2020/2021

MINI PROJECT

Image Cartoonifier

Major: Electrical Engineering

Prepared By:

Dana Tarhini 5640

Tala Karaki 5582

Supervised by:

Dr. Mohamad Aoude

Table of Contents

List of Figures

General Introduction

Python is an interpreted high-level, general purpose programmimg language. It is the pool of libraries and 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.

At the end of this article, the main goal is to build a python application that will transform an image into its cartoon using OpenCV. In the first chapter, all the needed libraries and functions will be explained. And then, in the second chapter, all the steps that should be followed to achieve the goal will be explained in details.

Chapter 1: Libraries and Functions for Image Processing

1.1. Introduction

There are many libraries used in python for many purposes. Each library contains a lot of functions. That's why, to accomplish any application in python, specific functions must be used, each function belongs to a specific library. So, choosing which libraries to import and which functions to use is an essential part of any coding project. The libraries and functions that are specific for image processing and for creating Graphical User Interface (GUI) for Desktop Applications will be listed and explained in this chapter.

1.2. Libraries

To process and modify an image, many libraries can be used:

1.2.1. Cv2 Library:

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. cv2. OpenCV-Python makes use of Numpy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This library is used for image processing.

import cv2

- cv2.imread(): This method loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix. This function has one parameter which is the path of the file.
- cv2. cvtColor(src,code[,dst[,dstCn): This method is used to convert an image from one color space to another.
  src: It is the image whose color space is to be changed.
  code: It is the color space conversion code.
  dst and dstCn are optional parameters.
- cv2.resize(src, dsize[, fx[, fy[, interpolation]]]): This method is used to resize an image which means changing the dimensions of it. The dimensions can be a width, height, or both. Also, the aspect ratio of the original image could be preserved in the resized image.
  Src: This parameter is required, and it is the source/input image.
  Dsize: This is the required parameter, and it is the desired size for the output image.
  The other parameters are optional.

- medianBlur(src, dst, ksize):This method is used to smoothen an image, we simply apply a blur effect. Here, the center pixel is assigned a mean value of all the pixels which fall under the kernel. In turn, creating a blur effect.

  Src: A matrix object representing the source (input image) for this operation.

  Dst: A matrix object representing the destination (output image) for this operation.

  ksize: A Size object representing the size of the kernel.

- In simple thresholding, the threshold value is global, i.e., it is same for all the pixels in the image.
  Adaptive thresholding is the method where the threshold value is calculate d for smaller regions and therefore, there will be different threshold values for different regions.
  Adaptive threshold operation on an image is done using the method adaptiveThreshold(src, dst, maxValue, adaptiveMethod, thresholdType, blockSize, C)
  Src : An object of the class Mat representing the source (input) image.
  Dst: An object of the class Mat representing the destination (output) image.
  MaxValue: A variable of double type representing the value that is to be given if pixel value is more than the threshold value.
  AdaptiveMethod: A variable of integer the type representing the adaptive method to be used. This will be either of the following two values

  - ADAPTIVE_THRESH_MEAN_C − threshold value is the mean of neighborhood area.
  - ADAPTIVE_THRESH_GAUSSIAN_C − threshold value is the weighted sum of neighborhood values where weights are a Gaussian window.

  ThresholdType: A variable of integer type representing the type of threshold to be used.

  BlockSize: A variable of the integer type representing size of the pixel neighborhood used to calculate the threshold value.

  C: A variable of double type representing the constant used in the both methods (subtracted from the mean or weighted mean).

- cv2.bilateralFilter(src, d, sigmaColor, sigmaSpace): This method is used to remove the noise, it can be taken as smoothening of an image to an extent. A lightened color image is prepared that will be masked with edges at the end to produce a cartoon image.

  d: Diameter of each pixel neighborhood.

  sigmaColor: Value of sigma in the color space. The greater the value, the colors farther to each other will start to get mixed.

  sigmaSpace: Value of sigma in the coordinate space. The greater its value, the more further pixels will mix together, given that their colors lie within the sigmaColor range.

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.

- cv2.bitwise_and(src1,src[,dst[,mask]): It is a function that performs bitwise AND processing as the name suggests. The AND of the values for each pixel of the input images src1 and src2 is the pixel value of the output image.
- cv2.imwrite(filename,image): This method is used to save an image to any storage device. This will save the image according to the specified format in current working directory.
  filename: A string representing the file name. The filename must include image format like .jpg, .png, etc.
  image: It is the image that is to be saved.

