ADIP – Assignment 02

Report

Mustafa Usman

Mu06166

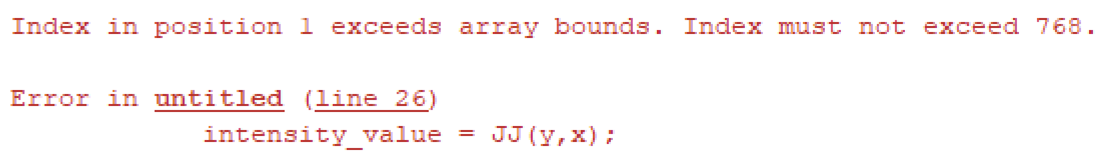
Introduction:

In this assignment we were given a jpg image and was asked to convert into an ascii art, where the main idea was converting the pixel intensities into its corresponding asci character.

Initially, I started the assignment on matlab and was convinced to complete the assignment on matlab however, array manipulation and caring for array bounds in matlab got a little confusing so had to switch to python-OpenCV for completion of this assignment which in result took some time and effort but I was able to complete the assignment and understand it completely. Both Matlab and python codes are provided with this assignment.

Code on Matlab:

- Initially we started off by reading the images using the imread command.
- Created a string variable which would hold all the given asci characters.
- Finding the length of asci variable using length function.
- Dividing 255 (which is the maximum intensity value) by length calculated in the last step to find the appropriate range of intensities depending on our given asci array.
- Creating temporary arrays like art and line to hold the character values after mapping.
- Implementing two nested loops to loop over the length and height of each given pixel and store their intensity value in the variable intensity value.
- After retrieving the intensity value we divide it by the range (XX) calculated before to find out the index for our asci array.
- We would then populate the recent created temp variables to hold these indexes.
- After the commencement of one loop we would intentionally append '\n' in order to identify a line break.
- Trying to create a text file and trying to write to on that same file using fopen command with 'wt' to write on the given file, given fprintf command I tried to populate to the given file but there a lot of issues were rising things were getting confusing and I switched to python.



```
Index in position 1 exceeds array bounds. Index must not exceed 768.

Error in untitled (line 26)
            intensity_value = JJ(y,x);
```

“Solution of MatLab is faulty”

Code on Python – OpenCV:

- Importing the libraries; CV2, math, time.
- A variable image is used to store and read the given images using imread
- Given size of the image was too big, resizing the given image using the built in function of cv2.resize(). We’re keeping the dimensions [350, 180]. To represent a nice little box.
- Retrieving the height and width [H,W] of the given image using the built in function image.shape[0/1], passing 0 or 1 to retrieve height or width respectively.
- Curating an array of self-given asci characters of length 9.
- Calculating the range, similarly by dividing the 255(max intensity value) by length of the asci character list. We’re using the function floor to make sure in rounds off to nearest integers and static casting all of this in int() to make sure the result that we’re getting in the type int.
- Creating two arrays, final_arr and final_arr2, they will use to populate html and text files respectively.
- Now we are going to loop over all the pixels of the given image by using nested loops over height and width that we extracted before.
- Creating two temporary string variables to store the values within the loop.
- Now in the loop we can easily extract the intensity value of each given pixel.
- `intensity = image[y,x]`
- Since we haven’t changed the image into grayscale we would be getting three different values of intensity based on the pixels rgb value. In order to map the intensity value from the asci character array we would be required to calculate the average of the intensity values of rgb..
- We’re also going to divide the intensity value by the range value that we calculated before in order to find the index of the character from our array to map the intensity.
- Accordingly we are also going to make use of static int casting and floor function to make sure we are dealing with type int.
- `index = int((math.floor(intensity[0]/range_) + math.floor(intensity[1]/range_) + math.floor(intensity[2]/range_))/3)-1`
- Now we’re left with an int value which can be used as an index value to extract an asci character from our given list of characters.
- Now inorder to map the intensities based on their rgb values we created a Boolean variable boolGrayScale and set the default to false.

