Tadhg Riordan
12309240

Computer Vision
Assignment 2 - Which Page

High Level Overview

- Test Data gathered
    - Attributes for all the correct features in the test data is found (slopes of sides, angles of corners, number of good matches and correct page).
    - SURF algorithm, FLANN matcher and perspective transformation used to gather this.
- Mean and standard deviation of the angles of the four corners in the test data calculated
    - This will be used to identify correct pages in the scene when the main part of the program is executed
- Main part of the program is executed
    - all pages matched against each scene in turn (again using SURF etc) and attributes gathered for each.
    - a function is called with these attributes which returns and prints the correct page number for that scene.

Description

The main part of my solution uses the SURF (Speeded Up Robust Features) algorithm. This relatively new algorithm is loosely based off of SIFT with more advantages; the object to be recognised in the scene can be scale/rotation invariant and the algorithm is relatively strong against ambient occlusion. In terms of the technical implementation, It first creates a number of keypoints in the object it is trying to find; these are distinctive parts of the image, such as edges, and the number of keypoints it finds is decided by the hessian value, which I set to 10. This is the 'detector' part of the algorithm. Next, all of the neighboring points around any keypoints are compiled into vectors called 'feature vectors'. This creates a 'descriptor', which must be distinctive for the matching procedure but must also deal with perspective and detector errors. To match the descriptors, I used the FLANN (Fast Approximate Nearest Neighbour Search) 'k' nearest neighbour matcher which finds the k best matches for each descriptor. k was obviously set to 2 to find the two best matches for that descriptor in both the page and scene. From this, a number of 'good matches' had to be found; this was done using the default Nearest Neighbour Distance Ratio between the two matches. When all the good matches are finally acquired, their keypoints are extracted, and a perspective transform was found and applied using the findHomography function with the RANSAC method and the perspectiveTransform method respectively. Lines are drawn around the object in the scene, and the result is displayed, along with the matches and page being used to search.

At this stage the corners of the correct object in all scenes can be found. I now had to find a way for the program to display which page was the correct one from this. I initially thought that the page with the highest number of good matches would suffice but this actually turned out not to be the case, as sometimes incorrect objects would have a higher ‘good matches’ value than correct objects. After trying some other ways with inconsistent results, ie. object ratio/side lengths/slope lengths, I eventually decided to use the mean and standard deviation of the four angles of the found feature. First, I gathered test data from 8 randomly chosen scenes and their corresponding pages, where I mapped the correct object to the correct scene. I would get the object feature with the most good matches first, and then see if each angle was within the test data’s angles mean and a margin which was decided by that same angle’s standard deviation.

