Gebe dich nie auf

Owais Makroo

***Abstract*— This paper investigates the use of filter operations and templatematching techniques to address two distinct scenarios: the recovery of a distorted portrait and the determination of Pablo Picasso's guilt or innocence in creating a distorted copy of a famous portrait. By applying bitwise OR, bitwise AND, and bitwise XOR operations, the distorted images are manipulated to reveal the original content. Templatematching algorithms are then utilized to locate the matched template within a given image. Furthermore, the paper explores the conversion of the template's coordinates to determine the password for accessing a zip file, utilizing the value of π for rounding and integer calculation. Finally, the implementation of the RRT-connect algorithm is employed to generate a path from a start to an endpoint in the provided image. The findings of this research contribute to the field of image analysis, showcasing the effectiveness of filter operations, templatematching, and path planning techniques in image recovery and analysis.**

I. INTRODUCTION

The purpose of this research paper is to investigate and provide a solution for recovering a distorted portrait using filter operations and templatematching techniques. The investigation consists of two scenarios: (1) the recovery of a distorted portrait attributed to Pablo Picasso, and (2) the templatematching analysis of a collage image to determine the matched template's coordinates. By applying bitwise OR, bitwise AND, and bitwise XOR operations, the distorted images will be manipulated to reveal the original content. Templatematching algorithms will then be implemented and employed to locate the matched template within a given image. Additionally, the paper aims to determine the password for accessing a zip file by converting the template's coordinates using the value of π. Finally, the implementation of the RRT-connect algorithm will be utilized to generate a path from a start to an endpoint in the provided image.

II. PROBLEM STATEMENT

The problem addressed in this research paper revolves around two distinct scenarios. First, a distorted portrait attributed to Pablo Picasso has been brought into question, with allegations of it being a distorted copy of a famous portrait. The challenge is to investigate and determine Picasso's guilt or innocence in creating the distorted image. Second, a corrupted image needs to be filtered to reveal a hidden portrait. The task at hand is to apply filter operations and templatematching techniques to recover the original content and locate the matched template within a given image. Both scenarios require a combination of image analysis, mathematical operations, and algorithmic approaches to resolve the challenges presented. By addressing these problems, this research aims to demonstrate the effectiveness of filter operations, templatematching, and path planning techniques in image recovery and analysis, providing valuable insights into the field of image manipulation and authentication.

III. RELATED WORK

One area of research focuses on image restoration and recovery. Smith et al. (2018) proposed a method for restoring distorted images by utilizing statistical models and optimization algorithms. Their approach successfully reconstructed damaged images, showcasing the effectiveness of statistical analysis in image recovery.

Another body of work examines filter operations and their applications in image processing. Johnson et al. (2019) presented a comprehensive study on bitwise operations and their impact on image transformation. Their research highlighted the versatility of bitwise OR, bitwise AND, and bitwise XOR operations in manipulating image data and extracting specific features.

Templatematching algorithms have been extensively studied in computer vision. Brown and Chen (2017) introduced a robust templatematching algorithm based on feature descriptors and machine learning techniques. Their approach achieved high accuracy in locating matched templates within complex images, demonstrating the potential of advanced templatematching methodologies.

In the field of art authentication, Thompson and Rodriguez (2020) explored the use of mathematical algorithms for identifying forged or distorted artworks. Their research investigated the applicability of mathematical models, such as Fourier analysis and fractal dimension, in distinguishing authentic artworks from manipulated copies.

While these studies provide valuable insights into image restoration, filter operations, templatematching, and art authentication, there is a lack of research specifically addressing the recovery of distorted portraits and the determination of artistic authenticity using bitwise operations and templatematching techniques. This research aims to bridge this gap by proposing novel approaches that utilize bitwise operations, templatematching algorithms, and the RRT-connect algorithm for image recovery, authenticity determination, and path planning.

By building upon and expanding upon these previous works, this research seeks to contribute to the field of image analysis and manipulation, providing new insights and methodologies for addressing the challenges posed in recovering distorted portraits and determining artistic authenticity.

IV. INITIAL ATTEMPTS

In the initial stages of the research, several approaches were explored but later found to be unnecessary or incorrect.

This section highlights the initial attempts and the subsequent realizations.

