-(Image processing project report)-

Under the supervision of : **DR / Azhar Ahmed Hamdy**

Presented by:

1- Yasser Salah Abdo Elsayed	**section: 5**
2- Abdelrahman Alaa Mohamed	**section: 3**
3- Abdelrahman Sobhy Mohamed	**section: 3**
4- Ahmed Fawzy Abdelaziz	**section: 1**
5- Mostafa Wahed Mohamed	**section: 5**
6- Mahmoud Mohamed Abdelhadi	**section: 5**
7- Ramiz Medhat Kamel	**section: 2**
8- Mostafa Ashraf Mohamed	**section: 5**

Table of contents:-

1- Problem definition..... (2)

2- Selected algorithms and their mathematical foundations .....(3)

3- Step-by-step implementation.....(5)

4- Challenges faced and solutions.....(8)

5- Results and evaluation.....(10)

1- problem definition:-

The project's main objective is to design and construct an executable program that uses MATLAP as a coding platform to do face detection on humans. The project must not include any high-level functions that contain complex face detection algorithms and in this specific instance like (cascade parameters) and (HAAR recognition algorithms).

Furthermore, the project must have an application development and team collaboration in order to finish it. The main concern for the project was to find an implementation to detect facial features with a simple algorithm that can run on MATLAB coding environment.

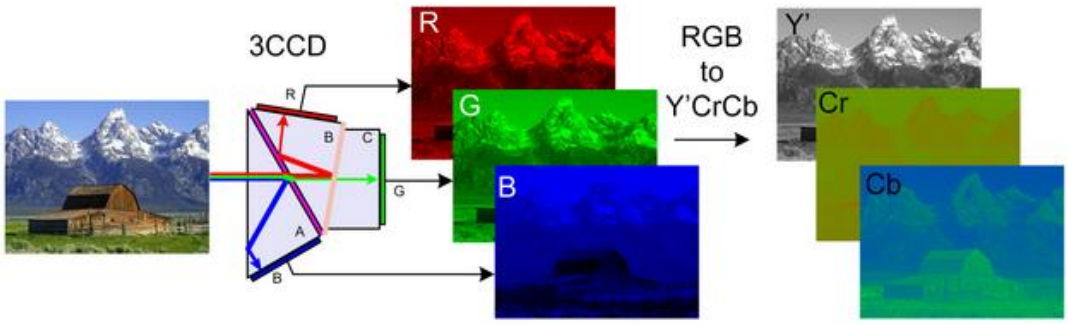
Face detection is a computer vision problem that involves finding faces in images. It is also the initial step for many face-related technologies, for instance, face verification, face modeling, head pose tracking, gender and age recognition, facial expression recognition, and many more.

2- Selected algorithms and their mathematical foundations:-

The team approved using low level analysis that focuses on color information in 4 different approaches :

1- **Eye color detection**
2- **Lips color detection**
3- **Skin color detection**
4- **Face border detection**

Each approach had a team working on, trying different coding methods to deal with the intensity of light and colors. There are several color models being used in face detection. Among them, the following are the significant ones: red, green, blue (RGB) model; hue, saturation, intensity (HSI) mode; and luminance, in-phase, quadrature (YIQ) model, and the Y-Cb-Cr color isolation system. The team agreed on using the Y-CB-CR color system instead in the common RGB system for several reasons including that the new system includes illumination control (Y) and red color saturation (CR) which is the base color reaction for skin with illuminated light on.

