 1;
end
end
end

if face
title('Face Detected');
else
title('No Face Detected');
end
```

Testing and results :

Test images	Results