18-551

Group 18:

License Plate Recognition

Final Report

Pierre Ponce | Stanley S. Wang| David L. Wang

Ponce | sswang | wang4

The purpose of this report is to explain the implementation of our project, "License Plate Recognition System". This report will begin with sections on motivation, past projects, and constraints. It will then proceed to describe our system in broad terms to provide a general overview of our project to the reader. It will then describe each subsystem in detail. For each subsystem, we have included explanations for why we chose our methods, performance of our methods, under what conditions would our methods fail, and how can we improve our methods. The report ends with a discussion on possible future work using more evolved image processing techniques and acknowledgements.

Table of Contents

Introduction

Monitoring vehicles for law enforcement and security purposes is a difficult problem because of the number of automobiles on the road today. An example is this lies in border patrol: it is time consuming for an officer to physically check the license plate of every car. Additionally, it is not feasible to employ a number of police officers to act as full-time license plate inspectors. Police patrols cannot just drive in their cars staring at the plates of other cars. There must exist a way for detecting and identifying license plates without constant human intervention. As a solution, we have implemented a system that can extract the license plate number of a vehicle from an image – given a set of constraints.

Past Projects

Although there have been no similar 18-551 projects in the past, we have been able to find a few research papers on the topic of license plate recognition systems as well as several commercial license plate recognition systems. Among the research papers: Hermida, Xulio; Fernandez, Fernando Martin; Rodriguez, Jose Luis Fernandez; Lijo, Fidel Pita Sande; and Miguel Perez Iglesias. "An OCR for V.L.P's (Vehicle License Plate)", ICSPAT, 1997. Among the commercially available systems: Hi-Tech Solutions, *See/Car LPR* (http://www.htsol.com/Profile.html); Electro-Optical Technologies, Inc., (http://users.erols.com/lnelson/lpir.html).

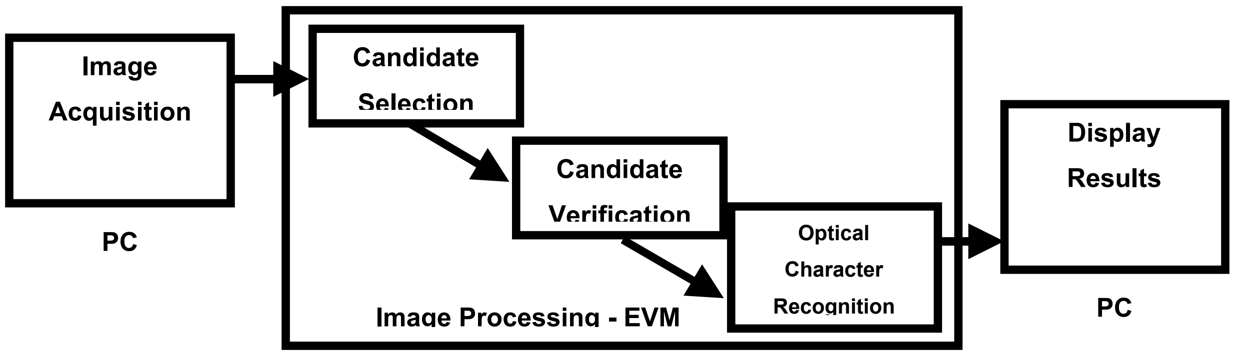
The Constraints

Due to the limited amount of time we have, a set of constraints have been placed on our system to make the project more manageable, they are as follows:

- Image of the vehicle taken from fixed angle.
- Image of the vehicle taken from fixed distance.
- Vehicle is stationary when the image was taken.
- Only standard Pennsylvanian license plates will be dealt with *(figure 1)*.

Figure 1, standard Pennsylvanian license plate

System Overview



Our license plate recognition system can be roughly broken down into the above block diagram. The image acquisition is done by hand via a digital camera and stored on the PC; an interface on the PC side then transfers the image from the PC to the EVM, where the actual image processing is occurred; at the end of image processing, the system returns to the PC the license plate number of the vehicle in the image. The details of each subsystem are to follow in the following sections.