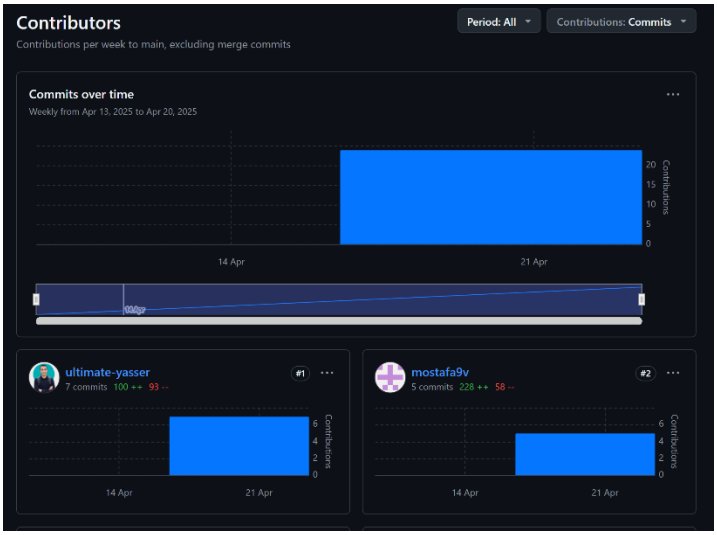


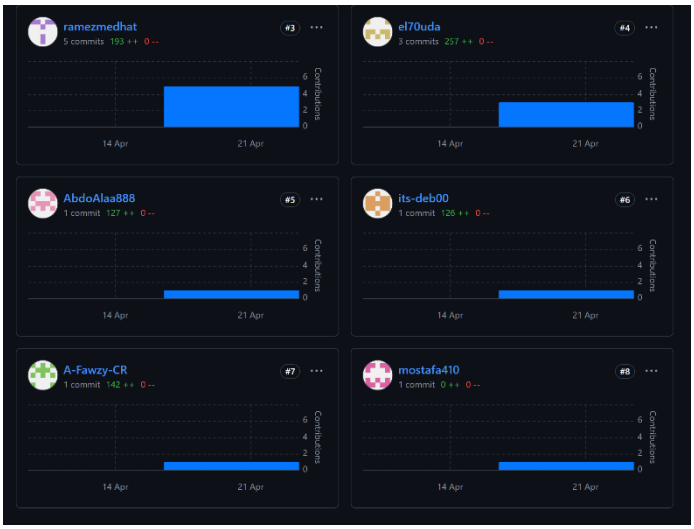
The mathematical representation for the color system used is that the light intensity value (Y) has resultant signals range from 16 to 235 and the color indicators (Cb and Cr range from 16 to 240). The values from 0 to 15 are called footroom, while the values from 236 to 255 are called headroom.

Human skin color information prominently builds a skin color cluster, which paves the way for faster face detection. Color processing is much faster, compared to other facial feature processing. Additionally, the color orientation is invariant under certain lighting conditions. Nonetheless, color information is sensitive to luminance change, and different cameras produce significantly different color values. For side viewed faces, the algorithm yields low accuracy.

3- Step-by-step implementation:-

The group was separated into three teams to work on the system then collaborate to put it all together. Team (1) was responsible for border detection testing and tracking. Team (2) worked on eye tracking and lip detection, while team (3) worked on GUI design and recognition testing.





initialization	test version 2 that adds sensitivity slider	18 hours ago
releases	Upload first release of the app	16 hours ago
team_1	Add files via upload	17 hours ago
team_2	Add files via upload	17 hours ago
team_3	Add files via upload	2 days ago
LICENSE	Initial commit	2 days ago
README.md	Initial commit	2 days ago

1- Oval detection and face border algorithm test:-

Team (1) worked with a unique approach to detect the area enclosed in a skin-like color and tested if it has a shape similar to the human face

```
        % Check if the shape is oval-like and not too bright
        isValid = ...
            eccentricity < 0.85 && ...  % Reject non-oval shapes
            solidity > 0.9 && ...       % Higher solidity for human-like shapes
            extent >= 0.5 && extent <= 0.85 && ...
            axisRatio >= 0.7 && axisRatio <= 1.3 && ...  % Aspect ratio of an oval
            brightness < 220 && ...  % Reject overly bright areas
            isSkinColor(roi);  % Check if the ROI contains human skin color
    end
```

which is oval and if so a circle or a rectangle is drawn around the object and detected as a face. The method was successful, but it was too dependent on the ideal human oval face which is not a realistic view.