1.2.2. EasyGUI Library

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. This library is used to open the filebox. It allows us to select any file from our system.

import easygui

- easygui. fileopenbox(): This method in easyGUI module which returns the path of the chosen file as a string.

1.2.3. NumPy Library

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. It's used to store the image. Images are stored and processed as numbers. These are taken as arrays.

import numpy as np

1.2.4. Imageio Library

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.

import imageio

1.2.5. Sys Library

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.

import sys

- sys.exit(): This method in Python is used to exit the process with specified status without calling cleanup handlers, flushing stdio buffers.

1.2.6. Matplotlib.pyplot Library

Matplotlib. pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

import matplotlib.pyplot as plt

- matplotlib.pyplot.imshow(): The imshow() function in pyplot module of matplotlib library is used to display data as an image; i.e. on a 2D regular raster.
- pyplot. Subplots() method provides a way to plot multiple plots on a single figure. Given the number of rows and columns , it returns a tuple (fig , ax), giving a single figure fig with an array of axes ax .
- plt.show() plots the whole plot at once after plotting on each subplot.

1.2.7. OS Library

The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality. The *os* and *os. path* modules include many functions to interact with the file system.

import os

- os.path.dirname(path): This method is used to get the directory name from the specified path.
  path: A path-like object representing a file system path.
- os.path.splitext(path): This method in Python is used to split the path name into a pair *root* and *ext*. Here, *ext* stands for extension and has the extension portion of the specified path while *root* is everything except *ext* part.
  path: A path-like object representing a file system path. A path-like object is either a str or bytes object representing a path.

- Os.path.join(path, *paths)

path: A path-like object representing a file system path.
*path: A path-like object representing a file system path. It represents the path components to be joined.
A path-like object is either a string or bytes object representing a path.
Note: The special syntax *args (here *paths) in function definitions in python is used to pass a variable number of arguments to a function.

1.2.8. Tkinter Library

Tkinter is a standard library in python used for creating Graphical User Interface (GUI) for Desktop Applications. With the help of Tkinter developing desktop applications is not a tough task. The primary GUI toolkit we will be using is Tk , which is Python's default GUI library.

import tkinter as tk
from tkinter import filedialog
from tkinter import *

- messagebox.Function_Name(title, message [, options])

Function_Name: This parameter is used to represents an appropriate message box function.

title: This parameter is a string which is shown as a title of a message box.

message: This parameter is the string to be displayed as a message on the message box.

options: There are two options that can be used are:

default: This option is used to specify the default button like ABORT, RETRY, or IGNORE in the message box.

parent: This option is used to specify the window on top of which the message box is to be displayed.

Function_Name:
There are functions or methods available in the messagebox widget.

showinfo(): Show some relevant information to the user.

showwarning(): Display the warning to the user.

showerror(): Display the error message to the user.

- geometry():This method is used to set the dimensions of the Tkinter window and is used to set the position of the main window on the user's desktop.
- Configure(): This method lets you change the options of a widget, and the available parameters depend on the widget you're configuring.
- Button (master, option=value, ...)

Master:This represents the parent window.
Options:These options can be used as key-value pairs separated by commas.

- mainloop() tells Python to run the Tkinter event loop. This method listens for events, such as button clicks or keypresses, and blocks any code that comes after it from running until the window it's called on is closed.

1.2.9. PIL Library

Python Imaging Library (PIL) is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats.

from PIL import ImageTk, Image

1.3. Conclusion

These are the information about libraries and functions that are needed in order to write a successful code for the cartoonify python application.

