Machine Readable Zone Detection Explanation

The whole code can be divided mainly into 10 parts.

Part – 1: Loading the image

For loading the image we use **imread** and then we verify whether the image is RGB or not. If the image is RGB, then the size of the image will have 3 elements. If the image is RGB type, then we will convert it into a grayscale image using **rgb2gray** function.

Part – 2: Rotating the image if required

If the image is not in portrait mode, then we will rotate it by 90 degrees using **imrotate**. This can be checked using the concept that in portrait mode the height of the image is greater than its width.

Part – 3: Scaling the image

If image size is more than 4 Mega Pixel, then we will rescale the image to width of 1000 pixels. We will maintain the aspect ratio by setting the height of the image to NaN which will make MATLAB determine its new height automatically.

Part – 4: Kernel creation

We make two kernels – rectangle shaped and square shaped so that later on they can be moved across the image to get information and filter the image as well. For this we have used **strel** (structured elements) function.

Part – 5: Gaussian filtering for noise removal

For noise removal (pre-processing) of the image, Gaussian filter has been used. Using the function **imtophat** and passing the rectangular kernel as the input for structured element, we can apply the filter.

Part – 6: Image gradient calculation using Sobel operator

We apply the Sobel operator for finding the gradient of the image. This helps in finding out the text area in the image which can later on help in finding the Machine Readable Zone in the passport. Then we rescale the X – Gradient image using min – max rescaling.

Part – 7: Closing operation for connecting the separate nearby areas

After min – max rescaling, a closing operation is applied on the resultant image. This helps in combining the small rectangular regions formed at the MRZ zone. This is done using **imclose** function. Later on the image is threshold using Otsu thresholding technique. For this we use **graythresh** function and **im2bw** function.

Part – 8: Another closing function

We perform another closing operation, this time using square kernel to close gaps between lines of the MRZ, then perform an erosion to break apart connected components. For these we use, **imclose** and make a line structured element for erosion.

Part – 9: Finding contours

In this part, we find the contours in the thresholded image. For this we use **bwlabel** which returns the labelled image. From the labelled image, bounding boxes are made for covering up the various regions created in the image at the end of part 8. **Regionprops** and **'BoundingBox'** are used for these functions. Later on, we iterate over the bounding boxes and find the one which has the aspect ratio and characteristic ratio above that a certain limit (5 and 0.6 respectively). To take care of some thresholding and other processing errors, the detected regions are extended by 3% on both sides. We also ensure that the rectangle at the bottom most section is selected as the final one (because the MRZ lies at the bottom of the passport). This takes care of the possibility of multiple Bounding box detection. Finally, if a rectangle is detected, then it is drawn on the original image and the Region of Interest is cropped using **imcrop**.

Part – 10: Reading text from Region of Interest

Once the region of interest is detected, we use the concept of Optical Character Recognition. For this we will use the built in function, **ocr**. Since this was introduced in MATLAB R2014a, the system requirements should be according to that only. One can also create a new trained list using **OCR Trainer** app and a set of database to recognize the MRZ text more efficiently. The recognized text is extracted and then displayed on the screen using **fprintf**.