**IMAGE TO CARICATURE**

­

Manvi Vrati[1], Pranjal Tulsiyan[2] and Saloni Chawla[3]

Mentor: Dr Praneet Saurabh[4]

Mody University of Science and technology, Lakshmangarh

[1] [manvivrati19.set@modyuniversity.ac.in](mailto:manvivrati19.set@modyuniversity.ac.in)

[2] [pranjaltulsiyan19.set@modyuniversity.ac.in](mailto:pranjaltulsiyan19.set@modyuniversity.ac.in)

[3] [salonisaloni19.set@modyuniversity.ac.in](mailto:salonisaloni19.set@modyuniversity.ac.in)

[4] [praneetsaurabh.set@modyuniversity.ac.in](mailto:praneetsaurabh.set@modyuniversity.ac.in)

***ABSTRACT-***

**Caricature is a comical representation of one’s image which emphasis on picturing the facial features boldly. This application can be useful for people who are interested in comic and webtoon writing to create realistic characters. This can also be used by programmers/ hackers or other people who put their Bitmoji’s as their profile pictures as they don’t wish to reveal their faces for privacy purposes but they wish their DP to be unique. Graphic designers also use caricatures to escape their boring routine. It turns a simple hobby to lifestyle. Now a days caricatures are in trend. The earlier method of transformation requires complicated computer graphics and skills. The idea of this paper is based on designated images which are converted to their caricature form. Amongst all the techniques usable, this application is made using CV2 and Tkinter.**

**In this project, we aim to transform images into cartoon. We will be using Python language on PyCharm. Conversion will be done by first uploading the image in the interface made by using Tkinter and then just by clicking on the Cartoonify Button. Hence, we get the caricature of the image ready to be saved in our devices. We will be using various ML libraries like OpenCV, Numpy, matplotlib, etc. We will put our knowledge of Machine Learning together to create this innovative program.**

**Keywords-** caricature,machine learning, python, image, OpenCV.

1. **INTRODUCTION**

Cartoon images play essential roles in our everyday lives especially in entertainment, education, and advertisement, that become increasingly intensive research in the field of multimedia and computer graphics. The automatic cartoon object extraction is very useful in many applications; one of the most important is the cartoon images retrieval, where the user for cartoon images retrieval system targets to get similar images from the original image. Today, a number of researchers have exploited the concepts related to content-based images retrieval (CBIR) to search for cartoon images containing particular object of interest. Several region-based retrieval methods proposed, for more details see some of the automatic methods, which discriminate the region of interest from the other less useful regions in an image, have been adapted to retrieve cartoon characters; they use partial features for recognizing regions and/or aspects which are suitable for cartoon characterization or gesture recognition. Some efforts go beyond extracting central objects, others used Salient Object Detection (SOD). In this paper, a simple automatic method for objects extraction from cartoon image is proposed; it is based on the assumption that the wanted object is founded within or close to the central part of image.

**Caricature** is a comical representation of one’s image which emphasis on picturing the facial features boldly. It can be viewed as a cartoon like image that appears two dimensional but realistic at the same time. Example of an image shown below (Figure 1.1) [Picture taken from https://www.unsplash.com], converted into its caricature (Figure 1.2) is as follows:



Figure 1.1. Original Image [Reference]



Figure 1.2. Caricature of above image [Figure 1.1.]

Presently, there are several applications that people use in their mobile phones, which can provide the facility of making an image in its cartoon form. But we aim to develop a desktop application that can take input from the already present files in a system and produce caricature images in just one click and very less time. For example, Snapchat is a mobile application that can provide filters for converting our image into caricature while clicking the selfie, but that filter is actually designed in such a way that the 2-D features to be applied are already defined, which are same for everyone. But in our project, we will extract the image edges and then convert it into its caricature, so the features defined boldly will be different in every different image chosen. There are several other systems that already exists, and can be browsed on internet for making caricatures.

Surfing through many websites that convert an image into their cartoonized version, people obtain the websites which provide high quality images as output are the paid versions and the free websites do not provides satisfactory results. Some websites produce low quality images.

