Robot Vision [06-25024]
Summative Assignment 2

May 3, 2023

Submission deadline:	**12:00pm (BST), Monday, 10 May 2023**
Instructor:	Dr. Hyung Jin Chang Prof. Hamid Deghahni Dr. Jianbo Jiao
Total marks:	100
Contribution to overall module mark:	25%
Submission Method:	This assignment must be submitted through Canvas.

Important note on plagiarism

Plagiarism can have very serious potential consequences. Please see `https://canvas.bham.ac.uk/courses/47172/pages/computer-science-programme-information?module_item_id=1648181` for further information. Plagiarism includes failure to provide proper attribution for ideas originating from external sources, and copying from other students or external sources. Changing the wording of copied text is still considered plagiarism. It is acceptable, and even encouraged, to discuss course content and assignments with your fellow students. However, you must write your answers to all assignments individually and you must not share those answers with other students.

Instructions (Please read carefully!)

This assessment is summative and contains two parts. In Part 1, you will carry out object/scene reconstruction with Structure from Motion. In Part 2, you will use Deep Learning Toolbox to perform image classification and image regression tasks with a model trained by yourself, a pre-trained model, and a fine-tuned model.

Your answer must be submitted to Canvas before the deadline in the form of a single zip archive file containing:

1. Your answers to the questions in prose and diagrams. This should take the form of a single PDF document with the answers for each question using the provided LaTeX template.
2. Your code and any accompanying files necessary to execute the code for any programming questions as specified in the LaTeX template.

and a separate PDF document with the answers for Turnitin checking (two files in total; one zip file and one PDF file). Some or all of the text of each question is emphasised using *italics*. This emphasis indicates a question that must be explicitly answered or a task that must be completed.

Part 1

Question 1.1 A panorama is formed by stitching together multiple images into one seamless image. In this task, you will need to implement *Feature Based Panoramic Image Stitching* in MATLAB.

Question 1.1.1 [10 marks] Three images of the Aston Webb building have been provided. The following steps need to be taken in order to create the panorama:

1. Use any preprocessing you like to manipulate the given images
2. Detect and match features between image I(n) and I(n - 1).
3. Estimate the geometric transformation, T(n), that maps I(n) to I(n - 1).
4. Compute the transformation that maps I(n) into the panorama image as T(1) * T(2) * ... * T(n - 1) * T(n).
5. Utilise the imWarp function to align the images into the panorama.

A guide on this process can be found here: https://uk.mathworks.com/help/vision/ug/feature-based-panoramic-image-stitching.html

Your solution to this task should include:

1. *Figure showing undistorted input images (report in PDF)*
2. *Figure showing complete panorama (report in PDF)*
3. *A written explanation of the steps taken in the report, stating which functions you used, why you used them and a short explanation of how they work. (report in PDF)*
4. *The calibration file (.mat file)*
5. *Code for Task 1.1.1 (.mat file)*
6. *All images needed for the code to function*

Question 1.1.2 [8 marks] The panorama which has been produced is not a uniform shape. Write an algorithm from scratch that iteratively crops the image so that no black areas are included. Your algorithm should preserve as much of the non-black areas of the image as possible and work with any provided panorama. See Fig 1 for an example of the expected result.

Your solution should include:

1. *A figure showing the original panorama overlaid with lines representing the cropped area (report in PDF)*
2. *The cropped panorama (report in PDF)*
3. *An explanation of your algorithm (report in PDF)*
4. *Code for Task 1.1.2*
5. *All files needed for Task 1.2 to function*

(a) 1.1.1: Input Images

(b) 1.1.1: Output Panorama

(c) 1.1.2: Cropped Panorama

Figure 1: Figures for Question 1.1

Question 1.2 Structure from Motion involves reconstructing the 3D structure of a scene from multiple images. Given the size of an object in the scene, it is possible to recover the actual sizes of everything in the scene given sufficient image quality.

Question 1.2.1 [10 marks] The aim of this task is to recreate a dense reconstruction of objects from two images. Two images of a tea canister are provided. The aim of the first part of this question is to reconstruct a 3D object from the images producing a point cloud.

