**Cartoon Faces**

Team members:

* Yasmine Alaa Ahmed.
* Mohamed Gamal.
* Berlnty Kerlos Fakhry.
* Mostafa Yasser Darwish.

Brief:

Detecting faces in an image, detecting eyes in then replacing yes and mouth with cartoon eyes and cartoon mouth, finally clustering the image to be more like a drawn one.

Sequence:

1. Read image and detect faces.
2. Detect image in each face

* Detect region of eyes geometrically.
* Thresholding of skin color.
* Detecting contours of eyes.
* Determining eyes’ coordinates.

1. Copying new eyes.

* Resizing cartoon eyes.
* Calculating their coordinates relating to the original image then copy them.

1. Copying new mouth.

* Resizing cartoon mouth.
* Calculating its coordinates relating to the original image then copy it.

1. Cartoonify the resulted image

* Detecting edges and removing noise.
* Appling k-means clustering.

Work load:

* Face detection, Eye and mouth installation: Yasmine Alaa , Berlnty Kerlos.
* Skin threshoulding and eye countouring: Berlnty Kerlos.
* Clustring: Yamine Alaa.
* Edge detection, detect eyes geometrically: Mohamed Gamal.

Trials:

1. Detecting Eyes

* Convert image to CrCb color space then Thresholding of skin color range.
* 
* Convert image to HSV color space then Thresholding of skin color range
* Convert image to HSV color space and equalizing the histogram space then Thresholding of skin color range.

1. Copying eyes

* In case of rotation of eyes black regions appears and part of the image is eroded.
* So, we use PNG images and convert it to BGRa where a is the opacity and don’t show the pixels where a < 0.5
* And to solve the problem of eroded regions we copied the image in larger matrix its dimensions are equal to the diagonals of the original one.

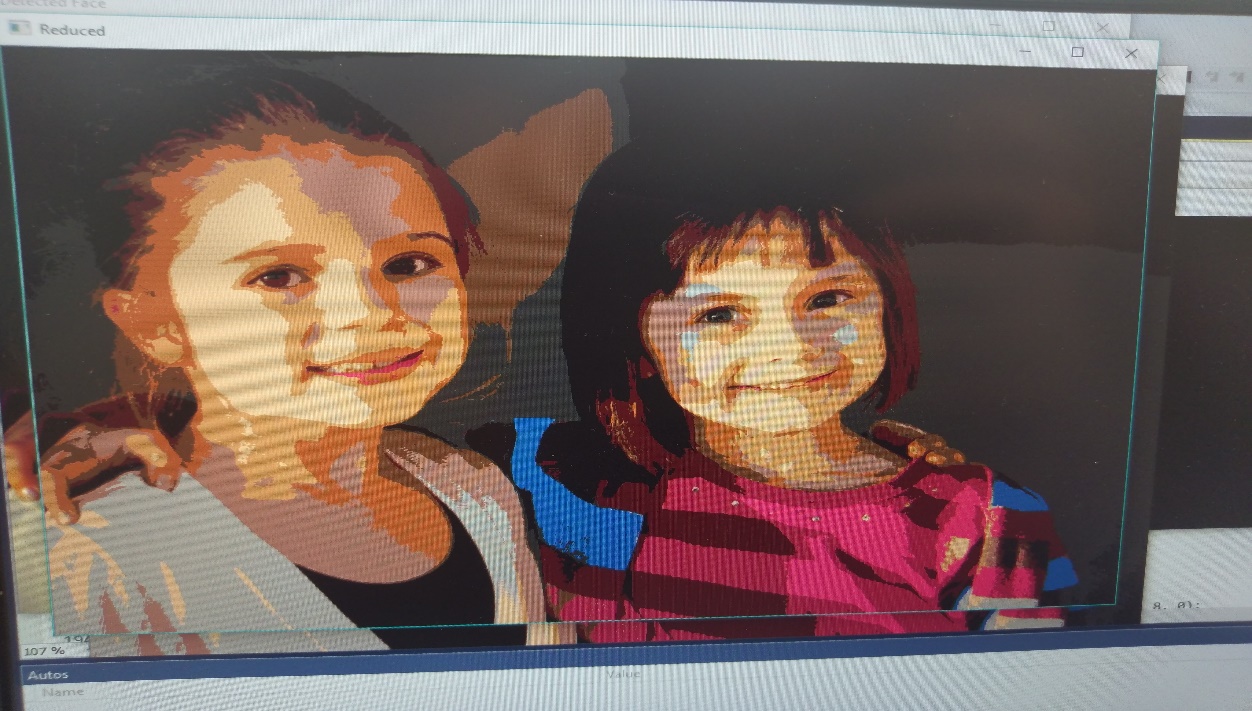
1. Cartoonifying:

* Detecting edges.

