REMOJIFY :
CONVERT IMAGE TO CARTOON
USING MACHINE LEARNING CONCEPTS

A PROJECT REPORT

Submitted By-

SACHIN VERMA (CSJMA18001390038)

DEEPAK VERMA (CSJMA18001390018)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY INSTITUTE OF ENGINEERING AND TECHNOLOGY

C.S.J.M. UNIVERSITY, KANPUR

May 2022

University Institute of Engineering and Technology
C. S. J. M. University, Kanpur

BONAFIDE CERTIFICATE

Certified that this project report **"REMOJIFY : CONVERT IMAGE TO CARTOON USING MACHINE LEARNING CONCEPTS"**

is the bonafide work of " **SACHIN VERMA AND DEEPAK VERMA"** who carried out the project work under my supervision.

SIGNATURE

Dr. Sandesh Gupta

SUPERVISOR

Lecturer
Computer Science and Engineering
UIET ,CSJM University kanpur

ABSTRACT

The use of cartoon characters has become a popular trend among the youth worldwide. This can be seen from their profile pictures on online social platforms to trending memes.

The process of converting real-life high-quality pictures into practical cartoon scenes is known as **cartoonization**.In the field of the research processing of an image consisting of identifying an object in an image, identify the dimensions, no of objects, changing the images to blur effect and such effects are highly appreciated in this modern era of media and communication.

There are multiple properties in the Image Processing. Each of the property estimates the image to be produced more with essence and sharper image.

Each Image is examined to various grid. Each picture element together is viewed as a 2-D Matrix. With each of the cell store different pixel values corresponding to each of the picture element.

ACKNOWLEDGEMENT

We take the opportunity to thanks to our enthusiastic supervisor of this project **Dr sandesh Gupta** sir for his valuable guidance and advice. He inspired us greatly to work in this project. His willingness to motivate us contributed tremendously to our project .

We are highly indebted for his guidance and constant supervision as well as providing necessary information regarding the project and also for his support in completing the project.

We would like to take this opportunity to thanks to the **Computer Science and Engineering Department of U.I.E.T** for offering this project. It gave us the opportunity to participate and learn about machine learning and Opencv.

We would like to thanks CSE department for providing us such a good environment and facilities to complete our project.

SACHIN VERMA (CSJMA18001390038)

DEEPAK VERMA (CSJMA18001390018)

TABLE OF CONTENTS

INTRODUCTION

Cartoon is a popular art form that has been widely applied in diverse scenes. Modern cartoon animation workflows allow artists to use a variety of sources to create content.

Some famous products have been created by turning real-world photography into usable cartoon scene materials, where the process is called image cartoonization.

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.

Cartoons are commonly used in various kinds of applications. As we know cartoons are artistically made it requires elegant and fine human artistic skills. While portraying Cartoons in any animated movies it gets time-consuming for the artist as they need to define the sketch of the cartoon properly to get a good result. We all know that animation plays an important role in the world of cinema, so to overcome the problem faced by the artist we have created a program with the help of GAN which not only converts images but also converts video into an animation.

We have given 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.

What is Open-CV?

Computer Vision :

Computer vision is a process by which we can understand the images and videos how they are stored and how we can manipulate and retrieve data from them.
Computer Vision is the base or mostly used for Artificial Intelligence. Computer-Vision is playing a major role in self-driving cars, robotics as well as in photo correction apps.

Python is the pool of libraries. It has numerous libraries for real-world applications. One such library is **Open-CV**. Open-CV 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.
Applications of Open-CV: There are lots of applications which are solved using Open-CV, some of them are listed below

- face recognition
- Automated inspection and surveillance
- number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channels advertisement recognition

THE_ALGORITHM

The process to create a **cartoon effect** image can be initially branched into
2 divisions

1: To detect, blur and bold the edges of the actual RGB color image .

2: To smooth, quantize and the conversion of the RGB image to greyscale.

The results involved in combining the image and help achieve the desired result.

1: IDENTIFYING THE EDGES

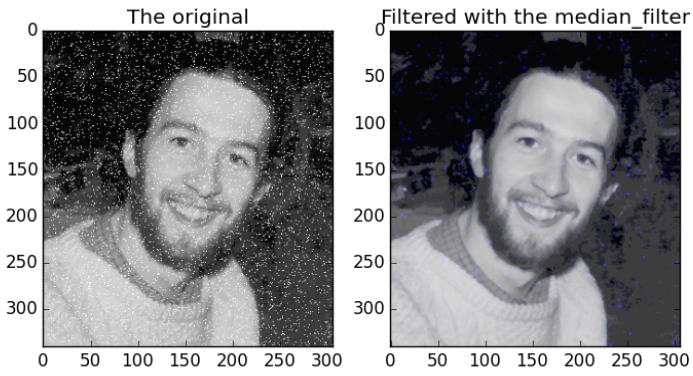
Edge detection is an image-processing technique, which is used to identify the boundaries (edges) of objects, or regions within an image. Edges are among the most important features associated with images. We come to know of the underlying structure of an image through its edges.

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**:

1: MEDIAN FILTERING .

The **median filter** is a non-linear digital filtering technique, often used to remove noise from an image or signal. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image). Median filtering is very widely used in digital image processing because, under certain conditions, it preserves edges while removing noise .

EXAMPLE:



2: EDGE DETECTION :

Edge detection is an image-processing technique, which is used to identify the boundaries (edges) of objects, or regions within an image. Edges are among the most important features associated with images. We come to know of the underlying structure of an image through its edges. Computer vision processing pipelines therefore extensively use edge detection in applications.

2.1: CANNY EDGE DETECTION.