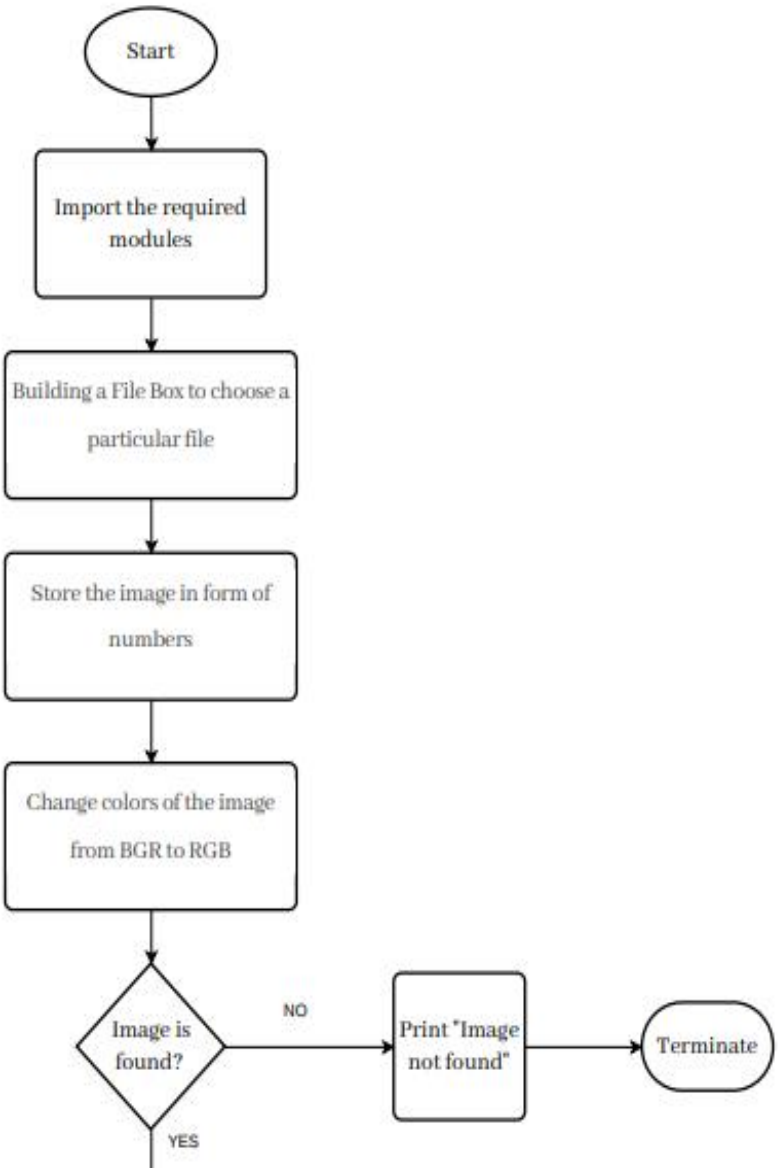
Chapter 2: Cartoonify an Image

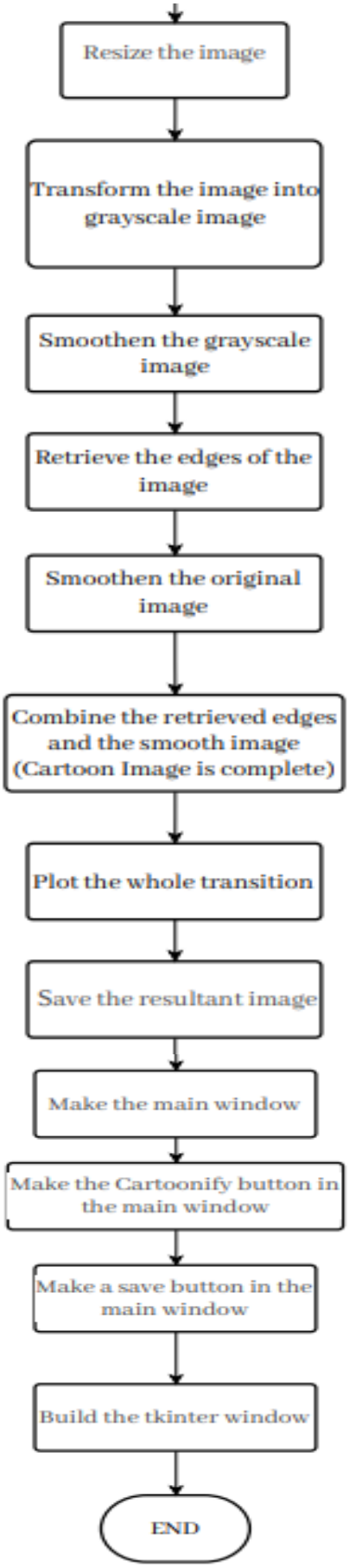
2.1. Introduction

After all the needed libraries and functions are clear. Now it's time to know the steps that should be made in order to transform a normal image into its cartoonified version.