In this project, we aim to transform images into cartoon. We will be using Python language on PyCharm. Conversion will be done by first uploading the image in the interface made by using Tkinter and then just by clicking on the Cartoonify Button. Hence, we get the caricature of the image ready to be saved in our devices. We will be using various ML libraries like OpenCV, Numpy, matplotlib, etc. We will put our knowledge of Machine Learning together to create this innovative program.

The paper is organised as follows. Section 2 covers literature survey. Section 3 represents the proposed approach. Section 4 Experimental result and analysis with comparison to existing systems while Section 5 concludes this paper.

1. **LITERATURE SURVEY**

A cartoon is a type of illustration that is typically drawn, sometimes animated, in an unrealistic or semi-realistic style. The specific meaning has evolved over time, but the modern usage usually refers to either: an image or series of images intended for satire, caricature, or humour; or a motion picture that relies on a sequence of illustrations for its animation. Someone who creates cartoons in the first sense is called a cartoonist, and in the second sense they are usually called an animator.

Image Translation Numerous methods based on GANs (Goodfellow et al. 2014) have been recently developed for image translation for paired (Isola et al. 2017; Chang et al. (2018), unpaired (Zhu et al. 2017a; Kim et al. 2017; Liu et al. (2017), and multimodal (Zhu et al. 2017b; Lee et al. 2018; (Huang et al. 2018) settings.

While most GANs based methods require large image sets for training, neural style transfer models (Gatys et al. 2015, 2016) only need a single style image as the reference. A number of methods have since been developed (Johnson et al. 2016; Li et al. 2017a; Chenet al. 2017; Liao et al. 2017; Li et al. 2017c) to improve style translation or runtime performance. In the photo-to-caricature task, however, neither GAN-based approaches or neural style transfer methods take the large shape discrepancy across domains into account. Unlike existing image translation methods, our algorithm enables shape exaggerations by utilizing an encoder–decoder architecture with the guidance of a face parsing map to densely model the shape transformation.

Caricature Rendering Learning to caricature from photos is mainly concerned with modelling shape transformation. However, it has not been widely explored due to the large domain gap between photos and caricatures. Some early methods (Akleman 1997; Akleman et al. 2000) rely on userdefined source and target shapes to compute the warping parameters, but the rendered images do not exhibit caricature styles. Several rule-based methods (Luo et al. 2002; Liaoet al. 2004; Brennan 2007) are developed to perform shape exaggeration for each facial component. Recently, CycleGAN-based image translation methods (Liet al. 2020; Zheng et al. 2019) have been proposed for caricature generation with facial landmarks as conditional constraints.

However, the geometric structures of the images generated with these algorithms are still close to the original photos, with limited exaggerated effects. To increase shape exaggeration, Cao et al. (2018) use a geometric exaggeration model by leveraging landmark positions to predict key points in the subspace formed by principal components. The WarpGAN model (Shi et al. 2019) learns to directly predict a set of control points used to warp the input photo based on unpaired adversarial learning.

Despite showing promising results, their shape exaggerations are still limited due to the use of sparse landmarks or control points. In contrast, we use a pixel-wise parsing map and model shape transformation, thereby generating plausible exaggerations while retaining the facial traits of the input image. Furthermore, due to the learned representations in the parsing space, users can explicitly manipulate the facial map to generate preferred shapes. In addition to 2D image-based methods, several 3D geometry-based approaches (Wu et al. 2018; Han et al.2018; Chen et al. 2020) have been developed to manipulate caricature images. Wu et al. (2018) introduce an intrinsic deformation representation which reconstructs a 3D caricature image from sparse 2D landmarks.

On the other hand, Han et al. (2018) utilize user-defined line sketches to edit a 3D caricature model for synthesizing caricature images. However, it is complex for users to define appropriate and exaggerated sketches. In addition, the generated images do not contain caricature textures.

