Code and Report

1. Introduction

The main goal of this practical was to gain practical experience processing signals in the time and frequency domain and the analysis of images. A secondary goal was to gain more familiarity with MATLAB as it is a common language used for image and signal processing. Here in this report, we will discuss the methodology and results for the audio processing and image analysis tasks for this practical. We will conclude with what has been learnt from this practical, what would have been done differently and further steps that could have been implemented with more time. For more information, please reference the 'CS4302_practical2.zip' folder which contains the code for this practical.

2. Audio Processing

The task in this section was to analyse the signals from three audio files, each with the same base signal of interest but with different background noises. The aim was to identify the frequency of the background noise and use a filter to remove it from the file.

2.1 Finding the frequency of background noise in each audio recording

The background noise of the audio recordings was identified as 600 Hz, 150 Hz and (250 Hz , 900 Hz) for audios 1, 2, and 3 respectively. These frequencies were identified using the following process. A Fast Fourier Transform was used to plot signals in the frequency domain (Figure 1b). The sampling frequency of the audio signals was halved in the graph to prevent aliasing. This is because the sampling rate must be at least double the frequency of the signal according to the Niquest Shannon theorem. The graphs in Figure 1a were reduced by 20-fold to improve the resolution of the x-axis so that the frequency could be read off more easily. The resulting graph was compared from all three recordings and differences were identified as background noise. These frequencies were read off the x-axis of the graphs.

2.2 Steps for frequency filtering in each audio recording

A band stop filter was used to remove frequencies that contained background noise from the recordings (Figure 1c,d). Two filters were applied to audio 3 as it contained two different frequencies of background noise.