2.2. Steps to cartoonify an image

To create a cartoon image, many steps should be completed. These steps are showed in the flowchart below.




Resize the image
Transform the image into grayscale image
Smoothen the grayscale image
Retrieve the edges of the image
Smoothen the original image
Combine the retrieved edges and the smooth image (Cartoon Image is complete)
Plot the whole transition
Save the resultant image
Make the main window
Make the Cartoonify button in the main window
Make a save button in the main window
Build the tkinter window
END

1. The following modules are imported:

CV2: Imported to use OpenCV for image processing
Easygui: Imported to open a file box. It's used to select any file from the system.
Numpy: Images are stored and processed as numbers. These are taken as arrays. It's used to deal with arrays.
Imageio: Used to read the file which is chosen by file box using a path.
Matplotlib: This library is used for visualization and plotting. Thus, it is imported to form the plot of images.
OS: For OS interaction. Here, to read the path and save images to that path.
Tkinter: imported to create a Graphical User Interface (GUI) for Desktop Applications.

2. Open a file-box to choose the file from the device. The path of the chosen file is returned as a string. The chosen file contains the path of the origin image. The figure below shows the original image.

Figure 1-Original Image

All the below steps are the part of function cartoonify (ImagePath)

3. The image is stored in the form of numbers and it is read as a numpy array, in which cell values depict R(red), G(green) and B(blue) values of a pixel. The colors are stored in this order in Pillow 'PIL' module which is used to open and manipulate the image.
   But the image is read with the OpenCV function `imread()`, where the colors are stored is BGR . So, in order to use both the Pillow and the OpenCV modules, the colors are converted from BGR and RGB using the OpenCV function `cvtColor()` .
   The image is resized ,using the resize() method in cv2 and display it using imshow() method, after each transformation to display all the images on a similar scale at last. This is done to get more clear insights into every single transformation step.

4. The image is transformed to Gray Scale image, like the old days images by using BGR2GRAY flag. This returns the image in grayscale.

Figure 2-Grayscale Image

5. The image is smoothened by applying a blur effect. This is done using medianBlur() 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.

Figure 3-Smoothened Grayscale Image

6. Here, the edges are retrieved and highlited. This is attained by the adaptive thresholding technique. The threshold value is the mean of the neighborhood pixel values area minus the constant C. C is a constant that is subtracted from the mean or weighted sum of the neighborhood pixels. Thresh_binary is the type of threshold applied, and the remaining parameters determine the block size.

Figure 4-Highlighted Edges of the Image

7. A lightened color image is prepared to be masked with edges at the end to produce a cartoon image. We use bilateralFilter which removes the noise. It can be taken as smoothening of an image to an extent.
   It's similar to BEAUTIFY or AI effect in cameras of modern mobile phones.

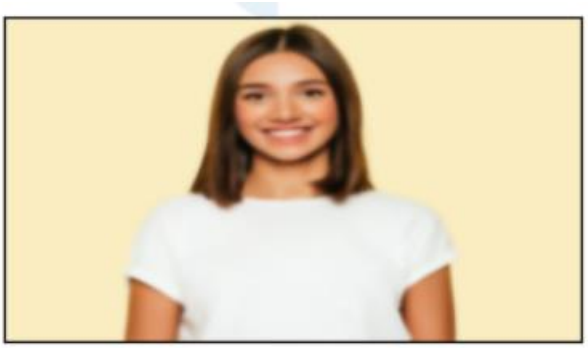


Figure 5-Lightened Color Image

8. The two specialties (Highlighted Edges & Smooth colors) are combined using MASKING. It's done using bitwise_and method on two images to mask them. That's how the edged image is masked on the “BEAUTIFY” image.
   The cartoon image is now complete.

Figure 6-Cartoon Image

9. A list of all the resized images is made. Each one of them is plotted using imshow() method. plt.show() plots the whole plot at once after we plot on each subplot.