Image Acquisition

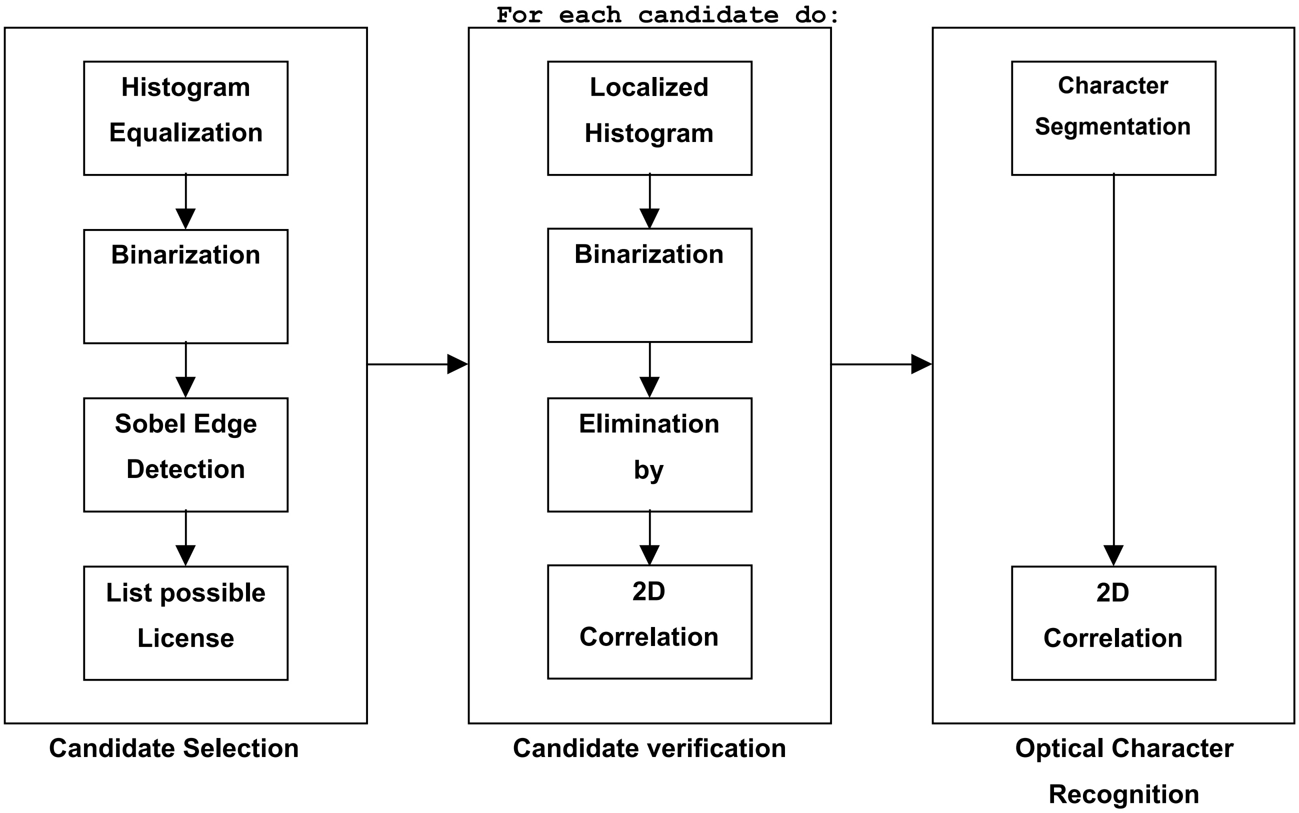
Description of the method:

The images of vehicles were taken with a Sony digital camera, borrowed from the MEMS lab, with a resolution of 480x640. On average, the images were taken seven feet away from the vehicle. They were stored in color JPEG format on the camera. We use Matlab to convert the color JPEG images into gray scale raw format on the PC. An interface on the PC side then transfers the images to the EVM for processing.

Rationale:

We chose the resolution of 480x640 because it is the highest resolution the digital camera would allow. We want to take the image as far away from the vehicle as possible such that it is more similar to a real-world system and more challenging. Given the 480x640 resolution, we can be seven feet away from the vehicle and still capture the license plate with adequately clarity.

Image Processing Overview:



Once the gray scale image is received by the EVM from the PC, the image processing process begins. Image processing can be further broken down into three phases: Candidate selection, candidate verification, and optical character recognition. First, the system uses edge detection to compile a list of candidates. Second, it chooses a candidate to be the license plate based on discrimination operations performed on the original gray scale image. Lastly, optical character recognition was performed on this region designated as the license plate. The details of the subsystems are discussed in the following sections.