2- Eye tracking and lips detection:-

On the other hand, team (2) tried working around isolating different facial gestures and unique objects like eyes and lips through working around detecting a color change in the area of detected skin. The eye tracking worked around detecting pair of black circles and test if there's a certain distance between them that is covered with the same skin tone detected earlier, while lips detection focused more on changing the color

```
% Analyze connected components
stats = regionprops(eyeCandidateMask, 'Centroid', 'BoundingBox', 'Eccentricity');
filteredCenters = [];

imgW = size(roi,2);
for i = 1:length(stats)
    bbox = stats(i).BoundingBox;
    aspectRatio = bbox(3) / bbox(4);  % width / height

    if aspectRatio > 0.4 && aspectRatio < 3.2 && stats(i).Eccentricity < 0.96
        cx = stats(i).Centroid(1);
        if cx > 0.10 * imgW && cx < 0.90 * imgW  % Looser side crop
            filteredCenters = [filteredCenters; stats(i).Centroid];
        end
    end
end
```

```
% Check for eye-like pairs
eyePairs = [];
for i = 1:size(filteredCenters, 1)
    for j = i+1:size(filteredCenters, 1)
        pt1 = filteredCenters(i,:);
        pt2 = filteredCenters(j,:);
        dist = norm(pt1 - pt2);
        if dist > 12 && dist < 130 && abs(pt1(2) - pt2(2)) < 35 && abs(pt1(1) - pt2(1)) > 10
            eyePairs = [eyePairs; pt1, pt2];
        end
    end
end
```

tone of the skin slightly around the lower part of the detected object and the exact position of it.

3- Gui and skin mask and recognition:-

Furthermore, team (3) focused more on app development and database and recognition testing. The final application design was successful, but the database proved to be more complicated than expected and would require high level functions to connect the detection code to the database which isn't allowed in this project, so only the application and the main user interface were introduced to the final project. However, skin mask technology was researched on the team as well to improve skin detection and study color sensitivity based on team (1) and team (2) work around color isolation.

```
function objects = detectSkinObjects(frame, Cr_min, Cr_max)
    imgYCbCr = rgb2ycbcr(frame);
    Cb = imgYCbCr(:,:,2);
    Cr = imgYCbCr(:,:,3);

skinMask = (Cb >= 77 & Cb <= 127) & (Cr >= Cr_min & Cr <= Cr_max);
    skinMask = medfilt2(skinMask, [5 5]);
    skinMask = imfill(skinMask, 'holes');
    skinMask = bwareaopen(skinMask, 100);

    stats = regionprops(skinMask, 'BoundingBox', 'Area');
    minArea = 4000;
    objects = struct('BoundingBox', {}, 'points', {});
    for i = 1:length(stats)
        if stats(i).Area > minArea
            objects(end+1).BoundingBox = stats(i).BoundingBox;
            objects(end).points = 0;
        end
    end
end
```

4- Challenges faced and solutions:-

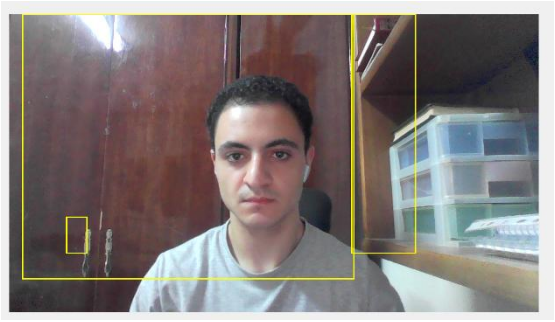
For each technique tried and used, a problem exists in the code or the lighting that eliminated the idea which will be explained in this section.

1- Inconsistency of face shapes:-

The system used in oval detection was faulty due to different face dimensions and had no direct result due to fixed values for the oval-like shape that needs to be detected , resulting in error as shown.

2- Eye tracking false detection:-

Working around dark color plates like black is tricky, cause black is part of many other colors like brown, so anything in the background can be detected as an eye if the system's sensitivity isn't one by one as shown here. And the code used in the eye tracking had a fixed red component ratio.



3- High lag and frame drop rate:-

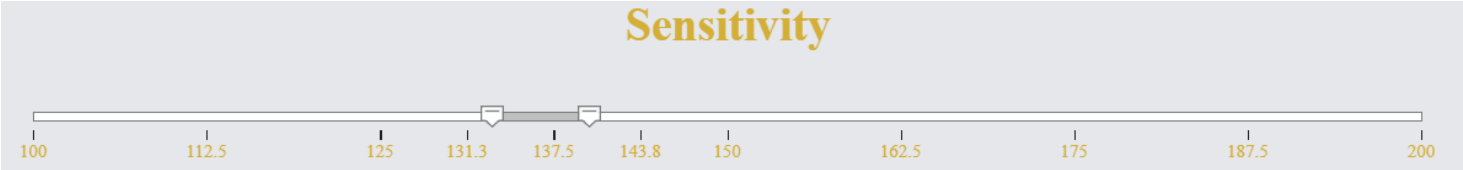
While working on the lip detection formula, the processing power was too high that the live feed capture was lagging, and the frame rate of the web cam feed dropped to 5 so the lip detection program was cancelled.

