CHAPTER 4
THE PROPOSED LICENSE PLATE DETECTION AND RECOGNITION METHOD

4.1 License Plate Location

4.1.1 Pre-processing

Initially, we applied a median filter with a (3x3) kernel size. A median filter is non-linear in nature; it selects the pixel value based on the surrounding neighborhood pixels and the size of the local region considered is determined by the kernel size (i.e. 3x3 pixels around our current pixel). The reason why the median filter was chosen was that it effectively removed the salt-and-pepper noise present in the background (refer to figure 18 for a better illustration) while preserving object edges. Another option considered was the bilateral filter; it can reduce noise very well and is edge preserving. However, the drawback is that it is very slow compared to most filters, with filters with a kernel size > 5 being unsuited to real-time applications [20] . Since each image frame in the video must undergo pre-processing, the effect on running time is significant hence the median filter was chosen over the bilateral filter.

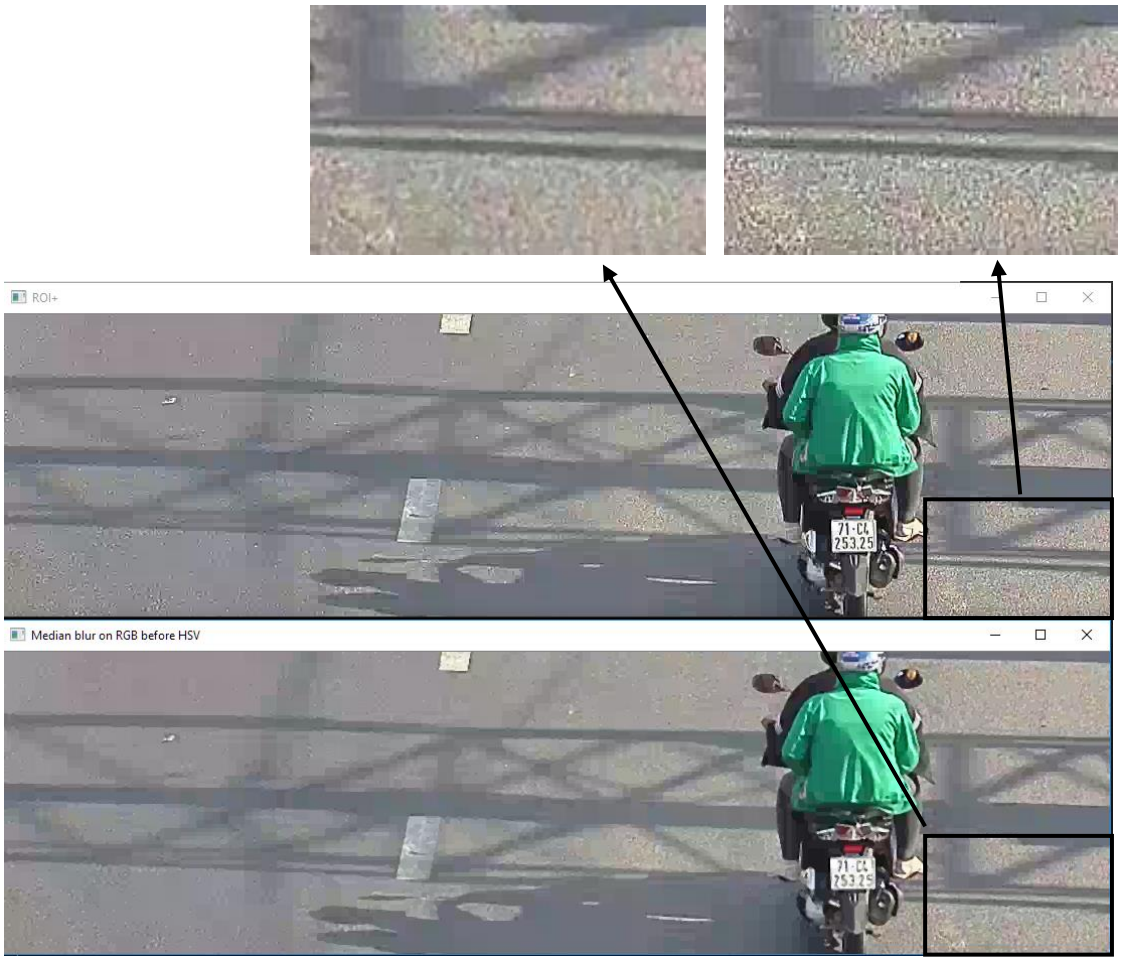

Figure 18 Top image is the original image while the bottom image is the image after applying the median filter

However, we discovered that the median filter has a significant impact on running time when applied to every frame; without the filter the frame per second value tripled. Hence we tried to find an alternative method, and we finally settled on an additional morphological opening step after color segmentation to remove the small white pixel noise on the road. This chosen method achieved acceptable results with considerable gains in performance.

Thus in the pre-processing step, we only applied a color model transformation. The outdoor conditions in the video make illumination a problem, hence the increased necessity to pre-process the illumination of images. The conversion of the BGR image to HSV reduces the effect of lighting changes, as mentioned earlier.

