Path Planning And Computer Vision Puzzle

Anuj Kakde (20CS30005)

Abstract— **This task is a puzzle which we need to solve completely by following the leads. The tasks start with an image called Level1.png, from which we need to derive some ASCII text. This text has instructions about what we need to do next.**

In the second part we had to get an image from the remaining elements of the Level1.png. I did this by adding the required elements of the image to a list and then converted it to a numpy array of proper shape. This image was a part of the larger image named zucky_elon.png.

In the next task we have to find where (top left coordinates) was the smaller image was taken from. I did this using template matching in Opencv library of python. The required x coordinate was 228.

Now we had to remove coloured noise from a monochrome image. I first read the maze image and then split it into three colour channels of blue, green and red. Out of the three, the blue channel contained the maze while the other two were just noise. Now as we know the colour of monochrome image is 228, I traversed over the entire image and set the pixel values of those who were already above 228 to 255 and the remaining to 0. This removed all the noise and I got a clear image as shown below. Then I saved this image as Maze.png. After solving the maze I got the password that would open the final treasure for me. The password was APPLE.

Then I got an image which was to be converted to audio. For this decoding I wrote a program which opened the image using cv2. Then I wrote the pic as binary file in "Treasure.txt". Then I changed its file type to mp3 and thus I got my final treasure.

I. INTRODUCTION

This question is a treasure hunt puzzle in which we have to follow the leads to reach to the final reward. Initially we are given an image which has an encoded ASCII text that we need to decrypt for getting further instructions. Next we had to recover an image from the same image. I had a hard time finding the image.

Towards the end of this hunt we were required to convert a grayscale image to mp3 file. The mp3 file was embedded in the image. This was quite a challenge for me as I searched many python audio modules but got no result. I could retrieve the file only after 3 days search.

II. PROBLEM STATEMENT

This Task is a Treasure hunt puzzle

The task starts with an image called Level1.png. As we know in a normal RGB scale the colour is represented by a number from 0 to 255 inclusive for each channel. We also know that an ASCII character can be represented as a binary number from 0 to 255 inclusive. The thin initial greyscale pixels of the image contain the information about what we have to do next.

After decoding the first image the further instructions were as follows.

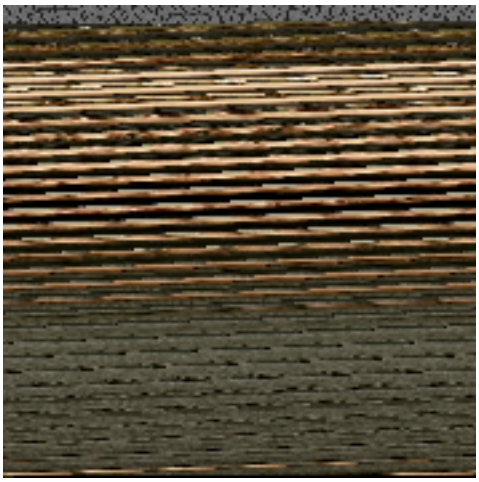

Fig. 1. Level1.png

"*Congrats on solving the first level of this task! You were able to figure out the ASCII text from the image, but wait! Your journey is not yet over. After the end of the text you will find a (200, 150, 3) coloured image. This image is a part of the bigger image called "zucky_elon.png". Find the top left coordinate (Image convention) from where this image was taken. The x coordinate represents the colour of a monochrome maze hidden in an image with coloured noise. Find the maze and solve the maze using any algorithm like dfs but better. Try comparing them and seeing how they perform like A*, RRT, RRT* for example. Once the maze is solved you will see a word. This word is a password to a password protected zip file which contains a png. Note that the password is case sensitive and all the alphabets in the password will be capital letters This is your treasure. To open the treasure you need to convert the image in to an audio file in a simple way like you did for this ASCII text. Once converted, open the .mp3 file and enjoy your treasure, you deserved it! A part of the image "zucky_elon.png" will begin immediately after the colon, image-lv2 :*"

III. RELATED WORK

In the first part of this task, I tried to open the image in notepad in order to decode it. Also for the last part of the problem, I tried every possible thing in python that I could. I searched and tried applying various sound modules in python but with no success. Only at the last moment finally I was able to decode the relatively simple part.

This task also required me to learn the numpy library of python.

IV. INITIAL ATTEMPTS

Initially, in the second part of the question where we had to decode the image present in Level1.png, I tried rearrange the image matrix but according to the question, image matrix would have 200x150 i.e. 30000 elements.

Only after careful observation, I found out that the ending elements were zeroes, which we did not have to take in the image.

Also my initial attempts of the final part of this question we mostly some or the other implementation of the sound libraries of python. In doing this I got a exposure to the various python modules.

V. FINAL APPROACH

This is the complete process by which I solved the puzzle

A. First Part

The first part (Task3a.py) was to decode the ASCII text from the Level1.png image shown in Fig 1. To do this I first read the image into my code using the cv2 library in gray scale mode. Then I used the chr() function to convert an integer to its corresponding character. After some trail and error I was able to see that the message is present in the first 8 elements of the pic variable.