To deform an artist-drawn caricature according to a given normal face expression, Chen et al. (2020) develop a method based on Wu et al. (2018) and Cao et al. (2018) to generate caricatures, which are then further re-rendered with dynamic texture generated from a conditional generative adversarial network. Spatial Transformer Network The spatial transformer networks (STNs) (Jaderberg et al. 2015) are developed to improve object recognition performance by reducing input geometric variations. Close to our work is the method proposed by Lin et al. (2018) in which low-dimensional warping parameters are learned to manipulate foreground objects for image composition.

In our task, we also introduce an STN to predict warping parameters to enable shape exaggeration on normal photos. In contrast, we need denser and more complex deformations instead of lowdimensional affine transformation (Jaderberg et al. 2015) or homography transformation (Linetal. 2018).

* 1. **Objective Of Research**
* The higher literature review reveals that there are varied gaps within the study of converting image to a cartoon image.
* An obvious disadvantage of smoothing is the fact that it does not only smooth noise, but also blurs important features such as edges and, thus, makes them harder to identify.
* Linear diffusion filtering dislocates edges when moving from finer to coarser scales.
* To implement multiple number of bilateral filters.
* To apply multiple number of values to the existing parameters
* There are other systems that use neural networks or GAN for making caricatures but many of them do not produce images that are close to reality.

There are several sites available on the internet, from where we can obtain either caricature images or cartoon images but non of the method produces output that is very close to the real life image provided. Our system will provide such an output which is the cartoonified version of the image provided and also close to the reality, so that it does not give a complete 2D feeling.

* 1. **Existing Systems**

We surfed through many websites that convert an image into their cartoonized version. The websites which provide high quality images as output are the paid versions and the free websites do not provide us with satisfactory results. The outputs of some such websites are given below. Some websites produce low quality images. So, we need a better method.

* ***Photocartoon.net***
* Watermark is present in the output image
* It has high contrast.
* The output image is much brighter than the original image.
* Quality of the image also decreases.
* ***Edit.cartoonize.net***
* Image is smoothened to a lesser extent.
* The cartoonified version is not up to the mark.
* Quality of the image also decreases.
* ***Befunky.com***
* Instead of the cartoonified output it is giving output as if it is an acrylic painting.
* Watermark is also present.
* ***Our Project- Output Image***
* The quality of the output image is not affected.
* There are no predefined features as it extracts edges according to the image.
* Watermark is not present.
  1. **Outputs Of Existing Models**

|  |
| --- |
| Figure 2.1. Original Image |
| Figure 2.1. **photocartoon.net image** |
| Figure 2.3. **Edit.cartoonize.net Image** |
| Figure 2.4. **Befunky.com Image** |
| Figure 4.5. **Image Output from ML Project** |

1. **PROPOSED WORK**

This project is designed for fun purpose. We can cartoonify image of people, animals, flowers or other objects, given that the image is already present in our device, on just one click. The application runs by extracting edges and smoothening the image to boldly express its unique features and creates a caricature which seems to be two dimensional but actually looks very real. This application uses machine learning libraries and creates a beautiful cartoon art of the image given in input. It is done by plotting points of images on matplotlib and also adding brighter colours to the final image.

* 1. **The Algorithm**

The process to create a cartoon effect image can be initially branched into 2 divisions.To detect, blur and bold the edges of the actual RGB colour image. Identifying the edges and then to smooth, quantize and the conversion of the RGB image to grayscale. The results involved in combining the image and help achieve the desired result.

Figure 3.1 System Design

* 1. **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 certain threshold is removed. Small outline identified by the detection method is removed from the final image.

* 1. **COLOURS TO THE RGB IMAGE**

The most important aspect is to eliminate the colour regions and apply cartoon effects. Through this algorithm, the colours are smoothened on multiple filtrations so as to create an equal colour region.

***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.

BF[I]p =

Here, is Normalisation Factor, is Space Weight and is Range weight

* 1. **RECOMBINE IMAGES**

The final task is to overlay the edges onto the colour image is when both the colour and edge image processing are complete.