Image Processing:
Candidate Selection → Histogram Equalization

Description of method:

Histogram equalization is an image transformation that computes a histogram of every intensity level in a given image and stretches it to obtain a more sparse range of intensities. This manipulation yields an image with higher contrast than the original. The process is based on the creation of a transfer function that maps the old intensity values to new intensity values. For instance, if we let T be the transfer function, 'T(34) = 44' denotes that each pixel value of 34 present in the image will be replaced by a 44. The transfer function is based on the computed probabilities of each intensity value appearing on the image. This has the end effect of spreading out the most commonly appearing pixel intensities to cover a larger range. The following images demonstrate the effect of histogram equalization on sample input images.

Rationale:

The candidate selection process utilizes the guaranteed contrast between the characters and the license plate. Therefore, it is crucial for us to preprocess the image in enhance contrast. To increase the contrast of the gray scale image from the PC, histogram equalization is used. As shown below, the histogram equalized image on the right has much better contrast, especially around the license plate region, than the original image on the left.

Before Histogram Equalization

After Histogram Equalization

Image Processing:
Candidate Selection → Binarization

Description of method:

For a given gray scale image, we examine the intensity value of each pixel. If it is above a threshold, we mark it as white; otherwise we mark it as black. The threshold chosen for the candidate selection process is 102 (given intensity values ranging from 0 to 255). This threshold is chosen based on examining and experimenting with our training set images.

Rationale:

Having only black and white pixels makes the image much easier to work with, as demonstrated below with the example of Sobel Edge detector output:

Output of Sobel edge detector of binarized image

Output of Sobel edge detector of gray scale image

As one can clearly see, a binarized image is much less confusing to the edge detector. Thus binarizing the image is desirable in candidate selection.

Possible Problems/Weaknesses:

Although the binarization threshold is chosen based on extensive experimentation, there is always the chance that the threshold we chose will not work well for some images, in spite of the histogram equalization.

Possible Solution:

Instead of using an arbitrary threshold, perhaps an adaptive threshold is possible.

Image Processing:
Candidate Selection → Sobel Edge Detection

Description of method:

Given the binarized image from the previous step, we perform vertical edge detection with *mask A* and horizontal edge detection with *mask B*. Since we are not interested in the direction of the edge, we take the absolute value of the output of the mask to obtain edges present in all four directions (Thanks to Pete for this idea). Wherever an edge is present we mark the pixel as a 1, otherwise, we mark the pixel as a 0.

$$\begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \qquad\qquad \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

Mask A Mask B

Rationale:

If the binarization threshold is appropriate, performing edge detection on the black and white image should result in a great deal of edges in the area of the license plate due to the characters. This is a property we can use to choose possible license plates.

Possible Problems/Weaknesses:

The 3x3 masks we currently use detect edges that are relatively thing comparing to the lines of the characters on a license plate. Therefore, the edge detector will detect much more thinner edges in addition to the edges of the characters of the license plate.

Possible Solution:

Construct masks that specifically detect lines that are roughly the thickness of the characters on a license plate to minimize false alarms.

Because we were strained for time toward the end, we were only able to use the quadruple “for” loop to performing the edge detection. Although we unrolled the inner loops, it is still not nearly as fast as it could be.

Possible Solution:

Do what we did in lab 3. Move chunks of image into on-chip memory and then performing the edge detection.

Image Processing:
Candidate Selection → Choosing Candidates

Description of method:

Knowing the approximate distance of the vehicle, we can estimate the size of the license plate. In our case, the license plate is roughly 23x79 pixels. To actually locate possible candidates, we start at the upper left hand corner of the Sobel edge detector output and count the number of edges within a 23 x 79 window. If the number of edges is within 900 - 1500, then we mark that coordinate as a possible candidate. The window is then moved in increments of 5 pixels to cover the entire Sobel edge detector output. At the end of candidate selection, we end up with a list of possible candidates for the license plate.