Canny Edge Detection is one of the most popular edge-detection methods in use today because it is so robust and flexible. The algorithm itself follows a three-stage process for extracting edges from an image. Add to it image blurring, a necessary preprocessing step to reduce noise.

This makes it a four-stage process, which includes:

1: Noise Reduction

Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian filter.

2: Finding Intensity Gradient of the Image

Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (Gx) and vertical direction (Gy). From these two images, we can find edge gradient and direction for each pixel as follows:

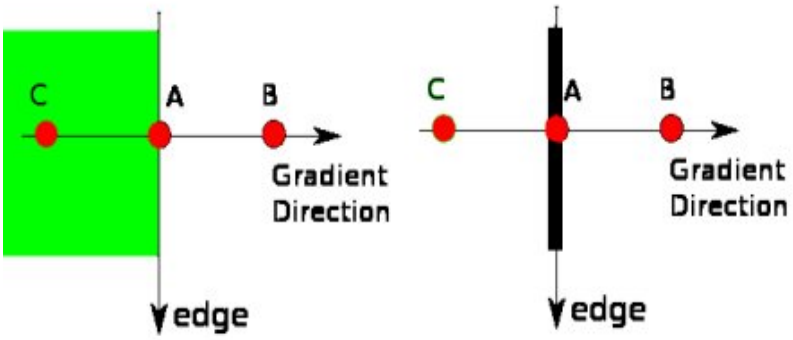
$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

3: Non-maximum Suppression

After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighbourhood in the direction of gradient. Check the image below:

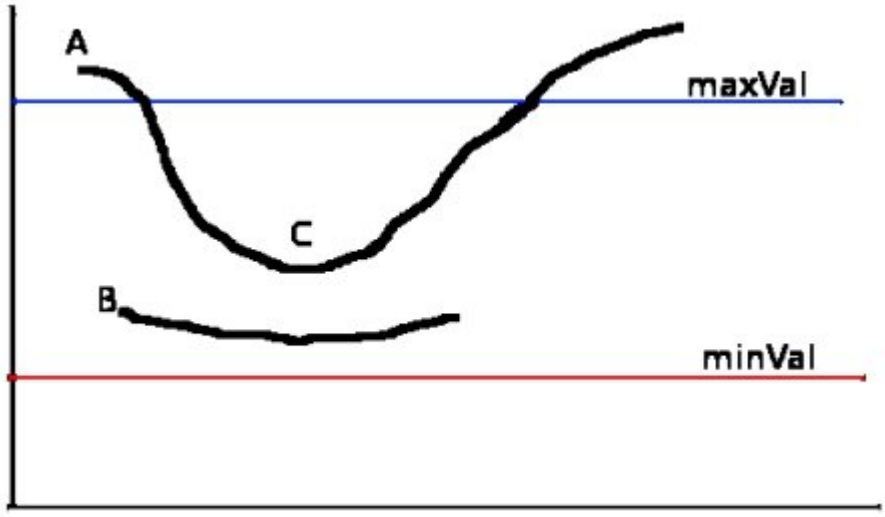


Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed (put to zero).

In short, the result you get is a binary image with "thin edges".

4: Hysteresis Thresholding

This stage decides which are all edges are really edges and which are not. For this, we need two threshold values, min Val and max Val. Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded. See the image below:



The edge A is above the maxVal, so considered as "sure-edge". Although edge C is below maxVal, it is connected to edge A, so that also considered as valid edge and we get that full curve. But edge B, although it is above minVal and is in same region as that of edge C, it is not connected to any "sure-edge", so that is discarded. So it is very important that we have to select minVal and maxVal accordingly to get the correct result.

2.2: CONTOURS .

A contour is a closed curve joining all the points with either the same intensity or color. They help in visualizing the shape of objects or figures in an image.

Normally to use this filter we first need to use either **canny or some thresholding**.

Using **contour detection**, we can detect the borders of objects, and localize them easily in an image.

It is often the first step for many interesting applications, such as image-foreground extraction, simple-image segmentation, detection and recognition.

EXAMPLE: Comparative image, input image and output with **contours overlaid**.

Application of Contours in Computer Vision

Motion Detection*:* In surveillance video, motion detection technology has numerous applications, ranging from indoor and outdoor security environments, traffic control, behaviour detection during sports activities, detection of unattended objects, and even compression of video.

Unattended object detection: Any unattended object in public places is generally considered as a suspicious object

3: 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.

- In short: A set of operations that process images based on shapes. Morphological operations apply a ***structuring element*** to an input image and generate an output image.

- The most basic morphological operations are: Erosion and Dilation. They have a wide array of uses, i.e. :

- Removing noise
- Isolation of individual elements and joining disparate elements in an image.
- Finding of intensity bumps or holes in an image

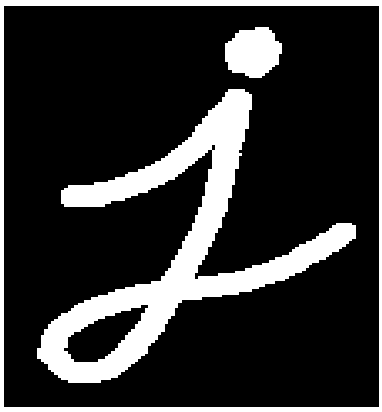
1: EROSION.

Erosion is a **morphological process** The basic idea of erosion is just like soil erosion only, it erodes away the boundaries of foreground object (Always try to keep foreground in white).