1. **EXPERIMENTATION**

The figure below depicts the flow of data in the proposed system. How we have imported the modules in our code file, then started creating a window and used these libraries to cartoonify image. Following are the steps:

* 1. **IMPORTING MODULES**

Following are the libraries and modules used in our project for creating a caricature image of a given image:

* + 1. **Module cv2**

OpenCV is a python open-source library used for image processing, ML and computer vision. cv2.imread(path, flag) is used to load image from specified file where the parameter path is the path of image to be read and flag is the way to read the image. Flags are also of 3 types:

* cv2.imread\_COLOR
* cv2.imread\_UNCHANGED
* cv2.imread\_GRAYSCALE
  + 1. **Module easygui**

Easygui is a module for very simple and easy GUI programming. It provides a user-friendly interface for simple GUI interaction. In this programmer does not need to know anything about framer, tkinter, widgets or lambda. EasyGUI is different from other GUI generators. It is not event-driven. All GUI interactions are invoked by simple function calls. This is used here to open a file box. It allows us to select any file from our system.

* + 1. **Module numpy**

NumPy is a fundamental package and a python library that provides a multidimensional array object, derived objects like marked array and matrices. It is also used for operations on arrays, mathematical logical shape, shape manipulation, selecting, sorting. It provides N- dimensional array, sophisticated functions, tools for integrating C/C++, useful linear algebra.

* + 1. **Module imageio**

Imageio is a library used in python programming with easy interfacing to read and write a wide range of image data like

animated images, volumetric data and scientific formats.

* For reading an image imageio.imread() method is used.
* For reading gif file imageio.get\_reader() method is used.
* For creating image file imageio.imwrite() method is used.
  + 1. **Module sys**

In python programming sys module is used to manipulate different parts of python runtime environment by various functions and variables. With this we can perform operations on interpreter as it gives access to those variables and functions that interact strongly with the interpreter. Also, it provides variables for controllinginput/output in a better way. This is done using 3 variables:

* stdin- used to get input from cmd directly and also automatically adds '\n' in the end of every sentence
* stdout- used to display output in any form directly in screen console
* stderr- sys.stderr is written whenever an exception occurs in Python
  + 1. **Module matplotlib**

matplotlib.pyplot is a plotting library used to interface to matplotlib. It provides a way of plotting just like MATLAB. Also, it opens figures on the screen and acts as figure GUI manager. pyplot is majorly used for interactive plots like Line plot, Histogram, Polar, 3D plots, etc. and simple cases of programmatic plot generation. Matplotlib is generally used for user interface toolkits like Tkinter, awxPython etc.

* + 1. **Module os**

As the name suggests OS module in python is used for interacting with operating system through functions. OS is one of the python's standard utility modules which provides a portable way of using OS dependent functionality. Its functions are:

* os.getcwd- confirms returns the presently working directory.
* os.mkdir- to create a new directory.
* os.chfir- to change current working directory to a newly created one.
* os.rmdir()- to remove any specified directory
* os.listdir()- to list all files and directories in the specified directory
  + 1. **Module tkinter**

Tkinter is standard graphical user interface (GUI) library for python. It provides various controls that are used to build GUI applications. Some of them are:

* Button- In python application button is used to add different kinds of button.
* Canvas- This widget is used to draw the canvas on window
* Entry- This is used to display single line text field.
* Label- It is a text field used to display information about other widgets.
* Menu- used to add menu items

Tkinter modules we used here:

* Messagebox widgets: used to display messageboxes for applications in python.
* Filedialogbox: It is intended to provide unique dialogues to be used for dealing files.

There are several predefined methods present in tkinter to make windows.

* + 1. **Module PIL**

PIL stands for python imaging library which is a free and open-source library in Python intended to support for opening, controlling and saving various image file formats. It provides powerful image processing capability and efficient internal representation. It is designed for fast access to data stored in a basic pixel format. It provides a strong foundation for any image processing tool.

* 1. **BUILDING A FILE BOX TO CHOOSE A PARTICULAR FILE**

In this step we have written the function ‘upload\_image’ to open the file box, i.e., the pop-up box to choose the file from the device, which opens every time we run the code. fileopenbox() is the method in easyGUI module which returns the path of the chosen file as a string.

