Cartoonization using Edge Detectors

Submitted to
Sarthak Padhi Sir

By
Kirti Padhi
B121025
CSE

Write the OpenCV code for photograph cartoonization using various edge detection operators. Show the output of the process using the following operators:

- **Prewitt's operator**
- **Sobel's operator**
- **Kirsch operators**
- **Canny operator**
- **LoG operator**

```
# importing modules
import cv2
import numpy as np
```

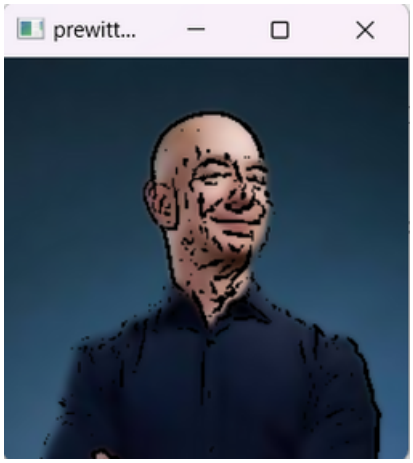
Prewitt's Operator

```
def prewitt_edge_detection(image):
    kernels = [np.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]]),
               np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]])]

    results = [cv2.filter2D(image, cv2.CV_64F, kernel) for kernel in kernels]
    magnitude = np.max(results, axis=0)

    # Normalize and convert to uint8
    magnitude = cv2.normalize(magnitude, None, 0, 255, cv2.NORM_MINMAX, cv2.CV_8U)
    magnitude=inverseAndthreshold(magnitude)

    return magnitude

pre_edges=prewitt_edge_detection(gray)
pre_cartoon=cartoon(pre_edges)
```

Original Image

Intermediate Image

Cartoonized Image



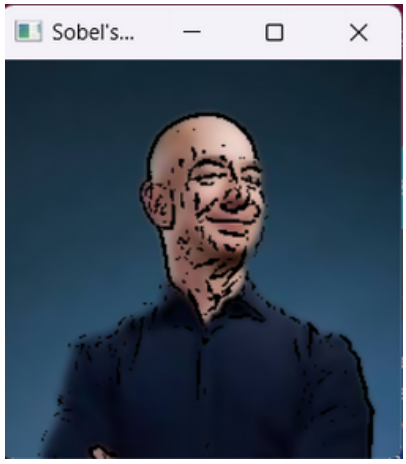
Sobel's Operator

```
def sobel_edge_detection(image):
    kernels = [np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]]),
               np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])]

    results = [cv2.filter2D(image, cv2.CV_64F, kernel) for kernel in kernels]
    magnitude = np.max(results, axis=0)

    # Normalize and convert to uint8
    magnitude = cv2.normalize(magnitude, None, 0, 255, cv2.NORM_MINMAX, cv2.CV_8U)
    magnitude=inverseAndthreshold(magnitude)

    return magnitude

s_edge=sobel_edge_detection(gray)
s_cartoon=cartoon(s_edge)
```

Original Image

Intermediate Image

Cartoonized Image

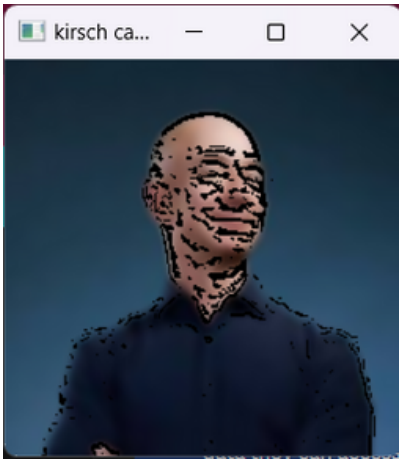


Kirsch Operator

```
def kirsch_edge_detection(image):
    kernels = [np.array([[-3, -3, 5], [-3, 0, 5], [-3, -3, 5]]),
               np.array([[-3, 5, 5], [-3, 0, 5], [-3, -3, -3]]),
               np.array([[5, 5, 5], [-3, 0, -3], [-3, -3, -3]]),
               np.array([[5, 5, -3], [5, 0, -3], [-3, -3, -3]]),
               np.array([[5, -3, -3], [5, 0, -3], [5, -3, -3]]),
               np.array([[-3, -3, -3], [5, 0, -3], [5, 5, 5]]),
               np.array([[-3, -3, -3], [-3, 0, -3], [5, 5, 5]]),
               np.array([[-3, -3, -3], [-3, 0, 5], [-3, 5, 5]])]

    results = [cv2.filter2D(image, cv2.CV_64F, kernel) for kernel in kernels]
    magnitude = np.max(results, axis=0)
```

```
    # Normalize and convert to uint8
    magnitude = cv2.normalize(magnitude, None, 0, 255, cv2.NORM_MINMAX, cv2.CV_8U)
    magnitude=inverseAndthreshold(magnitude)

    return magnitude

kirsch_edge=kirsch_edge_detection(gray)
k_cartoon=cartoon(kirsch_edge)
```