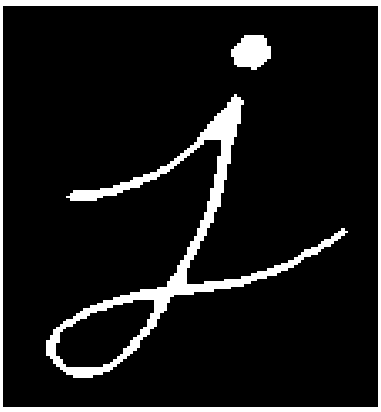
So what it does? The kernel slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero).
So what happens is that, all the pixels near boundary will be discarded depending upon the size of kernel.
So the thickness or size of the foreground object decreases or simply white region decreases in the image. It is useful for removing small white noises (as we have seen in color space chapter), detach two connected objects etc.
In simple terms, they are used to thicken or lessen boundary shapes in an image. We will use it to erosion to thicken the contour boundaries to make them stand out.

2: DILATION

It is just opposite of erosion. Here, a pixel element is '1' if at least one pixel under the kernel is '1'. So it increases the white region in the image or size of foreground object increases. Normally, in cases like noise removal, erosion is followed by dilation.
Because, erosion removes white noises, but it also shrinks our object. So we dilate it. Since noise is gone, they won't come back, but our object area increases. It is also useful in joining broken parts of an object.
EXAMPLE:



Dilation



Erosion

2: Colors 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.

1: BILATERAL FILTERING.

A **bilateral filter** is non-linear, edge-preserving and noise-reducing smoothing filter. The intensity value at each pixel in an image is replaced by a weighted average of
intensity values from nearby pixels
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.

EXAMPLE :

Original Image

Bilateral Filter

2: QUANTIZE COLORS .

The last step of the conversion involves the step of color quantization .

Color quantization is **the process of reducing the number of distinct colors in an image**. Normally, the intent is to preserve the color appearance of the image as much as possible, while reducing the number of colors, whether for memory limitations or compression.

To do color quantization, we apply the K-Means clustering algorithm .

2.1: K-Means Clustering Algorithm.

k-Means is an unsupervised algorithm from the machine learning approach that divides the unlabelled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

The unsupervised k-means algorithm has a loose relationship to the k-nearest neighbour classifier, a popular supervised machine learning technique for classification that is often confused with *k*-means due to the name. Applying the 1-nearest neighbour classifier to the cluster centres obtained by *k*-means classifies new data into the existing clusters. This is known

as nearest centroid classifier or Rocchio algorithm.

Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

EXAMPLE:

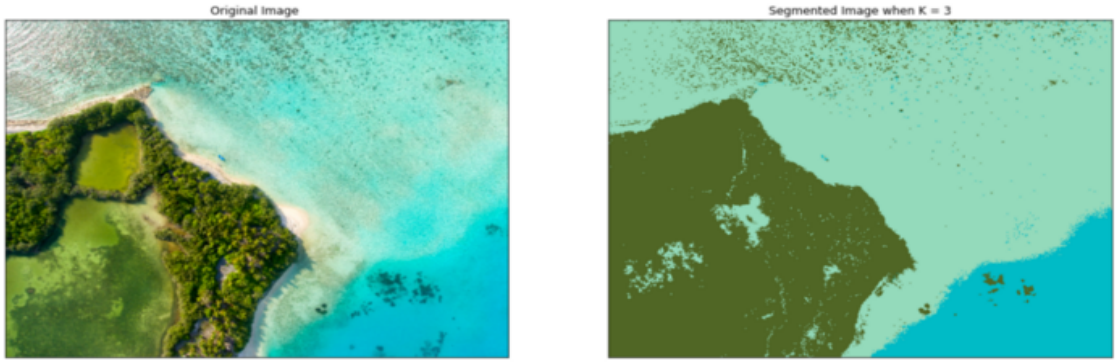


Image Segmentation when K=3

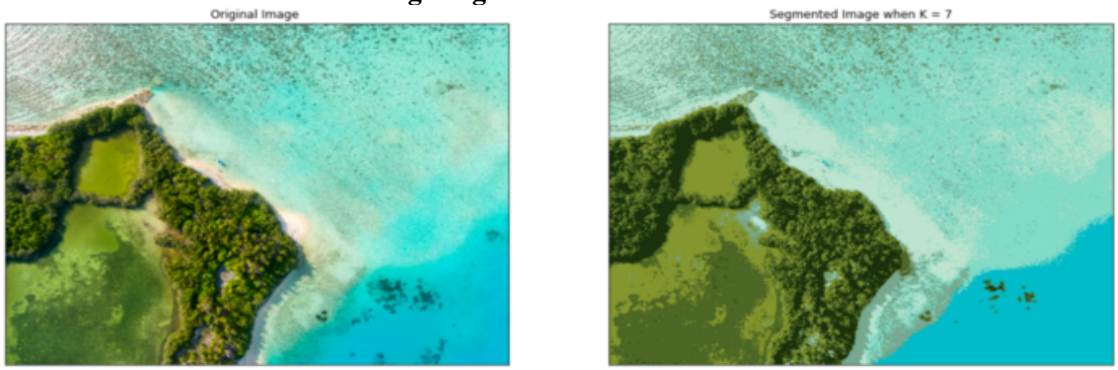


Image Segmentation when K=7

3: Recombine

The final task is to overlay the edges onto the color image is when both the color and edge image processing are complete . we do the bitwise and of the output images to get a cartoon effect in the original image

RESULTS

In the below results, the first image consists of the real world which is taken from a camera and later after using Cartoon GAN with the help of OpenCV we get the cartooned image of the original image as you can see below:-

ORIGINAL

CARTOONIFIED

GETTING CARTOONIZE VIDEO

METHODOLOGY:

To transform videos into animated or Cartoonized videos, we have additionally used the cv2(Computer vision application) which is a library in python the video is divided into frames depending on the specified time period fps=0.5s One can change the time period and No. of frames will differ. Later on, these frames will be brought together by os.join to convert into video files (.mp4 or .avi).