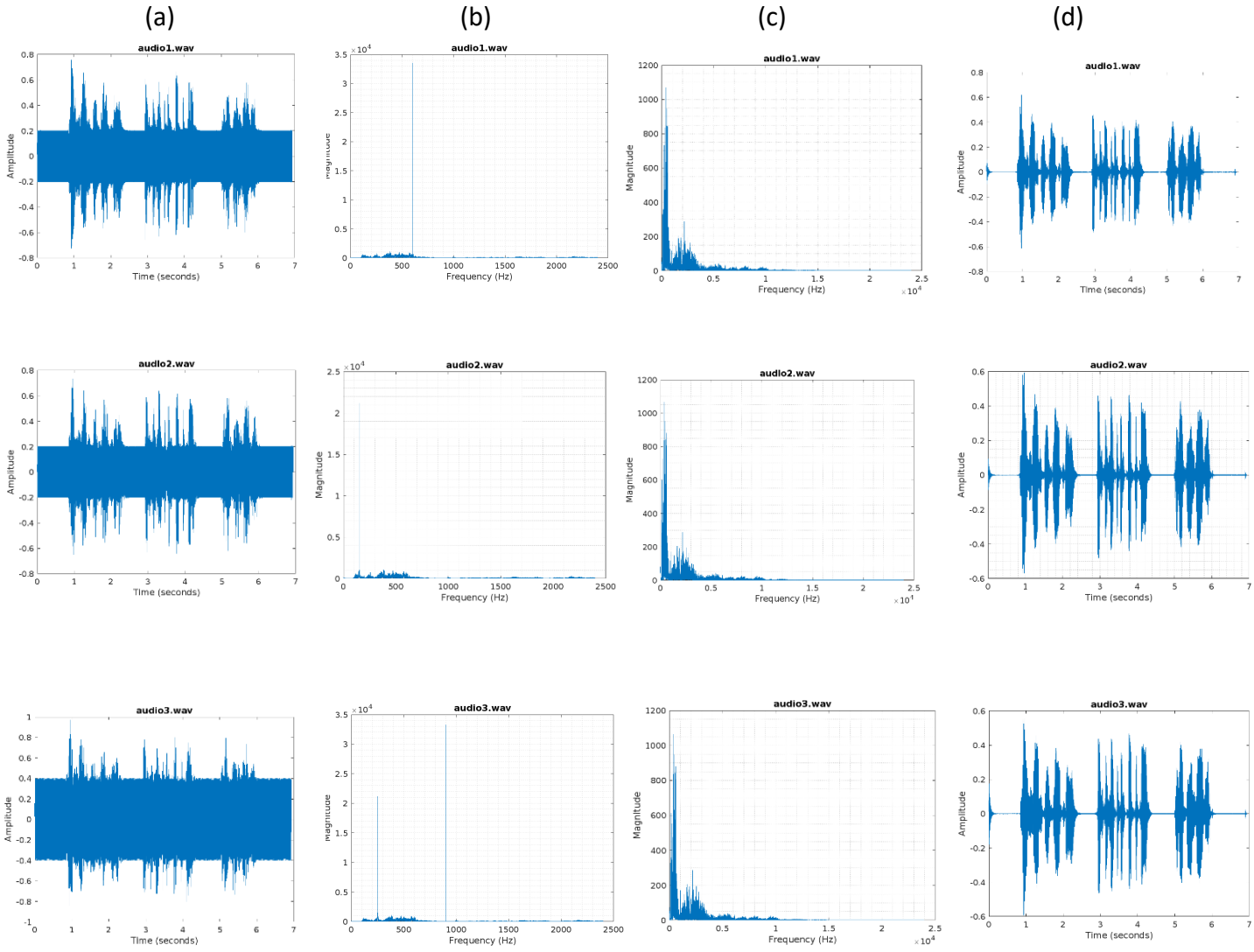


Figure *1 – step by step output of audio processing (a) amplitude of signal plotted against time. (b) magnitude of signal plotted in frequency domain. (c) magnitude of signal plotted in frequency domain with background noise removed (d) amplitude of signal plotted against time with background noise removed.*

2.3 Change in each audio recording

Background noise was removed from all audio recordings so that they only contained the base noise of interest. All recordings sounded similar after processing however audio1 did have an occasional beep noise which audio 2 and 3 did not have. Please refer to the files ‘noise_removed_audio(1/2/3).wav’ in the ‘Audio analysis’ folder to listen to the files.

3. Image Analysis

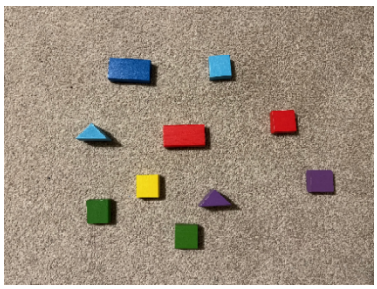
The task in this section was the analyse five images of coloured blocks on a neural surface that ranged in difficulty from easy to extreme. The aim was to count the total number of blocks in each image, the number of blocks of each colour, the number of blocks of each shape, and the number of blocks of each shape and colour.

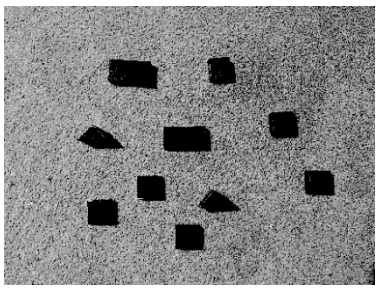
3.1 Count total number of blocks

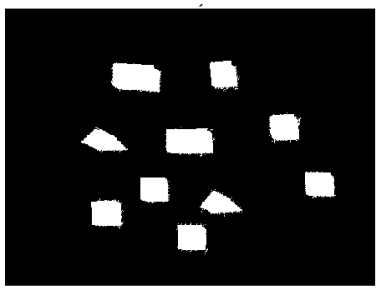
3.1.1 Methodology

Easy: Loaded image into MATLAB (Figure 1a). Separated images into R,G, B colour channels and marked pixel as black in all channels if one was above a given threshold in a given channel (Figure 1b). Pre-processed the image (converted to grayscale, found the compliment, binarised, filled in the gaps and removed artefacts. Used the bwconncomp() function to count the number of blocks (Figure 1 c).