Rationale:

Because of the characters, a license plate has a typical number of edges present. To extract a list of candidates, the most straightforward solution is to look for regions that have this typical number of edges. The range of 900 to 1500 was chosen based on experimentation: in all our training set images, no license plate has an edge value that is lower than 900 or higher than 1500. By moving the window in increments of 5 pixels allows this part of the system to run faster without sacrificing much accuracy.

Possible Problems/Weaknesses:

The candidate selection routine (locate) is the slowest part of the entire system because it has to sum the pixels values of a 23x79 window many times.

> **Possible Solution:**
>
> Since we are moving the window in increment of 5 pixels, there exists a lot of overlapping. Instead of summing the entire window every time, we can save portions of the current window and only compute a small portion of the next window and sum them together to save computation time. This was not implemented due to time constraints.

Image Processing:
Candidate Selection → Choosing Candidates (continued)

Although the range of 900 to 1500 always captures the true license plate as a candidate, the generous range also results in a large list of candidates at times. On average, an image has 56 candidates. However, a few images can have up to 200 candidates. This is a serious setback on performance for candidate verification.

Possible Solution:

Make the range adaptive. Perhaps start out with a narrow range and iterate through the system. If the system cannot find the characters, then increase the range.

Image Processing:
Candidate Verification → Local Histogram Eq./Binarization

Description of method:

A candidate region is copied from on-board memory into on-chip memory to improve the performance. The region is then locally histogram equalized and then binarized with a threshold value of 153 (given intensity values ranging from 0 to 255). This threshold value was derived by examining the specific grayscale intensities on the edges of characters on a set of license plates.

Rationale:

Given the list of candidates, we proceed to choose one to be the license plate. Black and white images are easier to work with, therefore, we chose to locally histogram equalize each candidate and then binarize. The local histogram equalization enhances contrast and yields much better binarization result. As shown below:

Each candidate region is copied from on-board memory to on-chip memory because we perform a good deal of operations on the region, therefore, the performance will increase if we were to copy the region to the faster on-chip memory first.

Image Processing:
Candidate Verification → Elimination by Heuristics

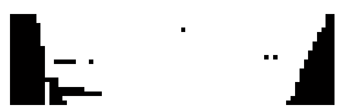
Description of method:

Three simple heuristics were used to eliminate obviously wrong candidates:

1. Check for large amount of white pixels on left or right:
   - Given 23x15 regions on the left and right end of the candidate, if either region contains more than 250 white pixels, the candidate is eliminated.

2. Check for large amount of black pixels on left or right:
   - Given 23x15 regions on the left and right end of the candidate, if either region contains less than 60 white pixels, the candidate is eliminated.

3. Check for thick white column in the middle of region:
   - If there exists a solid white column of more than four pixels wide in the candidate, it is eliminated.



Rationale:

We frequently have to discriminate against a large amount of candidates. Instead of computing the FFT and 2D correlation on each one, we can improve the performance somewhat by eliminating the obviously wrong candidates.

Image Processing:
Candidate Verification → Elimination by Heuristics (continued)

Possible Problems/Weaknesses:

Sometimes the license plate will have a large amount of white or black pixels on the left or right end. This is usually because the license plate is either too large or too small. If the true candidate were eliminated, the candidate verification phase would choose an inferior candidate to be the license plate, thus causing problems in OCR.

Possible Solution:

Make sure that the image was taken from the designated distance. Not much improvement could be made to the heuristics – they have been tweaked to death during our Matlab prototyping phase.