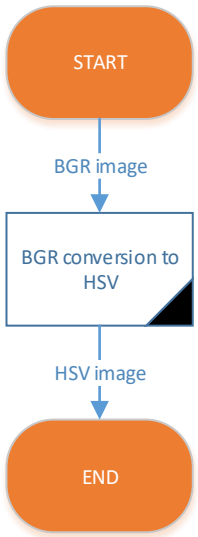

Figure 19 The pre-processing stage

4.1.2 License plate location

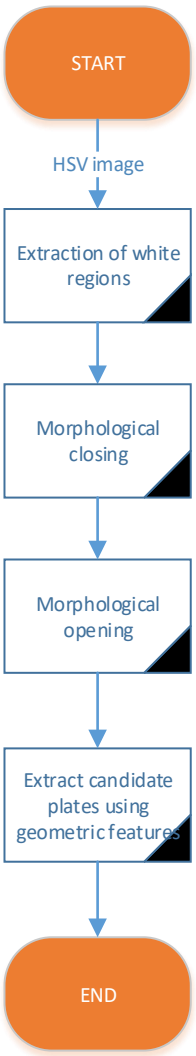

Figure 20 The candidate license plate location stage

We first identify the range to extract the white or light grey segments of the image. Our approach chose to use a rather wide range of acceptable colors because of the poor image quality and the presence of shadows that obscured the plate. The lower bound of the HSV values is (0, 0, 215) while the upper bound is (180, 30, 255). We then perform a morphological closing, with the aim of eliminating small holes or dark spots in the white objects. We then find the outline of the white objects in the image; applying a morphological closing increased the accuracy of plate region detection. In Figure 21,

the first image is the BGR image(for visualization purposes) of the region of interest. The image below is the region of interest after the white color extraction. There are still black regions inside the plate regions because of the black numbers. The bottommost window illustrates how the morphological closing closes these interior holes.

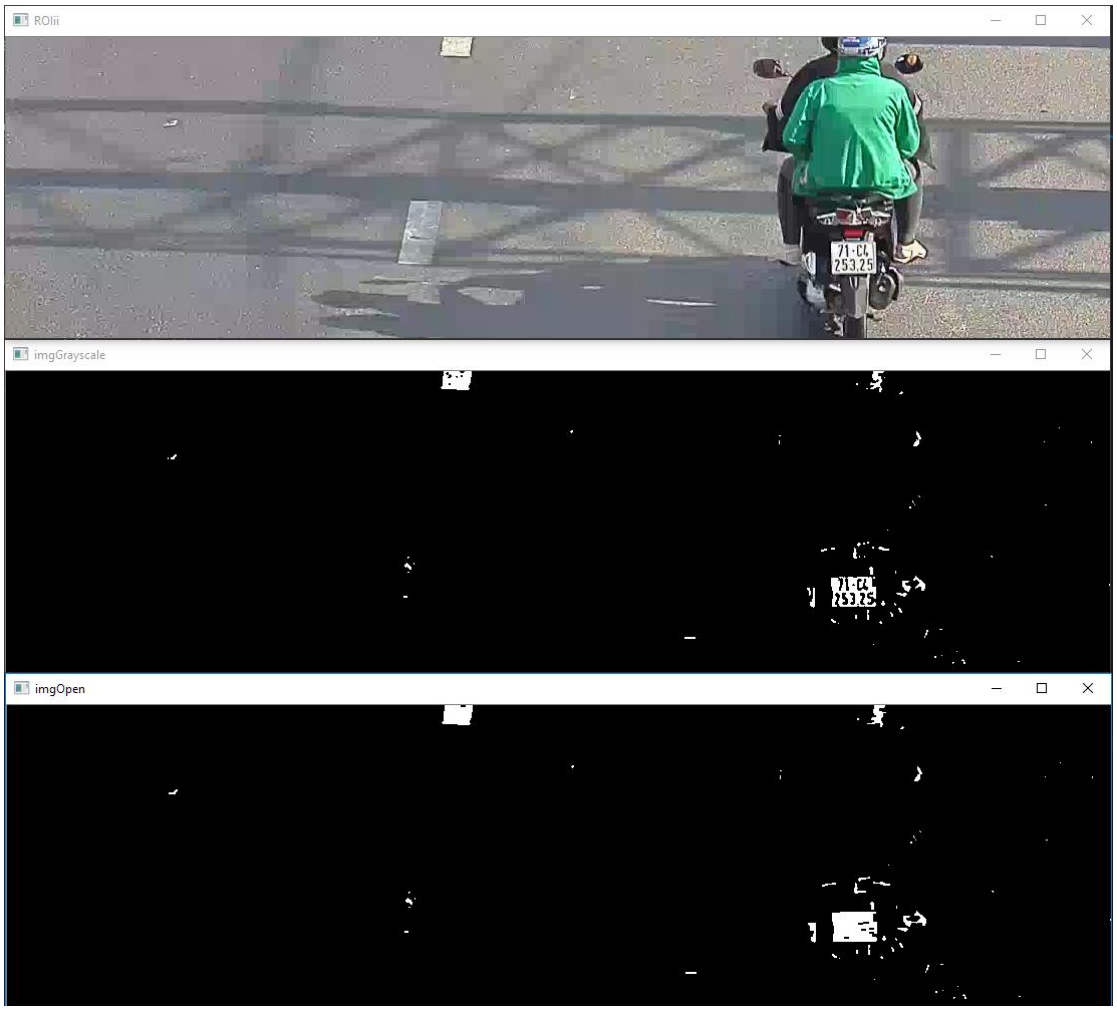

Figure 21 Visual illustration of the license plate detection method applied to a dataset

Contours that possess the following characteristics are filtered as potential plates:

- Contour Area: is the area of the white object calculated using Green's theorem, which utilizes line integrals to compute the area, with the area of a region D being:

$$A = \iint_D dA.$$

The contour area should not be too small or too large, with the chosen range being [300, 2500].

- Aspect ratio: is calculated by taking the bounding rectangle's width over height. This rectangle encloses the object's contours and has 0 degrees rotation hence it may not necessarily be the same as the minimum bounding area. Based on the dimensions of Vietnamese license plates, a perfectly extracted contour would have a width/height ratio of 1.4. Given the heavy noise in the image quality, the range was chosen to be [1.4, 1.7].

- Area of the bounding rectangle: the area of the enclosing rectangle must be in the range (700, 2500).

Since the bounding rectangle may not perfectly enclose the plate, padding was tried to achieve better complete plate extraction. We tried different padding values the image size, and finally we chose to add 1/10 of the height or width, which factors in the fact that plates can be of different sizes. The greater the height or width of the extracted image, the greater the possibility that the additional portions are actually noise. Figure 22 is an example of the benefit of padding, allowing us the extract the full characters in the upper line. However, in Figure 23, the padding only adds more noise to the image.

