Aerial Robotics Kharagpur Task Round Documentation Task 3 : PATH PLANNING AND COMPUTER VISION PUZZLE

Nikhil Giri*

***Abstract*— This Documentation gives a summary of how I approached Task 3 given to me during ARK Software Selection Round**

From the time when I was first given these tasks to the time i am writing this documentation I have learned and explored a lot of things. I learnt a lots of Path finding algorithms and also tried implementing them. These algorithms are essential for drones to judge and correct their path on basis of algorithms they encounter.

The Computer Vision puzzle had 4 different level. Each level had to be solved sequentially as the next as it had some hint about the next level. Also each level required some sort of Image processing concept application. After all these levels we end up with mp3 file created in a mind-blowing manner, But before hearing it take a small break and relax yourself as it might shock you a bit. That being said the treasure was not that beautiful the path taken to achieve it was nothing short of great.

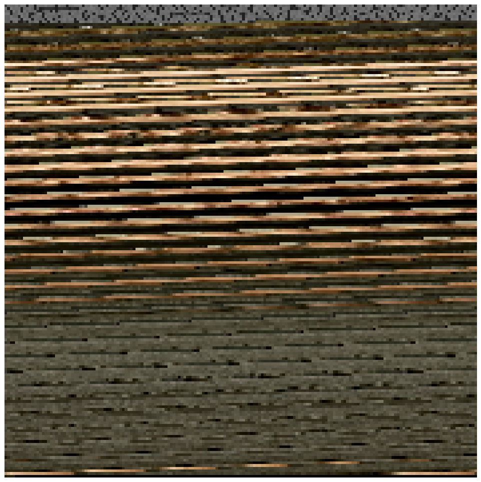
I. INTRODUCTION

In the beginning we are given a Simple PNG file. Thought the image makes no sense and seems random pixels at a glance. But there is a Hint stating that inside this noisy image there is a clue on how to proceed further in the Image processing Puzzle.

The Hint mentions that the pixel of the image given might be the encrypted values of a text message. This ASCII characters are hidden inside the pixel of this random noisy image. On decoding this clue the entire path and challenges become more or less clear. We have to start with RGB to Grayscale,int to ASCII text, Image pixel manipulation, then we apply concepts like path Finding Algorithms , Thresholding , Template Matching, int to Byte and File Handling.

II. PROBLEM STATEMENT

The task essentially is an image processing based treasure hunt. In this task, we have to use our image processing skills to decode the clues in each step.Initially we are given 4 Files - 3 images ”Level1.png” , ”maze-lvl3.png”, ”zucky-elon.png” and along with these 3 images we have a password protected zip file called ”Treasure.zip”. Using these images, we have to decode the path to the further treasure hunt.



Once we decode these images, we will get instructions on how to proceed and complete the treasure hunt. Each step of this treasure hunt level requires certain image processing skills and some levels also needs the knowledge of application of pathfinding algorithms in an image,

Initially, we are given a hint regarding how to decode ”Level1.png” image, which suggest that a value of pixel in an image can be between 0 to 255 and also ASCII characters can be represented in binary between 0 -255.SO, we can try to check the pixel values,as they may be ASCII text information which can provide the path and aim of the treasure hunt.

III. Related Work

For this task I learnt BFS, DFS, Dijkstra's Algorithm and A* algorithm and implemented it. I also learnt RRT and RRT* and tried implementing RRT in the Level three but Failed to complete it.

IV. Initial Attempts

Initially, My attempt was just implementing the hint. The hint suggested that in the image "Level1.png" pixel values might contain encrypted text which points to the next clue of the treasure hunt. My initial attempt was also my final attempt.

V. Final Approach

Now lets see, How I proceeded for the treasure hunt. The hunt was split into various levels and I will also discuss my final attempt level by level.

Level 1 : Hidden Message in Level1.png

As earlier mentioned the hint suggests that the values of the pixel in the "Level1.png" image file contains in formation or clues regarding the Treasure Hunt. So I first did a pixel by pixel operation and converted the level1.png image file into grayscale image by taking the mean of all 3 values a RGB pixel contains.