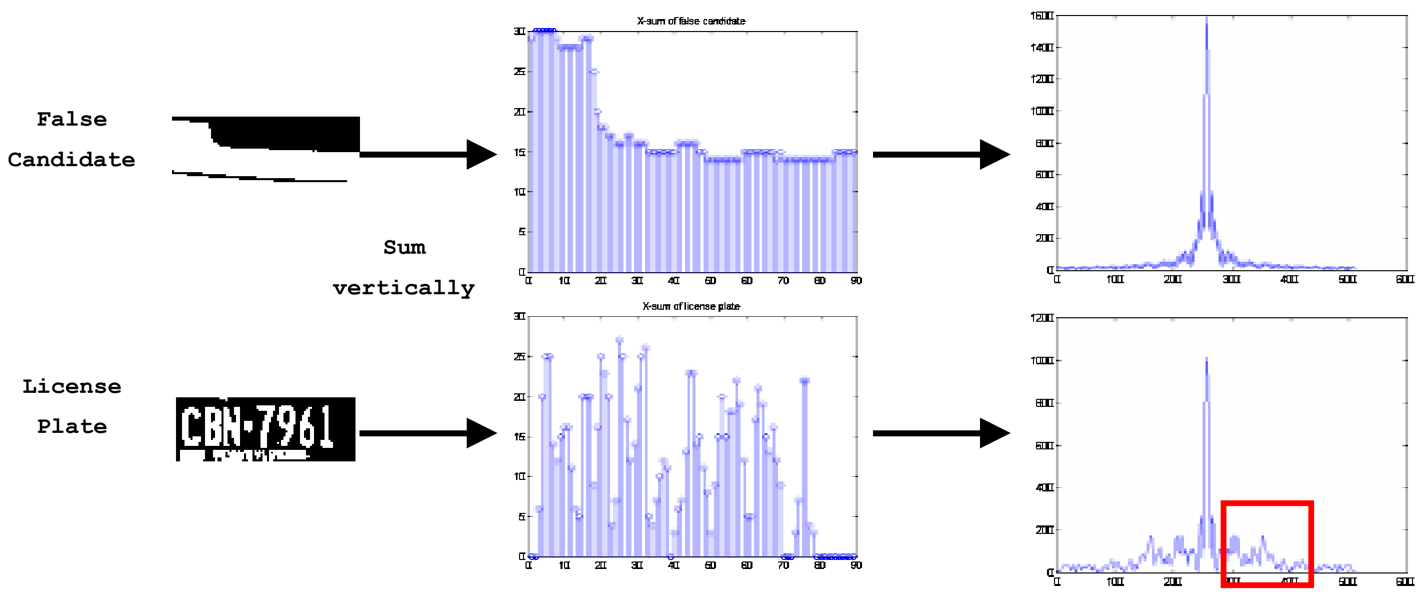
Image Processing:
Candidate Verification → Elimination by Periodicity

Description of method:

We collapse the candidate onto its x-axis and take the FFT of the resulting 1D signal to check for periodicity. A typical license plate has high frequencies (meaning .31 pi to .5 pi) present. Therefore, we examine the value of the FFT in this range and if they do not fall within a range of 9500 and 14000, we eliminate the candidate; the range is obtained though experimentation.

Rationale:

A license plate has a typical periodicity due to the regularly spaced characters, as shown below. This property can be used to further discriminate against the candidates.



Possible Problems/Weaknesses:

If the 1-d column sum is similar to that of the correct candidate, it will let it through. This happens when the candidate contains part of the license plate, perhaps 3-4 characters of the license plate.

Possible Solution:

Add 2-d Correlation to further eliminate candidates.

Image Processing:
Candidate Verification → Choose License Plate by 2D Correlation

Description of method:

The remaining candidates were correlated against a license plate template (shown below). The candidate with the highest correlation output was chosen to be the license plate.

Rationale:

In the earlier versions of the system, we had planned to scrutinize the FFT to choose the license plate. However, we realized that the FFT alone is not enough to discriminate between a candidate that has three-fourth of the license plate in it and the true license plate. Therefore, we added the 2D correlation routine.

Possible Problems/Weaknesses:

One of the major problems in 2-D correlation is that it is very sensitive to the size of the license plate within the picture. This could result in a higher correlation for a candidate that does not have the license plate rather the candidate with the license plate

Possible Solution:

Adaptive Scaling of the picture to make sure the license plate is always a certain size could be a possible solution. Better photography skills are also another solution so that the variance of the size of the license plate is very small, which is probably due to angle and distance of the camera shot.

Image Processing:
Character Recognition → Segmentation

Description of method:

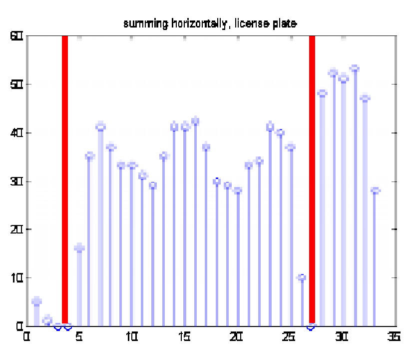
After the license plate has been chosen, the character segmentation begins. This routine starts off by extending the license plate region five pixels in each direction. In another words, the 23x79 region chosen to be the license plate would be 33x89. This was done to ensure that we have all the characters in the region.