Conversion of Video is basically similar to getting an animated video out of the normal one. The video is first divided into frames and saved into an array, then passed through the generator and discriminator with the help of OpenCV to get the animated(cartoonized video).

IMAGE EDITING WEB APPLICATION

TECHNOLOGY USED:

1: HTML:

The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as <img /> and <input /> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

2: CSS :

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML.CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts.This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

CSS EXAMPLE:

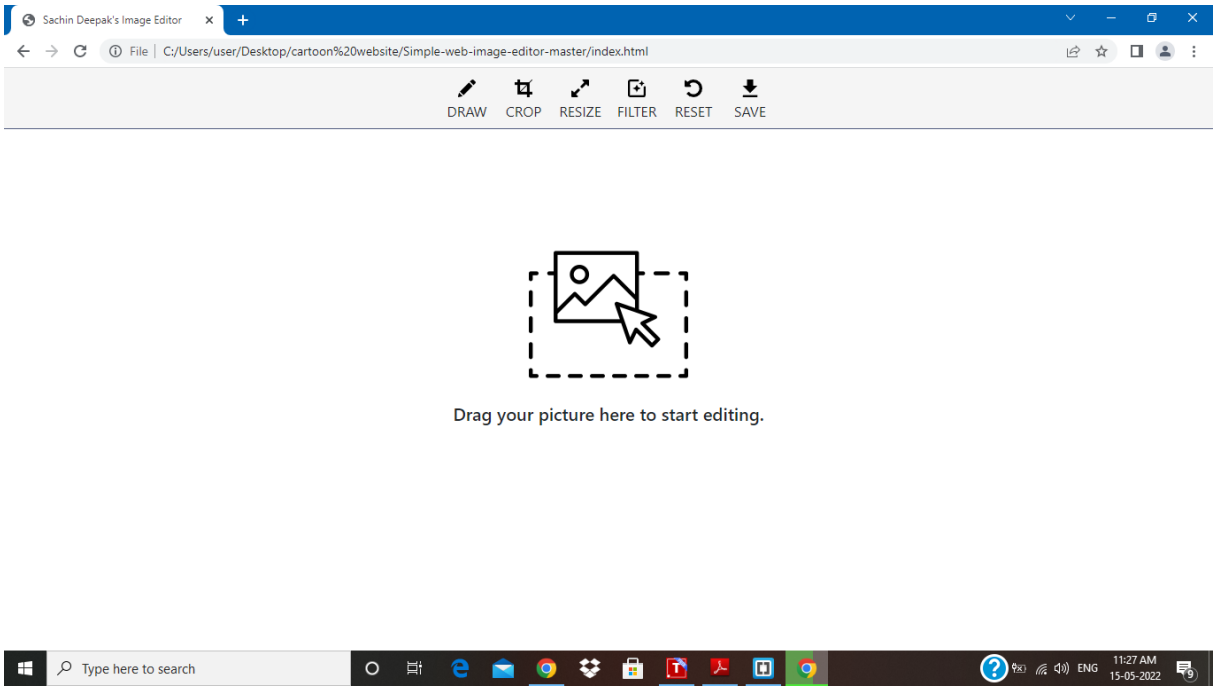
```
body {
  background-color: lightblue;
}

h1 {
  color: white;
  text-align: center;
}

p {
 font-family: verdana;
  font-size: 20px;
}
```

3 : JAVASCRIPT :

JavaScript often abbreviated JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS.Over 97% of websites use JavaScript on the client side for web page behavior ,often incorporating third-party libraries .All major web browsers have a dedicated JavaScript engine to execute the code on users' devices.
JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard .It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).
JavaScript engines were originally used only in web browsers, but are now core components of some servers and a variety of applications. The most popular runtime system for this usage is Node.js.
Although Java and JavaScript are similar in name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design.
EXAMPLE ;

```
<!DOCTYPE html>
<html>
<body>

<h2>My First Web Page</h2>
<p>My First Paragraph.</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>

</body>
</html>
```

SCREENSHOT:



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 picture of the internal structure of the wire. 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.

REFERENCES

[1] Y. Chen, Y.-K. Lai, Y.-J. Liu, "CartoonGAN: Generative Adversarial Network for photo cartoonization", International Conference on Image Processing, 2018

[2] Y. Chen, Y.-K. Lai, Y.-J. Liu, "Transforming photos to comics using convolutional neural networks", International Conference on Image Processing, 2017

[3] Zengchang Qin, Zhenbo Luo, Hua Wang, " Autopainter: Cartoon Image Generation from Sketch by Using Conditional Generative Adversarial Networks", International Conference on Image Processing, 2017

[4] J. Bruna, P. Sprechmann, and Y. LeCun., "Superresolution with deep convolutional sufficient statistics" In International Conference on Learning Representations (ICLR), 2016

[5] K. Beaulieu and D. Dalisay, "Machine Learning Mastery", Machine Learning Mastery, 2019. [Online]. Available: https://machinelearningmastery.com/.
[Accessed: 24- Nov- 2019].

[6] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," 2011.

[7] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing, Dec 2008.

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, "Microsoft coco: Common objects ´ in context," in European
conference on computer vision. Springer, 2014, pp. 740–755.

