Junye Chen (Junyec), Zhan Dong (zhand)
November 20, 2017

Project Milestone Report

1.Work Done so Far

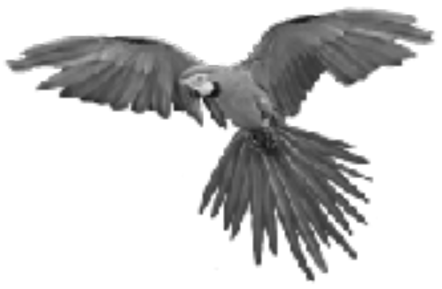
We've revised our proposal, and fixed our project on Nov.14th. In the past six days, we followed our schedule, and finished the sequential implementation of three of the important parts of our algorithm: greyscale conversion, edge detection using Sobel's algorithm, and character segmentation. These three parts will also be the main parts we will parallelize in the next few weeks. Besides, We've also learned more about the other parts of the project, and understood more about the other image processing steps we need to undergo to finish our project.

2.Results of the Work Done

i. Greyscale Conversion

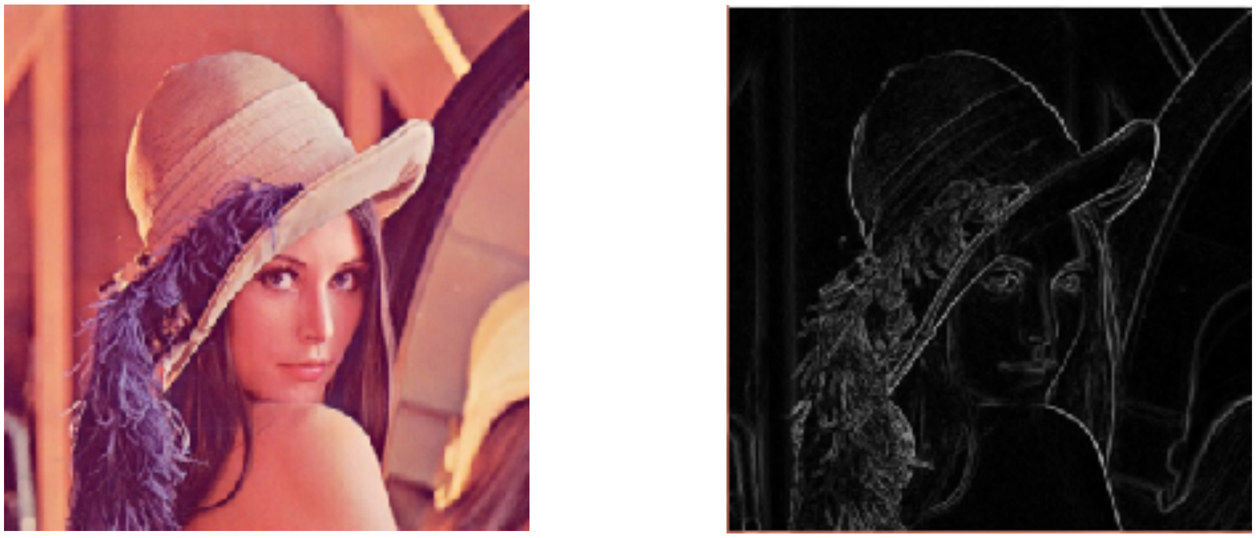
we've implemented our greyscale conversion using libpng library in c++, with the help of the sample code we found for this library. We've used one of the easiest ingredient for greyscaling, which is setting RGB values to the mean of all three. After that, we can still We will try more greyscale ingredients, and find the best in the next few weeks. The result of greyscale conversion is shown below:

Before


After

ii. Edge Detection

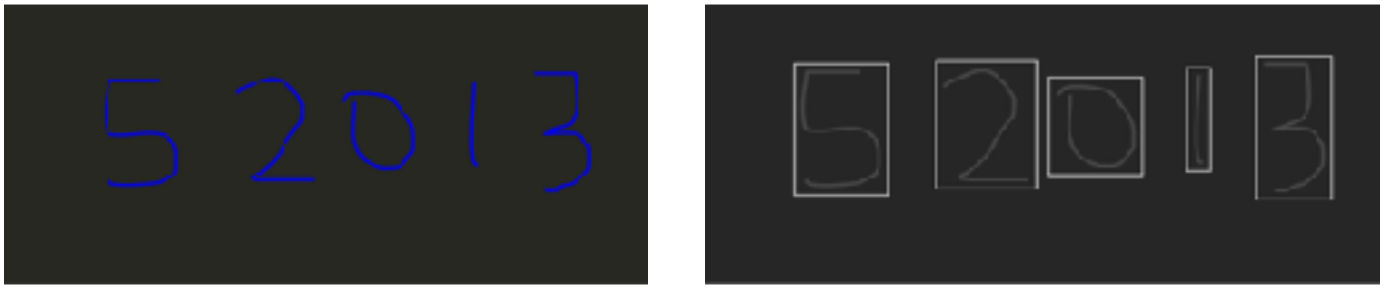
As suggested by the paper we read, we used Sobel's operator for our project to detect edges in the image. Sobel's operator is a simple but powerful edge-detection method, by applying filters to the graph, and find the edges where the pixels have huge feature changes. This algorithm is computational intense, and also has good locality. This implies that it might have a great potential for parallelization. We've also noticed that Sobel's operator works extremely well on images with high contrast and lightness differences. This might be the hint for our further improvement on the preprocessing of the image. The result of edge detection is shown below:



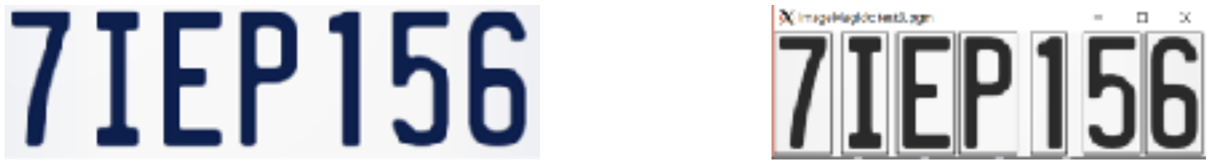
Edge Detection Sample

iii. Character Segmentation

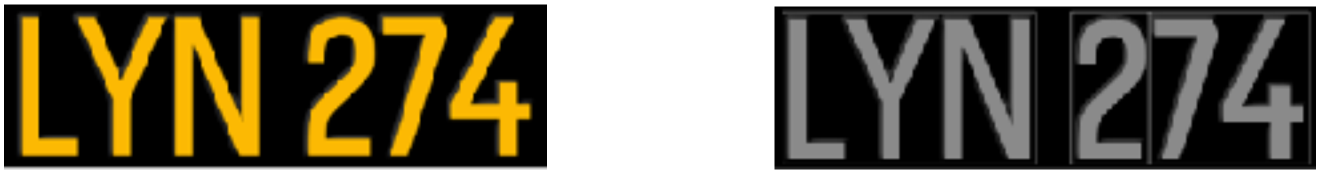
We use line sweeping algorithm to detect each character segmentation. We assume the input image is approximately binarized (which should be preprocessed correctly), which means the background color and the colors of actual characters have high contrasts. Since we also apply grayscale to the original image, the input 2d array has exactly one unsigned char value for each pixel. This algorithm detects the segments in most of the images correctly, but we find out there are some shortcomings in this algorithm. We will be fixing these bugs in the coming weeks, so that we can get the correct bounding boxes.



Correct Sample 1



Correct Sample 2



Wrong Sample 1

3.Goals and Deliverables

So far we are on the right track to our final goal: to build a parallel license plate recognition system. We've already implemented some important parts of the algorithm, but still have some parts left unimplemented. However, I believe that we still can finish our goals on time, since we've already implemented important parts of the algorithm, and can start dealing with the parallelization. We've also had a good understanding of our project, and believe that it is both realizable and meaningful.