(a) (b) (c)







```
The total number of blocks is  :
    10
```

Figure 1 – step by step processes counting total number of blocks in easy.jpeg image

Medium - Hard: Loaded image into MATLAB (Figure 2a) and used K-means clustering to segment image by colour (Figure 2b). Merged images in colour spaces of blocks into one image. Pre-processed image as above. Used the bwconncomp() function to count the number of blocks (Figure 2c). This technique was chosen over above as it removed shadows so was able to differentiate shapes more easily.

(a) (b) (c)

Figure 2 – step by step processes counting total number of blocks in hard.jpeg image

Very hard - Extreme: This was the same as the above except the number of blocks was counted in each colour space and the program found the total. This meant that the program was differentiate neighbouring blocks more easily. These two images contained blocks that were clumped together.

3.1.2 Results

This process produced accurate results for all five images (Figure 3).

Input image	Expected result	Actual result	Status
Easy.jpg	10	10	Pass
Medium.jpeg	20	20	Pass
Hard.jpeg	23	23	Pass
Very_hard.jpeg	27	27	Pass
Extreme.jpeg	27	27	Pass

Figure 3 – Table of outputs when counting number of blocks in images

3.2 Count number of blocks of each colour

3.2.1 Methodology

Easy - Extreme: Loaded image into MATLAB (Figure 4a) Converted image to Lab colour space and used K-means clustering to segment image by colour (Figure 4b). Image was converted to lab colour space to remove luminance and chrominance, so colours were clustered irrespective of brightness. K means clustering gives accurate segmentation by colour and to remove shadows so neighbouring blocks weren't merged. Pre-processed image and used the bwconncomp() function to count the number of blocks in each colour space containing blocks (Figure 4c). More difficult images required further segmentation and filling in to get desired shape.

(a) (b) (c)

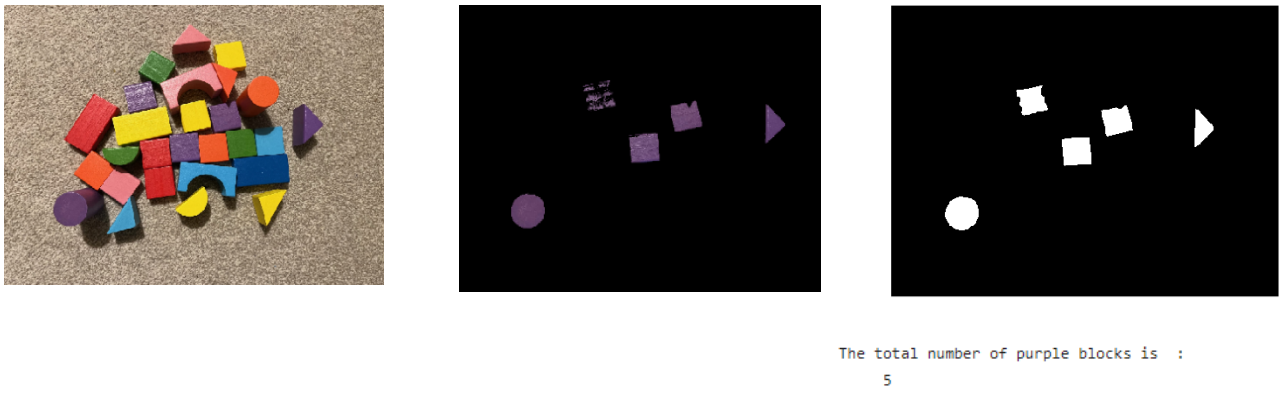


Figure 4 - step by step processes of counting number of blocks of each colour in extreme.jpeg image

3.1.2 Results

This process produced accurate results for all five images (Figure 5).