Your solution to this part should:

1. Use any pre-processing you like to manipulate the given images
2. A series of chessboard images are provided. Use them to generate a calibration file for the following questions using the camera calibrator app in MATLAB. The chessboard squares are 20mm a side. Use this file to remove the distortion from the given images.
3. Detect features using the minimum eigenvalue algorithm
4. Use a point tracker to track the matched features
5. Estimate the fundamental matrix from the matched points with any method, such as MSAC or RANSAC *etc.*, and check its validity
6. Triangulate over the matched points
7. Construct and display a point cloud
8. Obtain the colour values for the point cloud
9. The reconstruction has to be a dense reconstruction

Question 1.2.2 [7 marks] This question expands on task 1.2.1. Fit a cylinder to the previously generated point cloud and scale the point cloud according to the metric.

The can has a radius of 3.5cm. Include the following figures in your report: 1 view of the fitting of the cylinder, 1 view of the unfitted reconstruction, and an image of the tracked features.

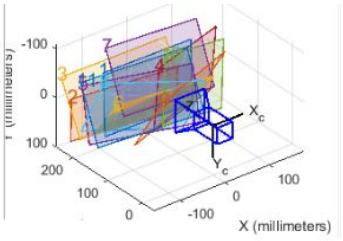
Your report should also include an estimate of the cans height and how this metric was formed. Due to variance in many of the MATLAB methods, you may need to run your code several times to get a good reconstruction, this is fine, however make sure that you save and include the metric fitted MATLAB reconstruction as a .fig file in your submission. When running the MATLAB file, these images should all appear as figures and the reconstruction as a movable graph.

Your solution should include:

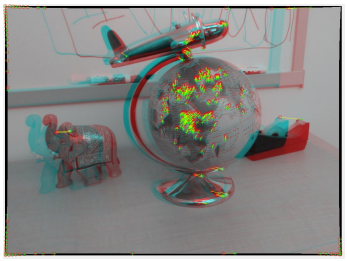
1. *Your code for Task 2 including all files needed for the code to run*
2. *Image displaying tracked features (report in PDF)*
3. *Unfitted reconstruction image (report in PDF)*
4. *Cylinder fitting image (report in PDF)*
5. *3 images of the metric fitted reconstruction (report in PDF)*
6. *Estimate of the can's height (report in PDF)*
7. *How you came to the estimate (report in PDF)*
8. *The reconstruction (.fig files)*