In our proposal, we mentioned that we hope to implement our own character recognition as "nice to haves", and try to parallelize that part. We will still hold that as a goal we wish to achieve if we have enough time. Since we don't have deep insight into the technical details of this part of the project, the parallelization for this part could be both interesting and challenging. If we end up having no time for this, we will use some existing libraries for character recognition, and call the functions in a parallel way. But we really hope that we can try tackling the problem on our own.

So for our presentation on the post session, we have the following goals:

i. **Algorithm Correctness: We expect our recognition system to be able to locate the plate region from the original image.**
ii. **Decent Accuracy: We expect our recognition system to recognize the plate numbers correctly with high probability.**
iii. **Decent Speedup: We expect our recognition system to have a good speedup against the sequential code, and other benchmarks such as some existing systems with the same function. Since we are using a multi-step algorithm, we also expect each step of our algorithm has a decent speedup against benchmarks.**

4. Content for the Poster Session

For the poster session, we will prepare the following content:

i. **We will prepare a set of test graphs and the results we get by providing the graphs as input**
ii. **We will prepare a set of images that shows how the image is processed after each step in our algorithm.**
iii. **We will prepare graphs that shows the speedup of our system against other benchmarks, as well as graphs that shows the speedup of each key step in our system against other benchmarks.**
iv. **We will check the accuracy of our system, as well as the reasons why our system doesn't work for some of the license plates.**

5. Concerns

The most imminent concern we have right now is related to our segmentation algorithm. This algorithm detects the segments in most of the images correctly, but we find out there are some shortcomings in this algorithm. Firstly, there could be noises at the background, which might have a high contrast with the background. The algorithm will detect those noises as segments as well, and fail to find the correct segments.Secondly, if two segments are too close to each other, the algorithm will fail to detect the end of one segment and incorrectly combine two separate segments.

We have proposed some ways to deal with these two issues. We might need to have some preprocessing techniques that remove noises as much as possible, and we should try connected component analysis on segmentation. The incorrect example 1 has two segments close to each other horizontally, but the actual pixels are not close to each other. If we define one segment as a group of pixels that have similar values and can be connected together, we would have avoided this issue. However, connected component analysis can be hard to implement and tricky to parallelize because it requires a lot of graph contraction, and we know working with graphs needs a lot of pointer chasing. Therefore, implementing this algorithm and effectively parallelizing it are the focus of our next step in our project.

Another concern I still have is related to the last step of the algorithm - character recognition. Our current plan is to recognize those characters using an algorithm based on templates, but the characters on US license plates seem to differ from each other a lot. That might results in a poor accuracy. A possible solution might be to use a more complicated algorithm for character recognition, but that will be sort of time-consuming.

A third concern we have is the minor steps we have after edge-detection to locate the plate-region in the image. The paper mentioned the technique we should adopt after finding the edges, but didn't provide much information. We need to do some further researches to figure out the correct way of doing this part.

6. Schedule for the Following Weeks.

Based on our progress so far, we come to the following revised schedule:

Date	Work to do	Person in Charge
Nov.20 - Nov. 23	Continue working on the implementation for character segmentation, improve its accuracy.	Junye
	Working on the minor steps after edge detection to find the plate region.	Zhan
Nov.24 - Nov.26	Working on character recognition with existing library, start parallelizing edge detection.	Junye
	Continue working on the minor steps. Start parallelizing the preprocessing step.	Zhan
Nov.27 - Nov.30	Continue working on parallelizing edge detection.	Junye

Date	Work to do	Person in Charge
Dec.1 - Dec.3	Continue parallelizing the preprocessing step.	Zhan
	Start parallelizing the segmentation step.	Junye, Zhan
	Continue parallelizing the segmentation step.	Junye, Zhan
	Trying different preprocessing techniques to improve the accuracy.	Zhan
	Continue working on parallelization. Start working on character recognition (If have time).	Junye
Dec.4 - Dec.7	Continue working on the work undone.	Junye, Zhan
	Working on character recognition(If have time).	Junye, Zhan
Dec.8 - Dec.12	Wrap up the algorithm.	Junye
	Do the speed tests.	Zhan
	Write the Report	Junye, Zhan