

Lab 1

# Image Cartoonifier

---

Menna Ghanem 3798

Nairouz Mahmoud 4010

## Introduction :

Our project is about images into its cartoonized version. To be able to do this process to we need two main steps: detecting edges in these photos and then combine them with a smoothed version of the original image.

## Steps:

1. First we used OpenCV library to read images `cv2.imread()` and also used for displaying them `cv2.imshow()`.
2. We used `cv2.cvtColor()` to convert the image from RGB to grayscale.
3. We applied a median filter of size (7\*7) to reduce noise using function `cv2.medianBlur()`
4. Then we applied laplacian filter to detect edges of the grayscale image by function `cv2.Laplacian()` and we tried different kernel sizes:
  - a. For example  $k=5$  tends to detect more details and more edges were obvious ,  $k=3$  detected stronger borders and  $k=1$  barely detected any edges.
5. We applied threshold to the output of laplacian filter to make edges either white or black with a threshold value 125 using function `cv2.threshold()`
6. We applied bilateral filter to the original image to smooth flat regions while keeping edges sharp. The image tends to be smoother as we increased the `sigmaColor` and `sigmaSpace` parameters. We used the function `cv2.bilateralFilter()` and applied it multiple times to get more cartoonized effect.
7. Last, we needed to combine the output of the bilateral filter and the pervious detected edges in one image to get the cartoonized image. We used `cv2.bitwise_and()` to combine them

## Code:

```
import cv2

import numpy as np

img=cv2.imread('C:/Users/M3MO/Desktop/test.PNG')

cv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

cv2.imshow('Gray',gray)

cv2.waitKey(0)

cv2.destroyAllWindows()

median = cv2.medianBlur(gray,7)

cv2.imshow('Smoothed Gray',median)

cv2.waitKey(0)

cv2.destroyAllWindows()

la = cv2.Laplacian(median,cv2.CV_8UC1,ksize = 5,scale = 1,delta = 0)

cv2.imshow('Laplacian filter',la)

cv2.waitKey(0)

cv2.destroyAllWindows()

ret3,th3 = cv2.threshold(la,125,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

cv2.imshow('Laplacian after threshold',th3)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

```

th4 = th3

print(th3.shape[0])

print(th3.shape[1])

for i in range(0,th3.shape[0]):

    for j in range(0,th3.shape[1]):

        if(th3[i][j]==0):

            th4[i][j]=255

        else:

            th4[i][j]=0

cv2.imshow('After flipping',th4)

cv2.waitKey(0)

cv2.destroyAllWindows()

for i in range(0,6):

    bi = cv2.bilateralFilter(img, d=9, sigmaColor=9, sigmaSpace=7)

cv2.imshow('bilateral filter',bi)

cv2.waitKey(0)

cv2.destroyAllWindows()

res=cv2.bitwise_and(bi,bi,mask=th3)

cv2.imshow('Cartoonized',res)

cv2.waitKey(0)