CODE USED FOR DIFFERENT TECHNIQUES

PYTHON CODE FOR IMAGE CARTOONIZATION:

```
import cv2
import scipy
from scipy import stats
import numpy as np
from collections import defaultdict

def update_c(C,hist):
  while True:
    groups=defaultdict(list)



    for i in range(len(hist)):
      if(hist[i] == 0):
        continue
      d=np.abs(C-i)
      index=np.argmin(d)
      groups[index].append(i)

    new_C=np.array(C)
    for i,indice in groups.items():
      if(np.sum(hist[indice])==0):
        continue
      new_C[i]=int(np.sum(indice*hist[indice])/np.sum(hist[indice]))

    if(np.sum(new_C-C)==0):
      break
    C=new_C
  return C,groups

# Calculates K Means clustering
def K_histogram(hist):
  alpha=0.001
```

```
  N=15
  C=np.array([128])
  while True:
    C,groups=update_c(C,hist)
    new_C=set()
    for i,indice in groups.items():
      if(len(indice)<N):
        new_C.add(C[i])
        continue
      z, pval=stats.normaltest(hist[indice])
      if(pval<alpha):
        left=0 if i==0 else C[i-1]
        right=len(hist)-1 if i ==len(C)-1 else C[i+1]
        delta=right-left
        if(delta >=3):
          c1=(C[i]+left)/2
          c2=(C[i]+right)/2
          new_C.add(c1)
          new_C.add(c2)
        else:
          new_C.add(C[i])
      else:
        new_C.add(C[i])
    if(len(new_C)==len(C)):
      break
    else:
      C=np.array(sorted(new_C))
  return C
# The main controlling function
def caartoon(img):
  kernel=np.ones((2,2), np.uint8)
  output=np.array(img)
  x,y,c=output.shape
  for i in range(c):
    output[:,:,i]=cv2.bilateralFilter(output[:,:,i],5,150,150)
```

```
  edge=cv2.Canny(output, 100, 200)
  output=cv2.cvtColor(output,cv2.COLOR_RGB2HSV)

  hists = []
  #now we are converting image from RGB to HSV for easy histogram functionality.
  hist,_=np.histogram(output[:,:,0],bins =np.arange(180+1))
  hists.append(hist)
  hist,_=np.histogram(output[:,:,1],bins =np.arange(256+1))
  hists.append(hist)
  hist,_=np.histogram(output[:,:,2],bins =np.arange(256+1))
  hists.append(hist)

  C=[]
  for h in hists:
    C.append(K_histogram(h))
  #print("centroids: {0}".format(C))

  output=output.reshape((-1,c))
  for i in range(c):
    channel=output[:,i]
    index=np.argmin(np.abs(channel[:, np.newaxis] - C[i]), axis=1)
    output[:,i]=C[i][index]
  output=output.reshape((x,y,c))
  output=cv2.cvtColor(output, cv2.COLOR_HSV2RGB)

  contours,_=cv2.findContours(edge,cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
  cv2.drawContours(output,contours,-1,0,thickness=1)
  #cartoon = cv2.bitwise_and(output, output, mask=contours)
  for i in range(3):
    output[:,:,i]=cv2.erode(output[:,:,i], kernel, iterations=1)
  #Laplacian = cv2.Laplacian(output,cv2.CV_8U, ksize=11)
  #output=output-Laplacian
  return output

if __name__=="__main__":
  output=caartoon(cv2.imread("sachin.jpg"))
```

```
 cv2.imwrite("sachin1.jpg", output)
 cv2.imshow("New generated",output)
 cv2.waitKey(0)
 cv2.destroyAllWindows()
```

PYTHON CODE FOR CARTOONIZED VIDEO:

```
import cv2
from cartoonize import caart

cap = cv2.VideoCapture(0)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH) + 0.5)
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT) + 0.5)
size = (width, height)
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('your_video.avi', fourcc, 10, size)

while(True):
 _, frame = cap.read()
 frame=caart(cv2.flip(frame,1))
 cv2.imshow('Recording...', frame)
 out.write(frame)
 if cv2.waitKey(1) & 0xFF == ord('q'):
   break

cap.release()
out.release()
cv2.destroyAllWindows()
```