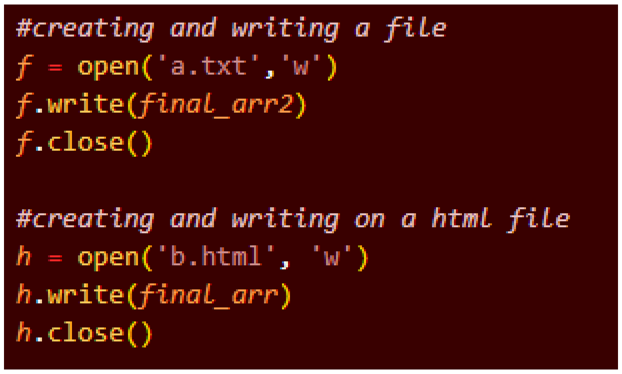
- In case we have a grayscale image we are going to populate our temporary variables with the regarding html tags to display the characters in grayscale.

```
if boolGrayscale:
            line += f'<samp style="color:
rgb({index},{index},{index})">{ch}</samp>'
```

- the samp html element us used to enclose inline text, where index is the range value and ch is our character that is to be insersted in place of that specific pixel.
- In case of a rgb value we're just replacing index with the intensity values of red, green and blue respectively in our last bit of code.

```
lse:
            line += f'<samp style="color:
rgb({intensity[2]},{intensity[1]},{intensity[0]})">{ch}</samp>'
```

- After extracting and appending these charcter values into the temporary variable we would append them in the final array list and in case of text file append '\n' at every linebreak and <br> in case of html files.
- Final step would be to create text and html files and populate them with our final arrays



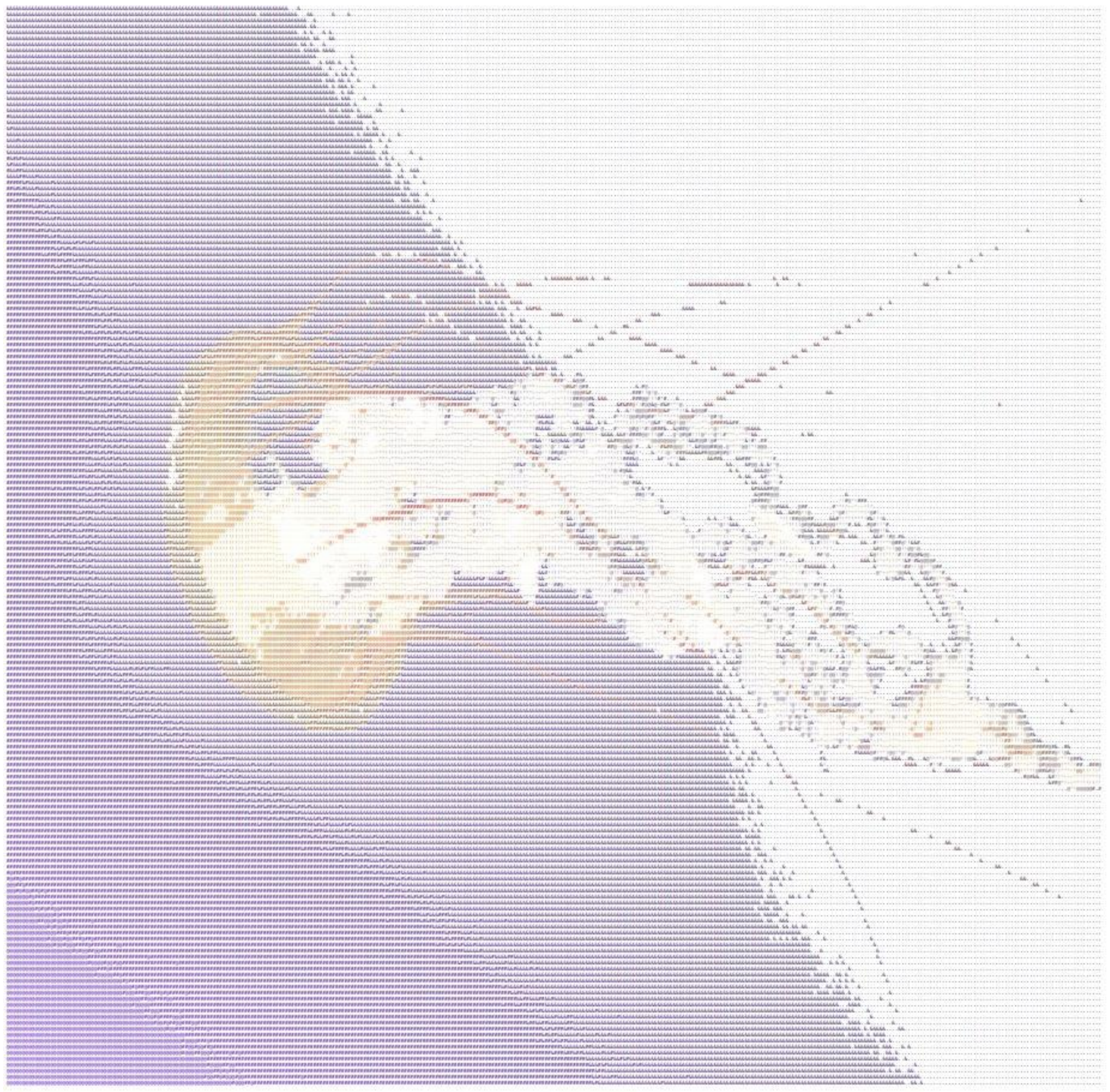
```
#creating and writing a file
f = open('a.txt','w')
f.write(final_arr2)
f.close()

#creating and writing on a html file
h = open('b.html', 'w')
h.write(final_arr)
h.close()
```