Figure 22 Positive effect of extracted plate image padding

Figure 23 Negative impact of extracted license plate image padding

4.2 License plate exact location

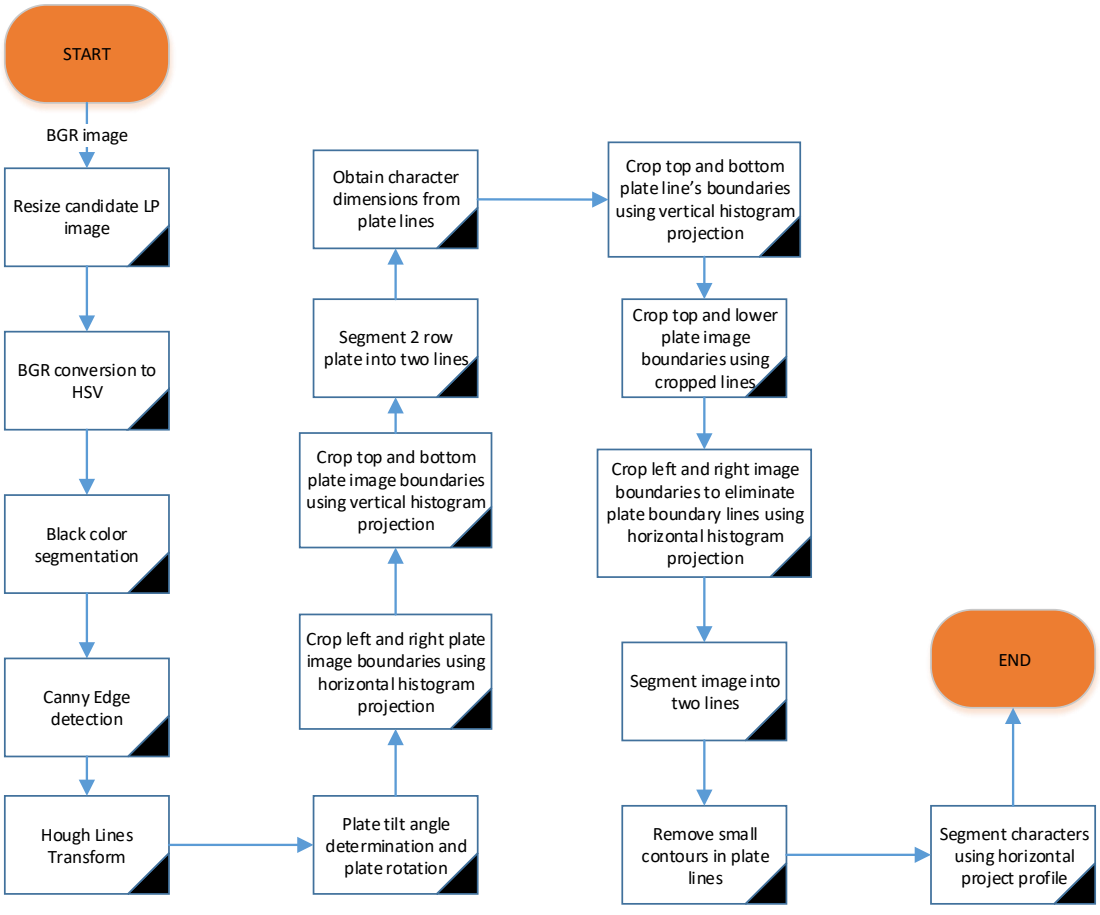

Figure 24 License plate exact location stage

In the pre-processing step, the input possible license plate is resized to (width=170, height=130) since plates can come in various sizes. The BGR plate image is then converted to HSV so that a black color segmentation operation can be carried out. We want to extract the black plate characters. Our approach chose to use a rather wide range of acceptable colors because of the poor image quality and the presence of light which makes the characters light in color. The lower bound of the HSV values is (0, 0, 0), while the upper bound is (180, 255, 160). In the figure below, the top image is the input BGR plate image, and the bottom image is the resulting image after resizing and black color segmentation. All the black colored pixels are in white.

Figure 25 Resized extracted license plate

After this pre-processing step, our next goal is to identify the rotation angle of slanted plates. This is an important step since vertical projection profiles of tilted plates do not exhibit a consistent pattern for character segmentation. We first apply Canny Edge detection to identify object boundaries. We need to pad the resulting edge detected image so that you can crop the black regions due to the rotation (if any). This is clearly illustrated in the third image in Figure 26. The rotated image has black regions as a result of the rotation and this will be noise in the histogram projection profile. We crop the padded borders after the rotation as shown in the fourth image in Figure 26.

Hough Lines transform is applied to extract the plate boundaries. We have found that the line representing the horizontal plate boundaries can be determined more accurately than the vertical plate boundaries. Angles less than 45 degrees are classified as belonging to the horizontal plate boundaries, and the average angle is of all these lines is computed. This angle is taken to be the tilt angle of the plate. Hough Lines Transform may determine more than 1 line and the underlying assumption with the computation of the average angle is that the number of lines that represent the plate boundaries is larger than false negatives. With this angle, we can correct slanted plates as shown in the 2nd and third image in the figure below.

Figure 26 Angle determination and plate rotation example

In projection profiling, the idea is to sum the pixels in a row or a column depending on whether it is vertical projection or horizontal projection. The more formal mathematical definition of vertical and horizontal projection is:

$$P_v[i] = \sum_{j=1}^{M} S[i,j]$$

Where vertical profile (represented by the vector Pv of size M) is sum of the black pixel to the vertical axis.

$$P_h[j] = \sum_{i=1}^{N} S[i,j]$$

Where horizontal profile (represented by the vector Ph of size N) is sum of the black pixel to the vertical axis.

Projections are used for character segmentation in many text recognition systems. Analyzing the horizontal projection of a plate we see that the space between characters has a sum of zero for the sum of black pixels. We can utilize this observation to extract the individual characters.

