Robot Vision [06-25024]
Summative Assignment 2

Name: Kai Meller
Username: klm222
Study program: MSc Artificial Intelligence and Machine Learning

May 12, 2023

Submission deadline:	**12:00pm (BST), Monday, 16 May 2022**
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.

Part 1

Question 1.1.1 Code adapted from Feature Based Panoramic Image Stitching - MATLAB tutorial.
Algorithm grayscales each of the three images, finds the interesting features using the SURF algorithm, and attempts to match these features to the previous image. Next, a projective geometric transformation from one image to the previous is estimated, and the total transformation relative to the first image is also calculated. Then, the center image is found, and each transformation is changed to be relative to this image instead of the first (leftmost) image. imwarp is used to transform the images and to create a binary mask, which are then put into the full panorama image over each other.

Figure 1: Montage of the three given images (top) and the stitched panorama (bottom).

Question 1.1.2 My cropping algorithm counts the number of black pixels in each row and column. Then the first column and row with mostly non-black pixels is chosen as the start location of the crop, and the first column and row with mostly black pixels after this region is chosen as the end location of the crop. The algorithm then repeats twice more to remove further black patches. The first crop has a threshold of 50%, the second 70%, and the third 90% to categorise the row/column as mostly black.

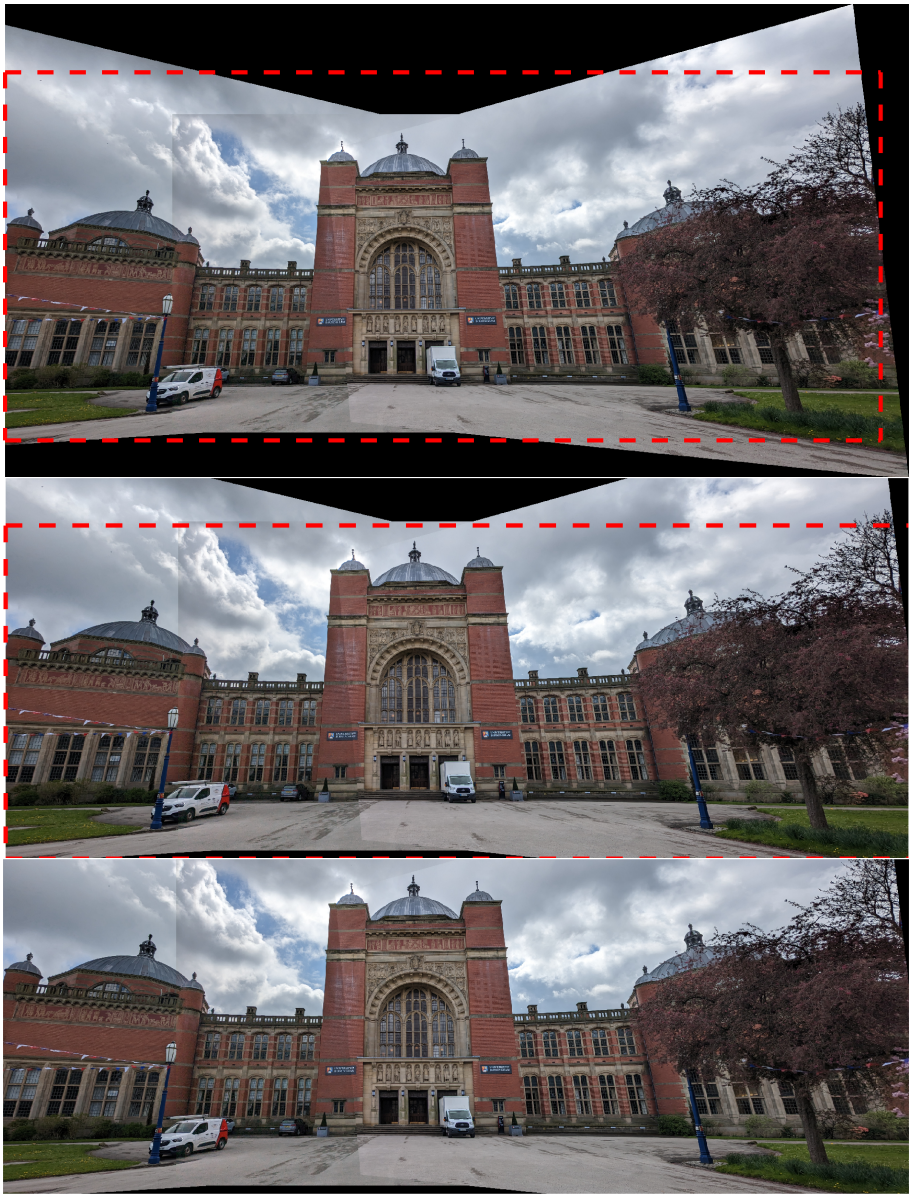

Figure 2: Each progressive stage of the crop, with the red dotted line showing what is cropped to get to the next stage. Final image is the final cropped panorama.

I found that multiple crops were necessary as each horizontal crop influenced each vertical crop, and vice versa. Empirically, I found 2 runs with an increasing threshold to find a balance between accuracy and runtime.

Question 1.2.1 Code adapted from Structure From Motion From Multiple Views - MATLAB tutorial.

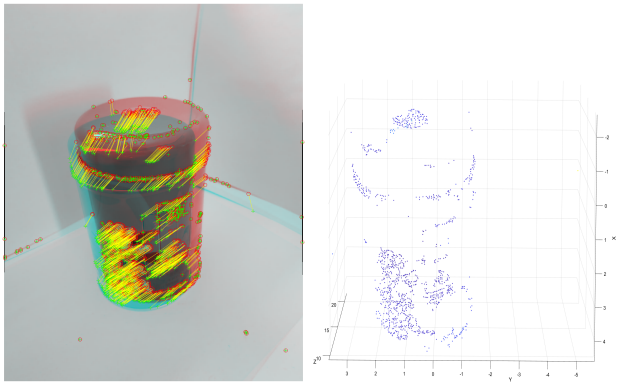

Figure 3: Features found by minimum eigenvalue algorithm tracked across the two images (left), point cloud from mapping features to 3d space using camera parameters (right).

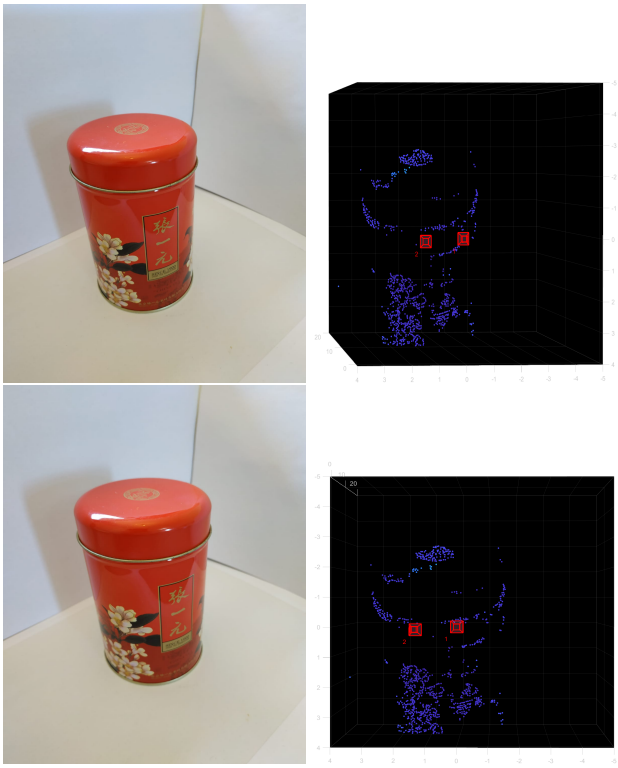

Figure 4: Original image (left), point cloud from perspective of original image (right).

Question 1.2.2 The estimated height was calculated by first finding a cylinder that represented the can in the original images. Then a point at the base and top of the cylinder were found in the figure environment, and the distance between the two points was calculated.

