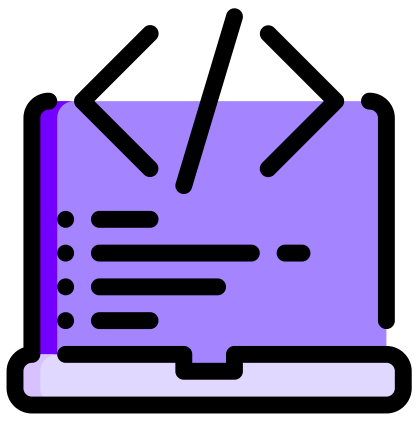
MACHINE LEARNING INDIA

Easy image

CARTOONIZATION WITH OPENCV

OpenCV

In just 25 lines of Python code!



We'd use OpenCV and Numpy!

Over the last few years **professional cartoonizer** softwares have popped up all over the place but they're rarely **free**. In order to achieve the basic cartoon effect, you don't need powerful rendering software or even years of experience. Yes, that's true!

All you need is essentially - a **bilateral filter** and some **edge detection**. The bilateral filter will reduce the color palette, necessary for the cartoon look, and edge detection will allow production of bold silhouettes.

Here's what we're gonna do:

- Apply a bilateral filter to reduce the color palette of the image.
- Convert the original color image to grayscale.
- Apply a median blur to reduce image noise in the resultant grayscale image.
- Create an edge mask from the grayscale image using adaptive thresholding.
- Combine the color image from step 1 with the edge mask from step 4.

The code: Part #1

In this part, we're setting the parameters, reading the image and resizing it.

File Edit Format Run Options Window Help

```
import cv2
import numpy as np

num_down = 2         # number of downsampling steps
num_bilateral = 7    # number of bilateral filtering steps

img_rgb = cv2.imread("IMG_20180206_145924758.jpg")
print(img_rgb.shape) #prints the dimension of the picture

#resizing so as to get optimal results after un sampling is done.
img_rgb=cv2.resize(img_rgb,(800,800))
```

Part #2:

In this part, we're downsampling the image and then applying bilateral filter the mentioned amount of times

```
#resizing so as to get optimal results after un sampling is done.
img_rgb=cv2.resize(img_rgb,(800,800))

# downsample image using Gaussian pyramid
img_color = img_rgb
for _ in range(num_down):
    img_color = cv2.pyrDown(img_color)

# repeatedly apply small bilateral filter instead of
# applying one large filter
for _ in range(num_bilateral):
    img_color = cv2.bilateralFilter(img_color, d=9,
                                    sigmaColor=9,
                                    sigmaSpace=7)
```

Part #3:

In this part, we upsample, convert the image to grayscale, apply median blur and then thresholding.

```
# upsample image to original size
for _ in range(num_down):
    img_color = cv2.pyrUp(img_color)

img_gray = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2GRAY)
img_blur = cv2.medianBlur(img_gray, 7)

img_edge = cv2.adaptiveThreshold(img_blur, 255,
                                 cv2.ADAPTIVE_THRESH_MEAN_C,
                                 cv2.THRESH_BINARY,
                                 blockSize=9,
                                 C=2)
```

Part #4:

In this part, we perform '**bitwise AND**' , and then **display** the resultant image. Do make sure to add in the end: **cv2.waitkey(0)**.

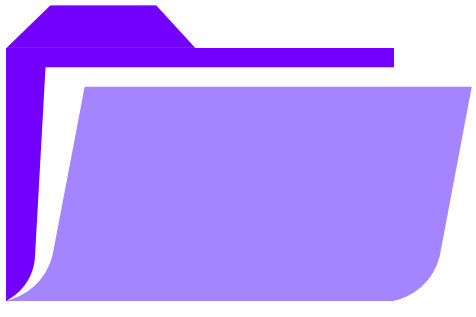
```
# convert back to color, bit-AND with color image
img_edge = cv2.cvtColor(img_edge, cv2.COLOR_GRAY2RGB)
img_cartoon = cv2.bitwise_and(img_color, img_edge)

# display
#cv2.imshow("cartoon", img_cartoon)
stack=np.hstack([img_rgb,img_cartoon])
cv2.imshow('Stacked Images',stack)
```

The output:

```
#resizing so as to get optimal results after un sampling is done.
img_rgb=
# downsa
img_colo
for  _ i
    img_
# repeat
# applyi
for  _ i
    img_
# upsamp
for  _ i
    img_
img_gray
img_blur
img_edge
```


```
# convert back to color, bit-AND with color image
img_edge = cv2.cvtColor(img_edge, cv2.COLOR_GRAY2RGB)
img_cartoon = cv2.bitwise_and(img_color, img_edge)
```



Content curators:

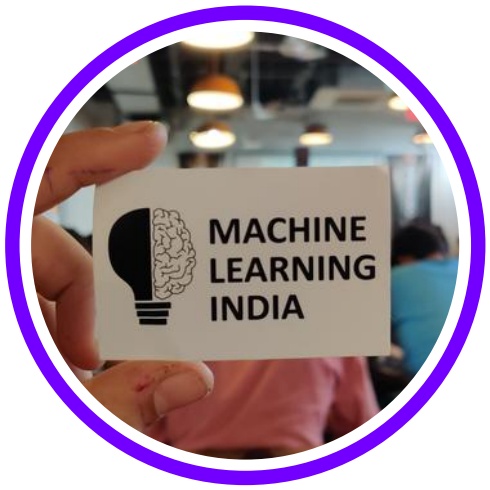
Bhavishya Pandit and Priyanka Kasture.

Notable references:

- How to create a cool cartoon effect with OpenCV and Python on **www.askaswiss.com**.

Important note:

The links to these resources will be put up on our Telegram. Channel ID: **@machinelearning24x7**.



Enjoying our micro-tutorials?

Let us know in the comments! If you like our content and find it **valuable**, do give us a **follow**! Your **love** and **support** inspires us to keep delivering the best we can! ❤️

Like.

Comment.

Share.