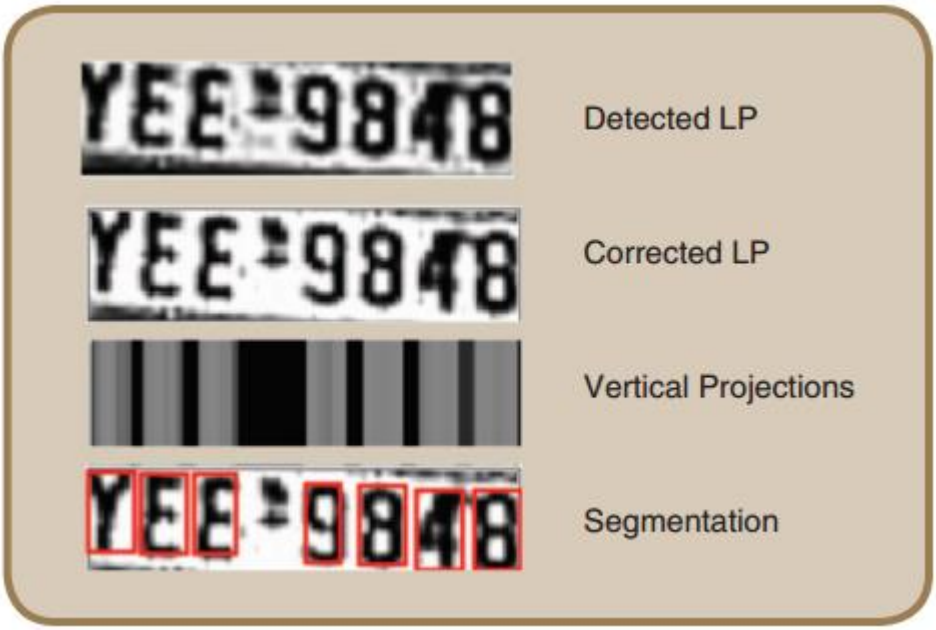

Figure 27 Histogram analysis for character segmentation example

It was noted that the projection profiles of the many of the extracted plates after the angle correction step still did not exhibit a noteworthy pattern that could be used to split the plate into two lines, as illustrated in Figure 27. In the figure below, the top left and bottom right images are the vertical and horizontal projection profiles of the plate image at the top right corner. Near the plate boundaries the plate should be completely white, hence the plate boundaries should have the least number of white pixels (corresponds to black pixels in the original plate image). If we used the histogram to identify 2 rows/columns with the two least number of white pixels in the plate image below, we would get the two circled rows/columns. However, these are not the correct plate boundaries (indicated by the green arrows).This noise in the initial cleaned plated is was found to be too significant in many of the plate images; the problem is exacerbated by the plate padding step. Thus one of the key ideas was to clean the image successively using histogram projections, slowly eliminating exterior plate regions that may be present due to the padding.

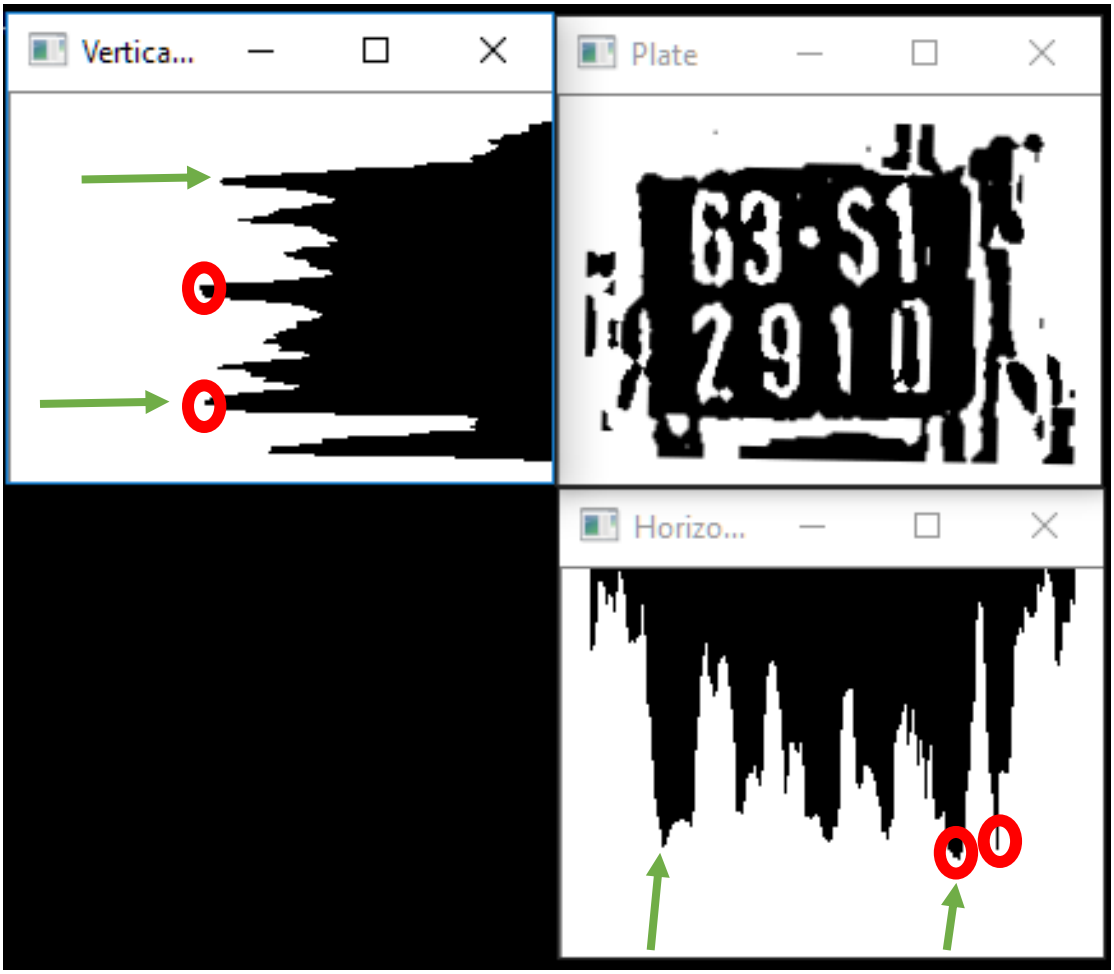

Figure 28 Analysis of vertical and horizontal histogram projections of an extracted license plate

