CMP2090M

Object-Oriented Programming Assignment Report

Kiran Thomasson, [Student ID: 15625218]

1. Introduction

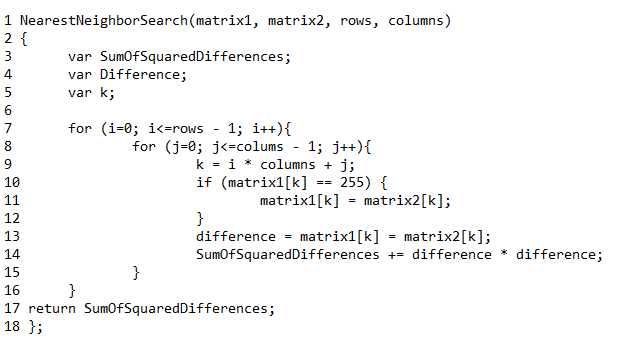
We were tasked to build a C++ solution which makes use of a Nearest Neighbour Search (NNS) algorithm to find an item from a reference image inside of a larger image. In this case, we were given a base image of wally and a scene in which to find him. NNS has many different ways of implementing and many different methods which can produce differing results. I decided to make use of the Sum of Squared Differences algorithm as it is best suited for dealing with numbers such as in a matrix for an image. I also decided to use the moving window approach in which the algorithm works through the image giving each window a SSD value and stores the lowest result. This is the closest match to the reference image, meaning locating the subject’s nearest match is a simple task.

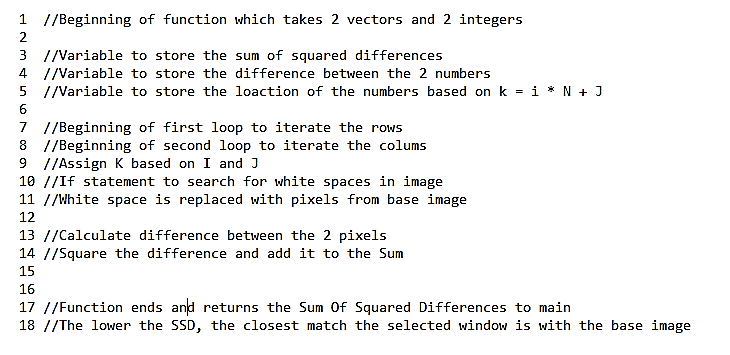
2. Programme Structure

The structure of my program makes use of many different OOP features including classes, objects, attributes and functions. //Talk about classes. I used objects of my classes in order to pass data into the main function. This allows my code to be efficient with memory usage as it allows objects to be deleted from memory when they are no longer in use. I also made use of functions in my program. One such function performs the sum of squared differences for each of the windows. It takes 2 matrices, the number of rows & columns and returns the result as a double value. The main function then takes he value returned and checks If it is lower than the lowest value recorded. If it is lower than the number along with the I and J integer values to record the location of the window when the SSD was lowest, thus this is a nearer match for wally. After the loops have finished the result will be the array containing the values of the lowest SSD and the location of the window in the form of I and J.

3. NNS Algorithm

Below is the pseudocode for my NNS algorithm. I also provide a line by line description of my algorithm. I chose to use a Moving-Window Sum of Squared Differences (SSD) algorithm to solve the problem given. I chose to use this algorithm because of its ability to take into account a window of a certain size and compare it to a reference image. It then provides a number based on the similarity of the 2 matrices, the lower the number is, the closer the 2 matrices, the more similar they are. With a result of 0 being an exact match. My algorithm uses a window the same size as Wally’s reference image and iterates through the base image, giving each area a SSD. It then passes these back to the main function which will record which positioning of the moving window gave the n nearest fit for to the reference of Wally.





4. Results

In this section detail the output of your programme: use pictures to illustrate these where possible.

4.1 Best matching

What is the best match to the provided template image? Have you found Wally?

4.2 N-best list

Extension task: describe the results of your programme to identify the (ranked) *n* best matching images (where *n* can be provided at run-time).

5. Discussion & Conclusion

Provide a discussion of your results (what worked well/badly and why)

References

If you use any references, place them here, ensuring you follow proper citation guidelines.