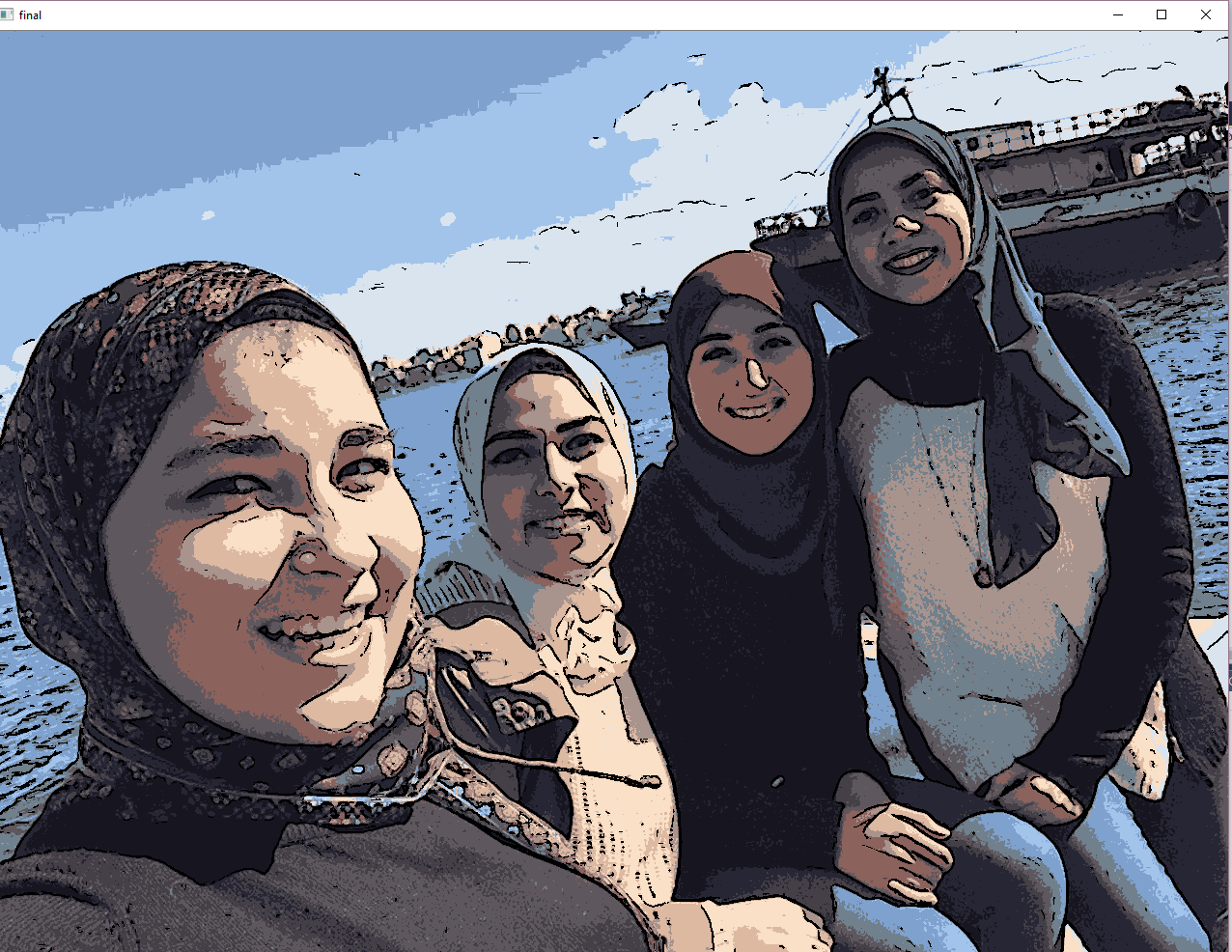


1. Clustering:

* calculating histogram.
* Determining peeks and their colors.
* Applying k-means clustring.



* Didn’t detect any eyes usin CrCb color space.



* Same image using HSV color space but problem was with the ilumination.
* Accepted output

|  |
| --- |
|  |
|  |
|  |

* Calculating wrong coordinates of mouth. 

**Clustering:**

-**Used algorithm**: Kmeans clustering function which is built in in open cv library

**-Problems:**

* There was a problem in mapping colors to each cluster but it is solved by taking the colors of centers to the cluster
* Choosing the number of clusters, solved by taking the number of peaks in the histogram

**Face detection:**

**-Used algorithm:** haarcascade classifier

**Eye installation:**

**-Used algorithm:** function resize() which resizes cartoon eye’s scale to fit face’s eye then we calculate the slope to rotate the eye with angle theta then we add it to the face with a known ratio

-**Problems:**

* Resizing the eye with variable ratio
* Neglecting cartoon eye’s background , solved by checking for the 4th channel of the image (alpha)
* Rotating the eye

**Detect eyes geometrically**

* Function parameters:
* Image matrix
* Center point
* Rectangle of image
* Description:

This function detect eyes as follows:

* It converts the image to gray scale level
* It loops over the image to draw the x-axis and y-axis.
* It calls function cut eyes which will be described next.

1. **Cut eyes function**

* function parameters
* Image matrix
* Center point
* Rectangle of image
* Description

This function generates and crop the eyes geometrically

* By using built in function Rect it takes the beginning point, and width and height of the cropped area from the image.

1. **Cartoonify Image function**

* Function parameters:
  + Source image
  + Destination image
* Description
  + This function converts image to gray scale, removes noise and detect edges.

1. **Remove Pepper Noise function**

This function removes dots and noise from the image.

Trials:

1. Detecting eyes by converting image to gray scale and looping on face area and storing the black dots as it belongs to eyes and take the average points.
2. Detecting eyes by converting image to hsv scale and choosing the eyes color to be shown only .
3. Detecting eyes geometrically and that way succeeded.