We first calculate the horizontal projection profile, which is then used to remove part if not all the external plate regions to the left and right of the actual plate. Below is the algorithm is used to determine the new left border of the horizontal plate borders. The idea is that we will find the row with the minimum number of white pixels until we encounter an increasing slope that spans for more than a specific number of rows. As we have shown earlier, the two rows with the minimum number of white pixels does not always correspond to the plate boundaries. An increasing slope that spans on for too many rows indicates a local/global maximum and we may have reached the histograms of plate characters, hence this is our stopping condition. There are two user-defined values in the algorithm: the width of the increasing slope at which we stop

searching for a lower minimum number of white pixels, which is defined as 4 and the requirement is that the minimum number of pixels must be at least less $7/10^{th}$ of the maximum number of white pixels in the histogram. We found that those two values gave the best results in cropping the horizontal image borders such that they were closer to the plate boundaries and did not cut off plate characters. The algorithm m to determine the right border to crop to is similar to the one we have outlined, the difference only is that we iterate from the left border to the right border instead.

Algorithm 1: Determining the left border to crop the horizontal image borders so they are closer to plate boundaries

Input: The horizontal histogram of the Plate Image

Output: The cropped image

procedure CropXaxisBorders

Begin

min_left ← GetHistogramValueAtColumn(0)

left_index ← 0

no_consec_dec_cols ← 0

for each column in the histogram **do**

currentColValue ← GetHistogramValueAtColumn(current)

if min_left < currentColValue **then**

min_left ←currentColValue

left_index ← current column index

no_consec_dec_cols ← 0

else

no_consec_dec_cols ← no_consec_dec_cols + 1

```
            endif
            if no_consec_dec_cols >= 4 and min_left < 0.7*GetLargestHistoVal() then
              break
        end for
        End
    end procedure
```

Figure 29 illustrates the results after the rough cleaning of the horizontal plate borders using the aforementioned algorithm. The green arrows point to the increasing slopes that resulted in the algorithm stopping its search for a border index. We can see that the resulting cropped image has much less noise than the before middle image in the diagram below.

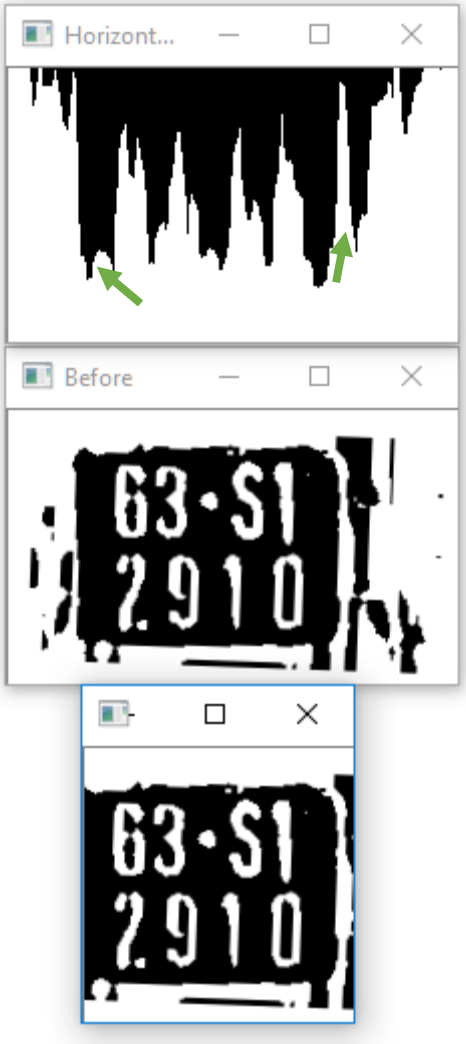

Figure 29 Cleaning the vertical plate borders of an extracted plate image

With the horizontal boundaries cut well, the less the noise in the horizontal histogram and hence the less the noise in the vertical histogram. This is because the surrounding area around the license plates is often dark in color: motorbike plates are often attached to a black plastic rear, and the roads are a dark grey in color. Hence, the exterior regions in the padded image will largely constitute of dark areas detected by the segmentation step. This means that the histogram projection values of the exterior regions will be high. When we crop the left and right borders well, the vertical histogram projection profile will have two very big white-filled parabolic-shaped

regions at the upper and lower portion of the histogram(see Figure 30). We then calculate the vertical projection profile, which is used to remove the exterior plate regions. Drawing the vertical projection profile, we noticed that we could utilize the fact that the line separating the two plate lines would give a very low number of white pixels value to design a better algorithm to crop the vertical plates. The idea was that we would identify the 3 rows with the 3 least number of white pixels and calculate the plate size from there. This plate size can tell us whether we have the chosen a good border to crop the image. In the figure below, we can see that the difference between the topmost red circled minimum and the middle red circled minimum gives one half plate size value and the difference between the 2nd red circled column and the lowest red circle column gives another half plate size value. We will 2 columns that give the larger half plate size value from the aforementioned two. With this value we can calculate the third bottom/top border. This step takes into account the fact that our top and lower borders may not perfectly correspond to the plate boundaries due to noise in the right and left of the image.