(a) Example Image 1

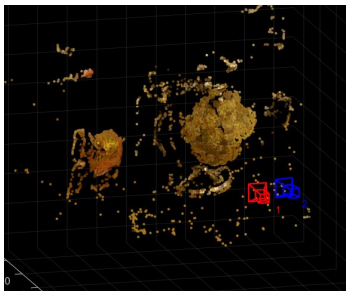
(b) Example Image 2



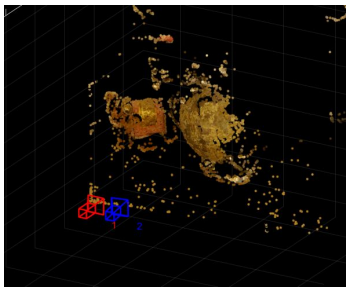
(c) Example camera calibration image



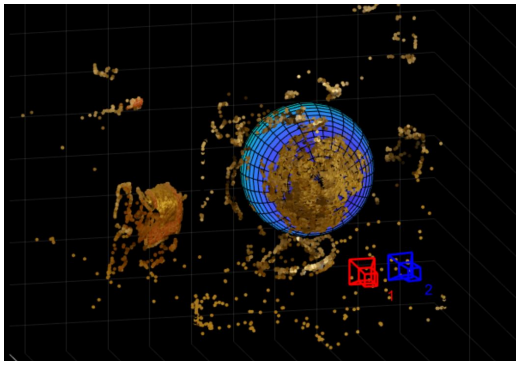
(d) Tracked Features



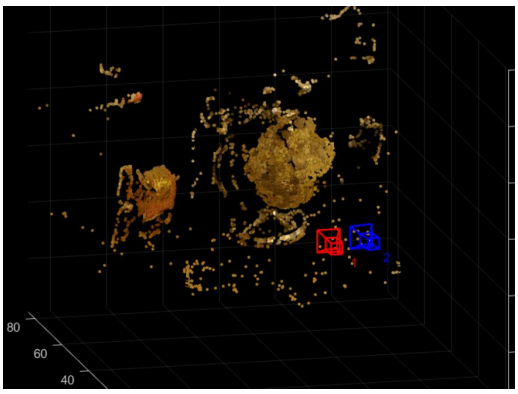
(e) Example unfitted reconstruction 1



(f) Example unfitted reconstruction 2



(g) Example Fitted Globe



(h) Example Metric Point Cloud

Figure 2: Figures for Question 1.2

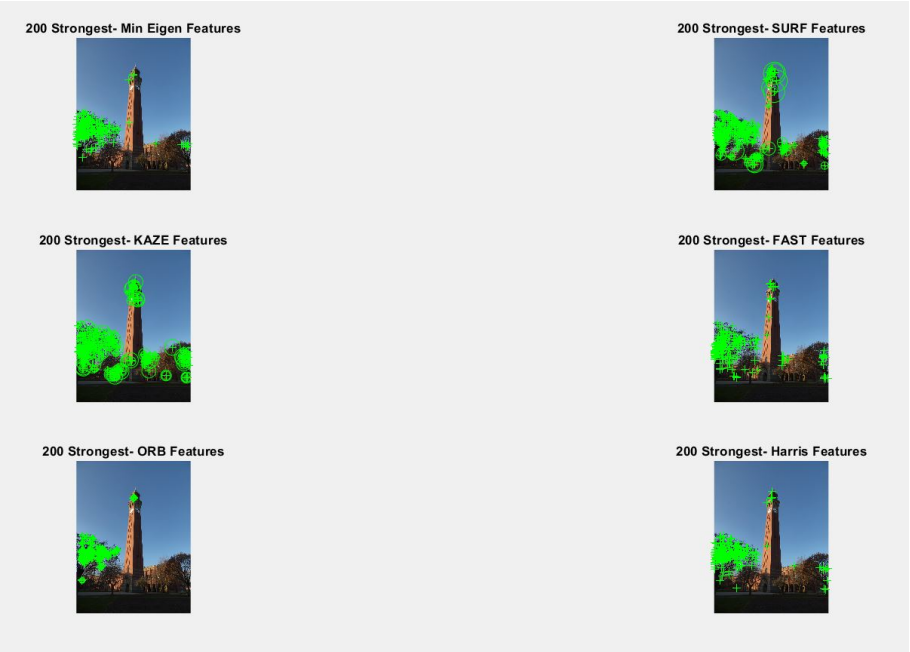
Question 1.3 Feature detection plays an important role in many computer vision tasks. In this question, we will be exploring and evaluating the different feature detection methods.

Question 1.3.1 [9 marks] An image has been provided. Plot on a 3 x 2 sub-plotted figure, the 200 strongest features when using the Minimum Eigenvalue, SURF, KAZE, FAST, ORB and the Harris-Stpehens algorithms. Include this sub-figure in your report and ensure that when the MATLAB files execute, it appears as a sub-figure.

Your solution should include:

1. *Your code for Task 2 and all files needed for the code to run*
2. *The generated subplot figure (report in PDF)*
3. *The generated subplot figure (.fig file)*

Question 1.3.2 [6 marks] Describe briefly 3 of the 6 feature detection methods previously explored. Give a brief overview of each of these methods and how they differ from each other. Discuss how these differences are represented in the results from Question 1.3.1. Include the answer to this section in your report.



(a) Expected results of Question 1.3

Part 2

Question 2.1 [16 marks]

Carry out a classification task for a subset of CIFAR-10/100 data using the Deep Learning Toolbox. The data and images needed for the task are in the 2_1 folder. Follow the instructions below to complete this task.
Submit your code as ***username_assignment2_part2_Q2_1.m.***

1. Load the images by unzipping **CIFAR-10.zip**. In total, there are 2500 images of size 32×32 and 10 categories in this dataset. The training data and test data for each category of images are stored in `/train` and `/val` folders, respectively. In the training dataset, there are 200 images in each category, whereas there are 50 images per category in the test dataset. The category of each image is given by its folder name. Simplified folder structure is shown in figure 3.

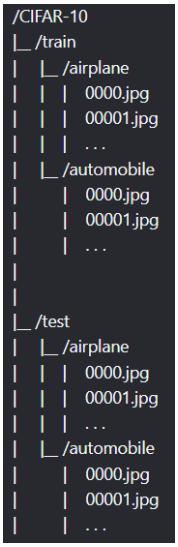


Figure 3: Simplified file structure of `imagenet.zip`.

