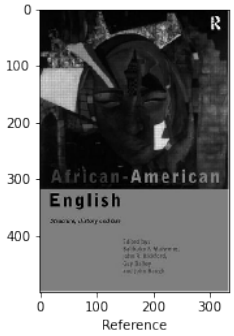
Assignment2

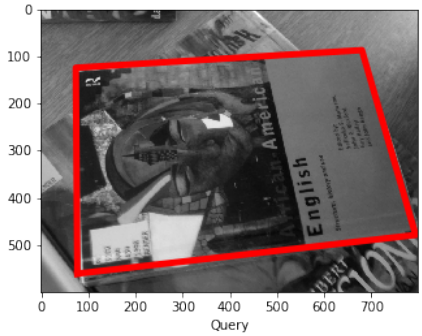
April 30, 2025

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

1 Question 1: Matching an object in a pair of images (60%)

In this question, the aim is to accurately locate a reference object in a query image, for example:





0. Download and read through the paper ORB: an efficient alternative to SIFT or SURF by Rublee et al. You don't need to understand all the details, but try to get an idea of how it works. ORB combines the FAST corner detector and the BRIEF descriptor. BRIEF is based on similar ideas to the SIFT descriptor we covered week 3, but with some changes for efficiency.

1. [Load images] Load the first (reference, query) image pair from the "book_covers" category using opencv (e.g. `img=cv2.imread()`). Check the parameter option in " cv2.imread()" to ensure that you read the gray scale image, since it is necessary for computing ORB features.

2. [Detect features] Create opencv ORB feature extractor by `orb=cv2.ORB_create()`. Then you can detect keypoints by `kp = orb.detect(img,None)`, and compute descriptors by `kp, des = orb.compute(img, kp)`. You need to do this for each image, and then you can use `cv2.drawKeypoints()` for visualization.

3. [Match features] As ORB is a binary feature, you need to use HAMMING distance for matching, e.g., `bf = cv2.BFMatcher(cv2.NORM_HAMMING)`. Then you are requried to do KNN matching (k=2) by using `bf.knnMatch()`. After that, you are required to use "ratio_test". Ratio test was used in SIFT to find good matches and was described in the lecture. By default, you can set `ratio=0.8`.

4. [Plot and analyze] You need to visualize the matches by using the `cv2.drawMatches()` function. Also you can change the ratio values, parameters in `cv2.ORB_create()`, and distance functions in `cv2.BFMatcher()`. Please discuss how these changes influence the match numbers.

Your explanation of what you have done, and your results, here

5. Estimate a homography transformation based on the matches, using `cv2.findHomography()`. Display the transformed outline of the first reference book cover image on the query image, to see how well they match.
    - We provide a function `draw_outline()` to help with the display, but you may need to edit it for your needs.
    - Try the 'least squre method' option to compute homography, and visualize the inliers by using `cv2.drawMatches()`. Explain your results.
    - Again, you don't need to compare results numerically at this stage. Comment on what you observe visually.

Your explanation of results here

Try the RANSAC option to compute homography. Change the RANSAC parameters, and explain your results. Print and analyze the inlier numbers.

Your explanation of what you have tried, and results here

6. Finally, try matching several different image pairs from the data provided, including at least one success and one failure case. For the failure case, test and explain what step in the feature matching has failed, and try to improve it. Display and discuss your findings.
    1. Hint 1: In general, the book covers should be the easiest to match, while the landmarks are the hardest.
    2. Hint 2: Explain why you chose each example shown, and what parameter settings were used.
    3. Hint 3: Possible failure points include the feature detector, the feature descriptor, the matching strategy, or a combination of these.

Your explanation of results here

2 Question 2: What am I looking at? (40%)

Your explanation of what you have done, and your results, here

5. Choose some extra query images of objects that do not occur in the reference dataset. Repeat step 4 with these images added to your query set. Accuracy is now measured by the percentage of query images correctly identified in the dataset, or correctly identified as not occurring in the dataset. Report how accuracy is altered by including these queries, and any changes you have made to improve performance.

Your explanation of results and any changes made here

---

6. Repeat step 4 and 5 for at least one other set of reference images from museum_paintings or landmarks, and compare the accuracy obtained. Analyse both your overall result and

individual image matches to diagnose where problems are occurring, and what you could do to improve performance. Test at least one of your proposed improvements and report its effect on accuracy.

Your description of what you have done, and explanation of results, here

2.1 Question 3: FUndametal Matrix, Epilines and Retrival (optional, assesed for granting up to 25% bonus marks for the A2)

In this question, the aim is to accuractely estimate the fundamental matrix given two views of a scene, visualise the corresponding epipolar lines and use the inlier count of fundametal matrix for retrival.

The steps are as follows:

1. Select two images of the same scene (query and reference) from the landmark dataset and find the matches as you have done in Question 1 (1.1-1.4).
2. Compute fundametal metrix with good matches (after appying ratio test) using the opencv function cv.findFundamentalMat(). Use both 8 point algorithm and RANSAC assisted 8 point algorithm to compute fundamental matrix.
3. Hint: You need minimum 8 matches to be able to use the function. Ignore pairs where 8 mathes are not found.
4. Visualise the epipolar lines for the matched features and analyse the results. You can use openCV function cv.computeCorrespondEpilines() to estimate the epilines. We have provided the code for drawing these epilines in function drawlines() that you can modify as required.

Your visualization for epilines goes here

---

4. Repeat the steps for some examples from the landmarks datasets.

Your visualization for additional epillines goes here

---

5. Find a query from landmarks data for which the retrival in Q2 failed. Attempt the retrival with replacing the Homography + RANSAC method of Q2 to Fundamental Matrix + RANSAC method using code written above. Does the change of the model makes retraival successful? Analyse and comment.

Your analysis goes here