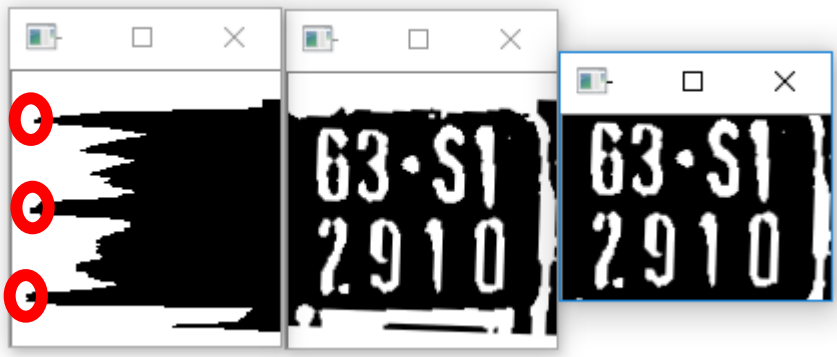

Figure 30 Analyzing the vertical projection profile of extracted plate image that has already undergone one plate border cleaning

There is also the case when the 3 minimum values only crop out half of the plate, like in Figure 31. The noise in the image resulted in the incorrect identification of the three indexes. In this case, we notice that the two half plate values would differ significantly from one another and this observation was used to identify whether we had only cut one half. If such was the case we would use the larger half plate size value to calculate the third bottom/top border. As you can see in the figure below, such an approach does allow us to clean the vertical plate boundaries, though not as accurate as in the ideal case. As we slowly crop the plate image, we progress towards an accurate extraction of the plate with the boundaries removed.

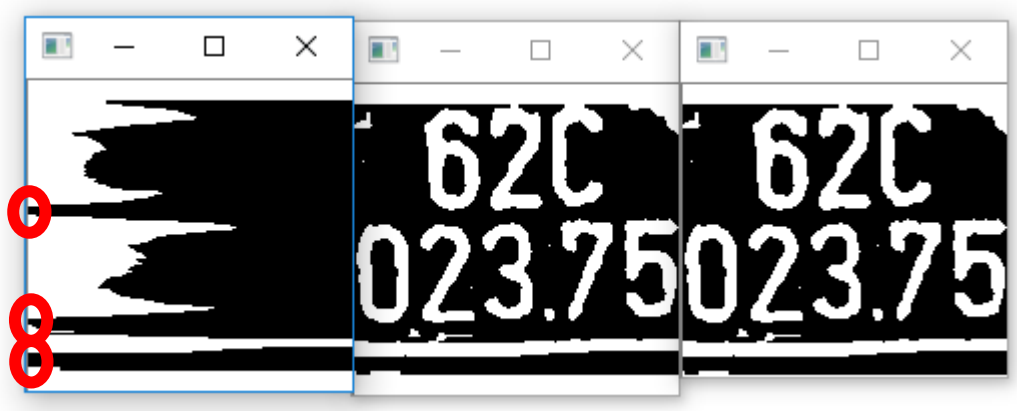

Figure 31 An exception case for the vertical plate borders

At this point, we found that the histogram profile of the plate was sufficiently clean enough for the two-row plate to be able to split into 2 lines-the upper line and the lower line. To deal with scenarios where the dividing line between the two plate lines is not the minimum, we introduce the idea of weights, with rows closer to the halfway point in the y-axis experiencing an increase in the adjusted number of white pixels value because of the larger factor value, while points further from the halfway mark will experience smaller increases or even a decrease in the adjusted number of white pixels value. Thus, there are two determinants: the number of minimum white pixels and the distance from the halfway point. Using two factors results in better plate line

segmentation. In the figures 32 and 33 below, the row with the minimum number of white pixels is at the closer borders to the edge of the plate boundaries and the weight factor corrects this so the plate is still split correctly.

Figure 32 An example of the line segmentation of a plate with a noisy vertical projection profile after undergoing two rounds of plate borders cleaning

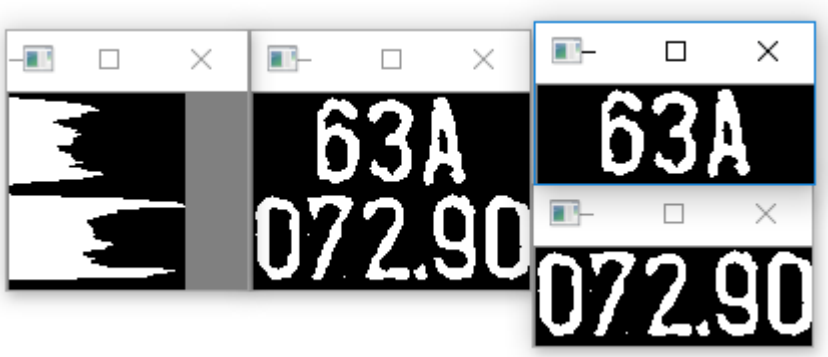

Figure 33 Another example of the line segmentation of a plate with a noisy vertical projection profile after undergoing two rounds of plate borders cleaning

We then extract contours that meet certain character requirements, which are:

- Width/height ratio: the bounding Rectangle's width/height ratio should be in the range (0.25,0.6)
- Bounding rectangle width and height: the width should be larger than 10 and the height should be larger than 30
- Contour area: the contour area should be larger than 30

We then group the possible characters because incorrectly defined contours may be identified. Those characters that have a height or width difference of 7 or less pixels and an aspect ratio difference of less than 0.11 are placed into the same groups. The underlying assumption is that the group with the most number of characters contains the plate line characters. The biggest group is taken to contain the height and width information of the plate line characters. For the upper plate, we can use the upper row index of the bounding rectangle to crop the top of the plate line and the lower row index of the bounding rectangle to crop the bottom of the plate line. In the upper plate line(Figure 34), we could only detect the number 6, since the other 2 numbers are joined with the white borders. There is only one group in this scenario and the upper and lower y-coordinates of the number 6 will be used to crop the upper and lower boundaries of the plate line to obtain the rightmost image in Figure 34. Similarly, the same operation is applied to the lower line(Figure 35). We are able to detect 5 characters and since this group is the largest, the plate line borders are cropped using the minimum upper boundary of a character and the maximum lower boundary of a character in the group. This results in the bottommost image in figure 35.

