Computer Vision Assignment 3 (CLab3) S1-2024

0 Overview

This assignment has **two** tasks in total, is graded out of **25** marks, and is worth **15%** of your final mark for this course.

0.1 Objectives

The goal of this assignment is to develop and assess proficiency at model fitting, including the Hough transform, and two-view homography estimation.

0.2 Permitted Python libraries

The Python libraries that you may use in this assignment are

- OpenCV (cv2);
- NumPy;
- Matplotlib;
- scikit-image (skimage);
- scikit-learn (sklearn);
- SciPy;
- Pillow (PIL); and
- os.

Use of other Python libraries will result in a score of 0 for the task in which the library was used.

0.3 Advice

1. Before writing your report, we recommend you watch the video "how to write a good lab report" on Wattle. The markers do not want to just see a collection of experimental results, but rather a coherent explanation and interpretation of the results, and key parts of your source code with detailed comments. Note that these are suggestions from a previous version of the course, not all of which apply to this assignment. In particular, they do not override any requirements in this document.
2. The requirements for submission are at the end of this document. Please ensure your submission meets the requirements.
3. The report is to be uploaded to the Wattle site before the due time. This course does not allow late submissions. That is, any submission after the deadline will receive a **mark of zero**. Therefore, it is important to factor upload time into your submission schedule and *submit early*; you can always resubmit an updated version later.
4. This is an individual assignment. All students must work individually when coding and writing the report.

0.4 Academic Integrity

You are expected to comply with the university policy on academic integrity and plagiarism. Please ensure you cite appropriately any resources that you use (lecture notes, papers, online documents, code), and complete all tasks independently. All academic integrity violations will be reported and can result in significant penalties. More importantly, working through the problems yourself will help you learn the material and will set you up for success in the final exam.

1 Task 1: Model Fitting with the Hough Circle Transform (10 marks)

In this task, you will use a Hough transform to detect circles in images. A circle centred at (a, b) with radius r can be parameterised by

$$\boxed{\begin{aligned} x &= a + r\cos\theta \\ y &= b + r\sin\theta \quad \forall\theta \in [0, 2\pi). \end{aligned}} \qquad \begin{aligned}(1)\\(2)\end{aligned}$$

Therefore, every point (x, y) that lies on a circle is associated with the circle parameters (a, b) via the following relationship

$$\boxed{\begin{aligned} a &= x - r\cos\theta \\ b &= y - r\sin\theta \quad \forall\theta. \end{aligned}} \qquad \begin{aligned}(3)\\(4)\end{aligned}$$

Since r is also an unknown, you will need to iterate over possible radius values within a range, i.e.,

$$\boxed{\begin{aligned} a &= x - r\cos\theta \\ b &= y - r\sin\theta \quad \forall\theta\ \forall r \in [r_{\text{min}}, r_{\text{max}}]. \end{aligned}} \qquad \begin{aligned}(5)\\(6)\end{aligned}$$

Each image point corresponds to a cone in the (a, b, r) parameter space; any image point could lie on an infinite number of different circles. Theoretically, this could generate infinitely many Hough votes. If several points are from the same circle, their cones will intersect somewhere in (a, b, r) space. The coordinates of the intersection represent the centre and the radius of the circle. Thus, to detect a circle using the Hough transform, we need to find locations in the (a, b, r) space with a large number of votes. Two images containing many circles are provided in Figure 1.

You need to write your code for Hough circle transform from scratch and apply your function to both images. You are not allowed to use any library function to detect circles in images.

1. Read in the RGB images `smarties.png` and `coins.png` as NumPy arrays and convert the RGB images into greyscale.
   (0 marks)
2. Apply the Canny edge detector on the two greyscale images and display the result in your report. You may need to try different thresholds for the detector to find the best setting. You may use OpenCV library functions for this question.
   (0.5 marks)
3. Design and implement a Hough transform function to identify the parameters of every circle within an image, adhering to the methodology outlined above. Your `hough_circle_transform` function should return a list of circle candidates, ordered from the candidate with the highest votes to the candidate with the lowest votes (above a threshold, see below), where each candidate is represented as a list of the form of $[a, b, r]$. Include the code listing for this function, with detailed comments, in your report.
   (3 marks)
   Hints:
   - Use an accumulator to store the votes from each pixel.
   - Iterate through each pixel in the edge-detected image, and for each edge pixel, increment the corresponding accumulator cells for all possible circle centres and radii.
   - Set an appropriate threshold value for the accumulator to filter out potential false positives and identify the most likely circles. The threshold could be, e.g., relative to the ratio of the number of votes in a cell to the maximum number of votes in a cell.
   - The `hough_circle_transform` function might include the following input parameters: input image, radius range, radius increment, circle centre increment, angle (theta) increment, and the accumulator threshold, for instance.
4. Design and implement a non-maximum suppression algorithm and use it to filter the list of circle candidates. Include the code listing for this function, with detailed comments, in your report.
   (2 marks)
   Hint: To perform non-maximum suppression, you can set a threshold in units of pixels and perform a pairwise comparison of all circle candidates, checking whether the absolute difference between a, b or r exceeds the designated threshold. When selecting which candidates to

