
# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

## 19CSE305 – MACHINE LEARNING

## PROJECT REPORT

## Topic: Cartoonify an image with OpenCV

**Submitted to:**

Dr. Athira M Nambiar

Asst. Professor

Department of CSE

Amrita School of Engineering

Chennai Campus

# Cartoonify an image with OpenCV

## Abstract:

This project is all about building a photo cartoonifyer using Python and OpenCV. In this project we are accepting input image and then extracting edges, gray image, then applying median blur with Bilateral Filter technique to convert input image to cartoon image.

## What is OpenCV?

Python is the pool of libraries. It has numerous libraries for real-world applications. One such library is OpenCV. OpenCV is a cross-platform library used for Computer Vision. It includes applications like video and image capturing and processing. It is majorly used in image transformation, object detection, face recognition, and many other stunning applications.

## What are we going to build?

At the end of this article, we aim to transform images into its cartoon. Yes, we will CARTOONIFY the images. Thus, we will build a python application that will transform an image into its cartoon using OpenCV.

## Introduction:

Social media is extensively used these days. And standing out in this online crowd has always been a to-do on every user's list on these social media platforms. Be it images, blog posts, artwork, tweets, memes, opinions and what not being used to seek attention of followers or friends to create influence or to connect with them on such social platforms. We aim to provide one such creative solution to their needs, which is applying cartoon like effects to their images. Users can later share these images on any social media platforms, messengers, keep it for themselves, share it with loved ones or do whatever they like with it. Nowadays almost everyone is registered in social networks. We keep online status updated every day, share photos and comments, follow our friends' news. To have a nice profile is a matter of prestige. You can use a photo of your own in a profile image, create an amusing avatar or turn your photo into a cartoon. With a pool of web applications available online, an image conversion to cartoon takes few clicks.

## Literature Review:

| S.NO | Title, Authors and year of paper | Observation |
|---|---|---|
| 1 | Cartooning an Image Using Opencv and Python Vaishali Sudarshan, Amritesh Singh 2020 | Process to create a cartoon effect image can be initially branched into 2 divisions |
| 2 | Technical Paper Presentation on Application of Cartoon like Effects to Actual Images, Chinmay Joshi, Devendra Jaiswal 2019 | This paper represents different techniques of converting image to cartoon. |
| 3 | Cartooning of an Image Pureti Anusha Ch.Sravani Y.Pavankumar T.Venkateswarlu G.Jahnavi A.Hemasri 2020 | This paper presented an efficient method for extracted cartoon objects. |

## THE ALGORITHM:

Process of converting an image to a cartoon

To convert an image to a cartoon, multiple transformations are done.

- Convert the image to a Grayscale image. Yes, like the old day's pictures.!
- The Grayscale image is smoothened
- Extract the edges in the image
- Form a color image and mask it with edges.

## Identifying the Edges:

Finding smooth outline that represents or bounds the shape of the image is an important property to achieve a quality image. All Edge processing tasks are:

**MEDIAN FILTER**

This filter helps in reducing the noise created during the downscaling the image and later converting the original image to cartoon image by applying the bilateral filter. Any extreme specks are smoothened over.

**EDGE DETECTION**

At first the noise of the image is removed within the image. Later the smoothened image is filtered using horizontal and vertical direction by dividing the cells of the picture element (both x and y dimensions.)

**MORPHOLOGICAL OPERATIONS**

This serves the purpose to Bolden and smoothen the outline of the edges variably. The pixels that are highlighted but seems far are removed. Hence the edge lines reduce to thinner outline.

**EDGE FILTERING**

Two divisions of the constituent regions, any region that pertain below a a certain threshold is removed. Small outline identified by the detection method is removed from the final image.