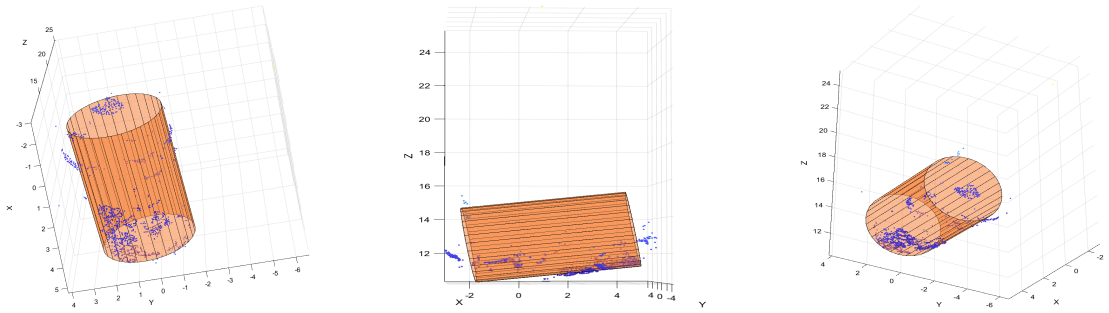

Figure 5: Cylinder fit to point cloud by pcfitcylinder with estimated height of 64mm.

Question 1.3.1 Below are the six feature detection algorithms, and the 200 strongest features detected by each of them.

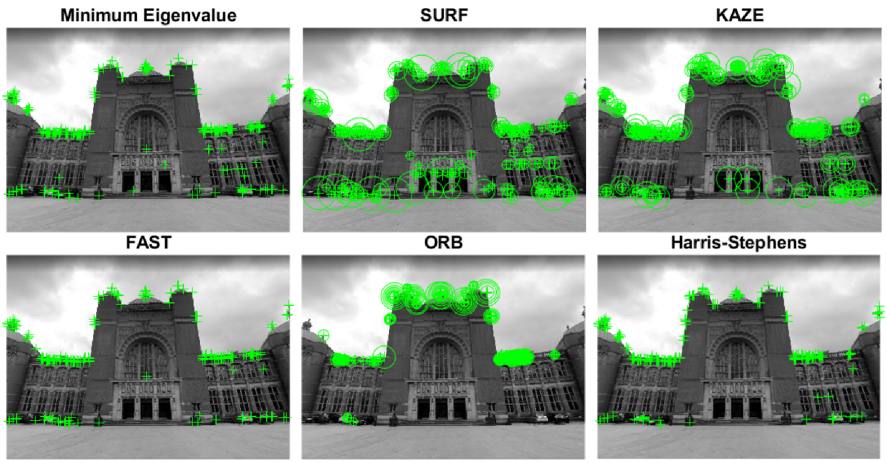

Figure 6: Aston-Webb.jpg with Minimum Eigenvalue (top left), SURF, (top center), KAZE (top right), FAST (bottom left), ORB (bottom center), and Harris-Stephens (bottom right) feature detection algorithms applied.

Question 1.3.2

- Harris-Stephens applies the Sobel operator to calculate the image derivatives in the x and y directions. If both eigenvalues of the Sobel operator matrix are large, then the center pixel is deemed to be a corner.
- SURF (Speeded Up Robust Features) applies Gaussian smoothing, and then the determinant of the Hessian represents how much the local pixels are changing, and so is greatest around points of interest.
- FAST (Features from Accelerated Segment test) corner detection works by looking at a circle around each pixel, and if the chosen pixel is brighter/darker than N of the pixels on the circle plus/minus a threshold, then that pixel is deemed to be a corner.

Harris-Stephens and FAST are both corner detectors, and so have similar results on the example image by selecting points along the sky/Aston-Webb and Aston-Webb/ground borders. The SURF algorithm has found more points on the face of the building than the other two algorithms as it has found places where there are large and distinct changes in a small area.

Part 2

Question 2.1 Code adapted from Train Deep Learning Network to Classify New Images - MATLAB Tutorial.