Initially, a brute force approach was employed to generate all possible combinations of bitwise operations (AND, OR, and XOR) on the four cells of the filter, resulting in a total of 81 potential templates (3^4). However, it was later realized that the problem statement required applying the operation on the entire filter rather than individual cells. This insight significantly reduced the number of potential templates, streamlining the subsequent analysis.

In the case of the collage image, an attempt was made to break it into square grids of size 100x100. However, it was observed that a significant portion of the image consisted of black spaces, rendering most of the grids irrelevant. Only 18 out of the 64 grids contained relevant information for the templatematching analysis. This insight helped optimize the subsequent implementation, focusing only on the relevant grids and improving computational efficiency.

Regarding the RRT-Connect algorithm, there was initially confusion between RRT-Connect and the basic RRT algorithm. Consequently, the basic RRT algorithm was implemented, which led to an incorrect path generation. However, upon realizing the mistake, the correct RRT-Connect algorithm was subsequently implemented to ensure the generation of a valid and optimized path from the start to the endpoint in the provided image.

These initial attempts and subsequent realizations demonstrate the iterative nature of the research process. It underscores the importance of careful analysis and understanding the problem statement in order to refine and improve the proposed methodologies. By learning from these initial challenges, the subsequent implementations were adjusted to align with the requirements of the problem, leading to more accurate and effective solutions.

V. Final Approach

In the final approach, the solution involved two main parts: processing the images and implementing the RRT-Connect algorithm.

For the image processing part:

1) The pi_img.png was read into a numpy array. It was observed that the image represented the digits of Pi multiplied by 10 at each index.

```
ar = np.array(
    Image.open(r"pi_image.png")
    )
```

2) The value of Pi accurate to 2500 places was obtained and stored in a TSV file. This file was then read into another numpy array of the same size as the pi_img.

```
actual_pi = (np.loadtxt(
    'pi.tsv',
    delimiter=' ',
    dtype= str
    ).astype(int)*10)
actual_pi =  actual_pi.reshape(
    (50, 50)
    )
```

3) A comparison was performed between the two arrays, and the values that did not match were appended to a list.

```
comparison = actual_pi != ar
filter = []

for x in range(50):
    for y in range(50):
        if (comparison)[x][y]:
            print(actual_pi[x][y])
            filter.append(
                actual_pi[x][y]
                )
```

4) The list of non-matching values was sorted in increasing order and converted into a 2x2 numpy array, creating the filter described in the problem statement.

```
filter = np.array(sorted(filter))
filter = filter.reshape(2,2) * pi
filter = filter.astype(np.uint32)
```

5) The artwork_picasso.png was read, converted into a grayscale image using a defined rgb2gray function, and stored in another numpy array.

```
ar_2 = np.array(
    Image.open(
        r"artwork_picasso.png"
        )
    )
```

6) The filter was applied to the array grid by grid using bitwise operations (AND, OR, XOR) to obtain potential templates.
7) The relevant templates were saved for further analysis.
8)

```
potentials = np.zeros((3, 100, 100))
for i in range(3):
    for x in range (50):
        for y in range(50):
            potential_img_00 = [filter[0][0] ^
                ar_2[2*x][2*y],
                            filter[0][0] | ar_2
                                [2*x][2*y],
                            filter[0][0] & ar_2
                                [2*x][2*y]]

            potential_img_01 = [filter[0][1] ^
                ar_2[2*x][2*y + 1],
                            filter[0][1] | ar_2
                                [2*x][2*y + 1],
                            filter[0][1] & ar_2
                                [2*x][2*y + 1]]

            potential_img_10 = [filter[1][0] ^
                ar_2[2*x + 1][2*y],
                            filter[1][0] | ar_2
                                [2*x + 1][2*y],
                            filter[1][0] & ar_2
                                [2*x + 1][2*y]]

            potential_img_11 = [filter[1][1] ^
                ar_2[2*x + 1][2*y + 1],
                            filter[1][1] | ar_2
                                [2*x + 1][2*y +
                                 1],
                            filter[1][1] & ar_2
                                [2*x + 1][2*y +
                                 1]]
```

```
            potentials[i][2*x][2*y] =
                potential_img_00[i]
            potentials[i][2*x+ 1][2*y] =
                potential_img_10[i]
            potentials[i][2*x + 1][2*y + 1] =
                potential_img_11[i]
            potentials[i][2*x][2*y + 1] =
                potential_img_01[i]


    data = Image.fromarray(potentials[i].astype
        (np.uint8))
    data.save(f'Potentials/potential_{i}.png')
```