Screenshots

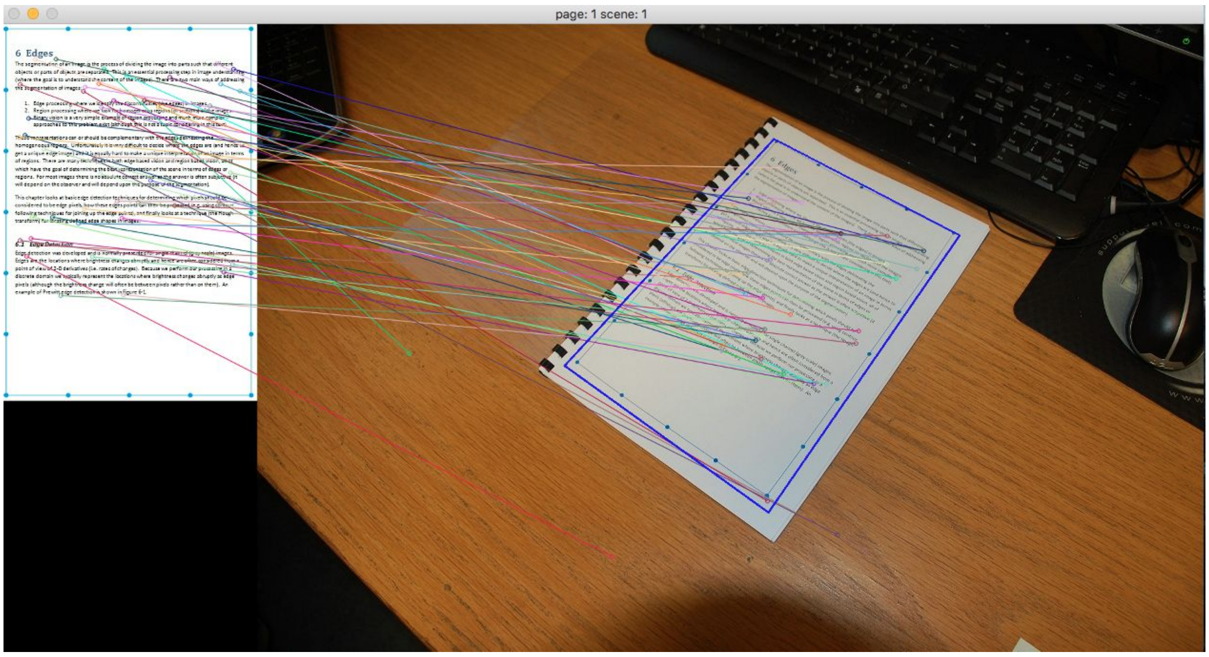

fig. 1 - correct page

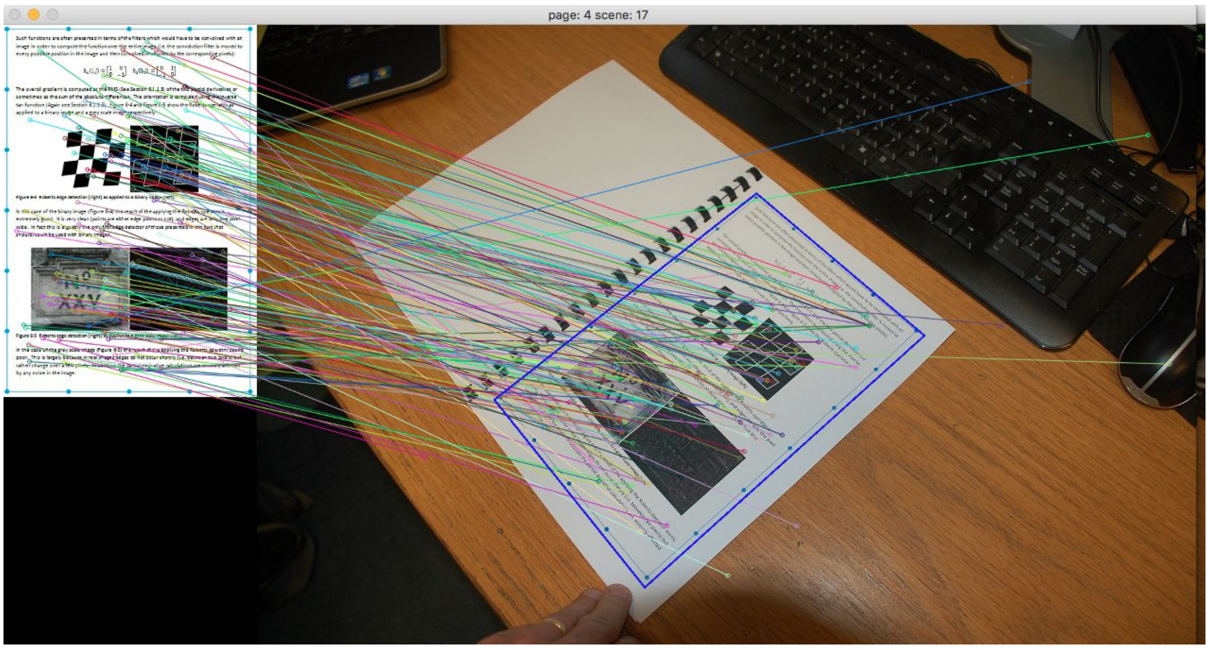

fig. 2 - correct page

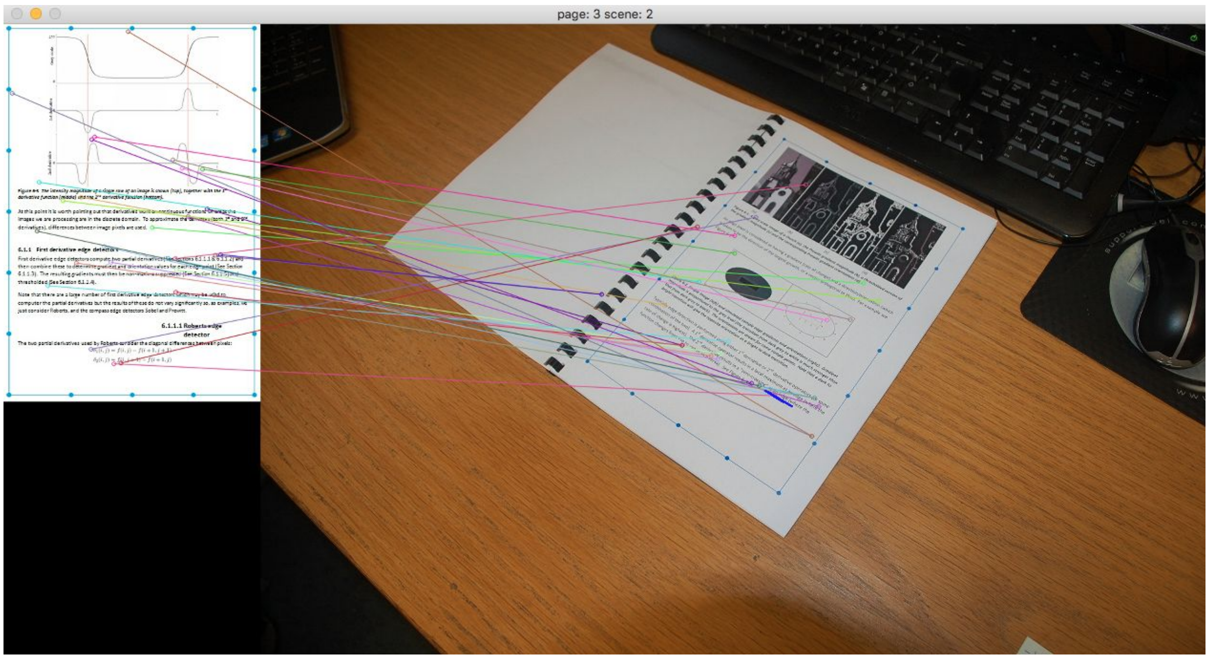

fig. 3 - incorrect page

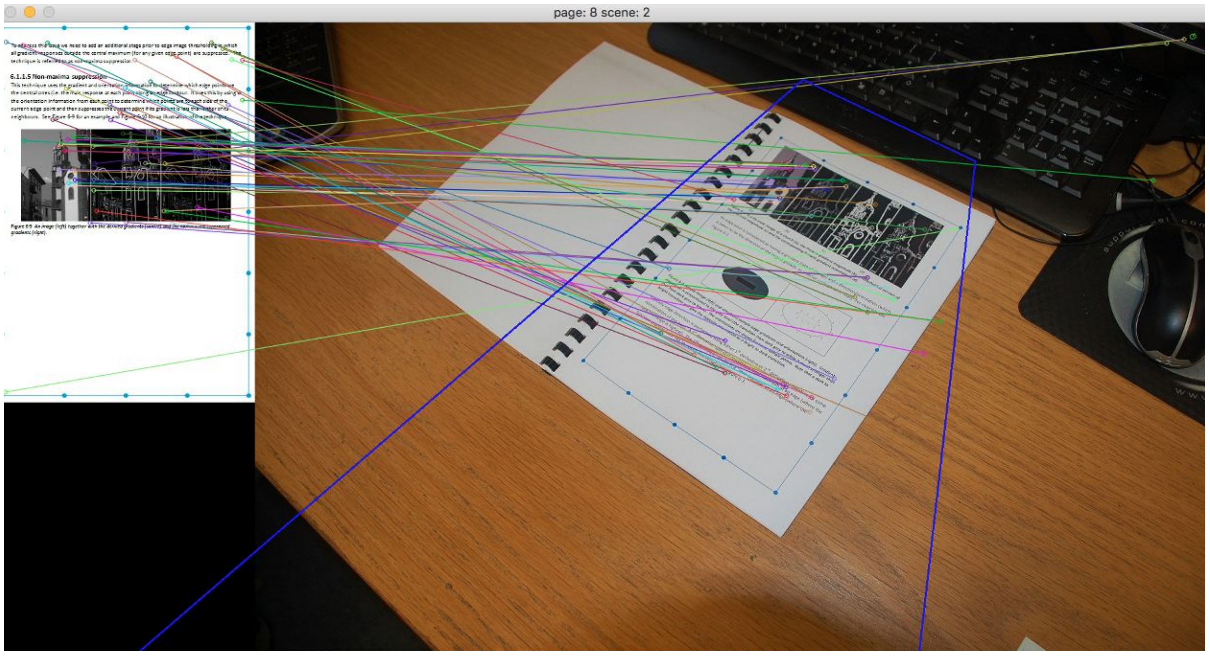

fig. 4 - incorrect page

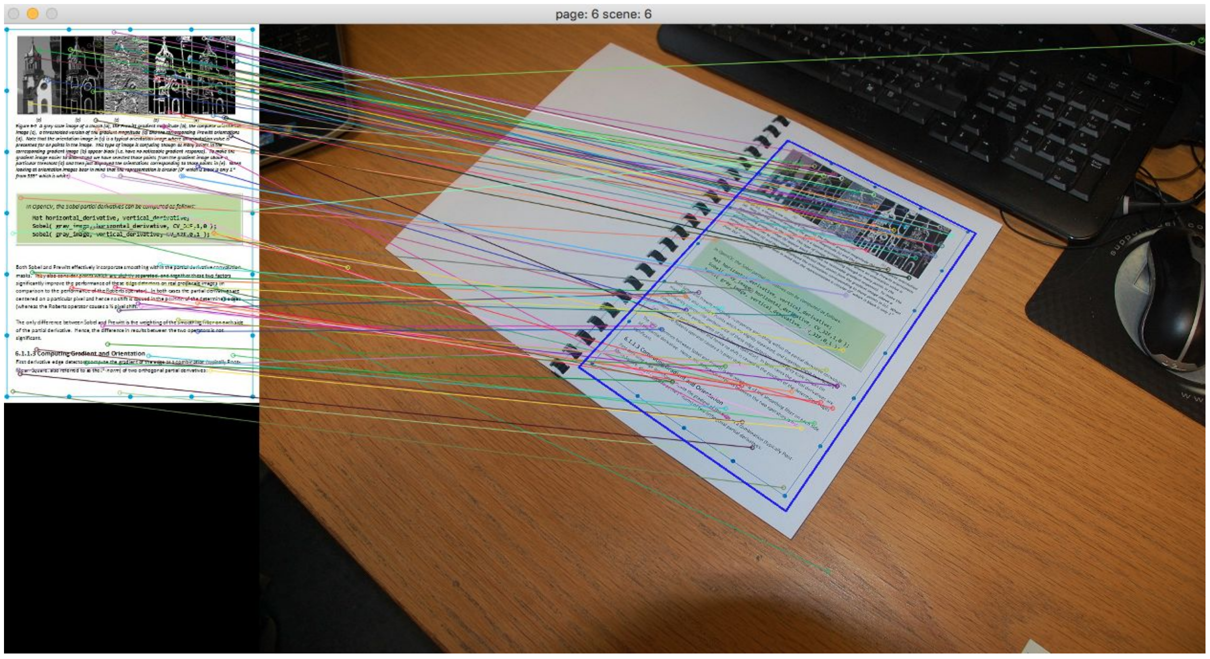

fig. 5 - correct page

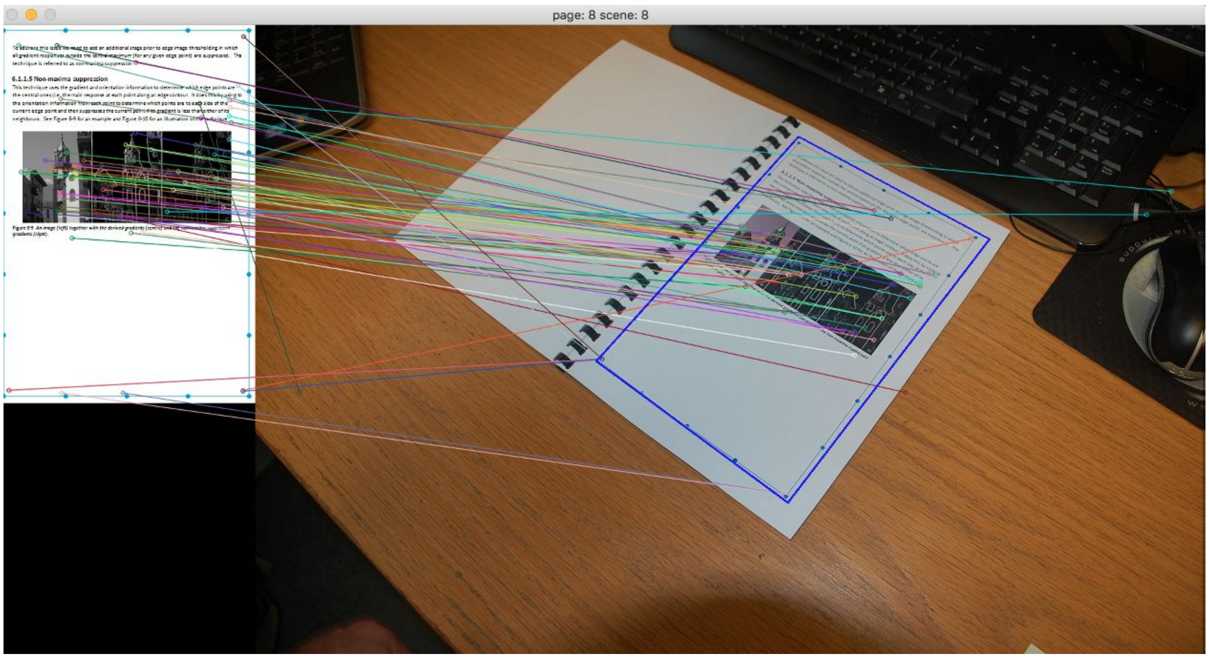

fig. 6 - correct page

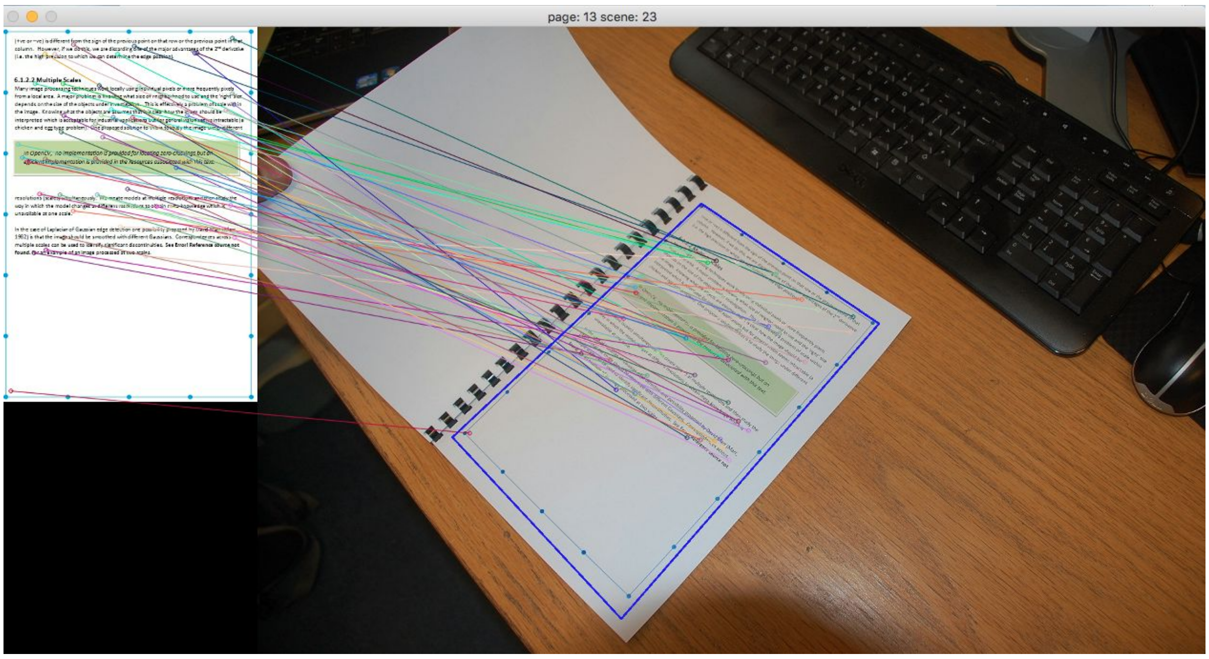

fig. 7 - correct page

Assumptions

- every scene had one of the pages from the set of pages we were given.