1. Data is unzipped and loaded into imageDatastores using given train/test split.
2. The vgg16 net is loaded, the final fully connected layer and the output layer are replaced with new layers, the data is resized to 224x224 to fit the vgg16 network, and the options are loaded. After several hours of bug-searching, I found that some of the network weights were NaN, which lead me to try changing some of the parameters given in the question. I found that the learning rate was too large and after I changed the learning rate from 1% to 0.1% the model was capable of learning. I chose to set the maxEpochs to be 6 as the loss got close to 0 around this time.

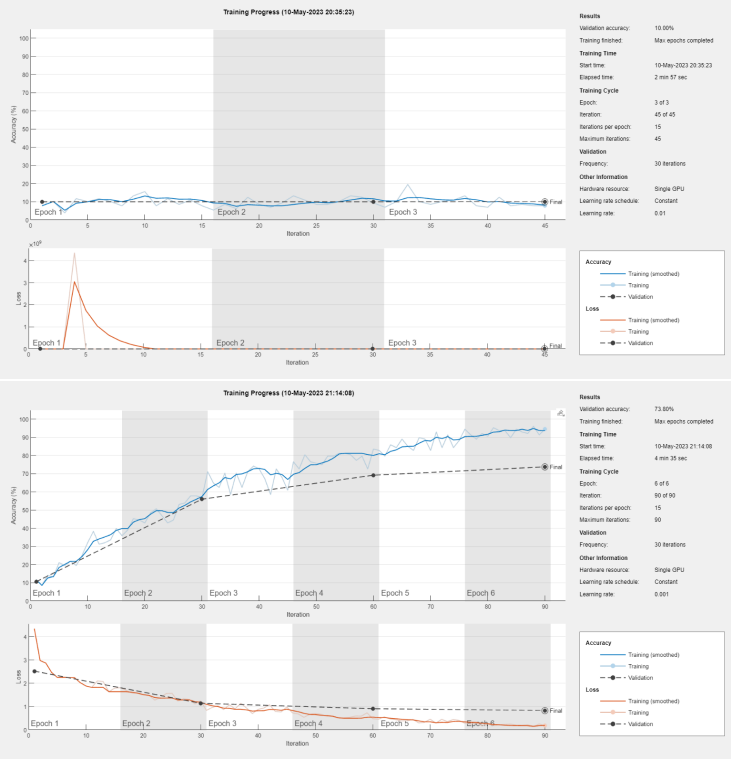

Figure 7: Learning rate set to 0.01 as given in question (top), learning rate set to 0.001 (bottom).

3. The model accuracy was determined to be 0.7380, cross-entropy was 2.3567, and the worst classified class was cat with only 20/50 correctly identified.

4. The additional data is loaded into the original imageDataStores, and the network is trained from the base vgg16 network, the same as before.

5. The 11-class model accuracy was determined to be 0.7727, cross-entropy was 2.3202, and the worst classified class was still cat with only 20/50 correctly identified.

6. When tasked with classifying the stable diffusion image, the network predicts bicycle with probability almost 1. The probability generated for the horse class is significantly more probable than any other class, but still negligible compared to the bicycle class. My guess is that the wheels in the image are the main feature that the network has learned, and that horse-like features it had learned may not have been present (e.g. the horses legs, or grass).

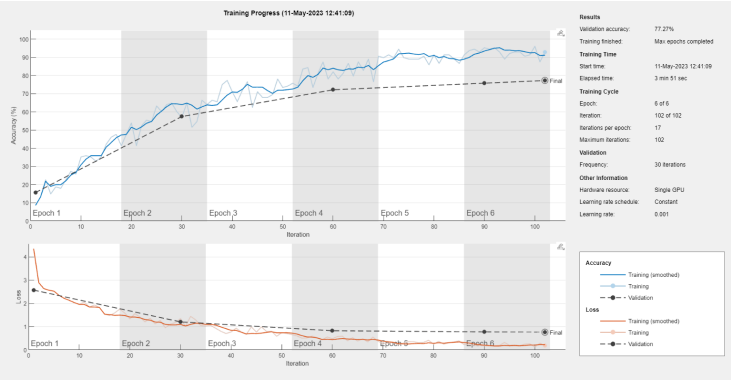

Figure 8: Network trained, again with altered learning rate of 0.001.

Question 2.2 Code to load MNIST data from Train Variational Autoencoder (VAE) to Generate Images.