The process of character segmentation can be broken down into the following four steps:

1. Vertical segmentation:

   The chosen license plate is collapsed onto its y-axis. We then search the row of characters by looking for at least fifteen continuous values that are within 22 (low threshold) and 50 (high threshold) in the 1D signal. If we cannot find fifteen such values continuously, we would decrease the low threshold to 21, then 20 and so forth. If the low threshold is decreased to 5 and we still have not found the row of characters, we would reset the low threshold to 22 and increase the high threshold to 55 and once again begin to decrease the low threshold until we find the row of characters. The iteration continues until the high threshold reaches 80, at which point the routine breaks and indicates that it cannot find the row of characters.

Original License Plate



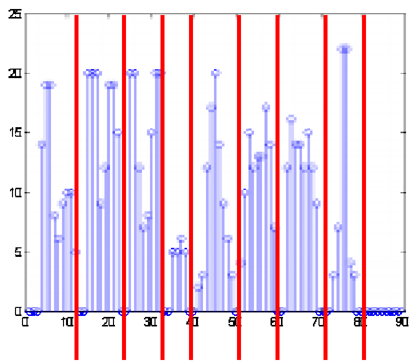
Collapsed onto Y-axis

Vertically segmented License plate

2. Horizontal segmentation

   The chosen license plate is collapsed onto its x-axis. We then look for possible characters by stepping through the 1D signal and record peaks that value more than 4 (threshold). If we find less than eight possible characters, we increase the threshold and step through the 1D signal again; if we find more than twelve characters, we decrease the threshold and step through the 1D signal again. We iterate until we either find eight to twelve characters or the threshold has reached zero or thirty three. Note that there are frequently junk on either side of the license plate (border, stickers, etc) and they would be listed as possible characters. This part does not attempt to distinguish between a sticker and a character; rather, it simply marks anything that could be a character.

Vertically segmented
License plate

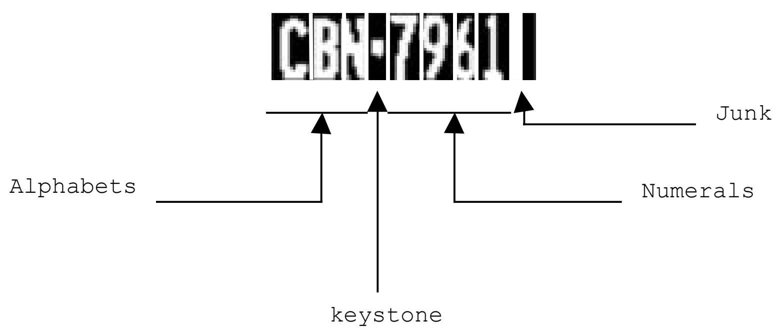


Collapsed onto Y-axis

Horizontally segmented
License plate

Image Processing:
Character Recognition → Segmentation (continued)

3. Choosing characters
   After horizontal segmentation, we have to distinguish possible characters from junk. We do so by looking for the keystone on the Pennsylvanian plate. The keystone is located by counting the number of white pixels in each possible character regions. If a region has less than 25 white pixels, then it is marked as the keystone. Once the keystone is located, the three character candidates to its left are designated as alphabets and the four character candidates to its right are designated as numerals.



4. Check for 'H' or 'L':
   The alphabets 'H' and 'L' are special cases. 'H' is always segmented as two characters while L is always cut short. The 'H' problem is solved by searching through the list of characters and look for two alphabets that are spaced only one pixel apart and coalesce them if they exist. The 'L' problem is solved by searching through the list of characters and looks for an alphabet that is less than four pixels wide and extends it by four more pixels to the right if one exists.

Rationale:

Since we chose to use 2D correlation to recognize the characters, it is essential that we segment the license plate into individual characters.

Image Processing: Character Recognition → Segmentation (continued)

Possible Problems/Weaknesses:

The character segmentation routine works extremely well on most of the images because it has thresholds that change based on the result from the previous iteration. However, the character segmentation routine is the most problematic part of the entire system because of this adaptive property. While vertical segmentation is quite robust, the horizontal segmentation loop occasionally gets stuck in an endless loop. The endless loop occurs if the act of increasing or decreasing the threshold by one results in changing the number of characters found from less than eight to more than twelve. In this case, the program would "oscillate" between two threshold values.

