1.

|  |  |  |
| --- | --- | --- |
| **Statistic** | **Formula** | **Value(s)** |
| Range |  | 205 |
| Normalised Range |  | 0.80 |
| Michelson Contrast |  | 0.84 |
| RMS Contrast |  | 66.08 |
| Simple Arithmetic Mean |  | 123.39 |
| Harmonic Mean |  | 82.98 |
| Geometric Mean |  | 103.40 |
| Midrange |  | 122.50 |
| 10% Winsorised Mean | ​L\_winsorised = (*x\_n*​…*x\_n*+1​ + *x\_n*+2​…*x\_n) / N*  *Where N = total number of pixels, n = number of smallest and largest pixels to be replaced* | 206.90 |
| 10% Trimmed Mean | Where N = total number of pixels |  |

|  |  |
| --- | --- |
|  |  |
| (i) | (ii) |
|  |  |
| (iii) | (iv) |

2.

3.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  | |  | |
|  |  |  |  |  |  |
| 215 | 225 | 175 | 225 | 20 | 125 |

|  |  |
| --- | --- |
| **Property** | **Value** |
| Total number of pixels selected | 3261 |
| Number of disconnected clusters selected | 18 |

|  |  |
| --- | --- |
| Text  Description automatically generated | A picture containing graphical user interface  Description automatically generated |
| (i) | (ii) |

4.

1. Hysteresis thresholding

Applying hysteresis thresholding didn’t turn out to be as efficient as the non-linear majority filter. So, we will use the latter one further.

ii) the non-linear **majority filter**

* Applying non-linear majority filter works only when the window size is as small as possible. It will be useful in the workflow to polish the image in the latter stage after we use morphological operators. In the table below, we can see that window size of 2x2 gives us an image with less cluster as the small black spots inside the number plate gets covered by white pixels as most of the pixel around them are white. As we keep on increasing the window size, the image gradually gets covered by black pixels as they dominate the whole image. So, when the window size is 44x44, we can’t see any white pixels.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Window Size** | **Total number of pixels selected** | | **Number of disconnected clusters selected** | **Binary Image** |
| 2x2 | 3214 | 5 | | Text  Description automatically generated |
| 3x3 | 3214 | 5 | | Text  Description automatically generated |
| 6x6 | 3075 | 4 | | Text  Description automatically generated with low confidence |
| 10x10 | 2782 | 3 | | Graphical user interface  Description automatically generated with medium confidence |
| 20x20 | 1941 | 3 | | A picture containing silhouette, night sky  Description automatically generated |
| 30x30 | 1088 | 5 | | A picture containing silhouette, night sky  Description automatically generated |
| 40x40 | 358 | 2 | | A picture containing silhouette, night sky  Description automatically generated |
| 43x43 | 177 | 2 | | A picture containing night sky  Description automatically generated |
| 44x44 | 0 | 0 | | Shape  Description automatically generated with low confidence |
| 45x45 | 0 | 0 | | Shape  Description automatically generated with low confidence |

5.

* As we can see on the table below, we can see that some of the morphological operators work well and some of them don’t. Erode doesn’t work. Dilate works perfectly as it thins out the letters in the number plate. Especially using the strel of diamond size 2 works very well because it also has the lowest number of disconnected clusters of just 4. Opening doesn’t work. Closing with strel of diamond size 2 might work depending on the circumstance but it has 10 disconnected clusters and it’s more inconvenient to work with.