- In order to calculate processing time in terms of milliseconds we imported the library of time subtracted start time by end time and multiplied by 1000.

Results:

Tulip: 0.190 ms

Jellyfish: 0.0021 ms



Python Code:

```
import cv2
import math
import time

boolGrayscale = False
image = cv2.imread("tulip.jpg")  #reading image
```

```
# image = cv2.imread("Jellyfish.jpg")
image = cv2.resize(image, dsize=(350,180)) #rescaling image

H, W = image.shape[0], image.shape[1]  #extracting dimensions of an image
ascii = ["&","#", "@", "!", "*", "+", "-", "^", "."]  #curating a list of ascii
characters
range_= int(math.floor(255/len(ascii)))  #our ascii list has 9 chracters dividing
by 255 (max inetnsity) to find range

final_arr = ''   #curating a final array
final_arr2 = ''

#looping over all the pixels
for y in range(H):
    line = ''  #a temporary variable
    line2=''
    for x in range(W):
        intensity = image[y,x]  #extracting intensities of a given pixel
        #Since we have three channels we'll find the mean of the intensity
        index = int((math.floor(intensity[0]/range_) +
math.floor(intensity[1]/range_) + math.floor(intensity[2]/range_))/3)-1
        # print(index)
        # if index == 0:
            # exit()
        ch = ascii[index]  #extracting the given charcater
        #providing html tags using samp tag
        if boolGrayscale:
            line += f'<samp style="color:
rgb({index},{index},{index})">{ch}</samp>'
        else:
            line += f'<samp style="color:
rgb({intensity[2]},{intensity[1]},{intensity[0]})">{ch}</samp>'
        line2 += ch
    line2 += '\n'
    final_arr += line + '</br>'  #adding to our final array with a line break
    final_arr2 += line2

#creating and writing a file
f = open('a.txt','w')
f.write(final_arr2)
f.close()

#creating and writing on a html file
h = open('b.html', 'w')
h.write(final_arr)
```

```
h.close()

start_time = time.time()

print("%s milliseconds" % ((time.time() - start_time)*1000))
```