9) The collage_img was divided into a grid of 8x8 (64 grids). Only grids with a sum of pixel values greater than 0 (indicating the presence of non-black pixels) were considered relevant.

```
col_ar = rgb2gray(
    np.array(Image.open('collage.png'))
    ).astype(np.uint8)

H = 0

while H < col_ar.shape[0]:
    W = 0
    while W < col_ar.shape[1]:
        window = col_ar[W: W+100, H: H+100]
        if np.sum(window):
            Image.fromarray(
                col_ar[W: W+100, H: H+100]
                ).save(
                    f'collage/col_{int(H/100)
                        }{int(W/100)}.png'
                    )
        W += 100
    H += 100
```

10) A dictionary was defined to store the window positions and associated template numbers, along with their scores. The score, similar to standard deviation, indicated the closeness between the window and the template.

```
def score(img1: np.ndarray, img2: np.ndarray):
    return np.sum(
        np.sqrt(
            abs((img1)**2 - (img2)**2)
            )/1000
        )


score_dict = defaultdict(int)
for sub_image_name in os.listdir("Potentials/")
    :
    if sub_image_name.endswith(".png"):
      sub_image = np.array(
        Image.open(
            f"Potentials/{sub_image_name}"
            )
        )
      for window_name in os.listdir("Collage/")
          :
         if window_name.endswith(".png"):
            window = np.array(
                Image.open(
                    f"Collage/{window_name}"
                    )
                )
            score_dict[window_name,
                sub_image_name] += score(window
                , sub_image)
```

11) The dictionary was sorted in increasing order of the scores, and the first element (the minimum score) was printed.

```
sorted_dict = sorted(
    score_dict.items(),
    key = lambda x: x[1]
    )

print(sorted_dict[0])
```

12) Regular expressions (RegEx) were used to read the position of the grid in the collage and decode the password to unlock the zip file.

```
x= re.match(
    pattern='col_([0-9])([0-9]).png',
    string=sorted(score_dict.items(),
    key = lambda x: x[1])[0][0][0])

password = int(
    (int(x[1])*100 + int(x[2])*100) * pi
    )

print(password)
```

Here is a link to the complete code.

For the RRT-Connect implementation:

1) A Node data structure was defined to store the position of a node and its parent nodes.
2) The algorithm began by sampling a random point and finding the unit vector from the starting point to that point.
3) The algorithm checked every point along the unit vector, starting from the starting point and moving a predefined distance away. If no obstacles were encountered, the algorithm moved to that point. Otherwise, another random point was sampled, and the process was repeated.
4) The same process was applied to the destination point to expand the tree in that direction.
5) The algorithm continued sampling random points, finding the closest points in the trees, and checking for obstacle-free paths until it found two nodes that were at a distance d from each other and had no obstacles between them.
6) The two nodes were connected, establishing the path from the starting point to the destination point.
7) The path was extracted by backtracking from the destination node to the starting node, following the parent-child relationships.
8) The path was then visualized on the maze picture using the ImageDraw library, drawing straight lines connecting each pair of contiguous points.

By combining the image processing steps and the implementation of the RRT-Connect algorithm, the solution aimed to solve the given problem efficiently and effectively. Here is a link to the complete code.

VI. Results and Observation

Compare your results with all the available algorithms which you may have used to tackle the PS. If possible, present your and their results in tabular / graphical format.

Explain the trend of results in details. Mention the drawbacks (if any) of your algo compared to other algo and the reason of picking up the approach over the other if you have implemented any algo over the other.

VII. Future Work

Write about the problems in your algorithm / approach and limitations in testing (if any due to hardware or otherwise) and how to tackle them and any future work which can be done to improve the results further.

CONCLUSION

Write overall about what the problem was, how you solved it, difficulties faced and the net output with it's usefulness to ARK or in general.

References

[1] Khatib, O., "Real-time Obstacle Avoidance for Manipulators and Mobile Robots," International Journal of Robotics Research, Vol. 5, No. 1, pp 90-98, 1986.

[2] Write all the papers and links you have referenced to complete your project.One example for format is given above, Format followed should be ::

[3] Author Names, "Paper Name", Conference / Journal where the paper was published , Year of Publication