CODE FOR IMAGE EDITING WEB APPLICATION

CODE 1 :INDEX.HTML

```
<!DOCTYPE HTML>
<html>

<head>
   <meta charset="utf-8" />
   <title>Sachin Deepak's  Image Editor</title>
```

```
    <script src="https://code.jquery.com/jquery-latest.js" crossorigin="anonymous"></script>
    <!--BOOTSRAP-->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.3/css/bootstrap.min.css"
integrity="sha384-Zug+QiDoJOrZ5t4lssLdxGhVrurbmBWopoEl+M6BdEfwnCJZtKxi1KgxUyJq13dy"
        crossorigin="anonymous">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min.js" integrity="sha384-
b/U6ypiBEHpOf/4+1nzFpr53nxSS+GLCkfwBdFNTxtclqqenISfwAzpKaMNFNmj4"
        crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta/js/bootstrap.min.js" integrity="sha384-
h0AbiXch4ZDo7tp9hKZ4TsHbi047NrKGLO3SEJAg45jXxnGIfYzk4Si90RDIqNm1"
        crossorigin="anonymous"></script>

    <link rel="stylesheet" href="./style.css">
    <script src="./script.js"></script>
</head>

<body>
    <audio id="audio">
        Your browser does not support the
        <code>audio</code> element.
    </audio>

    <header class="panel">
        <div class="controls">
            <div class="control-item">
                <div class="dropdown">
                    <div id="dropdownMenu2" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                        <li>
                            <img src="./assets/images/draw.svg" alt="cropButton">
                            <div class="text">DRAW</div>
                        </li>
                    </div>
                    <div class="dropdown-menu" aria-labelledby="dropdownMenu2">
                        <button class="dropdown-item" type="button" id="penDraw">Pen</button>
                        <button class="dropdown-item" type="button" id="lineDraw">Line</button>
                        <button class="dropdown-item" type="button" id="rectangleDraw">Rectangle</button>
                        <button class="dropdown-item" type="button" id="circleDraw">Circle</button>
                    </div>
                </div>
            </div>
```

```
<div class="control-item" id="cropButton">
  <li>
    <img src="./assets/images/crop-symbol.svg" alt="cropButton">
    <div class="text">CROP</div>
  </li>
</div>
<div class="control-item">
  <div data-toggle="modal" data-target="#resolutionModal">
    <li>
      <img src="./assets/images/resize.svg" alt="cropButton">
      <div class="text">RESIZE</div>
    </li>
  </div>
</div>
<div class="control-item">
  <div class="dropdown">
    <div id="dropdownMenu3" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
      <li>
        <img src="./assets/images/filter.svg" alt="cropButton">
        <div class="text">FILTER</div>
      </li>
    </div>
    <div class="dropdown-menu" aria-labelledby="dropdownMenu3">
      <button class="dropdown-item" type="button" id="grayscale">Grayscale</button>
      <button class="dropdown-item" type="button" id="threshold">Threshold</button>
      <button class="dropdown-item" type="button" id="sephia">Sephia</button>
      <button class="dropdown-item" type="button" id="invert">Invert Colors</button>
    </div>
  </div>
</div>

<div class="control-item" id="undoButton">
  <li>
    <img src="./assets/images/undo.svg" alt="cropButton">
    <div class="text">RESET</div>
  </li>
</div>
<div class="control-item">
  <a id="linkDownload" href="#" download="processedImage.png">
    <li>
```

```
                <img src="./assets/images/download.svg" alt="cropButton">
                <div class="text">SAVE</div>
              </li>
            </a>
          </div>
        </div>
      </header>
      <div class="img-area text-center">
        <!-- BEFORE DRAG IMAGE AREA-->
        <div id="default-drag-area">
          <img src="./media/49665-200.png" class="img-fluid" />
          <h5>Drag your picture here to start editing.</h5>
        </div>
        <!-- INITIAL IMAGE INVISIBLE IN ORDER TO UNDO CHANGES LATER-->
        <img id="initial-image" class="invisible" />
        <!-- PROCESSED IMAGE-->
        <canvas id="imageProcessed" class="invisible">
      </div>


      <!-- Modal resize resolution -->
      <div class="modal fade" id="resolutionModal" tabindex="-1" role="dialog" aria-labelledby="exampleModal-
Label" aria-hidden="true">
        <div class="modal-dialog" role="document">
          <div class="modal-content">
            <div class="modal-header">
              <h5 class="modal-title" id="exampleModalLabel">Resolution</h5>
              <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                <span aria-hidden="true">&times;</span>
              </button>
            </div>
            <div class="modal-body">
              <div id="modifyResolution">
                <div class="form-group">
                  <div class="custom-control custom-checkbox">
                    <input type="checkbox" class="custom-control-input" id="checkbox-img-ratio">
                    <label class="custom-control-label" for="checkbox-img-ratio">Keep the image ratio</la-
bel>
                  </div>
                </div>
```

```
                <div class="form-group">
                    <label for="width-size" class="col-form-label">Width:</label>
                    <input type="number" class="form-control" id="width-size">
                </div>
                <div class="form-group">
                    <label for="height-size" class="col-form-label">Height:</label>
                    <input type="number" class="form-control" id="height-size">
                </div>
            </div>
            <div id="uploadPictureError" class="invisible">
                <h3>You need to add a picture first!</h3>
                <h5>Use drag and drop.</h5>
            </div>
        </div>
        <div class="modal-footer">
            <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
            <button type="button" class="btn btn-primary" id="saveResolution">Save changes</button>
        </div>
    </div>
  </div>
</body>
</html>
```