If for a certain (i,j) in the image there is a pixel array which stores the 3 values of all RGB channel then the values of the pixel at same (i,j) at the gray scale image can be calculated using this formula:

$$gray - pixel = \frac{pixel[0] + pixel[1] + pixel[2]}{3}$$

After forming this new grayscale image we carry out the main objective that is extraction of the hidden clue.

```
for i in range(h):             #h is height of image
    for j in range(w):         #w is width of image
        print(chr(int(gray1[i][j])),end="")
    print()
```

Listing 1. Grayscale image values to ASCII text

The Result of the above code is :

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 algothrim 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 aplhabets 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 immediatly after the colon, image-lv2 :

Level 2 : The Hint analysis and approach to obtain the level 2 image

From the above text what we get is that first of all we have to find the first pixel of the second image and then obtain it. Once we obtain this new image we have find match it with the other image "zucky-elon.png" and obtain the x-coordinate of the first pixel(top-left) and that value will help to decode the third image.

So lets begin with the task to find the first pixel. The Clue suggests the new image starts after ':' so we can iterate throught the grayscale version of "Level1.png" and find the first pixel where the value of pixel is equal to the ASCII value of ':'.

```
img = cv2.imread("Level1.png")
h,w,z = img.shape
gray1 = np.zeros((h,w,1), dtype = "uint8")

for i in range(h):
    for j in range(w):
        gray1[i,j,0] = ((img[i,j,0]/3) +(img[i,j
    ,1]/3)+(img[i,j,2]/3))
row_num = 0
col_num = 0
exit = False
for i in range(h):             #h is height of image
    for j in range(w):         #w is width of image
        if chr(int(gray_level1[i][j])) ==':':
            col_num = (j+1)%w
            row_num = i + math.floor((j+1)/w)
            print("Starting pixel of second image :
  " ,row_num,",",col_num)
            exit = True
            break
    if exit == True:
        break
```

Listing 2. Position of starting pixel for Level2 image

Using the above program we find that the first pixel is at 6,94 in i,j convention. So we can now start extracting the "Level2" image form "Level1.png".

We know that size of Level2 image is (200, 150, 3) from the text obtained from "Level1.png" so we arrange the next 200*150 pixels including(6,94) in the same shape using the following code:

```
roi = np.zeros((200,150,3), dtype = "uint8")
x =row_num
y =col_num
for a in range(200):
    for b in range(150):
        roi[a][b] = img[x][y]
        y+=1
        if (y == w):
            x+=1
            y =0
```

Listing 3. Position of starting pixel for Level2 image

Now the Level2 image(ROI) comes out to be :

Where Roi is the region of interest in the task to match this level2 image with the ”zucky-elon.png” as the clue from Level1 said that this Level2 image is a part of the zucky-elon image. So we need an approach to find out the region from where this level2 image is taken and find the co-ordinate of the top-left pixel in the big image

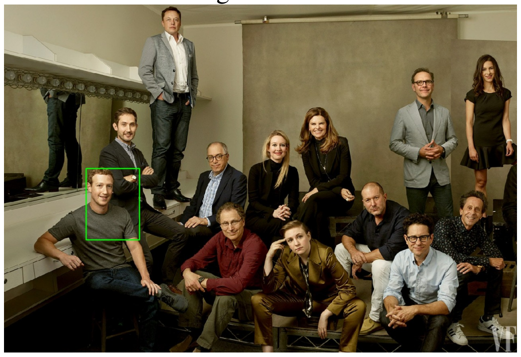
Now Lets see the program I used to find the X-coordinate by matching this level2 image with the given ”zucky-elon.png”

```python
target = cv2.imread("zucky_elon.png")
for i in range(h1-200-1):
    for j in range(w1-150-1):
        if(target[i][j][0] == roi[0][0][0]):
            match =True
            #print(target[i][j][0])
            for a in range(200):
                for b in range(150):
                    if (target[i+a][j+b][0] != roi[a][b][0]):
                        #print("not here")
                        match =False
                        break

        if match == True:
            m=i
            n=j
            print("(x, y) ==",n,m)
            cv2.rectangle(target, (n,m), (n+150,m+200), (0,255,0), 2)
            match = False
            break
cv2.namedWindow("Target",cv2.WINDOW_NORMAL)
cv2.imshow("Target",target)
```