* 1. **READING AND RESIZING IMAGE**

In caricature function 1st we are reading and resizing the image. imread is a method in cv2 which is used to store images in the form of numbers. The image is read as a numpy array, in which cell values depict R, G, and B values of a pixel. cvtColor(image, flag) is a method in cv2 which is used to transform an image into the colour-space mentioned as ‘flag’. We will resize the image after each transformation to display all the images on a similar scale at last.

* 1. **TRANSFORMING AN IMAGE TO GRAYSCALE AND SMOOTHENING IT**

To convert an image to a cartoon, several transformations are done. The image is converted to a Grayscale image. Then, this Grayscale image is smoothened. After this we try to extract the edges in the image. Finally, we form a color image and mask it with edges. This creates a beautiful cartoon image with edges and lightened color of the original image. Here, to convert the image in grayscale, we have used the BGR2GRAY flag. This returns the image in grayscale. To smoothen an image, we simply apply a blur effect. This is done using medianBlur() function. The center pixel is assigned a mean value of all the pixels. This in turn, creates a blur effect.

* 1. **RETREIVING THE EDGES OF AN IMAGE AND TO PREPARE A MASK IMAGE**

Caricature has 2 specialities: its highlighted edges and smooth colours. So, we need to bring out the edges and make them bold so that they highlight even more. For this we take mean of neighbourhood pixel values minus the constant. The type of threshold applied is THRESH\_BINARY. For masking the image, the bilateral filter is used that removes the noise from image. This is smoothening of an image. Similar to beautify image option we have in mobile cameras.



Figure 4.1. Masked image of a landscape

* 1. **GIVING A CARTOON EFFECT**

We need to combine the beautified image and the bold retrieved edges together to obtain a caricature image. Bitwise AND operation is performed on two images and this is called Masking. Plotting the transitions together using matplotlib, wefirst create axes and display images in each block.

* 1. **FUNCTIONALITY OF SAVE BUTTON**

To save the image we have created, we take the old image path to save image in the same folder and have predefined the name: “cartoonified\_image”. For taking the image path using the library OS.

* 1. **MAKING THE MAIN WINDOW BY TKINTER**

Making the window using tkinter is upon the user, it can be done according to our preferences. At the end of the file we need to write root.mainloop() so that the tkinter window appears as soon as the main program is executed.

* 1. **MAKING THE CARTOONIFY AND SAVE BUTTON IN THE MAIN WINDOW**

Creating the save function and caricature functions in the main program.

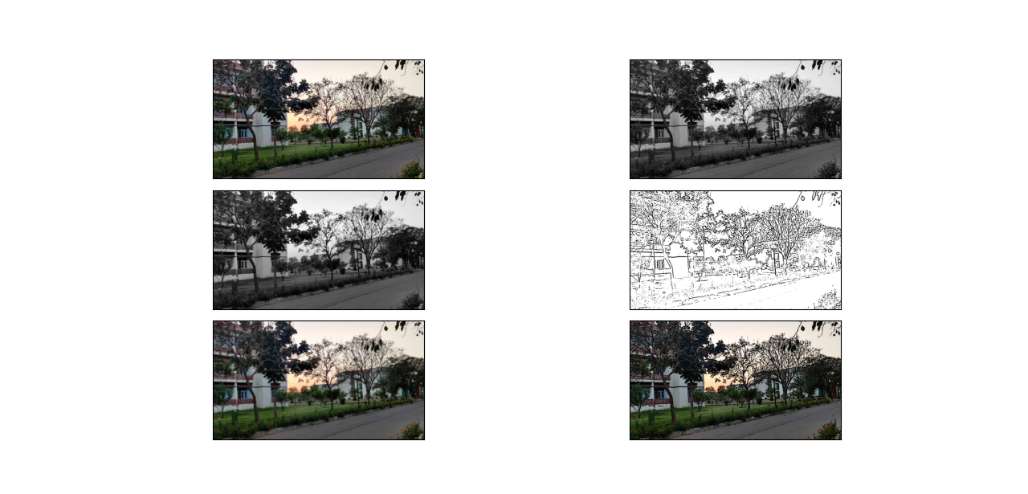


Figure 4.2. Recombined result at all stages of a landscape

* 1. **READING OF IMAGES AT VARIOUS STEPS BY MATPLOTLIB**
     1. **Reading An Image**

In Matplotlib, plotting of graph is performed using the imshow() function. Here we have grabbed the plot object.