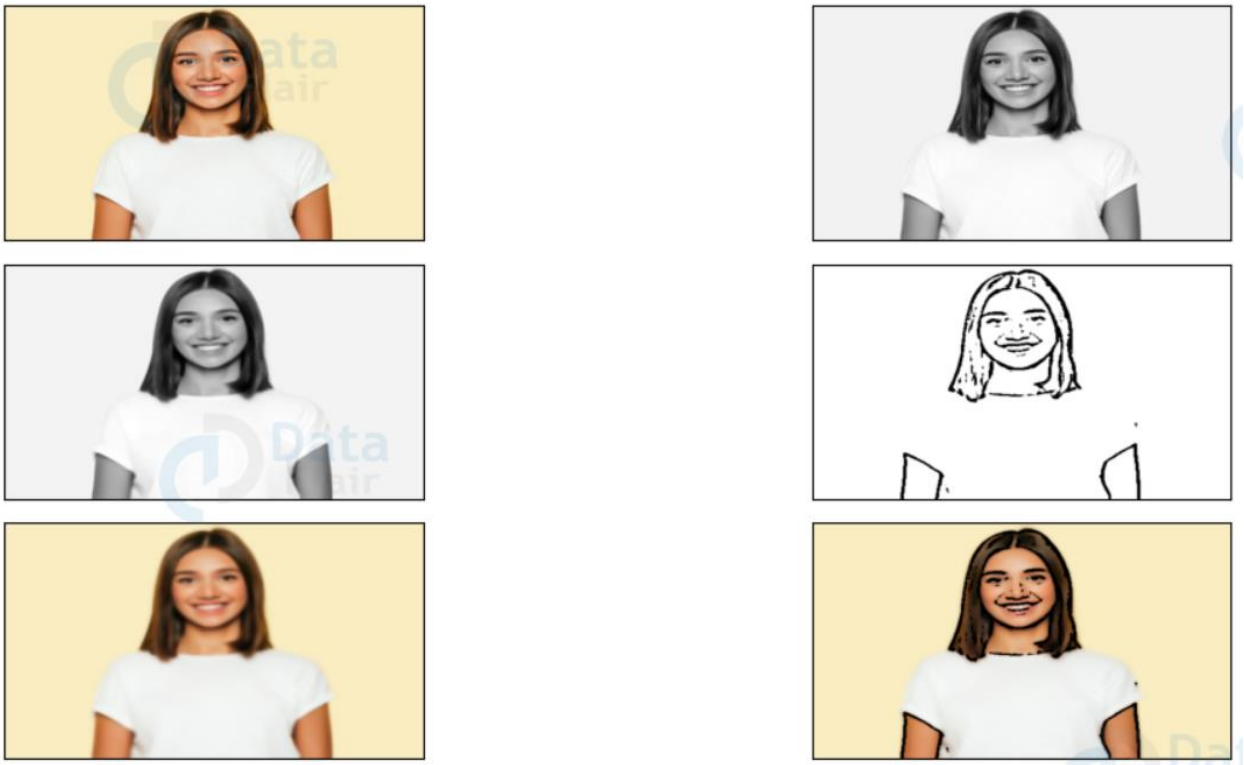


Figure 7-All resized images

10. The resultant image is then saved. For this, the old path is taken, and just the tail is changed (name of the old file) to a new name and the cartoonified image is stored with a new name in the same folder by appending the new name to the head part of the 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 the image.
    At last, the user is given confirmation that the image is saved with the name and path of the file.

11. The main window is made.

12. A button to cartoonify an image, and another to save it are made on the window.

13. Finally, mainloop() tkinter method is used to build the tkinter window.

Result:

An Image Cartoonifier python application is successfully developed with OpenCV in Python. Now, any original image can be transformed into a cartoon image using this application.

2.3. Conclusion

To convert an image to a cartoon, multiple transformations are done. Firstly, an image is converted to a Grayscale image. Then, the Grayscale image is smoothened, and the edges in the image are extracted. Finally, a color image is formed and masked with edges. This creates a beautiful cartoon image with edges and lightened color of the original image.

General Conclusion

An image cartoonifier application is finally made by doing many steps, like grayscale image, smooth image and highlighted edges. These steps couldn’t be done without the use of the particular functions and libraries. We think that this application can put a smile on anyone's face, and it is made only in the purpose of entertainment. But, it is important to point out, that with machine learning in python, many great ideas and applications can be achieved and we can say that ' The sky is the limit '.