2. Perform transfer learning using MATLAB's built-in model vgg16 with the data in the `/train` folder. Design appropriate structure to feed the data to the model. Adapt the model to classify only the 10 categories from the dataset. Use the stochastic gradient descent with momentum optimizer, set the mini batch size to 128, the initial learning rate to 0.01, validation frequency to 30, validation patience to 20, and shuffle the training data before each training epoch. Plot the training progress and include it into your written report. Save and submit the trained model into a `.MAT` file named `2-1_10.mat`.

3. Use the data in the `/test` folder, which was not used to train the mdoel, to test the performance of the trained model. Store the cross-entropy loss and accuracy in a table. Besides, use another variable to store the worst classified category. Save and submit them into a `.MAT` file named `2-1_10output.mat`. An example of the output is shown in figure 5.

   The formula of the average cross-entropy loss is given by

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i = \frac{1}{N}\sum_{i=1}^{N} -\sum_{c=1}^{M} y_{i,c} \cdot \log(p_{i,c})$$

   where

$$y_{i,c} = \begin{cases} 1, & \text{if } c \text{ is the correct classification for } i \\ 0, & \text{otherwise} \end{cases}$$

   i is image, N is the number of images, c is category, M is the number of categories, p is the predicted probability i is of c.
   The accuracy is given by

$$A = \frac{\text{count of correctly classified items}}{\text{count of all items}}$$

4. Load the images by unzipping **CIFAR-100.zip**. In total, there are 250 images of a category of bicycle of size 32×32. There are 200 and 50 images of such category in `/train` and `/val` folders, respectively. Modify the model trained in the previous task and retrain it using the data in the `/train` folder, allowing it to classify 11 categories (using the same training options as in the previous task). Plot the training progress and include it into your written report. Save and submit the retrained model into a `.MAT` file named `2-1_11.mat`.

5. Merge the data in `/CIFAR-10/test` folder and `/CIFAR-100/test` folder. Run the re-trained model on the merged test dataset. Again, save the cross-entropy loss, accuracy, and the worst classified category in the format shown in Figure 4. Save and submit them into a `.MAT` file named `2-1_11output.mat`.
6. `SF.JPG` is a 512×512 image obtained by Stable Diffusion[1]. Resize this image and use `2-1_11output.mat` to classify this image. Save and submit the prediceted result, the two highest predicted probabilities and their corresponding categories into a `.MAT` file named `2-1_SFoutput.mat`. An example of the output is shown in figure 5.

```
cross-entropy: 0.0000
accuracy: 1.00
worst catagory: ********
```

Figure 4: Example output of `2-1_11output.mat`.

```
predicted results: ********
********: 1.00
********: 1.00
```

Figure 5: Example output of `2-1_SFoutput.mat`.

Question 2.2 [16 marks] *Train a network using the MNIST handwritten digit database*[2] *for classification. The task can be solved using a CPU only execution environment. The data needed for the task are in the 2_2 folder.*
Submit your code as ***username_assignment2_part2_Q2_2.m****.*

1. The four files in the `/MNIST` folder are part of the MNIST dataset, including:
   (a) `train-images-idx3-ubyte` contains 60000 28×28 training images of handwritten digits.
   (b) `t10k-images-idx3-ubyte` contains 10000 28×28 handwritten digit test images.
   (c) `train-labels-idx1-ubyte` contains the labels of 60000 training images, i.e., the numbers they represent.
   (d) `t10k-labels-idx1-ubyte` contains the labels of 10000 test images, that is, the numbers they represent.

   Show 30 random example images (24 training images in `train-images-idx3-ubyte` and 6 validation images in `t10k-images-idx3-ubyte`) in a 5*6 subplot. Count the number of labels per each category and include the result into your report.
2. Augment the 24 randomly selected training images from the previous task. Use the imageDataAugmenter function with random rotation [-20, 30], using random reflection along the top-bottom direction and in the left-right direction, as well as using random X translation [-3, 2] and Y translation [-2, 4]. Show example images of the augmented data and include it into your report.
3. Define {1,3,5,7,9} as odd numbers and {0,2,4,6,8} as even numbers. Perform transfer learning to classify the odd and even numbers of the MNIST dataset. Use the augmented 24 images as the training dataset. Implement the network architecture shown in Figure 6. The input image size is 28*28, pool size and stride are 2. Use Adam optimizer, set the initial learning rate to 0.02, validation frequency to 50, validation patience to 20, and shuffle the training data before each training epoch (train the network for 35 epochs). Plot the training progress and include it into your written report. Save the trained model named `2-2_OE.mat` and submit it with your report. What is the accuracy of the model using 6 selected validation images?

