horizontal line

**Computer Vision**

Part I: Image Cartoonifying

**3rd May 2021**

| **Name** | **ID** |
| --- | --- |
| **Jamal Eldin Ahmed Khalaf** | **15** |
| **Yomna Gamal El-Din Mahmoud** | **63** |

# Applying Image Processing Filters For Image Cartoonifying

we want to make the real-world images look like they are genuinely from a cartoon. The basic idea is to fill the flat parts with some color and then draw thick lines on the strong edges. In other words, the flat areas should become much more flat and the edges should become much more distinct. We will detect edges and smooth the flat areas, then draw enhanced edges back on top to produce a cartoon or comic book effect.

# Implementation

## Image to Grayscale

First we read image as a grayscale image then it became as follow:

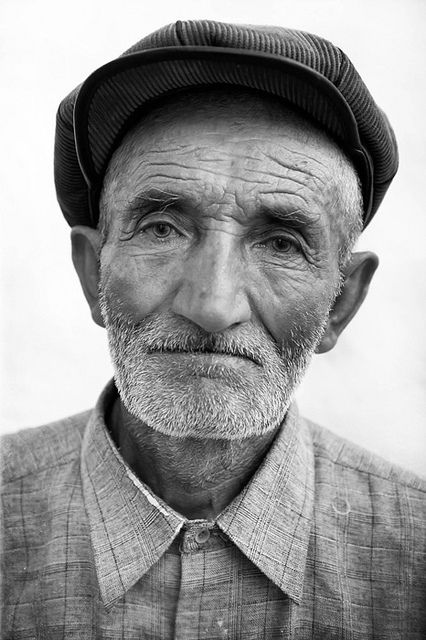
gray\_img = cv2.imread(imgpath, 0)

cv2\_imshow(gray\_img)

Original



Grayscale



## Noise Reduction Using Median Filter

median = cv2.medianBlur(gray\_img,9)  
to reduce the noise in the image before we use a Laplacian edge filter  


## Edge Detection Using Laplacian Filter

laplacian = cv2.Laplacian(median ,ddepth=cv2.CV\_8UC1,ksize=5)  
a Laplacian filter is used for edge detection  


## Edges Thresholding

It’s the final result of that part that will be added to the blurred coloured image to get a cartonifing image.  
Edgesimg = cv2.threshold( laplacian, 127, 255, cv2.THRESH\_BINARY\_INV)



## Bilateral Filter

It’s applied to the original coloured image.

blur = cv2.bilateralFilter( imgRGB,9,640, 426)



## 

## 

## 

## 

## 

## 

## 

## Creating Cartoon Effect

cartoon\_img = np.array(blur)

indexes = np.argwhere(Edgesimg[1] == 0)

for i in indexes:

cartoon\_img[i[0]][i[1]] = [0,0,0]

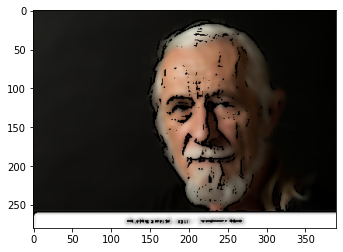
This part combine both images blurred one and image contains edges.

 **+** **= **

# 

# Sample tests (original → cartoonified)

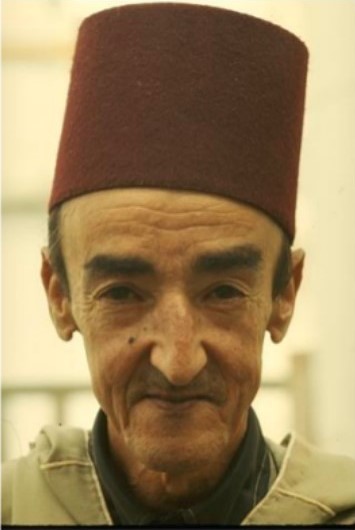
 **→ **

 **→ **

 **→ **

 **→ **

 **→ **

 **→ **