May18-37	
Client	Iowa State Parking Division
Advisor	Dr. Ahmed Kamal
Team Members – Roles	Derrick Lockwood – Project Manager Donavan Brooks – Backend Lead Riley Snyder - Webmaster Mason Schreck – Communications Lead John Ingwersen – Mobile Master Joe Krajcir – Quality Assurance
Team Email	Sdmay18-37@iastate.edu
Team Website	http://sdmay18-37.sd.ece.iastate.edu/
Version	1.0

Table of Contents

Introduction

Acknowledgement

Throughout the course of this project, the team will regularly consult Doctor Ahmed Kamal, who proposed the project. He is serving as our advisor during both the planning / architectural and implementation phases. For the acquisition of project-critical devices we currently have access to the standard $500 budget provided by the department. The team is also currently communicating with the Iowa State University parking division to potentially acquire another $500.

Problem Statement

The problem posed to the team by the parking division was the inability of faculty and staff to locate parking in a timely fashion. This is resulting in time being wasted to search for a parking spot as well as an excess of emissions. The problem occurs primarily during the workday hours 8:00 am – 5:00 pm, these peak hours are when the team is seeking to enact the most change.

To solve this problem the team is proposing to utilize a camera paired with a small computer located outdoors, and a second system to the "heavy lifting". The outdoor setup will use an IP (internet-protocol) camera to acquire images of specific parking spots. The small, outdoor computer will oversee sending the images acquired by the camera to the second remote system and ensure that each camera remains operational by frequent ping checks. The remote system will then check for a car in each spot and update our mobile applications accordingly. Each user will get live updates pushed to their mobile devices to provide a real-time aspect to the service, and to keep each end user appraised of their parking options.

Operating Environment

For each primary component of the system there are different operating environments. Our mobile applications will reside on the user's current mobile device and the domain over which our applications must function is the same that would be expected of any current industry standard mobile device. The pre-processing setup (camera paired with computing device) must be able to withstand outdoor conditions, which includes but is not limited to: snow, ice, thunderstorms, humid summers, and high winds. For the camera, we are seeking to acquire an IP66 rated off-the-shelf solution, this is an industry standard assured to be dust-proof and protected from high pressure water jets. The camera must also be battery-powered (similar to Arlo outdoor security cameras) that are able to operate through frigid Iowa winters. The team plans to do continuous research until prototyping begins to ensure the best device is acquired.

Intended Users

The intended use of this project is to produce a product that will cut down on emissions, time to park, and general confusion around locating a parking spot. This means that it will be an efficient way to increase productivity in everyone's daily life. The use will be when someone in the general public of the ISU population is looking for a parking spot across campus. They will research beforehand as to open spots in parking lots around campus and then be able to drive directly to the lot and spot(s) without having to search multiple parking lots for a place to park. Our user vase will include students, faculty, and campus visitors. Anyone will be able to download the applications for iOS or Android platforms and use it with or without an account.

Assumptions and Limitations

A limitation we have is the battery of the IP Cameras and the distance that the IP Camera can communicate with our preprocessing device. Another limitation is the weather, and how well our model is trained to detect cars in inclement weather. We assume we will be able to power the IP Camera(s) somehow, and be able to get multiple vantage points on the parking lot. Another assumption is that we will have the systems that we need to run the machine learning model efficiently.

Expected End Product

The main end product will consist of an Android, iOS, and web application that will be showing live parking lot updates with a comfortable UI. These apps will show users how open or full their favorite parking lot is so that they can make an informed decision and not waste time looking for a spot. The delivery date for this product will be at the end of the term.

There will also be a camera infrastructure in place in designated parking lots to capture this data. The camera will be able to withstand changes in weather or climate and still function and perform it's specified task. The delivery date for this will also be at the end of the term.

Proposed Approach

Functional Requirements

The functional requirements will be categorized by the three subsystems that will comprise the solution.

Pre-Processing:

- The Raspberry Pi (or comparable device) will maintain activity states of all paired cameras
- Each camera will 'report' a heartbeat status to the controlling device at least once a second

- Each device will communicate the images from each camera to the post-processing system at the rate of at least two relays per second
- Each device will maintain a connection status detailing the connection to Firebase to ensure that data can be manipulated as needed
- If three consecutive camera heartbeats are missed a warning email shall be generated detailing the device ID that has a 'dead' camera
- If the device connection status to Firebase is ever lost immediately generate an email report to the development team

Post-Processing:

- Object detection must take place in under two seconds at a confidence threshold above 70%
- The time taken to average the various prediction thresholds of a given spot must occur in under one second
- To update a spot to 'occupied' in Firebase the average prediction threshold of every camera that can see that specific spot must be above 75%

Mobile:

- Each mobile application will present a geographically accurate orientation of a lot with parking spot ID's
- Each mobile application will present a simple graphical representation of an occupied or unoccupied spot (colored red or green) to the user
- Each parking spot will be updated graphically in under one second after the Firebase update occurs (assuming stable network connection)

Technology Considerations

Our proposed technology is using the latest machine learning API to predict when a car is in an image that contains a parking spot. This API is called Tensorflow and is produced by Google and created in Python. We will be using this API to develop our model, train it, and produce a prediction of an image. Another piece of technology we are using by Google is Firebase. This is a reactive database that will allow our users to have real-time updates to their device they are viewing our application on. So if a parking spot updates they won't have to refresh or make another request to the server but rather it will give them the updated sports every time our detection calculates a car in the spot. Other technologies could include Google's TPU such that the predictions speeds will exponentially increase compared to running detections on a single CPU.