## Colours to the RGB Image:

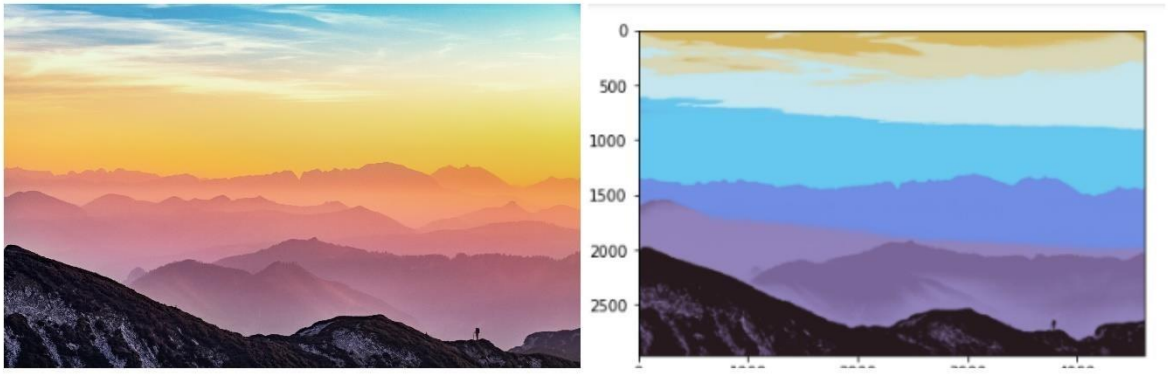
The most important aspect is to eliminate the color regions and apply cartoon effects. Through this algorithm, the colors are smoothened on multiple filtrations so as to create a equal color regions.

## BILATERAL FILTERING

The important role of this filter is to smooth the images without creating any sort of noise also while preserving the edges. Filtering is performed by reading an image from the file and storing it in a matrix object. Initially creating an empty matrix to store the result and applying bilateral filter. This totally depends on the kernel size and testing by running more no of iterations.

## QUANTIZE COLOURS

The last step of the conversion involves the step of reducing the number of colors in each pixel. This creates a beautiful cartoon image with edges and lightened colour of the original image.




## Code & Implementation:

### Importing the required modules

```
import cv2 #for image processing
import easygui #to open the filebox
import numpy as np #to store image
import imageio #to read image stored at particular path
```

```
import sys
import matplotlib.pyplot as plt
import os
import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image
```

### OPENCV:

In the above code we are importing the required modules for our project. This is a module from the OpenCV library, it will be used for the image processing.

### EASY GUI:

EasyGUI is a module for very simple, very easy GUI programming in Python. EasyGUI is different from other GUI generators in that EasyGUI is NOT event driven. Instead, all GUI interactions are invoked by simple function calls. Unlike other complicated GUI's EasyGUI is the simplest GUI till now.

### NUMPY:

NumPy is a module for Python. The name is an acronym for "Numeric Python" or "Numerical Python". NumPy enriches the programming language Python with powerful data structures, implementing multidimensional arrays and matrices.

### IMAGEIO:

Imageio is a Python library that provides an easy interface to read and write a wide range of image data, including animated images, volumetric data, and scientific formats. It is cross-platform.

## SYS:

The python sys module provides functions and variables which are used to manipulate different parts of the Python Runtime Environment. It lets us access system-specific parameters and functions.

**Making the GUI main window**

```
top=tk.Tk()
top.geometry('400x400')
top.title('Cartoonify Your Image !')
top.configure(background='white')
label=Label(top,background='#CDCDCD', font=('calibri',20,'bold'))
```

The Tk () function was called and assigned to the variable named 'top'. In this step, we will build the main window of our application. It will contain buttons, labels, and images. geometry (), title (),
. configure, and Label () are methods from Tkinter used in creating the window.

**Create a File Box to choose a particular file**

```
def upload():
    ImagePath=easygui.fileopenbox()
    cartoonify(ImagePath)
```

The above code opens the file box, i.e the pop-up box to choose the file from the device, which opens every time you run the code. fileopenbox() is the method in easyGUI module which returns the path of the chosen file as a string.

**How is an Image stored?**

```
def cartoonify(ImagePath):
    # read the image
    originalmage = cv2.imread(ImagePath)
    originalmage = cv2.cvtColor(originalmage, cv2.COLOR_BGR2RGB)
    #print(image)  # image is stored in form of numbers
```

In the above code, Imread is a method in cv2 which is used to store images in the form of numbers, it helps us to perform any operations on the image. The image is read as a numpy array, in which cell values depict R, G, and B values of a pixel

```
# confirm that image is chosen
if originalmage is None:
    print("Can not find any image. Choose appropriate file")
    sys.exit()
```

This code checks if the image is chosen or not.

```
ReSized1 = cv2.resize(originalmage, (960, 540))
#plt.imshow(ReSized1, cmap='gray')
```

We resize the image after each transformation to display all the images on a similar scale at last.