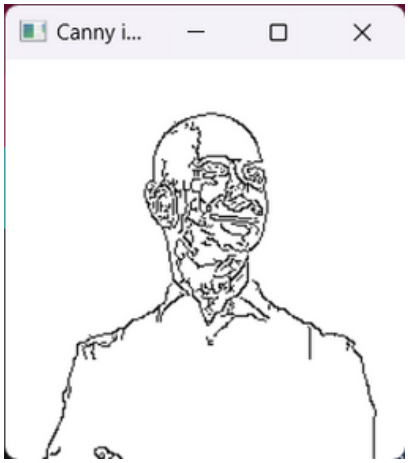
Original Image | Intermediate Image | Cartoonized Image

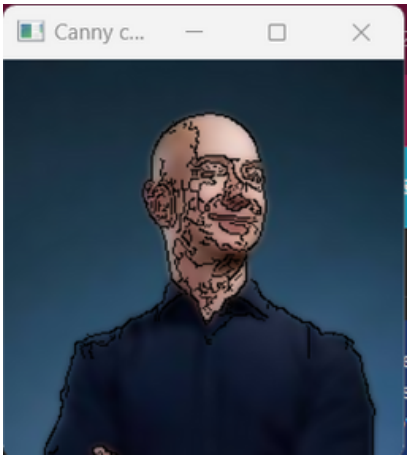


Canny Operator

```
def canny_edge_detection(image):
    edges = cv2.Canny(image, 50, 150)
    edges=inverseAndthreshold(edges)

    return edges

c_edge=canny_edge_detection(gray)
c_cartoon=cartoon(c_edge)
```

Original Image | Intermediate Image | Cartoonized Image





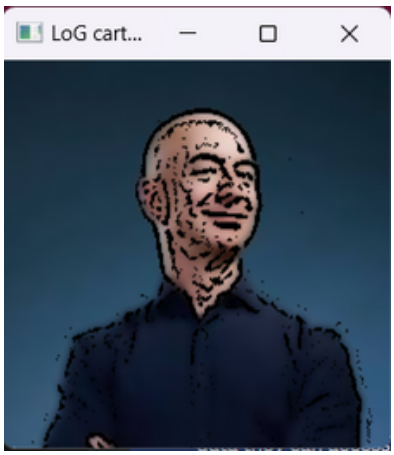
Laplacian of Gaussian Operator

```
def LoG_edge_detection(image):
    blurred = cv2.GaussianBlur(image, (5, 5), 0)

    # Apply Laplacian of Gaussian
    edges_log = cv2.Laplacian(blurred, cv2.CV_64F)

    # Normalize and convert to uint8
    edges_log = cv2.normalize(edges_log, None, 0, 255, cv2.NORM_MINMAX, cv2.CV_8U)
    edges_log=inverseAndthreshold(edges_log)

    return edges_log

l_edge=LoG_edge_detection(gray)
l_cartoon=cartoon(l_edge)
```

Original Image

Intermediate Image

Cartoonized Image



Other Functions and main function

```
def inverseAndthreshold(edge):
    edge=cv2.bitwise_not(edge)
    edge = cv2.adaptiveThreshold(edge,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY,9,9)

    return edge
```

```
def cartoon(edge):
    color=cv2.bilateralFilter(img,9,250,250)
    cartoon =  cv2.bitwise_and(color,color,mask=edge)

    return cartoon
```

```
img= cv2.imread("original.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

cv2.imshow("orginal image",img)
cv2.imshow("gray",gray)
cv2.imshow("prewitt intermediate",pre_edges)
cv2.imshow("Sobel's intermediate",s_edge)
cv2.imshow("kirsch intermediate",kirsch_edge)
cv2.imshow("Canny intermediate",c_edge)
cv2.imshow("LoG intermediate",l_edge)
cv2.imshow("prewitt cartoon",pre_cartoon)
cv2.imshow("Sobel's cartoon",s_cartoon)
cv2.imshow("kirsch cartoon",k_cartoon)
cv2.imshow("Canny cartoon",c_cartoon)
cv2.imshow("LoG cartoon",l_cartoon)

cv2.waitKey(0)
```

Source Code Link:

https://github.com/Kirti-kn/Cartoonization_edge_detector

THANK YOU