1. The images are loaded into 4D matrices, and the labels are loaded into tables. A random seed is generated for the training and test sets from the first 128 examples (to coincide with the read function later on). 24 examples from the training set, and 6 examples from the test set are shown.

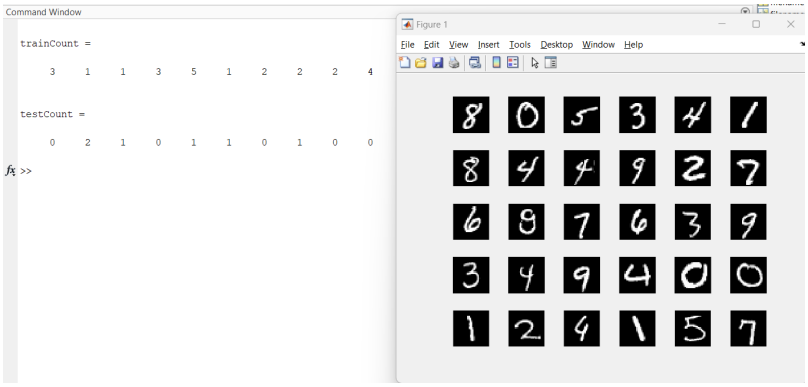

Figure 9: Random examples chosen - trainCount and testCount used to display the number of each class shown in example.

2. The example set of training data is augmented as described in the question, and resized to be 256*256*3 to work with the base network we will perform transfer learning on later. The first 128 examples are loaded into the table 'ims', and the random numbers generated earlier are used to find the matching images in the augmentedImageDatastore. These images are then shown in the figure below.

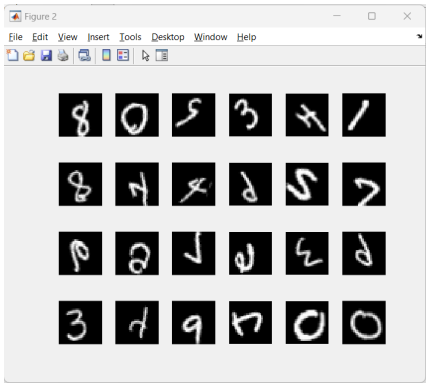

Figure 10: Same random examples as before have been augmented. Note : the test data is not augmented and not shown.

3. A new label set is created to hold the parity of the written numbers. I chose darknet-19 as the base network to perform transfer learning by borrowing the convolution and batchnorm layers. The ReLU, Max-Pooling, fully-connected and softmax layers were newly created. The network was then trained, achieving varied success depending on the run.

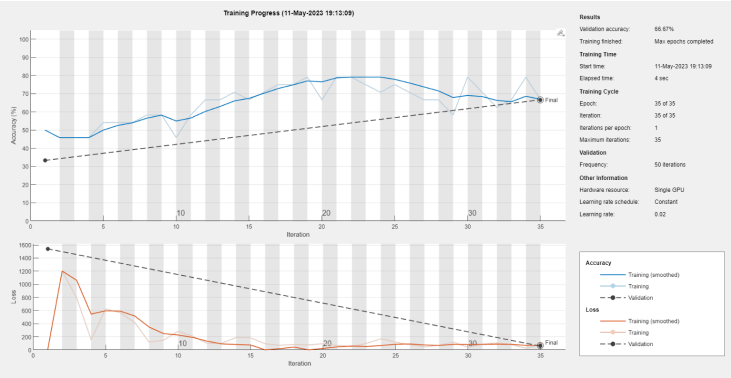

Figure 11: One run where the achieved validation accuracy was 66.6%.

4. I used MS Paint to create 3 of each digit, and separated them by parity into folders named 'true' and 'false' for odd and even respectively.
The accuracy of the network on these symbols was 50%, with the network guessing that the input

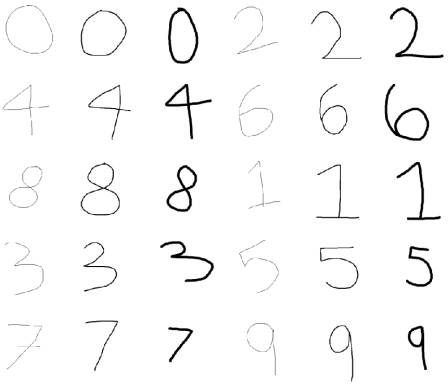

Figure 12: My handwritten digits.

was an even number everytime. My guess is that the network was trained on white on black images, and so wasn't capable of classifying the black on white images in my set.

