All the Single Characters: License Plate Localization and Character Segmentation for OCR

Chris Lim, Nhu Nguyen, Hillary Do, Annie Lin, Final Project TF: Aaron Graham-Horowitz

Our Demo Video: http://youtu.be/RqHCgnirCLQ Our Original Music: http://bit.ly/1iTYVo8

Overview: After an image is fed into our program, and we use a number of filters and algorithms (edge detection, license plate detection, etc) to try to identify if there are license plates in the image. If there is a license plate, we feed the individual characters of the license plate into an OCR algorithm to output the text of the plate. First, we sharpened the image using a filter from the Python Imaging Library. Next, we found the license plate by finding potential license plate candidates through edge detection and by keeping track of bounding boxes. After finding the license plate candidates, we check if they are actually license plates by analyzing the number of connected components within the bounding box, which is then cut some more to capture the middle text of the plate. Then, we threshold the image, so that the characters are clearer to identify and read. We segmented the license plate characters by creating an array of the number of dark pixels in the vertical component of a segment of the width of the plate (to find where the character begins and ends). After we have found these characters, we feed the individual character images into Tesseract OCR, and the text of what is on the identified license plate(s) should be outputted.

Planning: Draft Specification: http://bit.ly/1mkc39h, Final Specification: http://bit.ly/1hZdEgM, Functionality Checkpoint: http://bit.ly/1kuKJQc

Our initial draft specification and final specification are included. We basically underwent drastic changes compared to our specifications.We initially just planned to implement an Optical Character Recognition (OCR) program. However, after doing some research, we became interested in using edge detection to isolate license plates from cars, which is what we ended up implementing. In our Draft Specification, we planned out the functions needed for an OCR program. Although we did not implement an OCR, we were able to use a few of the ideas in the draft specification for our License Plate Detection program. Those ideas include partitioning the text, and finding the edges of the image.In our Final Specification, we planned to create a signature which contains all functions necessary for getting edge data and detecting the edges. We also had a signature which includes functions that match a template of characters to the input characters (this is when we still wanted to do OCR). Our goal for that week was to understand and implement the edge detection algorithm, which we attempted and succeeded for the most part.

After that week, we were able to apply edge detection on the image and isolate the license plate from the car. The week after, we made our edge detection functions detect license plates better, specifically isolating all license plates in images where there are more than one and eliminating components in the image that has the shape of a license plate. This week, we implemented a function which sharpens images, a function that segments the license plate horizontally to get just the license plate number, a function that does character segmentation from the resulted horizontally cut license plate. Then we put all of the segmented characters through the OCR function from Tesseract to output the desired text from the plate.

Design and implementation: Our experience with design, interfaces, languages, systems, and testing mostly come from CS50 and CS51. We were glad that we were organized through github to back up our code. Implementing the plate localization was difficult since there are so many nuances between different plates. Our initial implementation of edge detection went well, but, as we expanded our project into identifying and isolating the characters of license plates, we ran into some trouble when we try to run the code. The edges of a few license plates were identified as a letter or a number, thus the character segmentation was sometimes inaccurate at first. However, we experimented with different methods for isolating plates and characters, and found that thresholding the image helped a lot for us to analyze the image data to draw an accurate conclusion of where the elements that we looked for are. We also experimented with different filters from python libraries. We found that many libraries are useful, and we could use their filters to make our results more accurate. Our character segmentation function performed much better than expected, and remains pretty accurate.

- Order of how our code is put together:
    - License plate localization (~85% success rate) *flawless*
    - Sharpen the resulted image
    - Cutting the license plate such that it is ready for segmentation (middle part)
    - Segment the characters and then feed the single characters into OCR
    - Lastly, output text of the license plate (one version with the plate characters segmented and one without)

Reflection: We were pleasantly surprised by the Sobel filter; though the idea was confusing, the implementation was not as tedious as we were expecting. We were not-so-pleasantly surprised by how difficult it was to find and isolate the license with high accuracy. Is is hard to tell the computer what is or is not a license plate, especially since other elements in an image could have similar characteristics to a plate. Our decision to use edge detection resulted in our changing of the project, but it worked out well. We manage to change our project so that it focused more on preparing the image for OCR, not the actual implementation of OCR. If we had more time, we probably would have tried to figure out how to make our algorithms more accurate. Currently, we have a relatively high success rate, but it is still not 100% accurate since it involves multiple steps. If we were to redo this project from scratch, we would still use Python to code our algorithms, but we would have tried to better organize our code. We left that until the end, and that made things a bit hectic.

Advice for future students: The most important thing we learned, that everyone contemplating a proj(ay-z)ect like this should know(*les*), is that you should plan ahead and be sure about what you want to implement. Also, be efficient; keep your code well commented and organized. We found that meeting frequently allowed us to keep each other on task, distributing work among us periodically. Don't let your ego get ahead of you. Check your code a LOT before you turn(t) it in.