

Mid-Level Computer Vision with MATLAB

1. Introduction

This coursework provides the opportunity to design and implement a computer vision systems that can solve the correspondence problem. You will need to work on this project in your own time. If you require any clarification to the instructions please use the "Questions on Lab 2" forum on KEATS.

2. The Task

At the end of these instructions are 10 pairs of images. Each pair of images shows the same scene from a different viewpoint and/or under different lighting conditions or taken with different camera settings. Given the coordinates of locations in the first image of the pair, your code needs to find the corresponding locations in the second image of the pair. A zip file containing the training image pairs is available from the module's KEATS webpage. You can use these images in any way you wish, but they are provided primarily to allow you to develop your code and evaluate its performance.

3. System Requirements

Your code must include a m-file called "find_matches.m". The main function in this file should have the following definition:

```
function [pos2]=find_matches(I1,pos1,I2)
```

Hence, this function takes three input arguments:

- **I1** is a MATLAB variable storing a colour image (i.e. **I1** will be a 3-dimensional matrix that has been produced using a command such as: **I1=im2double(imread('bikes/img1.png'));**). This variable contains the first image in the pair.
- **pos1** is a n-by-2 matrix containing the coordinates of n points in the first image. These points will have been selected using an interest point detector. For example, using commands such as: **detector = detectHarrisFeatures(rgb2gray(I1)); pos1=detector.Location;**. To visualise the locations that need to be matched, you could use the following code: **imagesc(rgb2gray(I1)); plot(pos(:,1),pos1(:,2), 'y+', 'LineWidth',2);**
- **I2** is a MATLAB variable storing a colour image (i.e. **I2** will be a 3-dimensional matrix that has been produced using a command such as: **I2=im2double(imread('bikes/img2.png'));**). This variable contains the second image in the pair.

The single output variable, **pos2**, should be the same format and size as **pos1**. Each row of **pos2** should define the coordinates of the point in **I2** that corresponds to the point in **I1** defined by the coordinates in the corresponding row of **pos1**. In other words, **pos2** must be ordered so that corresponding rows of **pos1** and **pos2** correspond to the the coordinates of the same point in 3D space.

The **find_matches** function may call other functions. Unless they are official MATLAB functions, these additional functions should be included in the same m-file, or be include in the zip file you submit. Your code can also load data, as long as this data is included in the zip file you submit. Your code should run on a PC and not need access to special hardware, such as a GPU.

4. Submission

You should submit a single m-file, or a single zip file containing your code and any additional files that it requires to execute. If submitting a zip file you must ensure that when this file is unzipped that the **find_matches.m** file appears in the directory in which the zip file was unzipped, and not in a sub-directory.

Together with your code you should also submit a one paragraph description of the method you have used. This can be in the form of a separate pdf document or a block of comments at the top of your **find_matches.m** file. Within this description you must clearly acknowledge any code that you have copied or modified from online sources, books, or articles, and include full details of the source of this code by providing a URL or a citation. Re-using open-source software is good practice and demonstrates your research skills, however, not acknowledging other people's work is both intellectual property theft and plagiarism. Any submission that is identified to include other people's code which has not been clearly referenced will be reported for misconduct. You may **not** share code with fellow students, even if this is acknowledged. You can use any built-in MATLAB function or any official MATLAB toolbox without acknowledgement.

You need to submit through KEATS by the deadline specified on the module's KEATS webpage.

5. Terms and Conditions

Distribution or sharing of your submitted code, or any code derived from it, is forbidden. By submitting your coursework you agree to abide by these terms and conditions.

6. Assessment

The performance of your code will be assessed by testing the accuracy with which it finds corresponding points in 10 testing image pairs. The testing images are distinct from the training images given below, but like the training images contain pairs of images that differ from each other due to changes in viewpoint, lighting conditions or camera settings.

Marks will be awarded primarily for how accurately your code solves the correspondence problem. In addition, credit may be given for how sensible an approach you have chosen to implement and how efficiently and correctly that approach has been implemented. Marks will be deducted if your code needs to be fixed in any way to make it execute.

7. Training Images Pairs