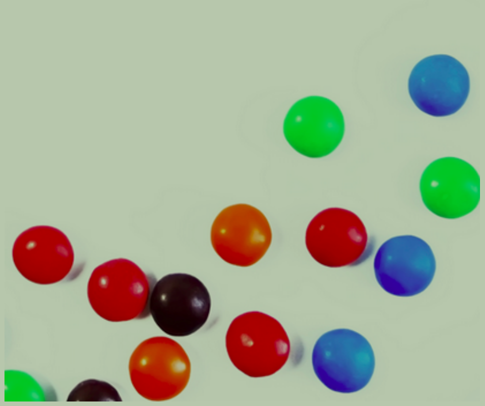
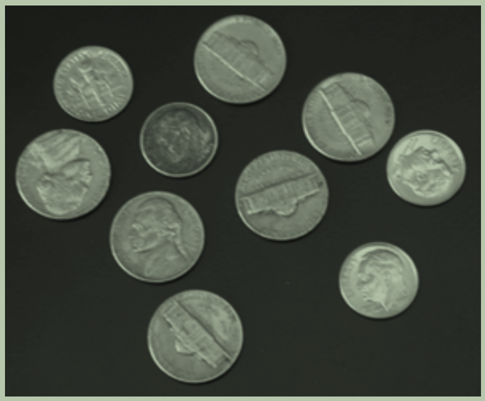

Figure 1: Images for Task 1.

keep, prefer those with a higher number of votes; i.e., select the local maxima. Depending on your design, this function may be called from inside or outside `hough_circle_transform`.
Commentary: Circle candidates from the previously designed Hough transform function may include many neighbouring (in parameter space) circles. Consequently, non-maximum suppression is essential to eliminate these duplicates.

5. Display the original images with the circles derived from the previous question superimposed. You may use the circle function from OpenCV.
(0.5 marks)

6. Analyse the images displayed in Part 5 above. Assess the algorithm's success at detecting all circles, reporting the algorithm's accuracy and its false positive and false negative rates. Provide insight into the effectiveness or shortcomings of the results.
(1 mark)
Hint: Consider how you might assess the algorithm's success qualitatively and quantitatively. What is your ground-truth and how did you obtain it?

7. Examine the impact of different parameters of the Hough circle transform (e.g., radius range, radius increment, circle centre increment, angle (theta) increment, and the thresholds). Provide all necessary results, observations, and analysis in this response.
(1 mark)

8. Report the time and memory efficiency of your `hough_circle_transform` and non-maximum suppression functions. Two marks are allocated for the accuracy and efficiency of your functions: better designs will receive higher marks.
(2 marks)
Hint: Consider algorithmic asymptotic analysis and the actual program runtime.

2 Task 2: Two-view Homography Estimation (10 marks)

A transformation from the homogeneous projective space P^3 to itself is called a homography. A homography can be represented by a 3×3 matrix with 8 degrees of freedom, being invariant to scale. Applying a homography therefore gives

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{7}$$

The goal of this task is to use the Direct Linear Transformation (DLT) algorithm to estimate a 3×3 homography matrix. You are given two images of a mountain as shown in Figure 2. Calculate the 3×3 homography matrix between the two images, from at least 4 corresponding points in both images, using the DLT algorithm. You are required to implement a function with the following syntax.

```
H = homography(u1, v1, u2, v2)

Usage:
  Computes the homography H using the Direct Linear Transformation

Arguments:
  u1, v1: normalised (u,v) coordinates from image 1
  u2, v2: normalised (u,v) coordinates from image 2

Output:
  H: the 3×3 homography matrix that warps normalised coordinates from
     image 1 into normalised coordinates from image 2
```

1. Read in the RGB images `mountain1.jpg` and `mountain2.jpg` as NumPy arrays.
   (0 marks)
2. Implement the homography function with the interface described above, which takes normalised coordinates as inputs. Include the code listing for this function, with detailed comments, in your report.
   (2 marks)
3. Implement a function `compute_normalisation_matrix` to generate normalisation transformation matrices for homogeneous point coordinates (refer to lecture slides). Implement a function `homography_w_normalisation` that integrates this normalisation step with the homography estimation function above. The output homography matrix H should transform *unnormalised* points from image 1 to image 2. Include the code listing for these functions, with detailed comments, in your report.
   (2 marks)
   Hint: You may need to include both input images in your function input (`img_1` and `img_2`).
4. Use the SIFT keypoint detector and feature descriptor to find matching feature points across both images. Sample half of these corresponding points ($\lceil \frac{N}{2} \rceil$), using a fixed random seed of 42, convert the coordinates into NumPy arrays, and store them as .npy files. You may use OpenCV functions for this task.
   (1 mark)
5. Implement a function `homography_w_normalisation_ransac` that estimates a homography with normalisation using the RANSAC algorithm. Apply the function to the point correspondences obtained from the previous question and report the 3×3 homography matrix H that is estimated by the function. Include the code listing for this function, with detailed comments, in your report.
   (2 marks)
   Hint: In `homography_w_normalisation_ransac`, other than function parameters for normalised homography estimation, you may need to include additional parameters for the RANSAC algorithm, e.g., distance threshold, inlier threshold, maximum iterations, etc. You may use functions implemented in previous questions.

