Computer Vision To Interpret Parking Signs

Carter Levin

[1] Lily Lab, Yale University

LILY Lab

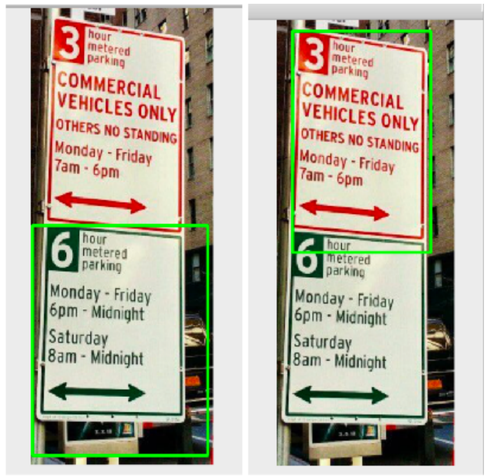
Introduction

Anyone who has tried to find parking in New York City knows that the struggle does not end once an open space is found. Often times, after parking, one will find themselves standing outside for several minutes struggling to discern the true meaning of several seemingly contradictory parking signs. This project attempts to solved this problem using computer vision and some natural language parsing. Street sign classification is a fairly well researched field, due to the interest in self driving cars, however relatively little attention has been given to reading parking signs. Analyzing parking signs introduces another layer of complexity since the system must not only do classification, but also do textual recognition and parsing. Additionally, in order to make the project practical and user friendly the front end was built as an iOS app which is used to simply take a picture of the parking sign in question. The image is then sent to a backend which processes the image and parses the text. The app then simply tells the user whether or not they can park at the moment they inquired.

Materials and Methods

The first important step was to recognizing each parking sign as an object both separate from it's background and distinct from the other signs surrounding it. OpenCV is used to detect the contours of both the red and green signs. Since these images are being taken "in the wild" it is necessary to merge contours close to each other and then crop using the surrounding rect. After the initial cropping of the image is finished each individual sign needs to be read using optical character recognition, which was done by the Google Vision Api. This bundle of text and metadata was then used to create a parsed set of "green hours" and "red hours." Due to the restrictions of a small testing dataset, the most effective approach was to simply hard code a set of parsing rules. This pipeline is hosted on a Google Virtual Machine and made accessible through an API. The client for this project is a simple iOS app, used to simply take a picture of the parking sign in question and upload it to an S3 bucket. After the image's url is sent to the backend and the parking signs are parsed the frontend simply checks the parsed set of dates against the current date and time.

Figure 1. The neon green denotes the rects drawn by openCV for both green and red signs respectively. The crop is imperfect but allows



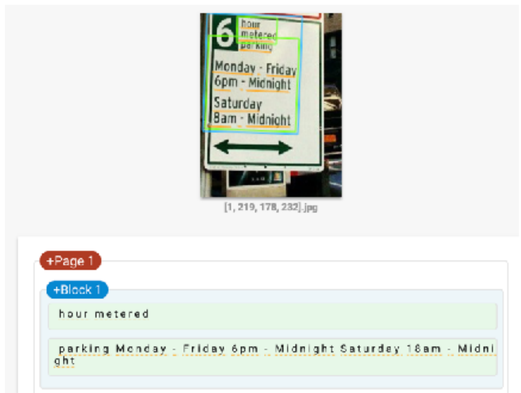
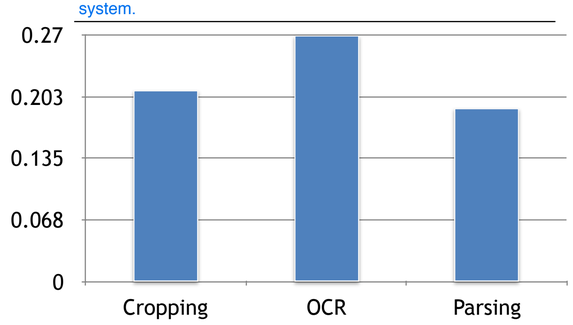


Figure 2. Text and metadata generated by the Google Vision API

Table 1. Table shows percent failure at each stage of the system.



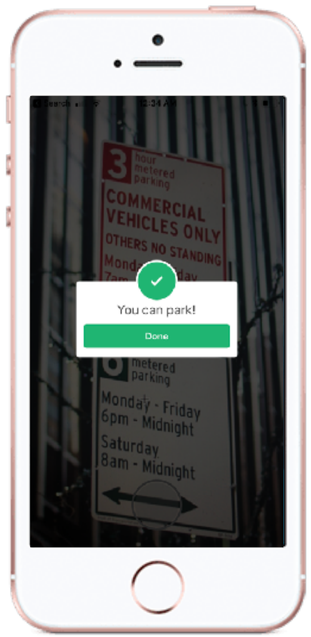


Figure 3. Screen shot of user facing iOS client app.

Results

Evaluating the accuracy of the system was somewhat difficult since it was hard to collect a large dataset and there was no easy way to generate a reference database of correct results. Ultimately a dataset was collected using a web scraper to download images related to "NYC parking signs" from the web. Out of the 300 images downloaded 37 were usable for testing. In all, the pictures provided a wide array of different sign formats and image quality. The server side algorithm was analyzed by running it against each image and inspecting each generated parse manually. The results indicated that one of the most significant points of failure is at the cropping stage. Most of the photos which failed at this step were taken at an angle, had a shadow cast across them or were of poor resolution.

Conclusion

Testing revealed that original image quality played a major role in the accuracy of the final result. While, the app performs very well when images are taken straight on with minimal glare and shadow, it is unrealistic to assume that users will be able to constantly capture such high quality images of signs in the real world. It will therefore need some refining if it is to be released and used practically. While more fundamental developments in the field of computer vision may be needed in order to truly produce a foolproof product, there are a number of potential tweaks that could be made to improve the current approach. One of the major points of failure for instance took place during the cropping stage. This is in part because RBG values for parking sign "red" and "green" vary depending on the lighting and can at times be picked up on in the background. A potential solution would be to use another graphical indicator to determine the top and bottom of a sign, such as the horizontal lines which separate them. Another way to improve the OCR step would be to create some graphical tools on the front end to instruct the user on how to take the best picture.

Acknowledgement

Thank you to Professor Radev and Yale University's department of Computer Science.