[1] https://en.wikipedia.org/wiki/Stable_Diffusion
[2] `https://en.wikipedia.org/wiki/MNIST_database`

4. Create your own test data (3 sets of digits handwritten by you, total 30 images, try to make them look as different as possible. You are free to use any device or software (*e.g.* Microsoft paint) to create the test data.), include it into your report as in Figure 7 and report the accuracy of your model on the test images. The test images must be submitted with your code, otherwise this part of the task will not be graded. Analyse your network with the built-in **analyzeNetwork** MATLAB function and include the analysis results as a figure into your report.

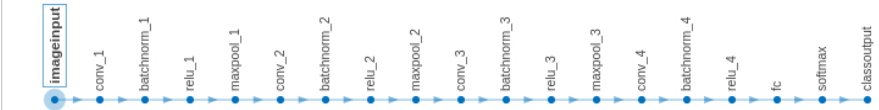


Figure 6: Network architecture

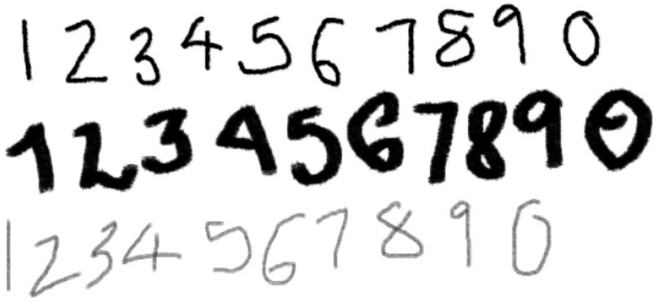


Figure 7: Handwritten digits

Question 2.3 [18 marks] *Use the provided network layer to carry out Facial Keypoints Detection task. In this task, the model needs to input an image and output a vector containing the coordinates of each keypoint. Typically, the length of this vector is equal to the number of keypoints in the image. Follow the instructions below to complete this task. The data and the model needed for the task are in the 2_3 folder.*
Submit your code as ***username_assignment2_part2_Q2_3.m****.*

1. Each predicted keypoint is specified by an (x,y) real-valued pair in the space of pixel indices. There are 15 keypoints, which represent the elements of the face.
   (a) `train.csv` contains a list of 100 training images. Each row contains the (x,y) coordinates of the 15 keypoints. The image data (96×96 for each image) is a list of pixels sorted by row in the last column.
   (b) `test.csv` contains a list of 25 test images. Each row contains the (x,y) coordinates of the 15 keypoints. The image data is a list of pixels sorted by row in the last column.
2. Perform transfer training using the data from the `train.csv`. Load `layers.mat` file as the architecture of the network. However, the layers has some problems/bugs so it cannot be correctly run. Identify and fix those problems and get the model run correctly. Use Adam optimizer, set the the maximum number of Epochs to 10, the mini batch size to 100, the initial learning rate to 0.01, the learning rate schedule to piecewise, the L2 Regularization to 0.001, the learning rate drop period to 3, the learning rate drop factor to 0.1, and shuffle the training data before each training epoch. Plot the training progress and include it into your written report. Save and submit the trained model into a `.MAT` file named `2-3_FKD.mat`.

Note that you are not allowed to change the main architecture of the model (e.g. the depth of the deep network).

3. Test the trained network using the data in the `test.csv` folder. Calculate the mean squared error for each test data.
4. Plot the image with the smallest mean square error in the test data and the corresponding feature points. Save and submit the image named `2-3_img`. An example of the image is shown in figure 8.

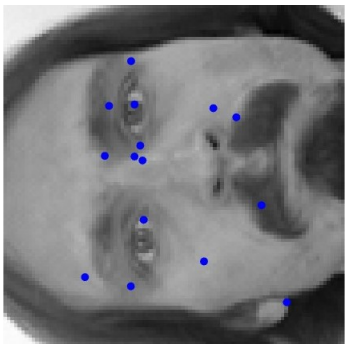


Figure 8: Example image and its feature points.