Figure 4.3. Reading image on matplotlib graph

* + 1. **Transforming An Image To Grayscale**

Grayscale image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information where pixel value varies from 0 to 255.



Figure 4.4. How a grayscale image looks

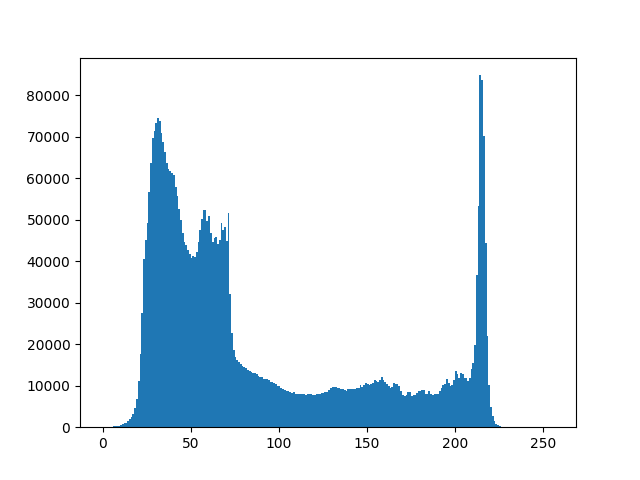


Figure 4.5. From the histogram, we can conclude that dark region is more than brighter region

* 1. **MATHEMATICAL READING OF IMAGE**

The image below is read mathematically in terms of arrays as:

|  |  |
| --- | --- |
| **INPUT IMAGE** | **IMAGE READ AS ARRAY** |
| Figure 4.6. Landscape Image | [[[ 73 50 66] [ 68 45 61] [ 65 43 56]  ...  [220 217 212] [220 217 212] [220 217 212]]  [[ 89 66 82] [ 89 67 80] [100 78 91]  ...  [220 217 212] [220 217 212] [220 217 212]]  [[ 79 59 71] [ 84 64 75] [ 78 58 69]  ...  [220 217 212] [220 217 212] [220 217 212]]  ...  [[ 38 45 37] [ 46 54 43] [ 58 66 55]  ...  [ 12 12 0] [ 14 21 5] [ 20 31 17]]  [[ 17 27 16] [ 21 31 20] [ 32 42 31]  ...  [ 25 25 0] [ 15 23 0] [ 17 30 10]]  [[ 8 18 7] [ 1 11 0] [ 1 12 0]  ...  [ 30 31 0] [ 14 24 0] [ 10 24 1]]] |

1. **CONCLUSION**

First of all, the basic tools to handle the titled problems of the thesis are incorporated. It includes origin and history of image processing, different types of uncertain environment, existing methods for cartoon imaging. Amid the previous three decades, the topic of image processing has gained vital name and recognition among researchers because of their frequent look in varied and widespread applications within the field of various branches of science and engineering. As an example, image processing is helpful to issues in signature recognition, digital video processing, Remote Sensing and finance. Conclusion and Future Directions. Firstly, we use high-resolution camera to take pictures. Secondly, we use OpenCV image processing functions to implement image pre-processing. Thirdly we use morphological opening and closing operations to segment image because of their blur image edges. The main attraction of the paper is to solve different types of images having one object, two object and three object which can’t be solved by any of the exiting methods but can be solved by our proposed method.

1. **REFRENCES**
2. Isabelle Belato, *Caricature*, Sept 13, 2021. Accessed on: Oct 8, 2021. [Online].Available: <https://en.wikipedia.org/wiki/Caricature>
3. “*Cartooning an Image using OpenCV-Python*,” Dec 22, 2020. Accessed on: Sept 25, 2021. [Online].