B. Second Part

In this part (Task3b.py) we had to get an image from the remaining elements of the Level1.png. To do this, I made an empty list and inserted the required 3000 elements in the list. Next I converted the list into a numpy array using the numpy.array() function. Now we have an numpy array which is of order 2, but we need the order of array to be 3. So I reshaped it using the arr.reshape() function to the required dimensions.

After doing this, I converted the array into an image and saved it using the PIL library. But the image I got had more amount of blue colour in it. This was due to the fact that OpenCV uses BGR while PIL uses RGB. So to counter that I converted the image from RGB to BGR and then saved it. This gave me the required image shown below.

C. Third Part

In the third part we need to find the top left coordinate (Image convention) from where the derived image was taken. In the code of this part (Task3c.py) I have first read the two images using cv2 module. Then I used the template matching function of the openCV library.

Template Matching is a method for searching and finding the location of a template image in a larger image. In cv2.matchTemplate(img_gray,template,cv2.TM_CCOEFF_NORMED) the first parameter is the main image, second parameter is the template to be matched and third parameter is the method used for matching. It returns a grayscale image, where each pixel denotes how much does the neighbourhood of that pixel match with template.

Fig. 2. Image derived from Level1.png

Threshold is the accuracy with which we want the template to match. Then we filter the out values using numpy.where() function. zip() returns an iterable, an object that can be used to loop over. It creates tuples of the input arguments. By observing the list loc, I found that the first element to be zipped is the required point. Thus the required x coordinate is 228.

D. Fourth Part

Now we need to remove noise from the image maze_lv3.png. I did this in the program Task3d.py. Here I first read the maze image and then split it into three colour channels of blue, green and red. Out of the three, the blue channel contained the maze while the other two were just noise.

Now as we know the colour of monochrome image is 228, I traversed over the entire image and set the pixel values of those who were already above 228 to 255 and the remaining to 0.

This removed all the noise and I got a clear image as shown below. Then I saved this image as Maze.png.

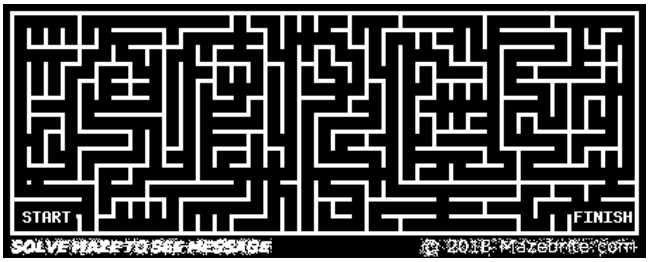

Fig. 3. Maze.png

1) Fifth Part: After solving the maze I got the password (APPLE). Then I got an image which was to be converted to audio. The image represented audio in the same way as the image of the first task represented ASCII text. For this

decoding I wrote a program (Task3e.py) which opened the image using cv2. Then I wrote the pic as binary file in "Treasure.txt". Next I opened the text file using a media player and voila, the audio file started playing. First time I felt good after being Rick Rolled :D

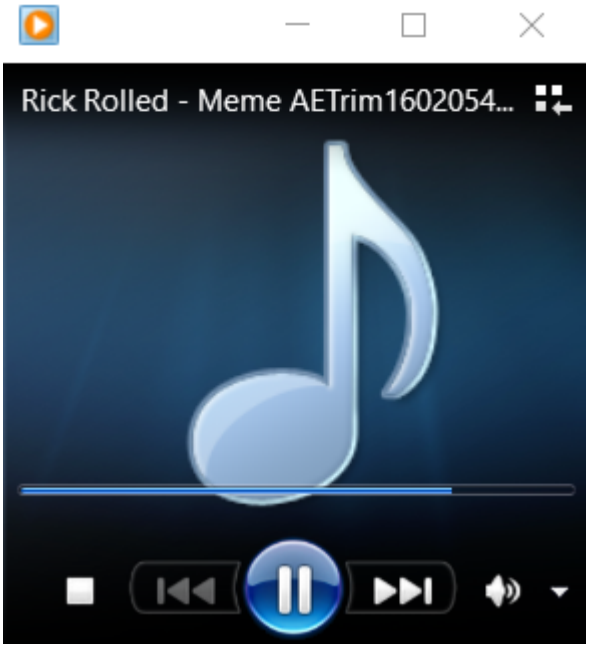

Fig. 4. Final Audio

VI. Results and Observation

Thus the puzzle/Treasure hunt is successfully completed and the final mp3 file has been derived.

VII. Future Work

To complete this task, I did not need to use any search algorithm. Still I read about some path finding algorithms. After this I would like to read more about them and implement them myself.

CONCLUSION

It was the most interesting task among all others. This problem tested the skill of finding the solution of any problem, however tough it may be.

References

[1] https://www.geeksforgeeks.org/template-matching-using-opencv-in-python/
[2] https://stackoverflow.com/questions/56449024/explanation-of-a-few-lines-template-matching-in-python-using-opencv