

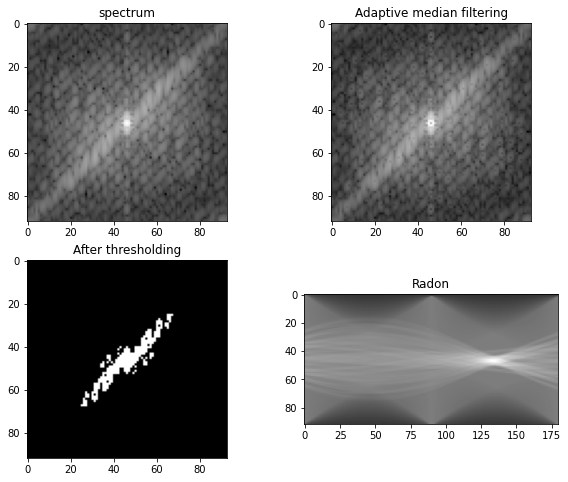
|  |  |
| --- | --- |
| License Plate  Recognition |  |
| Detecting the area of a license plate, extracting the characters out of it and subjecting the resultant image to standard OCR techniques  By Jainan Tandel (B19224), Yasir Khan (B19248), Aniruddha Prakash (B19206) |  |

|  |  |
| --- | --- |
| Introduction Our approach arose from quite basic thinking: what does one see when looking at a number plate? Obviously, some part of the car, some part of the surroundings, and then the number plate itself come to mind.  So, to recognize the number code from the plate, we had to first isolate the area in the image where the number plate was, extract the license plate, deblur it and then apply standard OCR techniques like EasyOCR to the extracted (license plate) image.  We operated primarily on (converted) grayscale images. The deblurring is discussed in the next chapter of the report. Only the recognition part is discussed here. Detailed Approach **Number Plate Area Extraction**   1. **Closing and Otsu’s method:** We coarsely identify background and foreground areas in the image by closing the original image with a square kernel and then thresholding it with Otsu’s method, which will help us obtain areas in the image that **may** contain the license plate. The bright areas obtained after this are coarse regions of interest. 2. **Blackhat Morphology:** We assumed that all number plates have a *light background with dark text* on it, which led us to perform blackhat morphology on the entire image to highlight such kind of patterns. 3. **Edge Detection:** We detect the edges of the image obtained in (2) by Sobel operator. The license plate area is expected to have significantly more edges than other areas of the image. 4. **Smoothing Image:** To combine the areas in the image obtained in (3) which may contain the number plate, we smooth it using a Gaussian blur, and then close the result with a rectangular kernel along with Otsu thresholding. 5. **Erode/dilate:** The results obtained in (4) are quite coarse. To improve the accuracy and get rid of “false areas” of number plates, we erode and dilate the image, which better highlights the license plate area. 6. **Bitwise AND:** We took the common light regions between the results of (5) and (1). Then we perform a series of erode/dilate operations on this result to clean up the image and further refine the probable areas for the license plate.     **Note:** We assumed that all number plates have a light background and dark text.   1. **Contour Detection:** We employ cv2.findContours() to find the white regions obtained as “final” above and store a pre-determined number of largest contours. Then, using cv2.approxPolyDP() we found an approximately 4-sided polygon which should resemble a number plate. Another approach we used in some images was to approximate the aspect ratio of each contour detected to match it with known dimensions of common license plates. 2. **License Plate Extraction:** We then extracted the license plate from the original grayscale image by picking up the pixels bound by the limits yielded in (7).   **Deblurring**  After obtaining the image of **only** the number plate, we deblurred the image to provide cleaner data for EasyOCR to work on. The deblurring methods are discussed in another section.  **Character Recognition**  In order to provide the ‘cleanest’ images to the OCR model, we used deblurring to enhance the readability of the number plate, seeing that cars are often in motion. The deblurring part is discussed in the next section.  EasyOCR and PyTesseract were compared for their performances. We found that the EasyOCR model delivered better performance.    Figure 1: Detected – PGOMNILZ    Figure 2: Detected - DAN 54P  Figure 3: Detected - HH 15BD8877      Figure 4: Detected – CRAIG    Figure 5: Detected - KL54A2670, TRUTH, LIFE    Figure 6: Detected - nothing |  |



|  |  |
| --- | --- |
| Blind Deblurring | |
| Since the motion of the car is generally unknown, we have attempted implement blind deblurring on the image before feeding it to the OCR model. |  |

|  |  |
| --- | --- |
| Introduction Looking at an image, the human eye can easily tell *where* the object is headed according to its motion blur. Teaching the same to a computer is a monumental task, which is why we have sought to remove the motion blur from the license plate.  Doing so helps increase the general accuracy of the process. Detailed Approach **Abstract**  After identifying the licence plate area, we attempt to remove the effect of any motion blurring on the image, using Richardson Lucy algorithm. However, the algorithm requires a known PSF to be able to deconvolve the image. To estimate this unknown PSF, which is a motion blur kernel, we need to estimate two things: the blur angle, and the blur length. There are several approaches to estimating the blur angle; we have estimated the blur angle using Radon transformation of the image spectrum.  **Steps**   1. **Padding the license plate image:** To avoid angle conversion, and the resulting errors, which can originate be trying to find the blur angle of a rectangular image, we have white-padded the image to make it a square. 2. **Finding windowed filtered binary spectrum:** To find the blur angle, we firstly take the frequency spectrum, and binarize the resultant spectrum after applying adaptive median filtering.   Finally, we apply Hann windowing to remove the frequencies due to boundary artifacts   1. **Radon transform to find blur angle:** Then, we simply find the Radon transformation of the image, and find the angle which corresponds to the highest average among its top 3 values, to find the dominating angle in the image, which in our case, corresponds to the blur angle. 2. **Find the rotated log spectrum:** Now, knowing the blur angle, we rotate the log spectrum of the blurred image, by that angle to make the blur horizontal. This results in the information of the blur length to be captured in the horizontal direction only, hence, we take the average of the log spectrum along this direction, and take the distance between the two local minima beside the maximum to be our estimate for the blur length in terms of the image width. 3. **Deconvolve the image using RL iterations:** Now that we know the blur angle and length, we simply make the motion kernel of these estimated parameters, and deconvolve the image using Regularized Richardson-Lucy algorithm with this kernel as the psf. |  |
| Blind Deblurring: Obtaining original image from the blurred image without prior knowledge of the PSF |
|  |







|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | Bibliography |  |
|  |  |  |

|  |  |
| --- | --- |
|  | Links  1. [PyImageSearch](https://www.pyimagesearch.com/2020/09/21/opencv-automatic-license-number-plate-recognition-anpr-with-python/) 2. <https://github.com/sarnold/adaptive-median/blob/master/AMF.ipynb> 3. <https://www.researchgate.net/figure/Blur-length-estimation-processes-a-The-original-blurred-image-b-The-rotated-blurred_fig3_344053535> 4. <https://arxiv.org/ftp/arxiv/papers/1802/1802.06214.pdf> 5. <https://scialert.net/fulltext/?doi=itj.2011.1709.1716> 6. <https://www.proquest.com/openview/12e652e95baeef1ca0c9f86093674a5f/1?pq-origsite=gscholar&cbl=4998668> 7. <http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/DampledLR94.pdf> 8. <https://en.wikipedia.org/wiki/Richardson%E2%80%93Lucy_deconvolution>  Other Resources Documentation, class notes, etc. |
| The given links and resources were helpful in developing the project. |