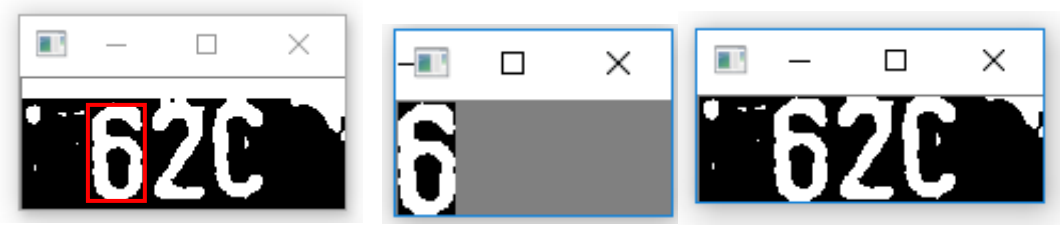

Figure 34 Refining the top and bottom borders of the extracted upper plate using candidate character contours

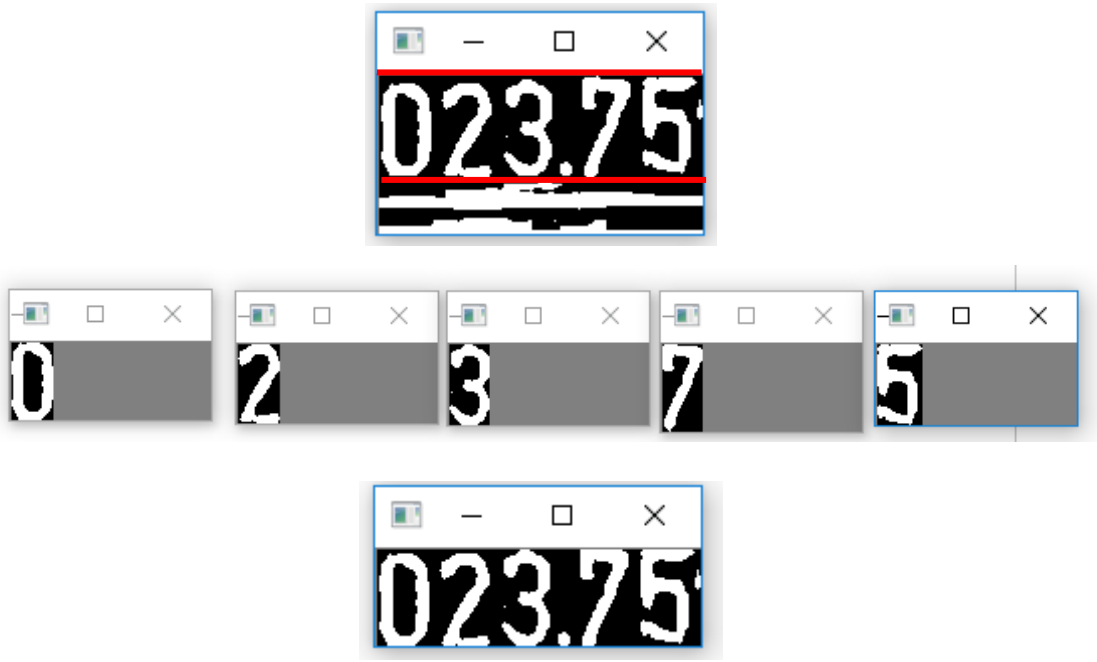

Figure 35 Refining the top and bottom borders of the extracted lower plate using candidate character contours

As illustrated in the figure below, we can use the upper row of the bounding rectangle in the upper plane line and the lower row of the bounding rectangle in the lower plate line (two middle images in the Figure 36) to obtain a much cleaner cut of the input plate image (the third image on the far right has a much cleaner vertical cut than the original at the far left).

Figure 36 Cropping the input plate image using the cropped upper and lower lines

Sometimes the broken long lines of the plate boundaries are still present (see the boxed red and green rectangles in the middle image in Figure 37). We will eliminate these misleading broken long lines at the vertical plate boundaries using the contour height: these lines will have a bounding rectangle height spans more than 3/5 of the plate height. The cropping x coordinate for the left border will be the midpoint of the bounding rectangle to the left of the plate's midpoint closest to the midpoint. Similarly, the cropping x coordinate for the right border will the second rectangle to the left. We then can use these tighter boundaries to crop the left and right sides of the 2 plate lines, obtaining the third image in Figure 37.

Figure 37 Removing plate borderlines

We then remove small contours from the plate lines to reduce the noise in the histogram projections. Small contours are those with a bounding rectangle area of less than 30 (2nd set of images from the left in Figure 38). We also apply a morphological opening after removing the contours, which also helps to remove noise further (3rd set of images from the left in Figure 38). In Figure 38, we can clearly see that the rightmost set of images are much cleaner than their counterparts on the far left.

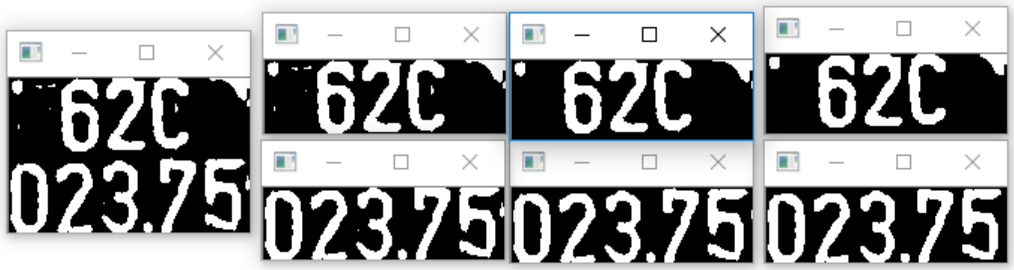

Figure 38 Removing small contours to reduce noise in the histogram projection

After that, we segment the characters based on the rows with zero white pixels in the histogram projection. The segmented characters obtained are those in Figure 39.