CODE 2: SCRIPT.JS

```
$(function () {

  // DRAG AND DROP FUNCTIONALITY
  $('.img-area')
    .on('dragover',  (e)=> {
      e.preventDefault();
    })
    .on('drop', (e)=> {
      e.preventDefault();
      var files = e.originalEvent.dataTransfer.files;
      if (files.length > 0) {
        var reader = new FileReader();
        reader.onload =  (e)=> {
          // the image saved in an img component in case we want to restore to the initial image (img is hid-
den)
          $("#initial-image")
```

```
                .load( ()=> {
                    //Hide the original image and text in drag area
                    $("#default-drag-area").addClass("invisible");
                    // Display the canvas
                    $("#imageProcessed").removeClass("invisible");
                    // Draw canvas
                    drawInitialImage();
                })
                .attr("src", e.target.result);
            };
            reader.readAsDataURL(files[0]);
            play("./media/waterdrop.wav");
        }
    });

    // Download Image (SAVE BUTTON)
    $("#linkDownload").click((e) => {
        clearEventsDraw();
        // Check if is any image uploaded
        if ($('#imageProcessed').hasClass("invisible")) {
            e.preventDefault();
        } else {
            var imageUrl = $("#imageProcessed")[0].toDataURL("image/png").replace("image/png", "image/octet-
stream");
            $("#linkDownload").attr("href", imageUrl);
        }
    })

    // RESOLUTION CHANGER MODAL SHOW EVENT
    $('#resolutionModal').on('show.bs.modal',  (e)=> {
        // Check if is any picture uploaded
        if ($('#imageProcessed').hasClass("invisible")) {
            // No picture show error message in modal
            $("#modifyResolution").addClass("invisible");
            $("#uploadPictureError").removeClass("invisible");
        } else {
            // Show edit resolution
            $("#uploadPictureError").addClass("invisible");
            $("#modifyResolution").removeClass("invisible");
            var canvas = $("#imageProcessed")[0];
```

```
        // Update values with current resolution
        $("#width-size").val(canvas.width);
        $("#height-size").val(canvas.height);

        // Keep image ratio if selected
        var ratio = canvas.width / canvas.height
        $("#width-size").keyup( (e) => {
            if ($("#checkbox-img-ratio").is(":checked")) {
                $("#height-size").val($("#width-size")[0].value / ratio)
            }
        })
        $("#height-size").keyup( () => {
            if ($("#checkbox-img-ratio").is(":checked")) {
                $("#width-size").val($("#height-size")[0].value * ratio)
            }
        })

        // Redraw image with new resolution values
        $("#saveResolution").click(() => {
            var context = canvas.getContext("2d");
            var image = new Image;
            image.src = canvas.toDataURL("image/png");
            canvas.width = $("#width-size").val();
            canvas.height = $("#height-size").val();
            image.onload =  ()=> {
                context.drawImage(image, 0, 0, canvas.width, canvas.height);
                play("./media/drawing.wav");
            };
            $('#resolutionModal').modal('hide');
        })
    }
    clearEventsDraw();
})

// UNDO CHANGES BUTTON
$("#undoButton").click(() => {
    clearEventsDraw();
    drawInitialImage();
    play("./media/drawing.wav");
```

```
})

// CROP
$("#cropButton").click(() => {
    clearEventsDraw();

    var canvas = $("#imageProcessed")[0];
    var context = canvas.getContext("2d");
    var position = $("#imageProcessed").offset();

    var started = false;
    var image;

    // Start coord from the source image relative to image
    var x = 0;
    var y = 0;
    // Position of the mouse used to get the width and height later
    var sx = 0;
    var sy = 0;
    // Width and height of the crop
    var width = 500;
    var height = 700;

    $("#imageProcessed")
        .mousedown((event) => {
            started = true;
            sx = event.pageX;
            sy = event.pageY;
            x = sx - position.left;
            y = sy - position.top;
            image = new Image;
            image.src = canvas.toDataURL("image/png");
        })
        .mousemove((event)=>{
            if (started) {
                // Draw rectangle
                var x1 = Math.min(event.pageX - position.left, x);
                var y1 = Math.min(event.pageY - position.top, y);
                width = Math.abs(event.pageX - position.left - x1);
                height = Math.abs(event.pageY - position.top - y1);
```

```
            redrawCurrentImage(image);

            if (!width || !height) {
               return;
            }
            context.strokeRect(x1-1, y1-1, width+2, height+2);
         }
      })
      .mouseup((event) => {
         started = false;
         width = event.pageX - sx;
         height = event.pageY - sy;

         var image = new Image;
         image.src = canvas.toDataURL("image/png");
         image.onload =  () =>{
            context.clearRect(0, 0, canvas.width, canvas.height);
            canvas.width = width;
            canvas.height = height;
            context.drawImage(image, x, y, width, height, 0, 0, width, height);
            play("./media/drawing.wav");
            $("#imageProcessed").unbind("mousedown");
            $("#imageProcessed").unbind("mouseup");
            $("#imageProcessed").unbind("mousemove");
         };
      });
})

// Draw with pen
$("#penDraw").click(() => {
   clearEventsDraw();

   var started = false;
   var canvas = $("#imageProcessed")[0];
   var context = canvas.getContext("2d");

   $("#imageProcessed")
      .mousedown((event) => {
         context.beginPath();
```

```
            context.moveTo(event.offsetX, event.offsetY);
            started = true;
        })
        .mousemove((event) => {
            if (started) {
                context.lineTo(event.offsetX, event.offsetY);
                context.stroke();
            }
        })
        .mouseup((event) => {
            if (started) {
                started = false;
            }
        });
})

// Draw lines
$("#lineDraw").click(() => {
    clearEventsDraw();

    var started = false;
    var canvas = $("#imageProcessed")[0];
    var context = canvas.getContext("2d");
    var position = $("#imageProcessed").offset();
    var image;
    var x = 0;
    var y = 0;

    $("#imageProcessed")
        .mousedown((event) => {
            x = event.pageX - position.left;
            y = event.pageY - position.top;
            started = true;
            image = new Image;
            image.src = canvas.toDataURL("image/png");
        })
        .mousemove((event) => {
            if (started) {
                redrawCurrentImage(image);
                context.beginPath();
```

```
            context.moveTo(x, y);
            context.lineTo(event.pageX - position.left, event.pageY - position.top);
            context.stroke();
            context.closePath();
        }
    })
    .mouseup((event) => {
        if (started) {
            started = false;
        }
    });
})
// Rectangle draw
$("#rectangleDraw").click(() => {
    clearEventsDraw();

    var started = false;
    var canvas = $("#imageProcessed")[0];
    var context = canvas.getContext("2d");
    var position = $("#imageProcessed").offset();
    var image;
    var x = 0;
    var y = 0;
    var width = 0;
    var height = 0;
    $("#imageProcessed")
        .mousedown((event) => {
            x = event.pageX - position.left;
            y = event.pageY - position.top;
            started = true;
            image = new Image;
            image.src = canvas.toDataURL("image/png");
        })
        .mousemove((event) => {
            if (started) {
                x = Math.min(event.pageX - position.left, x);
                y = Math.min(event.pageY - position.top, y);
                width = Math.abs(event.pageX - position.left - x);
                height = Math.abs(event.pageY - position.top - y);
```

```
            redrawCurrentImage(image);

            if (!width || !height) {
                return;
            }
            context.strokeRect(x, y, width, height);
        }
    })
    .mouseup((event) => {
        if (started) {
            started = false;
        }
    });
})
// Circle draw
$("#circleDraw").click(() => {
    clearEventsDraw();

    var started = false;
    var canvas = $("#imageProcessed")[0];
    var context = canvas.getContext("2d");
    var position = $("#imageProcessed").offset();
    var image;
    var startX = 0;
    var startY = 0;
    $("#imageProcessed")
        .mousedown((event) => {
            startX = event.pageX - position.left;
            startY = event.pageY - position.top;
            started = true;
            image = new Image;
            image.src = canvas.toDataURL("image/png");
        })
        .mousemove((event) => {
            if (started) {
                redrawCurrentImage(image);

                var x = event.pageX - position.left;
                var y = event.pageY - position.top;
```

```
            context.beginPath();
            context.moveTo(startX, startY + (y-startY)/2);
            context.bezierCurveTo(startX, startY, x, startY, x, startY + (y-startY)/2);
            context.bezierCurveTo(x, y, startX, y, startX, startY + (y-startY)/2);
            context.closePath();
            context.stroke();

        }
    })
    .mouseup((event) => {
        if (started) {
            started = false;
        }
    });
})
// Image filters

$("#grayscale").click((event)=>{
    var canvas = $("#imageProcessed")[0];
    var context = canvas.getContext("2d");
    let imageData = context.getImageData(0, 0, context.canvas.width, context.canvas.height);
    let pixels = imageData.data;
    for (let i = 0; i < pixels.length; i += 4)
        pixels[i] = pixels[i + 1] = pixels[i + 2] = Math.round((pixels[i] + pixels[i + 1] + pixels[i + 2]) / 3);
    context.putImageData(imageData, 0, 0);
})

$("#threshold").click((event)=>{
    var canvas = $("#imageProcessed")[0];
    var context = canvas.getContext("2d");
    let imageData = context.getImageData(0, 0, context.canvas.width, context.canvas.height);
    let pixels = imageData.data;
    for (let i = 0; i < pixels.length; i += 4) {
        var r = pixels[i],
            g = pixels[i+1],
            b = pixels[i+2];
        var v = (0.2126*r + 0.7152*g + 0.0722*b >= 180) ? 255 : 0;
        pixels[i] = pixels[i+1] = pixels[i+2] = v
    }
    context.putImageData(imageData, 0, 0);
```

```
    })


    $("#sephia").click((event)=>{
        var canvas = $("#imageProcessed")[0];
        var context = canvas.getContext("2d");
        let imageData = context.getImageData(0, 0, context.canvas.width, context.canvas.height);
        let pixels = imageData.data;
                    for(var i = 0; i < pixels.length; i+=4) {
                            var r = pixels[i],
                                    g = pixels[i+1],
                                    b = pixels[i+2];
                            pixels[i] = (r * .393) + (g *.769) + (b * .189)
                            pixels[i+1] = (r * .349) + (g *.686) + (b * .168)
                            pixels[i+2] = (r * .272) + (g *.534) + (b * .131)
                    }
        context.putImageData(imageData, 0, 0);
    })

    $("#invert").click((event)=>{
        var canvas = $("#imageProcessed")[0];
        var context = canvas.getContext("2d");
        let imageData = context.getImageData(0, 0, context.canvas.width, context.canvas.height);
        let pixels = imageData.data;
                    for(var i = 0; i < pixels.length; i+=4)
                    {
                            var r = pixels[i],
                                    g = pixels[i+1],
                                    b = pixels[i+2];
                            pixels[i] = 255-r;
                       pixels[i+1] = 255-g;
                            pixels[i+2] = 255-b;
                    }
        context.putImageData(imageData, 0, 0);
    })

});

function drawInitialImage() {
    var canvas = $("#imageProcessed")[0];
```

```
    var context = canvas.getContext("2d");
    var image = document.getElementById("initial-image");
    canvas.width = image.width;
    canvas.height = image.height;
    context.drawImage(image, 0, 0, canvas.width, canvas.height);
}

function redrawCurrentImage(image) {
    var canvas = $("#imageProcessed")[0];
    var context = canvas.getContext("2d");
    canvas.width = image.width;
    canvas.height = image.height;
    context.drawImage(image, 0, 0, canvas.width, canvas.height);
}

function play(path) {
    var audio = $('#audio')[0];
    audio.src = path;
    audio.load();
    audio.play();
}

function clearEventsDraw() {
    $("#imageProcessed").unbind("mousedown");
    $("#imageProcessed").unbind("mouseup");
    $("#imageProcessed").unbind("mousemove");
}
```