Possible Solution:

The temporary solution we have is to simply check for "oscillation" and once it occurs, break from the loop. The criterion for oscillation, usually a high number of iterations, is still subject to evaluation and assessment.

To graders:

The character segmentation routine is very confusing to explain in words. However, we have included the source code in the end. If you have any questions please feel free to contact tech support at wang4@andrew.cmu.edu. Our technical support specialist will do his best to clarify.

Image Processing:
Character Recognition → OCR by 2D Correlation

Description of method:

Given the chosen license plate and the coordinates that indicate where the characters are, we begin the OCR process by 2D correlation. We correlate each character with either the alphabet or the numeral templates then choose the value of each character based on the result of the correlation. The first three characters on the standard Pennsylvanian License Plates are alphabets; therefore, we correlate each one of them with the 26 alphabet templates. The latter four characters on the standard Pennsylvanian License Plates are numerals; therefore, we correlate each one of them with the 10 numeral templates. The result OCR is chosen based on the maximum values of the correlation for each character. However, we realized that certain characters, such as "B", "D", and "R", are frequently confused. As a quick solution, we implanted a scheme to display possible alternatives. Those characters that are identified as easily misinterpreted are subjected to a correlation comparison with an array of other characters that is historically known to be easily confused with the originals. If there is a possibility that a letter or number can be confused with more than one character, the characters are listed in the output in the order of decreasing likelihood. This likelihood is based on the correlation values of the character with the various templates, where high correlation denotes a good possibility. However, this scheme only occurs if the given letter does not have a very high correlation value (does not land above a nominal threshold).

Rationale:

OCR by 2D correlation is the option that seems to strike the best balance between performance and difficulty in implementation. Details can be observed in the character isolation portion of the source code.

Possible Problems/Weaknesses:

OCR by 2D correlation is sensitive to the size of the license plate, which meant bigger or smaller alphabets and numbers in the picture. The 2-d correlation was very sensitive to this and frequently gave back wrong results due to different size license plates.

Possible Solution:

Normalize the size of the image somehow.

Sensitive to the template we chose. We found that when we changed the templates size or even edited the templates it yielded different results.

Possible Solution:

More exhaustive experimentation with character templates.

Display Results / Demo

Our demonstration showcased how the output for the license plate recognition system can look. It basically had a prominent display of what the system considers to be the license plate characters, along with several likely candidates in the event that the license plate was incorrectly identified. The candidates are chosen depending on the specific characters on the plate since there are some characters that are more easily confused than others. An additional output for the system is the license plate candidate that the system identified. This was given in the form of a binary file with binary pixel values.

The demonstration was in the form of a simple command-line style output. This can easily be ported into a more graphical interface simultaneously displaying the license plate image, the input image, and the license plate. The dashes represent “wild-card” operators where previously established characters can be placed based on position.

CRN 7961
L-- ----
-B- ----
--M ----
--- --0-
--- --8-

Memory Allocation

On-chip:

Variable Name	Description	Type	Dimension	Size (bytes)
list	list of candidates	unsigned short	200x2	800
result	result of verification	unsigned short	1x3	6
region	candidate storage	unsigned char	23x79	1817
charTemp[26]	alphabet templates	signed char	21x10	212
numTemp[10]	numerical templates	signed char	21x10	212
licTempFrame	license plate template	unsigned char	24x79	1896
fftCol	result of fft of vertical sum	float	1x2048	8192
fftRow	result of fft of hor. sum	float	1x2048	8192
			Total	21327

On-board:

Variable Name	Description	Type	Dimension	Size (bytes)
gframe	gray scale picture	unsigned char	480x640	307200
bwframe	binarized picture	unsigned char	480x640	307200
bwsob	Sobel output	unsigned char	480x640	307200
			Total	921600

We made an effort to put all the templates on-chip in order to improve performance. Note that for the character templates (charTemp[] and numTemp[]) the actual size is only 210 bytes. We had to transfer 212 bytes because HPI would only transfer in multiples of four.