Input image	Expected result	Actual result	Status
Easy.jpg	Red: 2 Light blue: 2 Yellow: 1 Purple: 2 Green: 2 Dark blue:1	Red: 2 Light blue: 2 Yellow: 1 Purple: 2 Green: 2 Dark blue:1	Pass
Medium.jpeg	Red: 2 Light blue: 2 Yellow: 5 Purple: 4 Green: 2 Dark blue:1 Orange: 4	Red: 2 Light blue: 2 Yellow: 5 Purple: 4 Green: 2 Dark blue:1 Orange: 4	Pass
Hard.jpeg	Red: 2 Light blue: 3 Yellow: 5 Purple: 4 Pink: 1 Green: 3 Dark blue: 1 Orange: 4	Red: 2 Light blue: 3 Yellow: 5 Purple: 4 Pink: 1 Green: 3 Dark blue: 1 Orange: 4	Pass
Very_hard.jpeg	Red: 3 Light blue: 3 Yellow: 5 Purple: 5 Pink: 3 Green: 3 Dark blue: 1 Orange: 4	Red: 3 Light blue: 3 Yellow: 5 Purple: 5 Pink: 3 Green: 3 Dark blue: 1 Orange: 4	Pass
Extreme.jpeg	Red: 3 Light blue: 3 Yellow: 5 Purple: 5 Pink: 3 Green: 3 Dark blue: 1 Orange: 4	Red: 3 Light blue: 3 Yellow: 5 Purple: 5 Pink: 3 Green: 3 Dark blue: 1 Orange: 4	Pass

Figure 5 – Table of outputs when counting number of blocks of each colour in images

3.3 Count number of blocks for each shape

3.3.1 Methodology

Easy: Took resulting image processed in 3.1 (Figure 6a) and used bwboundaries() to select each object in the image (Figure 6b). Classified each object based on unique characteristics of the shape (Figure 6c). Used regionsprops() and minboundrect() functions to calculate the proportion of the shapes area compared to the area of the minimum bounding box. Blocks were classified as triangles if this proportion was below a given threshold. Differentiated between square and rectangle by calculating ratio of width to height of minimum bounding box.

(a) (b) (c)

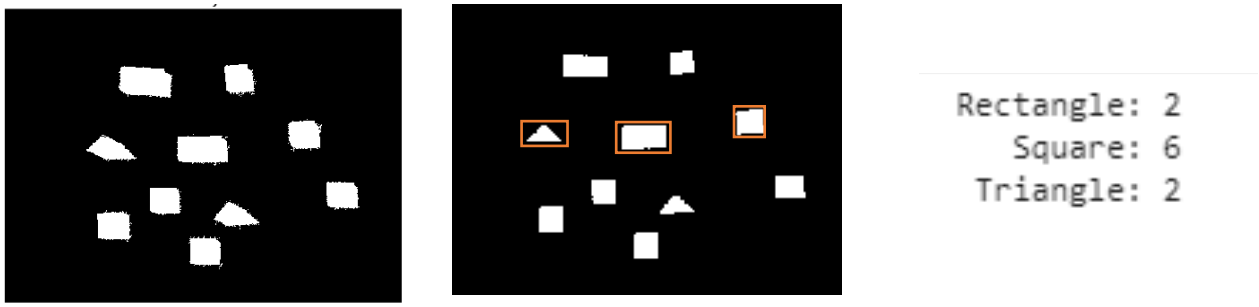


Figure 6 – step by step processes counting total number of blocks of each shape in easy.jpeg image

Medium: In addition to the above, a circularity measure was used to determine whether the block was a circle. Semicircles were classified if the area of the shape took up a lower proportion of the bounding box than a rectangle but a higher proportion than a triangle. (Figure 7a).

Hard: In addition to the above, bridges were classified if shapes were smaller than the bounding box and had a higher circularity measure than triangles and semicircles (Figure 7b).