cv2.destroyAllWindows()

```

## Samples:

Original image



Grayscale image



Smoothed Grayscale



Laplacian filter k=5



After Thresholding with 125



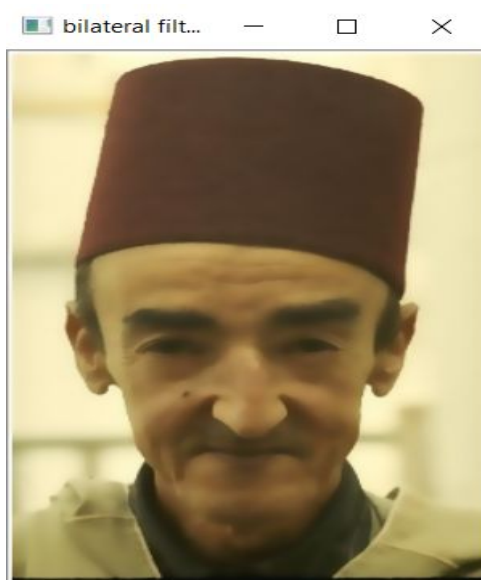
Detected Edges



**Bilateral Filter**



**Bilateral filter with other parameters**



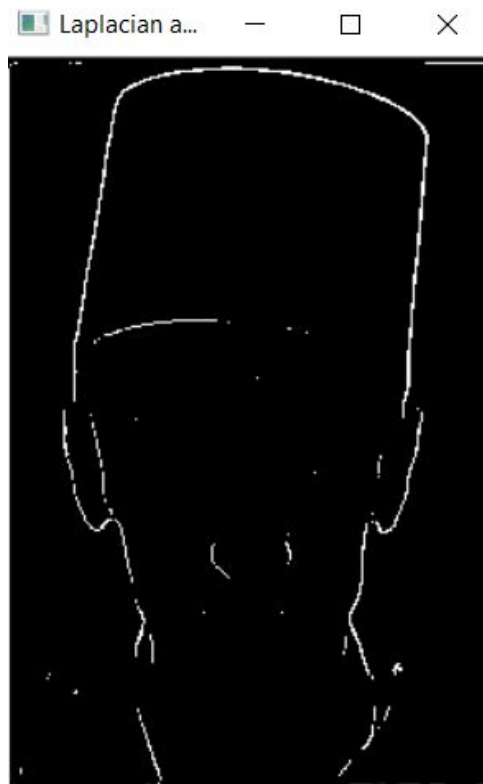