Listing 4. Template matching

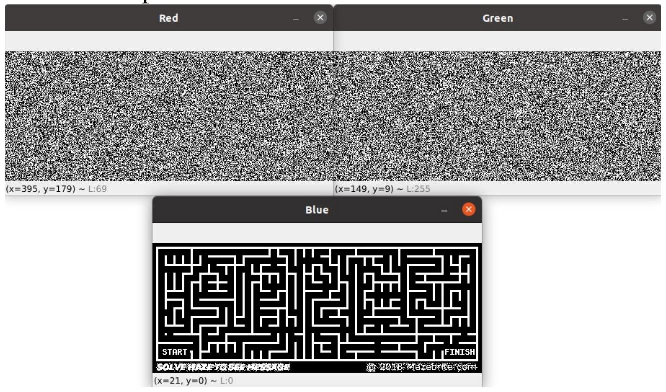
Now the Matching process prints out ***(x,y) == 230,460*** and gives out the following answer :



Thus we can clearly see that the result of matching process is correct and the x-coordinate we desired is ***230***

Level 3 : Extracting the maze and solving it

Now that we have the x-coordinate (230) , Lets analyse what the clue from level 1 further says. It represents a monochrome image so I considered each channel and applied a threshold such that if value of the pixel is equal to 230 then it the pixel values should be changed to 255 i.e. white in color. Else all pixels were given a value of 0 i.e. Black. Here is the output:

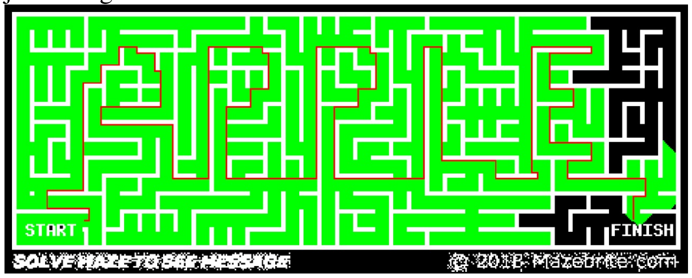


As we can see the monochrome channel we required was the Blue layer. The thresholding resulted into a well defined Maze and Now all we have to do is to implement various Path finding algorithm and Find their output. The Output will be a word which be the password of the password protected Zip File.

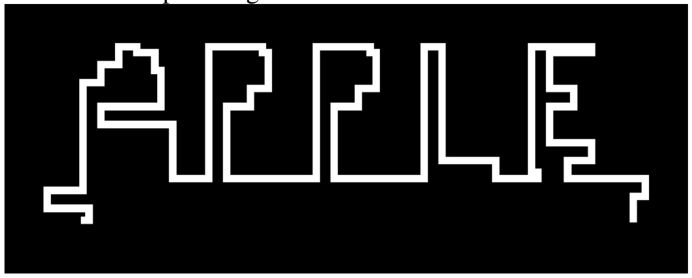
For path finding Algorithms I implemented a total of 5 Algorithms(4 +1 variation) they are :

- BFS
- DFS
- Dijkstra
- A* – Euclidean distance
- A* – Manhattan distance

We will discuss and compare them in the Result and Observation section. Right now lets ee the output of the Dijkstra Algorithm.



Now on thresholding and performing morphological operations on the output we get:

