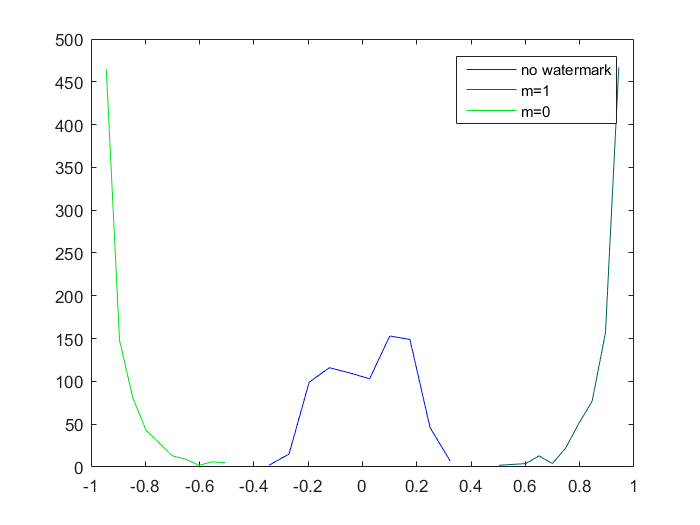
**Assessed Practical 4 – Jordan McDonald (40063974)**

1.



This graph shows the distribution of results from the detection process using block correlation coefficient and contains results where the watermark message bit equals one, zero or contains no watermark at all. The left peak shows the amount of images detected where m = 0, the right peak shows images detected with m = 1 and the central plot shows images with no watermark, overall using an alpha of one we can see a roughly even distribution of images detected correctly which means threshold is categorising the images correctly in most cases. The x axis represents the correlation coefficient value from each image and despite a lot of the values not equalling exactly one, zero or -1 the threshold can still classify values if they are close enough. It should be noted lowering the alpha will cause these lines to further overlap which will reduce effectiveness since the strength is reduced. With the current alpha value there is no overlap which assists in a correct plot of the values.

2.

|  |  |  |  |
| --- | --- | --- | --- |
| Alpha Value | Cac (no wm) | Ca0 (m = 0) | Ca1 (m = 1) |
| 0 | 0.91375 | 0.0300 | 0.05625 |
| 0.2 | 0.99125 | 0.405 | 0.385 |
| 0.5 | 0.93125 | 0.998750 | 0.998750 |
| 0.7 | 0.9900 | 1 | 1 |
| 1 | 0.97150 | 1 | 1 |

From the table we determine a few things, if there is no alpha value (0) then the watermark has no impact on the image as seen by the very low detection rate. This shows that detection fully dependent on a suitable alpha value, we can see this as the alpha value is gradually increased – each increase leads to a greater detection rate up until an alpha of 0.7 which leads to perfect detection and shows that all the values fall within the defined thresholds. This also shows that 0.7 is a good choice for the alpha value, and possibly 0.6 would lead to the best balance between fidelity robustness and detection rate. We can also see the detection rate for no watermark images is fairly constant with the values deviating slightly, which shows that alpha has no impact on detection in regards to the Cac no watermark array. In summary an increase in alpha will result in better detection rates but may impact fidelity if it is too high.

3.

a) This loop structure performs the following actions (a block wise scan), it iterates initial the rows (height) and then the columns (width) in order to isolate each 8x8 square in the image as a sequence of eight digits, seen by the use of ‘1:8’ which constructs an array. ‘vo’ in this context represents the watermark array, the instruction within the second loop *(orIm(i:i+7,j:j+7)* takes the original image and creates an array square of size 8x8 by taking the value and I and getting the next seven pixels and getting the value of j and the next subsequent pixels. On the next iteration I and j are incremented and a new square is found and the value of this is added to the watermarked array. This whole operation takes a larger image and compresses it into an 8x8 square. (See below for an example showing how extracted watermark is constructed)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

b) The for loop sets up a construct that creates an array of i:8 in sequence until the width or height of the image has been met, these two loops allow a formation of an 8x8 square which can be used to operate on the watermark. The code ‘wmIm(i:i+7,j:j+7)’ uses the values of I and j to construct an 8x8 block array on each iteration of the loop. The rest of the line of code ‘orIm(i:i+7,j:j+7) + vw - vo’ performs a similar block wise operation as described before except from the original image, the value of the marking space ‘vw’ minus the value of the normalised watermark ‘vo’ is then added to the 8x8 array block formed from the original image and then assigned to the block formed by the watermarked image .In summary this line of code performs a block wise scan of the image and adds a watermark in each block, this will result in a fully formed watermarked image.