CMP2090M

OBJECT-ORIENTED PROGRAMMING ASSIGNMENT REPORT

Alex Howe, HOW15618835

1. INTRODUCTION

This report details my thought processes and ideas I had during the creation of my OOP Assignment and an evaluation of my programme.
Initially, I decided to break the assignment task into smaller steps. The obvious one was performing the Nearest Neighbour Search, however, there were other problems I had to solve to get to that stage, such as reading the text files into the programme and outputting the PGM image.
As I didn't know what a PGM image file was and how it behaved at first, I researched into this and discovered that PGM images were essentially a file of values, from 0 to the maximum, with some various details about that file ("P5" to denote it being a PGM, the height and width, and maximum possible value). As we were given the function to write a PGM file it was relatively easy to understand how to change a text file of decimal values into a PGM image.

2. PROGRAMME STRUCTURE

As we were given most of the class hierarchies we were required to use, setting these up wasn't too difficult. However, there was one class whose relevance and functionality in relation to the other three classes I didn't fully understand; the Matrix class.
As the matrix I needed to use was a two-dimensional list of numbers, I decided that the STL Vectors collection would be an appropriate way of storing the values. I also decided that, as the data being handled in the Image classes would be very similar to the matrix, I would have the Matrix class inherit from Base_Image. This way I could directly use the functions in the Matrix class on each Image object. In my Matrix constructor, I resized the vector according to the provided row and column sizes. I mostly only used the Base_Image class to contain the function to read the text files into their respective matrices. I heavily modified the provided function to perform this task, to both accommodate for the change of data structure, and to generally integrate it into my own code.
I also needed to create a function which returned the height and width of the 2D Vectors, and for this I created a virtual function in the Base_Image class, which was overridden in the two Image subclasses. They simply returned the values for the rows and columns respectively.
The last function to be called inside the Main method was findWally. Inside this was the nearest neighbour search which will be investigated further in the next section. The function also called lowestVal, which searched through the list of differences generated by the NNS algorithm. There are two variables passed by reference into it, which are assigned the values of the column and row of the lowest value using nested for loops to traverse the vector. The function also returned the value of the lowest value.

The function showWally was then called which effectively drew a black rectangle starting from the coordinates of the top-left corner of where Wally was most likely to be, by changing the values stored in the Scene object to 0 in the relevant positions. Finally, the txttoPGM function was called, writing the vector into a file and adding the appropriate headers in.

3. NNS Algorithm

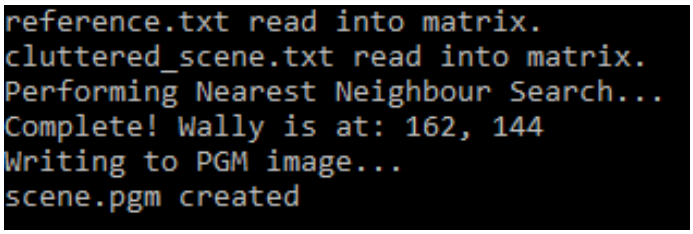
The Nearest Neighbour Search is an algorithm to find the closest match to a given dataset, in this case, the Wally reference image, in another, bigger, dataset – this being the cluttered Where's Wally scene.
I performed this using four nested for loops: the outer two being to navigate through the big scene, and the inner two for searching through a section of the scene the same size as the Wally image. Inside all of these I found the absolute difference between each pixel of the reference and the same pixel in the scene image. After the internal for loops had completed, I took the summation of these absolute differences and stored it in another vector called 'vals' and reset the sum of absolute differences in preparation for the process to repeat again.

Pseudocode:

```
for ExtRows = 0 to (sceneHeight – wallyHeight) {
   for ExtCols = 0 to (sceneWidth – wallyWidth) {
      for IntRows = 0 to wallyHeight {
         for IntCols = 0 to wallyWidth {
            sumOfDiffs += abs(scene[ExtRows + IntRows, ExtCols + IntCols] – wally[IntRows, IntCols])
      vals(ExtRows, ExtCols) = sumOfDiffs
      sumOfDiffs = 0
```

4. Results

The programme I created updates the user on its progress through the console. It outputs when the text documents have been read into matrices, notifies the user when the Nearest Neighbour Search begins, tells the user which pixel with which Wally was found (correct to the top-left corner of the image), and details the creation of the PGM file. The PGM image is created using the edited matrix and shows a rectangle around where Wally is predicted to be.



4.1 Best matching

In my programme, Wally was found at the coordinates 162, 144. Due to the nature of my application, this is the top-left corner of where Wally is. Looking at the PGM image that was generated, this is the exact location of Wally in the scene, meaning the Nearest Neighbour Search was a success.

4.2 N-best list

Unfortunately, I did not implement the N-best list of Wally matches. I initially had plans to, which is why I decided to store all the lowest values of the summations of absolute values in a separate vector, however I was unsure of how to find the n^{th} closest matches to Wally and thus could not fully complete the extension.

5. Discussion & Conclusion

As stated in section 4.1 I believe this application performed as it should and is, therefore, a success. However, I feel that my programme could have been much more efficient.
In terms of Memory Management, I could definitely have done better – the memory usage is approximately 15MB, however much of this were the three vectors I created. I attempted to minimise the memory usage by switching from using doubles to floats, however I could have done more. Time-wise it took approximately 17 seconds (Release, x64) to run to completion. I also could have implemented the extension task but was unsure of how to do so, as detailed in the previous section.
On the other hand, the class structure and abstraction of my programme were strong points in my opinion.

References

Poskanzer, J. (1991). *PGM Format Specification*. [online] netpbm.sourceforge.net. Available at: http://netpbm.sourceforge.net/doc/pgm.html

Andoni, A., Indyk, P. and Razenshteyn, I. (2018). *Approximate Nearest Neighbor Search in High Dimensions*. [online] Arxiv.org. Available at: https://arxiv.org/abs/1806.09823