CODE 3 : STYLE.CSS:

```
container-fluid,
html,
body,
.full-screen {
  height: 100%;
}

.panel {
```

```
  width: 100%;
  background-color: #f5f5f5;
  border-bottom: 1px solid #545e81;
  position: fixed;
  height: 69px;
}

.controls {
  display: flex;
  justify-content: center;
}

.control-item {
  padding: 8px 0px;
  cursor: pointer;
}

li {
  list-style: none;
  display: flex;
  -webkit-box-orient: vertical;
  -webkit-box-direction: normal;
  -ms-flex-direction: column;
  flex-direction: column;
  -webkit-box-pack: justify;
  -ms-flex-pack: justify;
  justify-content: space-between;
  -webkit-box-align: center;
  -ms-flex-align: center;
  align-items: center;
  text-align: center;
  margin: 5px 8px;
  width: 48px;
  height: 48px;
}

li:hover {
  color: #101f58;
}
```

```
li img {
  height: 20px;
  width: 20px;
  margin-bottom: 4px;
}

a {
  color: inherit;
}
a:hover {
  text-decoration: none;
  color: inherit;
}

.img-area {
  padding-top: 69px;
  min-height: 100%;
}
#default-drag-area {
  padding-top: 8%;
}
.invisible {
  display: none;
}
input[type="number"]::-webkit-inner-spin-button,
input[type="number"]::-webkit-outer-spin-button {
  -webkit-appearance: none;
  margin: 0;
}
```