|  |  |  |  |
| --- | --- | --- | --- |
| **Strel Shape and Size** | **Total number of pixels selected** | **Number of disconnected clusters selected** | **Binary Image** |
| **Erode**  **Strel**  **(‘Diamond’, 2)** | **1202** | **10** | **Text  Description automatically generated** |
| **Erode**  **Strel**  **(‘Diamond’, 3)** | **1202** | **10** | **Text  Description automatically generated** |
| **Erode**  **Strel**  **(‘Disk’,1,4)** | **2057** | **18** | **Text  Description automatically generated** |
| **Erode**  **Strel**  **(‘Disk’,2,4)** | **1202** | **10** | **Text  Description automatically generated** |
| **Erode**  **Strel**  **(‘Line’,5,45)** | **2139** | **16** | **Text  Description automatically generated** |
| **Erode**  **Strel**  **(‘Square’,3)** | **1831** | **12** | **Text  Description automatically generated** |
| **Erode**  **Strel**  **(‘Square’,6)** | **578** | **11** | **A picture containing silhouette, night sky  Description automatically generated** |
| **Dilate**  **Strel**  **(‘Diamond’, 2)** | **5625** | **4** | **Graphical user interface, text, application  Description automatically generated** |
| **Dilate**  **Strel**  **(‘Diamond’, 3)** | **5625** | **4** | **Graphical user interface, text, application  Description automatically generated** |
| **Dilate**  **Strel**  **(‘Disk’,1,4)** | **4493** | **7** | **Text  Description automatically generated** |
| **Dilate**  **Strel**  **(‘Disk’,2,4)** | **5625** | **4** | **Graphical user interface, text, application  Description automatically generated** |
| **Dilate**  **Strel**  **(‘Line’,5,45)** | **4433** | **9** | **Graphical user interface, text  Description automatically generated** |
| **Dilate**  **Strel**  **(‘Square’,3)** | **4757** | **6** | **Text  Description automatically generated** |
| **Dilate**  **Strel**  **(‘Square’,6)** | **6760** | **1** | **Graphical user interface, application  Description automatically generated** |
| **Open**  **Strel**  **(‘Diamond’, 2)** | **2658** | **7** | **Graphical user interface  Description automatically generated with medium confidence** |
| **Open**  **Strel**  **(‘Diamond’, 3)** | **2658** | **7** | **Graphical user interface  Description automatically generated with medium confidence** |
| **Open**  **Strel**  **(‘Disk’,1,4)** | **2979** | **5** | **A picture containing text  Description automatically generated** |
| **Open**  **Strel**  **(‘Disk’,2,4)** | **2658** | **7** | **Graphical user interface  Description automatically generated with medium confidence** |
| **Open**  **Strel**  **(‘Line’,5,45)** | **2987** | **9** | **Text  Description automatically generated** |
| **Open**  **Strel**  **(‘Square’,3)** | **2893** | **9** | **Graphical user interface  Description automatically generated with medium confidence** |
| **Open**  **Strel**  **(‘Square’,6)** | **1973** | **10** | **Text  Description automatically generated** |
| **Close**  **Strel**  **(‘Diamond’, 2)** | **3485** | **10** | **Text  Description automatically generated** |
| **Close**  **Strel**  **(‘Diamond’, 3)** | **3485** | **10** | **Text  Description automatically generated** |
| **Close**  **Strel**  **(‘Disk’,1,4)** | **3391** | **14** | **Text  Description automatically generated** |
| **Close**  **Strel**  **(‘Disk’,2,4)** | **3485** | **10** | **Text  Description automatically generated** |
| **Close**  **Strel**  **(‘Line’,5,45)** | **3395** | **15** | **Text  Description automatically generated** |
| **Close**  **Strel**  **(‘Square’,3)** | **3433** | **10** | **Text  Description automatically generated** |
| **Close**  **Strel**  **(‘Square’,6)** | **3711** | **6** | **Text  Description automatically generated** |
|  |  |  |  |

6.

The function correct identifies the registration plate. Confidence for the first word containing 4 letters is 76% and for the second word containing 3 letters is 87%.

I used the workflow where we binarize the image, then dilate it, and take the majority filter. After that, we store a pattern resembling the registration plate and we extract the text given from the Ocr function and match it to the pattern. If it matches, we get a perfect result. In this case, it matches perfectly and we store it in variable ‘plateNum’.

Graphical user interface

Description automatically generated

7.

I have collected 4 videos namely 1.mp4, 2.mp4, 3.mp4, and 4.mp4. Then, I took each frame of the video and passed it in the function which identifies the registration plate and returns the plate number as well as a binary image (with only the registration plate) with bounding boxes and confidence level. I create a new video with the images that were returned.

From the evaluation of these 4 videos, I found out that if the orientation of the registration plate is straight (or horizontal), then the OCR function correctly recognizes the plate number. However, if the orientation is changed, then it fails to do so. In future work, we might be able to correct the orientation of the registration plate in the image before passing it to the OCR function. One of the bigger challenges was that there were colors in the video that were very similar to the one on the registration plate like the yellow lines on the street, street signs, the leaves of the trees, and sometimes also various objects like houses, signs, etc on the streets. The function I built picked these colors up as well. In future work, we might be able to properly identify which of these objects belong to the car and which of them belong to the environment or surrounding and remove the surrounding ones. Also, future work might include optimization of the code because evaluating a 9-15 sec video that has around 250-500 frames on a Macbook Air takes between 5-10 minutes including the time it takes to create a video out of those output binary images. Maybe taking 30 images would have been easier but I wanted to push the boundary by using a video which made me tackle the edge cases as much as I could using the time I had. I implemented a pattern recognition technique on question 6 which extracted on the text that was in the pattern of the registration plate numbers in the UK but had to take that part out on this question because of limited time as it created quite a few errors. We could add that on the future to make the algorithm more robust and precise in detection the plate number.

8.

Image Processing Onramp: <https://matlabacademy.mathworks.com/progress/share/certificate.html?id=eef62964-e17c-4a6e-bddc-703c9678e0f9&>

MATLAB Onramp:

<https://matlabacademy.mathworks.com/progress/share/certificate.html?id=ae1c4910-cce7-4700-b8b0-24b8c3822c19&>