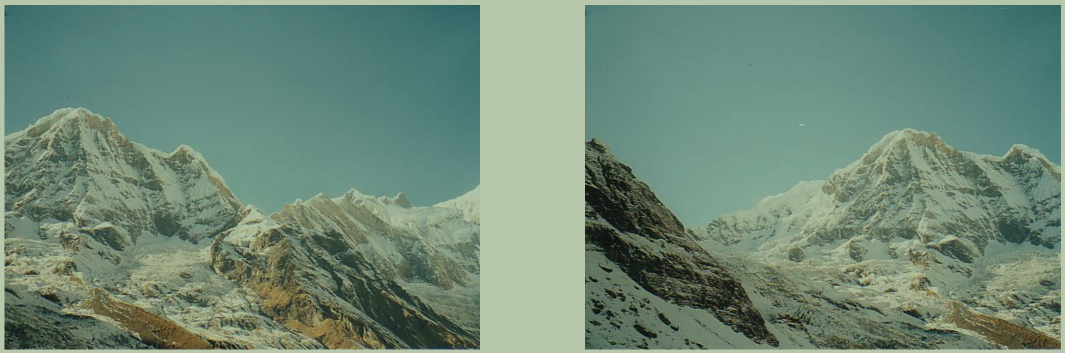

Figure 2: Images for Task 2.

Commentary: The correspondences obtained from the SIFT features may contain errors (outliers). The RANSAC algorithm can be used to fit a homography model that is robust to these outliers, resulting in a better estimate.

6. Warp the `mountain1.jpg` image according to the calculated homography matrix and stitch the warped image with the `mountain2.jpg` image for a more complete view of the mountain. Include the code listing for this warping function, with detailed comments, in your report.
   (1 mark)

7. Discuss the factors that affect the rectified results theoretically and empirically.
   (2 marks)
   Hint: To obtain ground-truth correspondences, you may want to manually choose matching points from images and use the `homography_w_normalisation` function to calculate the homography matrix. For more details on how to manually pick points from images, please read the ginput documentation.

3 Submission Quality (5 marks)

As well as its correctness (the “what”), your report is also judged on its presentation quality (the “how”). Marks are allocated for:

- Clear and concise explanation and interpretation, and high-quality presentation (2 marks)
- Correct report formatting (2 marks)
- Code quality, including clarity and commenting (1 mark)

4 Submission Requirements

4.1 Files

Upload a **PDF file of your report** and a **ZIP file of your code** by the due date. You must use the following file names: **uXXXXXXX.pdf** and **uXXXXXXX.zip**, replacing uXXXXXXX with your university ID. Your ZIP file must contain all your *.py and/or *.ipynb files.

Important: submit your report (PDF) and code (ZIP file) **separately** on the **correct tab** of the submission box.

4.2 Report

At the top of the first page, include the assignment title, your name and your university ID. The report may be written in LaTeX or other word processing software, but must be exported to PDF. Handwritten submissions are not permitted. *Concise explanations on how you solve the tasks are expected, including any assumptions you used. Concise description and interpretation of the results are also expected.*

4.2.1 Formatting Instructions.

Use:

1. numbered headings, as given in this document;
2. numbered answers, corresponding to those in this document;
3. Times New Roman font;
4. single-spaced font;
5. 12pt font for the body text;
6. references at the end, where needed; and
7. images where appropriate to explain your answers.

If not responding to a particular question or task (and so getting zero marks for that part), still include the associated heading and question number, but leave the content blank. For example, your report might look like:

1 Task 1: Basic Image I/O

1.1 Question 1

< Blank >

1.2 Question 2

Documentation, observations, results, analysis, etc.

. . . etc.

7.2.2 Figures and Tables.

These are critical for communicating your results clearly. Always refer to these from the body text of the report (e.g., “As shown in Figure 2, the results indicate that. . . ”). Use:

1. descriptive captions;
2. axis labels;
3. a legend;
4. appropriate axis scaling (e.g., perhaps a log scale is more informative?);
5. appropriate annotations;

6. 10pt Times New Roman font for captions;
7. a sufficient size for visibility; and
8. your own material, or public-domain and cited images only.