(a) (b)

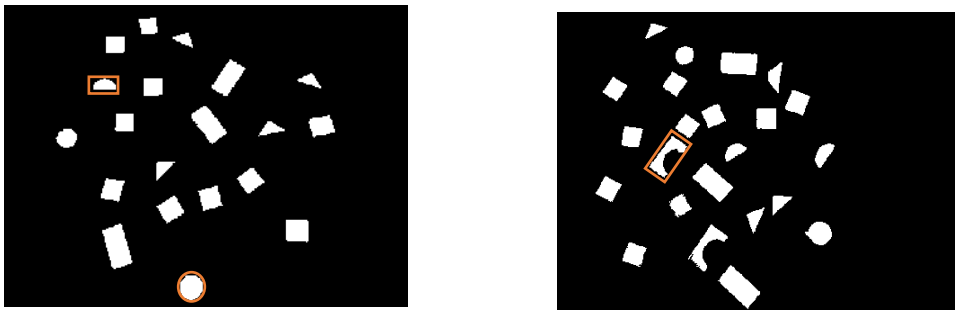


Figure 7 – (a) classifying semicircles and circles in medium.jpeg (b) classifying bridges in hard.jpeg.

Very hard - Extreme: This was the same as the above process except images were classified in the images that had been segmented by colour. A total of each shape from all images was calculated. This was chosen over above as was able to differentiate neighbouring blocks more easily. These two images contained blocks that were clumped together.

3.3.2 Results

This process only gave accurate results for one out of the five images (Figure 8). Accuracy reduced as images got more difficult.

Input image	Expected result	Actual result	Status
Easy.jpg	Rectangle: 2 Square: 6 Triangle: 2	Rectangle: 2 Square: 6 Triangle: 2	Pass
Medium.jpeg	Rectangle: 3 Square: 10 Triangle: 4 Circle: 2 Semicircle: 1	Rectangle: 3 Square: 10 Triangle: 4 Circle: 1 Semicircle: 2	Fail: mistook circle for semi: circle
Hard.jpeg	Rectangle: 3 Square: 10 Triangle: 4 Circle: 2 Semicircle: 2 Bridge: 2	Rectangle: 3 Square: 10 Triangle: 3 Circle: 1 Semicircle: 4 Bridge: 2	Fail
Very_hard.jpeg	Bridge: 2 Circle: 2 Triangle: 5 Square: 13 Semicircle: 2 Rectangle: 3	Bridge: 2 Circle: 4 Triangle: 4 Square: 11 Semicircle: 3Rectangle: 3	Fail
Extreme.jpeg	Bridge: 2 Rectangle: 3 Circle: 2 Semicircle: 2 Triangle: 5 Square: 13	Bridge: 3 Rectangle: 3 Circle: 1 Semicircle: 4 Triangle: 4 Square: 12	Fail

Figure 8 - Table of outputs when counting number of blocks of each shape in images

3.4 Count number of blocks of each shape with same colour

3.4.1 Methodology

Easy - extreme: This process was a combination of the methods used in 3.2 and 3.3. This process was chosen as it gave accurate segmentation of blocks into colours and was able to classify the majority of shapes correctly.

3.4.1 Results

This process gave some accurate results for all images but only easy.jpeg received accurate results for all colours. Common issues were mistaking square for circle and circle for semi: circle (Figure 9)