## Image transformation to grayscale

```
#converting an image to grayscale
grayScaleImage= cv2.cvtColor(originalmage, cv2.COLOR_BGR2GRAY)
ReSized2 = cv2.resize(grayScaleImage, (960, 540))
#plt.imshow(ReSized2, cmap='gray')
```

cvtColor(image, flag) is a method in cv2 which is used to transform an image into the color-space mentioned as ‘flag’. Here, our first step is to convert the image into grayscale. Thus, we use the BGR2GRAY flag. This returns the image in grayscale. A grayscale image is stored as grayscale Image. **Smoothening a grayscale image**

```
#applying median blur to smoothen an image
smoothGrayScale = cv2.medianBlur(grayScaleImage, 5)
ReSized3 = cv2.resize(smoothGrayScale, (960, 540))
#plt.imshow(ReSized3, cmap='gray')
```

To smoothen an image, we simply apply a blur effect. This is done using median Blur () function. Here, the center pixel is assigned a mean value of all the pixels which fall under the kernel. In turn, creating a blur effect.

## Cartoon effect specialties: Smooth colors

```
#applying bilateral filter to remove noise
#and keep edge sharp as required
colorImage = cv2.bilateralFilter(originalmage, 9, 300, 300)
ReSized5 = cv2.resize(colorImage, (960, 540))
#plt.imshow(ReSized5, cmap='gray')
```

In the above code, we finally work on the second specialty. We prepare a lightened color image that we mask with edges at the end to produce a cartoon image. We use bilateral Filter which removes the noise. It can be taken as smoothening of an image to an extent.

The third parameter is the diameter of the pixel neighborhood, i.e, the number of pixels around a certain pixel which will determine its value. The fourth and Fifth parameter defines sigma Color and sigma Space. These parameters are used to give a sigma effect, i.e., make an image look vicious and like water paint, removing the roughness in colors.

**Giving a Cartoon Effect**

```
#masking edged image with our "BEAUTIFY" image
cartoonImage = cv2.bitwise_and(colorImage, colorImage, mask=getEdge)

ReSized6 = cv2.resize(cartoonImage, (960, 540))
#plt.imshow(ReSized6, cmap='gray')
```

We need to combine the two specialties to get a cartoon image. This will be done using MASKING. We perform bitwise and on two images to mask them.

## Plotting all the transitions together

```
# Plotting the whole transition
images=[ReSized1, ReSized2, ReSized3, ReSized4, ReSized5, ReSized6]

fig, axes = plt.subplots(3,2, figsize=(8,8), subplot_kw={'xticks':[], 'yticks':[]}, gridspec_kw=dict(hspace=0.1, wspace=0.1))
for i, ax in enumerate(axes.flat):
    ax.imshow(images[i], cmap='gray')

save1=Button(top,text="Save cartoon image",command=lambda: save(ReSized6, ImagePath),padx=30,pady=5)
save1.configure(background='#364156', foreground='white',font=('calibri',10,'bold'))
save1.pack(side=TOP,pady=50)

plt.show()
```

To plot all the images, we first make a list of all the images. The list here is named “images” and contains all the resized images. Now, we create axes like subl=plots in a plot and display one-one images in each block on the axis using imshow() method.

plt.show() plots the whole plot at once after we plot on each subplot.

## Making a Save button in the GUI main window

```
def save(ReSized6, ImagePath):
    #saving an image using imwrite()
    newName="cartoonified_Image"
    path1 = os.path.dirname(ImagePath)
    extension=os.path.splitext(ImagePath)[1]
    path = os.path.join(path1, newName+extension)
    cv2.imwrite(path, cv2.cvtColor(ReSized6, cv2.COLOR_RGB2BGR))
    I= "Image saved by name " + newName +" at "+ path
    tk.messagebox.showinfo(title=None, message=I)
```

Here, the idea is to save the resultant image. For this, we take the old path, and just change the tail (name of the old file) to a new name and store the cartoonified image with a new name in the same folder by appending the new name to the head part of the file.

