I am bad at Spot The Difference. It's a game where there are 2 photos, one with a number of changes from the original, and you have to find all the differences as quickly as possible. The difficulty ranges from extremely obvious changes to minute differences in the photo.

I made the mistake of buying one of these books to test a theory that I had about the similarities between finding the differences in spot the difference, and finding Wally in the 'Where's Wally' series - that theory being that there is a pattern to the placement of the differences due to it being made by a person, and people are inherently pattern-following, even subconsciously.

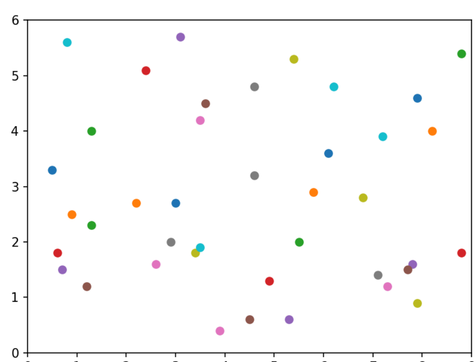
Here's how I did it.

I heard about a little project that a guy named Dr Randal Olson did on the American version of Where's Wally (aptly named 'Where's Waldo') that aimed to optimise the path that your eyes should take to find Wally in the shortest time. His method used a genetic AI model to find the best path between the points that Wally is found on the page.

I saw this and wondered what else this can be used for… and then I got the idea of Spot The Difference books. I thought this was perfect because it shares many of the same features as the Where's Wally series, such as needing to quickly find something that's potentially very difficult to spot, and it being made by a person. As mentioned earlier, people are very pattern-prone and so this will likely mean that there's a subtle pattern to the placements of the differences in this book, and it could possibly also apply to other books from other authors too, though this is not guaranteed.

So, to start my journey, I found and bought a Spot The Difference book, and I measured the position of the points in the first 5 9cm x 6cm photo. This gave me the coordinates of each difference in these images - giving a total of 40 points. While this is a small sample of the total 640 differences in the book, it should be enough to start seeing a pattern.

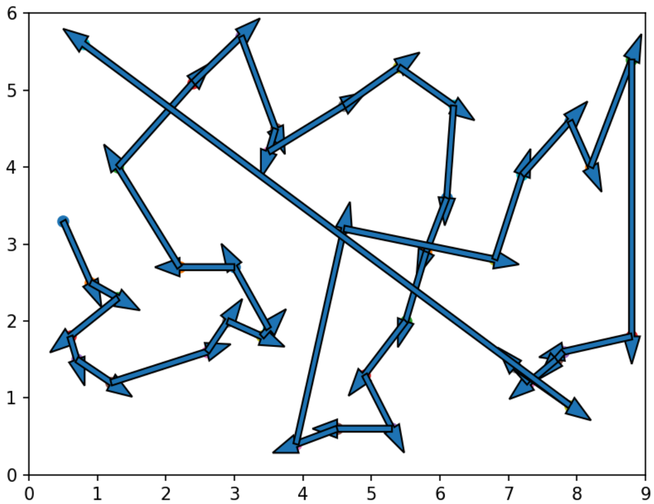
I then imported these coordinates into a .csv file for use in a python program that would analyse the positions and give me the most likely points that they would be in. I used matplotlib to draw a graph of these points :



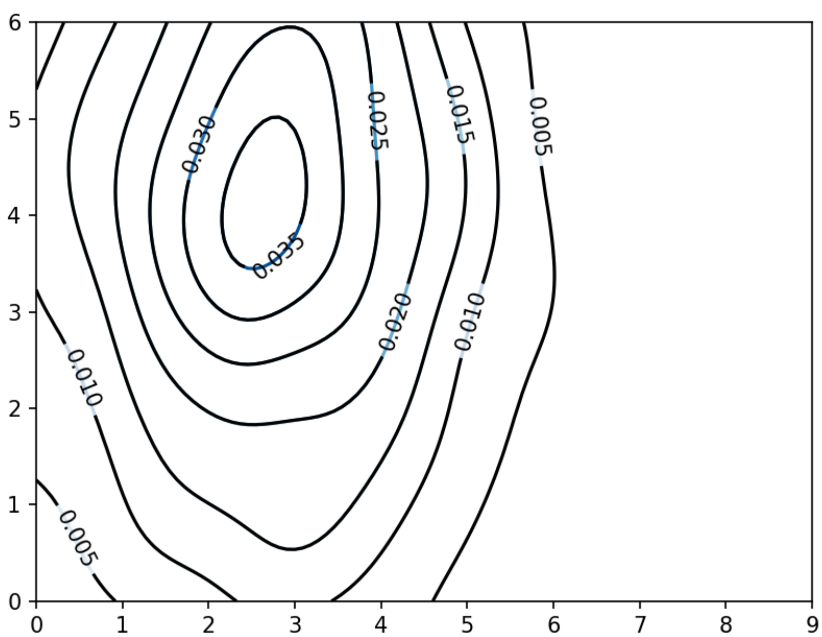
This seems like a rather random assortment of positions but I knew that there had to be a pattern to it.

I then used a Dijkstra algorithm to find the path between closest points, this being the fastest path to follow between the points, and though this gave an almost chaotic and inhuman path to follow that would be near impossible to remember and follow accurately, it did give me some information.

Using the matplotlib ‘arrow()’ function, I drew the points and the Dijkstra path on a graph for a visual representation. Here's what it showed :



This, again, is a seemingly chaotic path to take so, taking large inspiration from Dr Randal Olson’s work, I used a kernel density estimation to find the most likely areas for the points to be, and this gave this graph :



This density graph shows us the most likely position of the difference, and in this case it's in the upper left quadrant of the page, with the second most likely being the lower left. This is likely due to some of the pictures having 4 to 6 differences in those quadrants, with only 2 to 4 in the right half of the page. This can also be due to a small sample size giving less accurate data. Using this, as well as the Dijkstra path, I concluded that the best path to follow to find the differences is to look in the upper left quadrant, focusing on the centre and then the edges of the quadrant, before moving to the lower left quadrant. After the differences are found, move to the right side of the page and carefully look from bottom to top, right to left - this is because we are used to reading in the opposite direction and this is a psychological trick used by people such as special forces to spot tiny differences.

Overall, the current data is not super accurate, as I used a sample size of 6.25% of the population of differences, but it is enough to create a rudimentary method that allows the user to find the differences in each photo faster. I was able to cut times to find the differences by ~30 - 40 seconds using this method.

In order to improve this, I would use multiple books, and all points from each book to then have a much larger data set and get more accurate results. I would also use a another algorithm (instead of Dijkstra's) as this is likely to get a better result.