4- High level functionality for recognition:-

As previously mentioned, doing face recognition requires a data base that has a link to the active code that capture image from the web cam, hence, doing such a code with a low level function of detection such as skin mask or eye detection or oval shape detection would be difficult, due to several reasons mainly caused by complexity of the detection required for the data base to match faces together on a live feed.

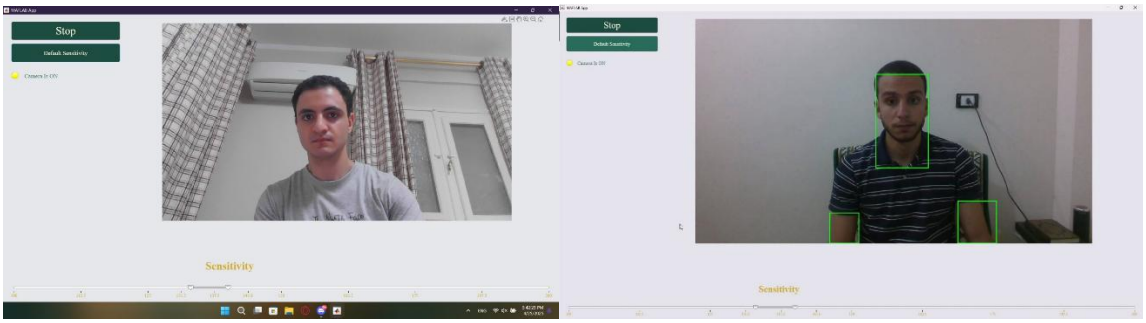
Solutions:-

Working around all the problems mentioned, we figured that using a simple skin mask technique would be the most accurate way to do a low-level face detection program, and to make it even more accurate, the value of red saturation (Cr value) in the captured image won't be constant, instead will be controlled by the user to change by himself. This specific change was applied due to different skin tones and lighting conditions for different users so that the user can set the bar to the value that starts detecting his face, while the default value is there and works on average for most users, some extreme light conditions would require sliding the sensitivity bar then letting go of it to change the Cr value that can be detected as shown .

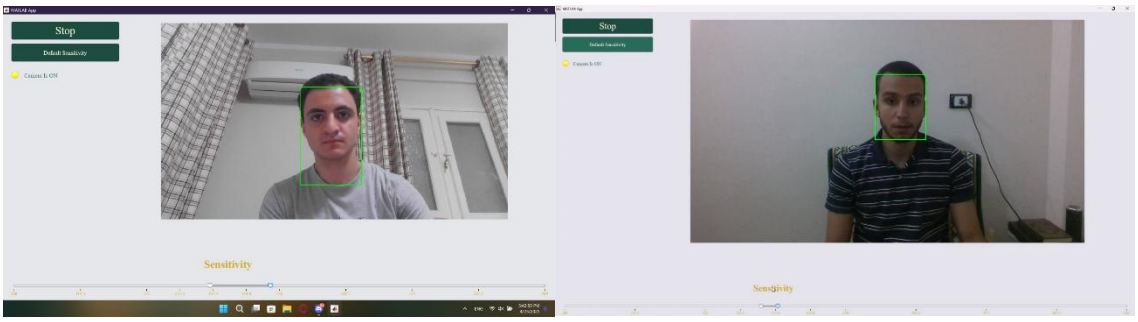


5- Results and evaluation:-

As shown in the following figures, this is the effect of default face detection algorithm on two users without changing the sensitivity, as said the default sensitivity can work but can also be faulty.



But, by changing and customizing the value of sensitivity, the program works perfectly on both users as shown in the next figures.



In conclusion, the program is executable and can be customized according to the preferences of each individual user, making it both successful and user-friendly. Recommendations for the next developers are to use a high-level function to have accurate face detection, and to increase the accuracy of the program by reducing the bugs and introducing new mechanics to the skin mask technique.

Theoretical reference:-

Hasan, M. K., Ahsan, Md. S., Abdullah-Al-Mamun, Newaz, S. H., & Lee, G. M. (2021). Human face detection techniques: A comprehensive review and future research directions. *Electronics*, *10*(19), 2354. https://doi.org/10.3390/electronics10192354