- Open pages in the scene are all at roughly the same angle.

Performance Metrics

25 total images were used in this program, where 8 of those were used as test data. therefore the remaining 17 can be tested on performance.

- True Positives(TP) - a page is visible and recognised correctly
    - Total: 17

- False Positive(FP) - an incorrectly recognised page, where EITHER no page was visible OR a different page was visible
    - Total: 0
- True Negative(TN) - no page visible and no page identified
    - Total: 0
- False Negative(FN) = a page is visible but no page was found
    - Total: 0

$$Recall = \frac{TP}{TP + FN} = 1$$

$$Precision = \frac{TP}{TP + FP} = 1$$

$$Accuracy = \frac{TP + TN}{TotalSamples} = 1$$

$$Specfcity = \frac{TN}{FP + TN} = 0$$

$$F_\beta = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall} = 1$$

Observations

- I found after performing this function with a few images that it was better to keep the hessian value low. While being more computationally intensive and thus taking more time, the increase in keypoints was essential in a program such as this, where the feature being found in the scene was very similar across all objects. The resulting feature in the scene gave me the correct object in 100% of the cases necessary to be tested - so it was ultimately a trade off between correctness and time complexity.
- I noticed that in rare cases the same input garnered some slightly different outputs. I think this may have something to do with how the features are computed in the SURF algorithm, but for the most part it was consistent.
- I realised after implementation that a much better way to decide the correct page would be to perform an affine transformation on the found feature in the scene, and then perform something like template matching on it between the transformation and the page, returning the highest value between all the pages. this would remove the assumption that open pages in the scene are all at roughly the same angle as the found feature would be transformed to a standardised orientation.