Available: <https://www.geeksforgeeks.org/cartooning-an-image-using-opencv-python/>

1. Tazki Anida Asrul, *Turn Photos into Cartoons Using Python*, Jan 2, 2021. Accessed on: Sept 14, 2021. [Online].

Available: <https://towardsdatascience.com/turn-photos-into-cartoons-using-python-bb1a9f578a7e>

1. Nikita Jain, “*Understanding cartoon emotion using integrated deep neural network on large dataset*”, April 21, 2021. Accessed on: Nov 19, 2021. [Online].

Available: <https://link.springer.com/article/10.1007/s00521-021-06003-9>

1. “*Python | Bilateral Filtering*” Oct 31, 2021. Accessed on: November 23, 2021. [Online].

Available:<https://www.geeksforgeeks.org/python-bilateral-filtering/>

1. Aljaž Jeromel & Borut Žalik, “*An efficient lossy cartoon image*”, August 29, 2019. Accessed on: Nov 24, 2021. [Online].

Available: <https://link.springer.com/article/10.1007/s11042-019-08126-7>

1. Wenqing Chu, “*Learning to caricate via semantic shape transform*”, July 9, 2021. Accessed on: Nov 20, 2021. [Online]. Available: <https://link.springer.com/article/10.1007/s11263-021-01489-1>
2. Vaishali Sundarshan, Amritest Singh, “*Cartoonifying an image using OpenCV and Python”*, January 24, 2021, Accessed on: Nov 19, 2021. [Online].

Available: <https://www.academia.edu/44964756/Cartooning_an_Image_Using_Opencv_and_Python>

1. Mohommad Razi Ul Haq, “*What is OpenCV in Python*”, March 05, 2021. Accessed on: November 20, 2021. [Online].

Available:[What is OpenCV in Python? (educative.io)](https://www.educative.io/edpresso/what-is-opencv-in-python?utm_term=&utm_campaign=%5BTest%5D+Dynamic+Verticals&utm_source=adwords&utm_medium=ppc&hsa_acc=5451446008&hsa_cam=14045073269&hsa_grp=128822123241&hsa_ad=535845844735&hsa_src=g&hsa_tgt=dsa-1394252596758&hsa_kw=&hsa_mt=&hsa_net=adwords&hsa_ver=3&gclid=Cj0KCQiAy4eNBhCaARIsAFDVtI2F1xzLOEc2HCQ_IA7oWwlreyGx3uG9MNmc2mJRRTx64VkEZxW9-_YaAvcUEALw_wcB)

1. Kellie McBride, “*Cartoon Vs. Caricature*”, February 28, 2006. Accessed on: November 21, 2021. [Online]. Available:<https://www.poynter.org/reporting-editing/2006/cartoon-vs-caricature/>
2. “*Pyplot in Matplotlib*”, Accessed on: November 20, 2021. [Online].

Available: [Pyplot in Matplotlib - GeeksforGeeks (ampproject.org)](https://www-geeksforgeeks-org.cdn.ampproject.org/v/s/www.geeksforgeeks.org/pyplot-in-matplotlib/amp/?amp_js_v=a6&amp_gsa=1&usqp=mq331AQKKAFQArABIIACAw%3D%3D#aoh=16380218424052&referrer=https%3A%2F%2Fwww.google.com&amp_tf=From%20%251%24s&ampshare=https%3A%2F%2Fwww.geeksforgeeks.org%2Fpyplot-in-matplotlib%2F)

1. Image references, Accessed on: November 15, 2021. [Online].

Available: [500+ Wild Animal Pictures [HD] | Download Free Images on Unsplash](https://unsplash.com/s/photos/wild-animal?modal=%7B%22tag%22%3A%22Filters%22%7D)

1. About OpenCV, March 06, 2019. Accessed on: November 15, 2021. [Online].

Available: <https://opencv.org/about/#:~:text=OpenCV%20(Open%20Source%20Computer%20Vision,perception%20in%20the%20commercial%20products>.