Performance, Accuracy

The accuracy of the license plate recognition system was evaluated by its performance on a set of 33 test input images. The input images contain the license plate, the back of the corresponding car, and the surrounding areas outside the car. We measured the success rate of the system by the rate at which it correctly identified the license plate and its individual alphanumeric characters. In general, numerals were much easier to identify than letters. This is most likely due to the larger differences (less potential ambiguity) found between numerical templates.

The license plate was located correctly approximately 73% of the time. Incorrect identification of the license plate is characterized by an incorrect identification of the spatial coordinates that define the location of the license plate region. This includes a complete miss (for example, identifying a patch of asphalt as a license plate) or a partial miss where only a portion of the target license plate was identified.

The success rate of identifying numerals was approximately 96% (93 right out of 96 numbers), while that of letters was around 70% (50 right out of 72 letters).

There were some characters that were more susceptible to error than others. We suspect that this can be attributed to the similarity between different templates. A possible solution to this situation is to redesign the character templates to accentuate the difference between similar characters. Currently, the templates' pixel values are either –20 or 30.

Performance, Speed

The possibility of converting the license plate recognition system into a commercial application depends heavily on the running time of the system. This overall system is very modular in that each functional block can be examined and analyzed independently of the others. This proves to be very convenient for profiling and optimizing the overall system. Our analysis of the system is done on a per-image basis. Although we examine each aspect of the overall system, optimizations where mostly mathematical operations are concentrated on those blocks where the input images are processed. These blocks are executed once for each of the input images.

The following table displays the performance figures of each major functional block (after all implemented optimizations):

Task Name	Cycles (in millions)	Time (sec)
(overhead)	50.6	0.337
gray_to_bin	20.3	0.1353
sobel1	57.3	0.3822
sobel2	49.7	0.3313
zero_edges	0.04	0.0003
locate	512.5	3.1457
chkpd	139.4	0.9293
isolateChar	1.3	0.0087
ocr	456	3.04
output	47.1	0.314
Total Image Processing	1283.64	8.2868

(italics denote blocks executed once per image)

These values were obtained from a final file compilation using 'level 3' optimization using the standard C6000 C compiler. Initially, all functions were enabling using 'level 1' optimization yielding running times (cycle counts) 160% larger on average throughout each function. Several additional other optimization strategies were attempted and gave appreciable results. Among those strategies was the use of loop unrolling of the Sobel functions, which gave promising results for future optimization attempts.

Prior to unrolling the inner two loops of the Sobel filter functions, the average execution cycles for each were 98.8 million and 90.8 million cycles, respectively. Unrolling the loops yielded performance figures of 57.3 million and 49.7 million cycles, respectively, corresponding to an average cycle reduction of *43.5%*. This translates to a half-second reduction in running time for optimizing a relatively small segment of the system. This finding warrants the consideration of unrolling other segments of code with similar nested loop structures, particularly in cycle-rich tasks.

Major performance improvements are expected through further investigation of optimization techniques concentrating mainly on memory issues. Most arithmetic operations and memory accesses occur using data stored in external memory, which is much slower in terms of access than internal memory. Buffering segments of external memory into internal memory can make speed improvements. The use of internal memory paging will surely enable improvements in performance. This has been proved with prior work in mask filtering optimization techniques.

Future Work

Due to time constraints and the lack of experience in image processing in our group, we are unable to make this license plate recognition system as functional as it could be. There are numerous improvements that could be made, such as:

- Instead of breaking out of the program after character segmentation fails (which usually indicate that we have chosen the wrong candidate), the program should have a system of selecting the next best candidate and continue to perform OCR on successive candidates.
- Use more evolved image processing techniques to improve the accuracy of the system. For example, we could use binary morphology to eliminate edges that are thinner than the characters of the license plate.
- Increase the resolution of the images. Currently, the only digital camera we could obtain limits us to 480x640.
- Expand the system to work with variable angle and distance.
- Expand the system to work with non-Pennsylvanian License Plates.

Acknowledgements

Code:

- We got the FFT ASM code from the TI website at:
  http://www.ti.com/sc/docs/products/dsp/c6000/62bench.htm

Ideas:

- Professor Casasent for the candidate selection idea.
- Pete Boettcher for taking the absolute value of Sobel output idea.

Assistance:

- Thanks to Steve and Pete for staying in the lab and help us out.