Figure 39 Final horizontal histogram projection to segment characters

4.3 Method for Traffic Signs Recognition

The last phase is character recognition. The extracted characters are resized to standardise the size, the features are then extracted and passed to the classifier. We chose to use the Support Vector Machine (SVM) because it is easy to use, produces accurate predictions with the correct feature extraction, high performance and is well integrated with OpenCV.

4.3.1 Creating the training datasets:

One source we used was the correctly extracted characters from the 15:00-1600h video (from cameras installed at Trung Luong Intersection (Ngã 3 Trung Lương))to create classes with 150 images. For characters where there are not enough extracted

data to form 150 images, we replicated the existing data in the class until we had enough images. We had to ensure that each class had the same number of images to ensure the prediction was not skewed towards classes with more images. Some characters occurred more frequently than others and we found that 150 images was a reachable target.

Figure 39 Data from 15:00-16:00 Trung Luong Intersection video

Aside from the video, we also photographed license plates from the IU parking lot, after gaining permission from the persons in charge. We also extracted characters from these plates.

Figure 40 IU Parking Lot images

A third source was the characters we generated using the same text font as that used in the plates. We were able to obtain the matching font from the internet and these characters represent the ideal case.

Figure 41 Optimal data images created using fonts

The choice of which images to include in the dataset affected the prediction quality. If we included the characters of poor quality (blurred, few identifiable characteristics for classification), then our SVM would more likely return character predictions for non-character input data.

We will train two classifiers, one to recognize the numbers and one to recognize the letters. In the numbers SVM classifier, we will have 11 classes, 10 classes for the numbers 0 to 9 and the 11^{th} class is for negatives (non-numbers). In the letters SVM classifier, we have 21 classes, 20 classes for the letters A,

B,C,D,E,F,H,K,L,M,N,P,R,S,T,U,V,X,Y,Z and the 21st class is for negatives (non-letters). The images are grayscale images with the characters in white and the background being black.

For the feature extraction, we used Histogram of Oriented Gradients (HOG). HOG is a widely used method for extracting object features which then can be used to identify whether an input image is the same object. The object features that are computed and stored are the gradient orientations of the pixels in an image. The image is divided into squares of a user-defined size (referred to as cells), and the gradient direction histograms are built for each of these cells. The histograms are normalized based on the intensity over image blocks. Normalization results in increased accuracy because it makes the histogram of gradients less susceptible to lighting changes and shadow variations in an object [21]. For the HOG, the input character is first resized to (64, 64). We played around with different window stride sizes, block size and block stride. The smaller the size chosen, the greater the number of histogram of features calculated hence the slower the prediction will run. After trying around with different possible combinations, we found that the following sizes did not lead to any visible decrease in prediction accuracy: the window stride size is (32, 32), block size is equal to the (16, 16) and the block stride chosen is (8, 8).

4.3.2 Importing the datasets in OpenCV

- **Step 1:** Two matrixes are needed to store the data passed to the SVM classifier:

  The first matrix contains the extracted features of all the training character images. The number of rows in the matrix is equal to the number of characters passed to the training, the column size is the area of the binary image. Say a binary image has the size (60x30) then the column size is 1800.

  The second matrix contains the labels for the corresponding feature in the first matrix. It has the same number of rows as the first

matrix, so the label of the feature at the ith row in the first matrix is at the ith row in the second matrix. Column size is only 1 since the labels are an integer value. If say our training data only has two classes of data, one set representing birds, another representing non-birds the labels used are 1 and -1 correspondingly.

- **Step 2:** Tuning the SVM parameters

  svm_type: since we have 10 possible numerals and 20 characters to train, we chose to use C_SVC, which is used when the number of classes we need to identify is higher than 2.

  kernel_type: since the amount of training image samples in each is small, only 150 images per class, the LINEAR kernel type was chosen; no mapping is done and linear regression is carried out in the original feature space.

  term_crit: is the terminate condition for the training step. It can either be the tolerance or a specified maximum number of iterations. The value used is TermCriteria(TermCriteria::MAX_ITER, 100, 1e-6).

  Cvalue: the smaller the Cvalue the larger the error rate of classification, the larger the Cvalue, the smaller the error rate, but the harder it is to calculate weights that separate the classes to that accuracy.

- **Step 3:** Train the SVM classifier with the aforementioned parameters, with the results saved to an XML file

4.3.3 Training the SVM classifier

After training, we save the SVM classifier we trained, specifically the support vectors and the chosen SVM parameters in an XML like below:

HOG_SVM_letters_negatives_LINEAR_(winStride_32)_small_trainingDatav3.xml

```
<?xml version="1.0"?>
<opencv_storage>
<opencv_ml_svm>
  <format>3</format>
  <svmType>C_SVC</svmType>
  <kernel>
    <type>LINEAR</type></kernel>
  <C>1.</C>
  <term_criteria><iterations>100</iterations></term_criteria>
  <var_count>144</var_count>
  <class_count>21</class_count>
  <class_labels type_id="opencv-matrix">
    <rows>21</rows>
    <cols>1</cols>
    <dt>i</dt>
    <data>
      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20</data></class_labels>
  <sv_total>210</sv_total>
  <support_vectors>
    <_>
      -2.65129954e-001 -1.07364669e-001 9.10163950e-003 1.37065738e-001
      -2.63553828e-001 -7.61295334e-002 -7.22899288e-002
      -7.35465139e-002 -2.90973753e-001 -2.18840670e-002 9.44733918e-002
      1.32168874e-001 3.04935992e-001 2.04539429e-002 -1.94516897e-001
      -1.84964612e-001 -1.85252845e-001 -6.11608922e-002 4.51923497e-002
      1.18029892e-001 1.71718329e-001 2.97325462e-001 -1.64181113e-001
```

Figure 42 SVM output xml file