For this, we extract the head part of the file path by os.path.dirname() method. Similarly, os.path.splitext(ImagePath)[1] is used to extract the extension of the file from the path. Here, newName stores “Cartoonified_Image” as the name of a new file. os.path.join(path1, newName + extension) joins the head of path to the newname and extension. This forms the complete path for the new file. imwrite() method of cv2 is used to save the file at the path mentioned. cv2.cvtColor(ReSized6, cv2.COLOR_RGB2BGR) is used to assure that no color get extracted or highlighted while we save our image. Thus, at last, the user is given confirmation that the image is saved with the name and path of the file.

## Making the Cartoonify & Live button in the GUI main window

```
load=Button(top,text="Cartoonify an Image",command=upload,padx=10,pady=5)
live=Button(top,text="Live Cartoonify",command=onlive,padx=10,pady=5)
load.configure(background='#364156', foreground='white',font=('calibri',10,'bold'))
live.configure(background='#364156', foreground='white',font=('calibri',10,'bold'))
live.pack(side=TOP,pady=20)
load.pack(side=TOP,pady=20)
```

The above code creates the buttons to load image and to take a live photo in the main window.

```
def liv():
    cap=cv2.VideoCapture(0)
    while(1):
        ret, frame =cap.read()
        cv2.imshow('the_input_frame',frame)
```

In this code we capture the live feed from our camera using cv2.VideoCaputure(). we save the image in frame by using cap.read() and take a photo from camera and we display the image taken from the camera using cv2.imshow() module.

## CV2.IMSHOW:

cv2.imshow() method is used to display an image in a window. The window automatically fits to the image size.

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

edges = cv2.Canny(frame,100,10)
cv2.imshow('edges',edges)

color = cv2.bilateralFilter(frame, d=7, sigmaColor=6,sigmaSpace=9)
def color_quantization(img, k):
```

From the above code we get gray image and image with edge detection and the bilateral filter is used to reduce the noise in the image and these images are displayed live from your camera.

```
    def color_quantization(img, k):
# Defining input data for clustering
          data = np.float32(img).reshape((-1, 3))
# Defining criteria
          criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 20, 1.0)
# Applying cv2.kmeans function
          ret, label, center = cv2.kmeans(data, k, None, criteria, 10, cv2.KMEANS_RANDOM_CENTERS)
          center = np.uint8(center)
          result = center[label.flatten()]
          result = result.reshape(img.shape)
          return result

    img_1 = color_quantization(frame, 2)
    cartoon = cv2.bitwise_and(color, color, img_1,mask=edges)
    cv2.imshow('cartoon',cartoon)
#     edges=cv2.Canny(frame,100,100)
    k=cv2.waitKey(5) & 0xFF
    if k== 27:
        break
cap.release()
```

In the above code we have done colour quanization using k-means clustering algorithm. We have transformed the image into date using np. float32(img). And we have defined a algorithm to termination criteria i.e the maximum number of iterations or the desired accuracy using criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 20, 1.0).

By applying k-mean clustering algorithm we can make the image with k colours.

We converted the ceter to unit8 using center = np. uint8(center). And replaced the pixel values with their center values.We take the quantized image and turn it into cartoon.

**Main function to build the GUI window**

```
def up():
    top.mainloop()

def default():
    return "choose correct option"

switcher = {
    1: liv,
    2: up
    }

def switch(value):
    return switcher.get(value, default)()
switch(2)
```

In this code we use a switch statement to change the process from upload an image to live camera image.

**Output Screenshots:**

## Conclusion:

We have successfully developed Image Cartoonifier with OpenCV in Python. This is the magic of OpenCV which let us do wonder and it is just a piece of what OpenCV can offer**.**

## References:


1. **https://www.geeksforgeeks.org/cartooning-an-image-usingopencv-python/**
2. **https://towardsdatascience.com/using-opencv-to-catoonize-animage-1211473941b6**
3. **https://www.researchgate.net/publication/333686497_Application_of_Cartoon_Like_Effects_to_Actual_Images**
4. **http://www.jctjournal.com/gallery/13-dec2020.pdf**
5. **https://www.geeksforgeeks.org/cartooning-an-image-usingopencv-python/**