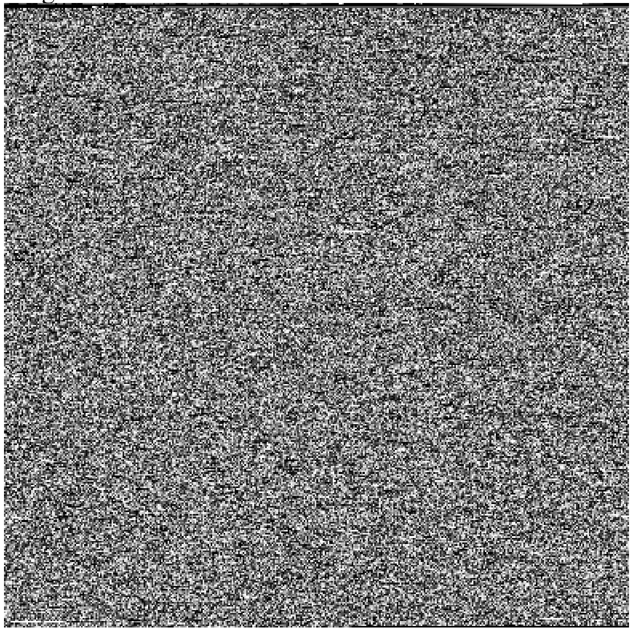


It is evident that the password for the ZIP file is " APPLE" .

Level 4 : The Final puzzle - visual to audio

The moment of truth, What lies behind the gates of the "treasure.zip" file. I had the password- "APPLE". So lets put it in.

As soon as the file opens we see another image "treasure-mp3.png". This is the last image to decode. Once we decode it we get our treasure.



On further reading the clues from the "Level1.png" file we have to convert this image into audio. First i thought it was not going to be that difficult but little did i knew then. Thought the final solution is pretty simple and was quite painful that i wasted so much time overthinking when the solution as this small. So i tried various method from using various packages such as scipy.io.wavfile to create a wav file to trying to create the mp3 by thinking that the 3 channels of the image represent 3 different factors of sound such as frequency,wavelength and amplitude (I accept that, I was desperate and now in hindsight this approach makes no sense). After various attempts i tried file handling and tried converting the image as grayscale and converting each pixel as integer as we did in First image and write them in a new file with extension mp3 or wav. After that did not work i got the idea to instead of doing so as integer, lets write it as binary, after even that did not work after various moderations, I tried to write it as a byte and that led me to the final solution.

```
img = cv2.imread("treasure_mp3.png")
h,w ,z= img.shape


if(z != 3):
    print("Please upload a colored image")
    exit()

gray1 = np.zeros((h,w,1), dtype = "uint8")

arr = []

for i in range(h):
    for j in range(w):
        gray1[i,j,0] = ((img[i,j,0]/3) +(img[i,j
,1]/3)+(img[i,j,2]/3))
        arr.append(gray1[i][j][0])


songo = bytearray(arr)
file1 = open("Song.mp3","wb+")
file1.write(songo)
file1.close()
```

Listing 5. PNG to MP3

I created array of bytes using BYTEARRAY function and the wrote it in a file in wb+ mode to write it as a binary and voila the result as a working 6 seconds long audio. The final treasure was achieved.

To be fair I tried the approach of the first image on this image which gave me a text which had some data about encryption of the mp3 file and it had the name of the audio file. So I wasn't that surprised when I heard the 6 seconds long mp3 file. The pain of Rick roll was much much smaller than the sense of achievement.

VI. Results and Observation

While the Result of the task was a 6-seconds long Mp3 file lets now discuss the performance of different path finding Algorithms.

Algorithm	Total Nodes analyzed	Final Path Length
BFS	46018	1695
DFS	22883	8151
Dijkstra	46018	1695
A*-Euclidean	45309	1695
A*-Manhattan	39962	1871

As we can see BFS, Dijkstra, A*(1) give us the smallest path to the end of the maze. But they also search big number of nodes which essentially slows then down. Whereas DFS and A*(2) give very big paths but both of them are very fast. In DFS though the path length is ridiculously big it is also the fastest whereas A*(2) is sort of a compromise between speed and accuracy as though the heuristic favors shorter path, the distance from end i.e. the factor which determine speed preference has a greater weight than the A*(1).

VII. Future Work

In my implementation of RRT, the program was taking too much time and the nodes generated could only reach till middle of the graph and not go beyond it. The maze was bit too complex for random searching. So,i plan implement RRT and RRT* in the future.

Conclusion

In the end I would say that the puzzle was quite intuitive and I had lots of fun solving it. This task also shows use that Image processing is quite power full toll with wide range of implementation.

References

[1] Amit Patel, , ”Heuristics, From Amit’s thoughts on Path Finding”, 2020.
[2] Wikipedia,”A* search algorithm”,2021
[3] Wikipedia,”Rapidly-exploring random tree”,2021