**Cartoonized image**



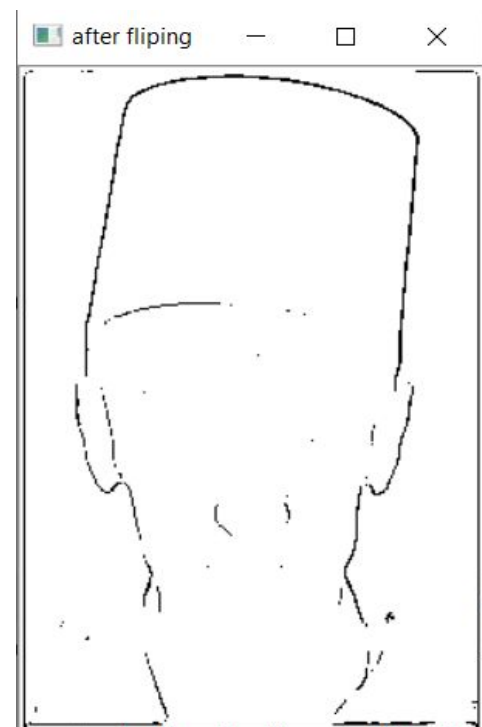
**Laplacian with k=3**



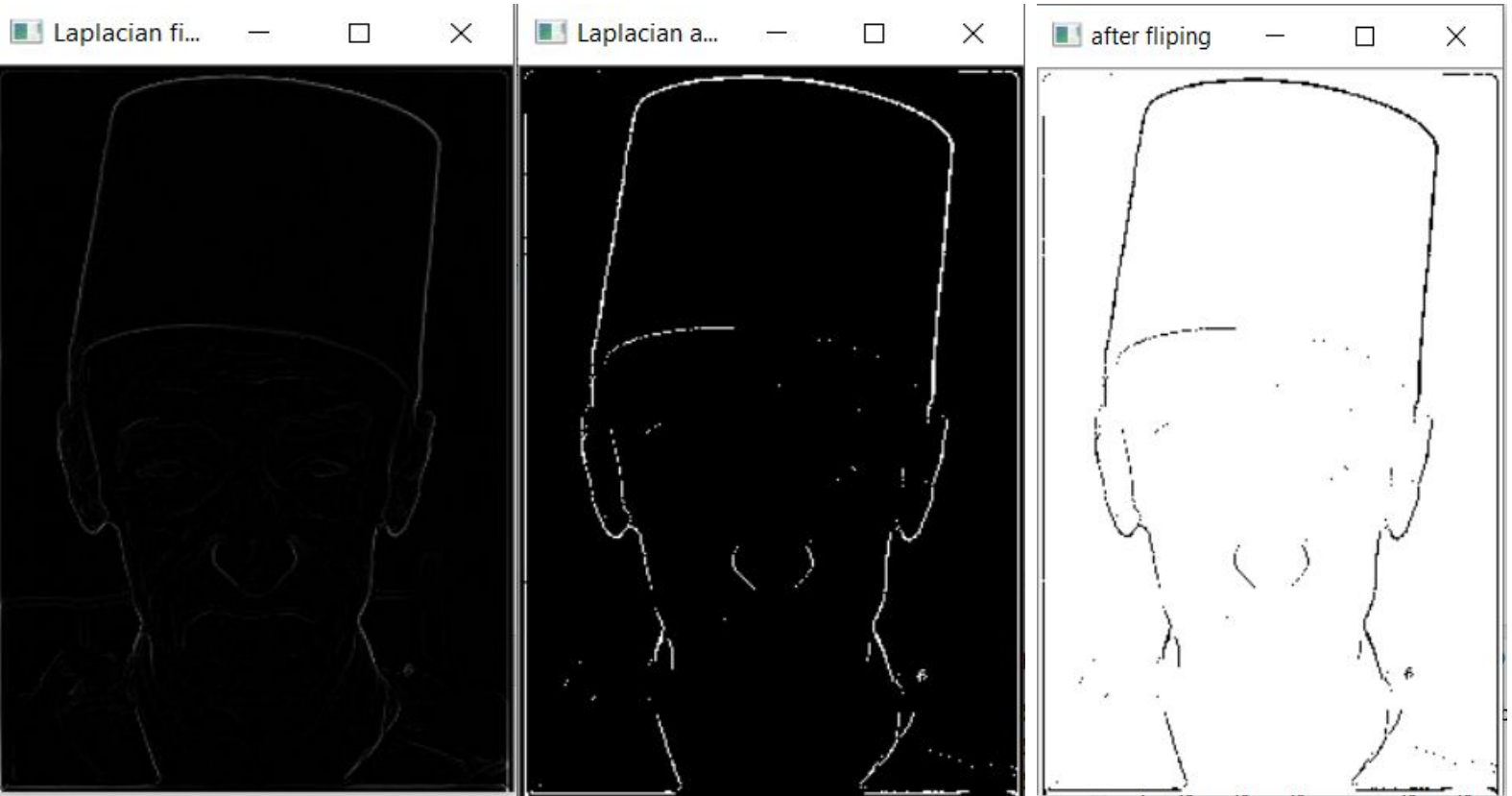
**Thresholding**



**Detected edges**

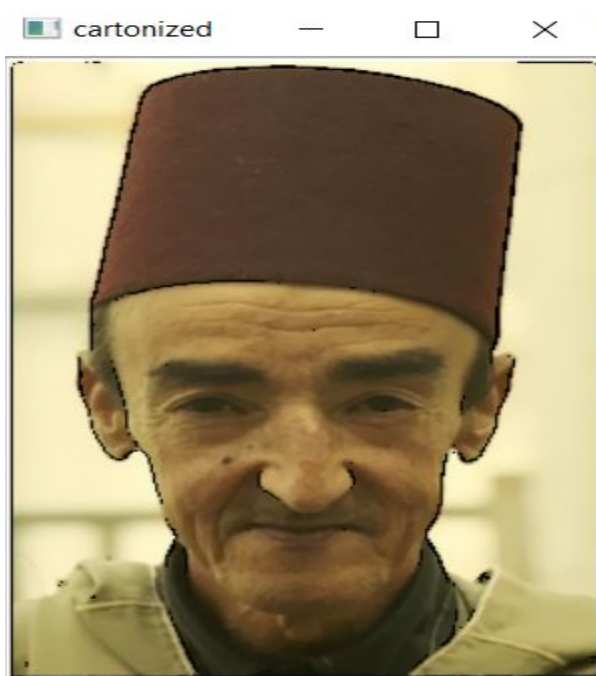
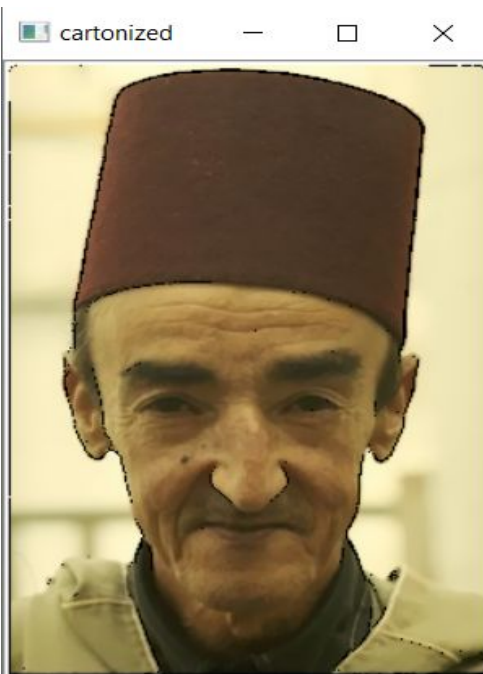


K=1



Cartoonized for k=1

Cartoonized for k=3





## Other Samples :