Input image	Colour	Expected result	Actual result	Status
Easy.jpg	Red	Rectangle: 1, Square: 1	Rectangle: 1 Square: 1	Pass
	Light blue	Square: 1 Triangle: 1	Square: 1 Triangle: 1	Pass
	Green	Square: 2	Square: 2	Pass
	Yellow	Square: 1	Square: 1	Pass
	Purple	Square: 1 Triangle: 1	Square: 1 Triangle: 1	Pass
	Dark blue	Rectangle: 1	Rectangle: 1	Pass
Medium.jpeg	Red	Rectangle: 1 Square: 1	Rectangle: 1 Square: 1	Pass
	Light blue	Square: 1 Triangle: 1	Square: 1 Triangle: 1	Pass
	Green	Square: 2	Square: 2	Pass
	Yellow	Square: 2 Rectangle: 1 Semicircle: 1 Triangle: 1	Square: 2 Rectangle: 1 Semicircle: 1 Triangle: 1	Pass
	Purple	Square: 2 Circle: 1 Triangle: 1	Square: 2 Semicircle: 1 Triangle: 1	Fail: mistook circle for semicircle
	Dark blue	Rectangle: 1	Rectangle: 1	Pass
	Orange	Square: 2 Triangle: 1 Circle: 1	Square: 2 Triangle: 1 Circle: 1	Pass
Hard.jpeg	Red	Rectangle: 1 Square: 1	Rectangle: 1 Square: 1	Pass
	Light blue	Bridge: 1 Square: 1 Triangle: 1	Bridge: 1 Square: 1 Triangle: 1	Pass
	Green	Square: 2 Semicircle: 1	Square: 2 Semicircle: 1	Pass
	Yellow	Square: 2 Semicircle: 1 Triangle: 1 Rectangle: 1	Square: 2 Semicircle: 1 Triangle: 1 Rectangle: 1	Pass
	Purple	Circle: 1 Triangle: 1 Square: 2	Circle: 1 Triangle: 1 Square: 2	Pass
	Dark blue	Rectangle: 1	Rectangle: 1	Pass
	Orange	Square: 2 Triangle: 1 Circle: 1	Square: 2 Semicircle: 2	Fail: triangle and circle mistaken for semicircle
	Pink	Bridge: 1	Bridge: 1	Pass
Very_hard.jpeg	Red	Rectangle: 1 Square: 2	Rectangle: 1 Square: 2	Pass
	Light blue	Square: 1 Triangle: 1 Bridge:1	Square: 1 Triangle: 1 Bridge: 1	Pass
	Green	Square: 2 Semicircle: 1	Square: 2 Semicircle: 1	Pass
	Yellow	Square: 2 Rectangle: 1 Triangle: 1 Semicircle: 1	Circle:1 Square:1 Rectangle:1 Triangle: 1 Semicircle: 1	Fail: mistook square for circle
	Purple	Square: 3 Triangle: 1 Circle: 1	Square: 2 Triangle: 1 Circle: 2	Fail: mistook square for circle
	Dark blue	Rectangle: 1	Rectangle: 1	Pass
	Orange	Square: 2 Circle: 1 Triangle: 1	Square: 2 Circle: 1 Semicircle: 1	Fail: mistook triangle for semicircle
	Pink	Bridge: 1 Square: 1 Triangle: 1	Bridge: 1 Square: 1 Triangle: 1	Pass

Figure 9 - Table of outputs when counting number of blocks of each shape with same colour in images

4. Conclusion

During this practical I gained several practical skills in signal processing and MATLAB including: converting signals from the time to frequency domain using fft(); removing background noise using a band stop filter; pre-processing images to prepare them for analysis; k-means clustering of colours in images using imsegmeans(); and deriving properties of object shape using regionprops().

If I was to do this practical again, perhaps I would use marker-based watershed to segment the blocks in the image. This is because the segmentation of objects was occasionally inaccurate in this program, especially in more difficult images. Additional processing was occasionally required to get the correct result. Furthermore, the program struggled to differentiate neighbouring objects that were clumped together in the more difficult images and objects had to be counted in separate colour spaces. Marker based segmentation may have provided a more elegant solution not both issues.

If I had more time, I would consider using a convolutional neural network to complete the image analysis section of this practical. Although it would require a larger dataset of images to train it I believe the model would offer a more accurate solution that is more transferable between images.