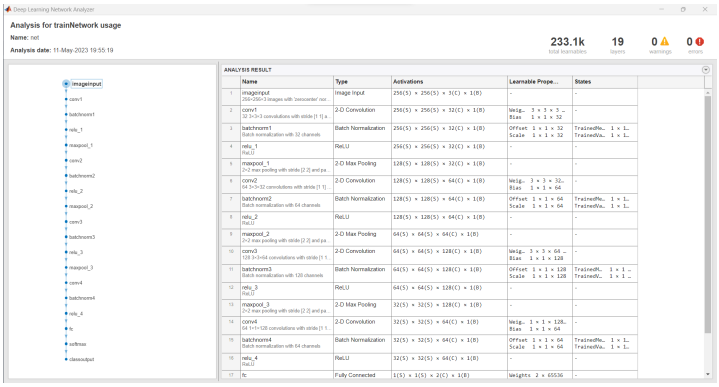

Figure 13: analyzeNetwork(mynet).

Question 2.3 1. The data is loaded from the layers.mat file given.

2. The given image matrices are given an additional colour channel to make them standard format. The input layer is corrected by changing the inputsize to [96,96,1]. The penultimate fully-connected and ReLU layers are also removed as there is no batchnorm and max-pooling layer between them and the final fully-connected and ReLU layers. The final softmax layer is removed and replaced with a regression output layer to produce the results.
   After these adjustments, the network is trained with the given options.

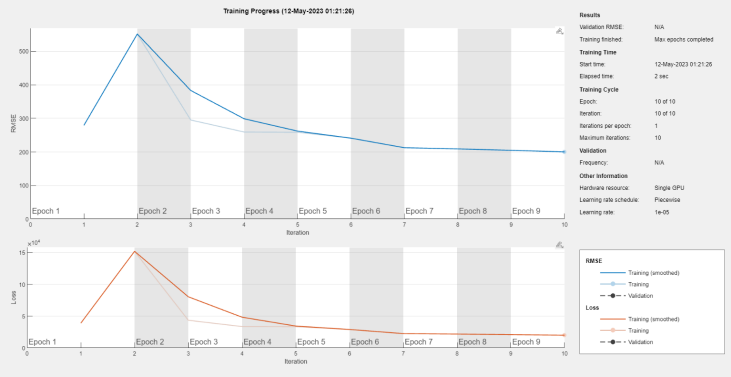

Figure 14: Network training.

3. The Mean Squared Error is calculated as the mean distance that each predicted keypoint is from the actual keypoint. The average MSE over then examples is 49 pixels, meaning that on average the networks prediction for any single keypoint is 49 pixels away from the ground truth.
4. The image with smallest MSE is shown below. Clearly the shape of the face has been learned by the model, but the relation of the face to the image is incorrect - with several points even outside the image.

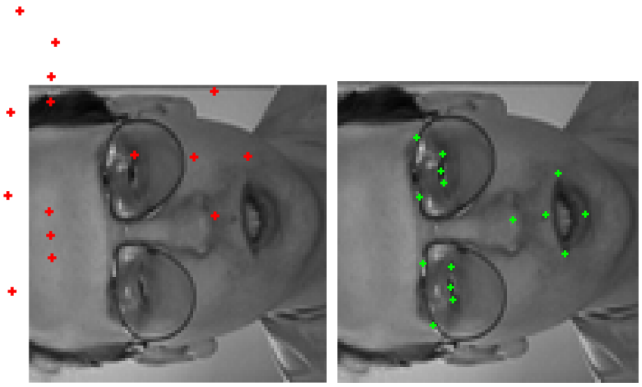

Figure 15: Predicted keypoint locations (left) actual keypoint locations (right).