Technical Approach Considerations

Our technical approach includes a data stream such that it starts with an IP Camera which transmits our images to a pre-processing device. This device splits the image into several other images centered around each parking space. This will then format these smaller images meaning reduce the resolution and run a grayscale filter plus a histogram over the image. Then these images will go to post-processing for the prediction. Here the post-processing machine will take the model we have created and run a prediction over this model. The post-processing machine will be readily updated will be readily updated with newer and better models as we progress the learning and development of these models that will best suit the needs of our predictions. Then the small images plus the prediction of it being a car or not will be averaged across the other cameras plus the average of the past five images and predictions. If the value has changed it sends this change to Firebase to be distributed out to the mobile platform.

Testing Requirements Considerations

The testing requirements of the project include mechanical testing, pre-processing testing, post-processing testing, and client side testing. Mechanical testing will include the best vantage point for each IP Camera to sit. This is to ensure the best angle of attack for the pre-processing unit to split the image into several other images. We will need to go through a testing phase of the best positions in a parking lot and then make adjustments as we start development depending on what we can get from the IP Camera. The pre-processing testing will include tests around the quality of the images, whether the image is split correctly enough to find a car in the spot, and what happens when the IP Camera stops sending images. This testing will be done during the development stage of the project and whenever our system needs to be installed. The post-processing testing will come from self evaluations of the model. We will be able to take the model and test the accuracy of it with an image we know to have a vehicle in it. Thus we will be able to make accurate predictions if we get a high accuracy with our model. Last but not least we will be doing client side testing, meaning we will have rigorous tests on iOS, Android, and the Web platforms. We will ensure 100% code coverage such that our tests cover every piece of our code and develop in an agile review process way.

Security Considerations

The security considerations we mainly have is around the images of the IP Cameras. We have talked with our client and are still getting details as to any other security concerns he might have. The security consideration around the images of the IP Cameras is since it is taking pictures of cars license plates will be on each car and so the security around the images being sent out to do the predictions.

Safety Requirements

Making a mobile application meant to be operated with a vehicle is something that raises many concerns for safety. We do not want users to be distracted by the mobile experience when they are driving. This means we have to make sure the information is easily readable and readily available for the user. One option we have is to make the information avalible in an audio format where the driver can listen to the lot stats be read aloud. If the information is behind multiple walls and creates confusion for the user than this could provide unwanted complexity while driving and distractions.

Previous Work / Existing Literature

After settling upon our strategy and a basic architecture for implementation we began to search for similar existing projects or solutions to the problem. We discovered essentially the same concept had been carried out by Andrews Sobral titled "Automatic Parking Lot Classification"[1] and the link is provided below for convenience. This solution is only trained for sunny, cloudy and rainy days and does not seem to handle snow, and seems to run with a lot less hardware and processing involved. This solution also does not seem to have a front-end component let alone a mobile solution.

A second project[2] that was quickly discovered is more of a mirror of what we are wishing to carry out. We plan to present a little more sophisticated parking lot description to allow end-users to easily distinguish which spots are vacant without memorizing spot identification numbers. This solution is more closely aligned with our preliminary architectural design decisions, and unlike the aforementioned project, does have a mobile and web interface. As a team we are seeking to provide a sleek application for iOS and Android with easy-to-understand lot layouts with spots colored red or green to show vacancies. We also are seeking to implement computing a confidence threshold in a manner to increase the reliability of object detection.

[1] https://www.behance.net/gallery/29828109/Deep-Learning-Automatic-Parking-Lot-Classification
[2] https://arxiv.org/abs/1606.09367

Possible Risk / Risk Mitigation

The biggest risk would be failure of the devices out in the parking lot or the mobile applications failing. Either outcome would result in the user not being able to get live parking updates. We can prepare for this by creating a good enclosed camera environment or by getting a weather proof camera. The applications errors can be prevented by thorough testing to ensure all bugs are caught ahead of deployment.

Other risk would be the detection software not being accurate in which case we would be providing wrong information to our user base. This would also be very bad and cause users to no longer use

our product. Again, we can run the software many times to improve the machine learning process to nearly always get the right results.

Milestones and Evaluation Criteria

1. Basic Infrastructure Setup
    a. Have components necessary for implementation (IP Cameras, Raspberry Pi for pre-processing)
    b. Identified optimal position to setup IP Cameras for partitioning of images by parking spot
    c. Setup our proposed architecture in a parking lot at ISU
    d. Basic communication with server, IP Camera, and pre-processing unit. Images being received by IP Camera and being received by the backend
2. Successful Pre-Processing of Images
    a. Images are being properly cut by parking spot, scaled, cropped and recolored by our pre-processing unit
    b. Images from IP Camera are being received by the backend for post-processing
    c. Evaluation whether a car exists in an image or not
3. Trained Model
    a. Have a finalized model for image recognition and object detection of vehicles in Tensorflow
    b. Model has been trained by being passed a multitude of images. These images include different weather conditions and different make and models of vehicles
    c. Object detection in images has reached a confidence threshold of at least 70%
4. Working Prototype / Finished iOS, Web and Android Platforms
    a. Web, iOS, AND Android platforms are receiving updates and accurate information about the parking lots that we have setup for testing, and showing them visually
    b. The average of various prediction thresholds of a given spot from multiple images is accurate
    c. Information about parking spot is being updated in Firebase
5. Final Product
    a. Worked through bugs that we have found through real-world and end-to-end testing
    b. Ensured that we have identified and found solutions for potential edge cases and failures
    c. We have met all functional and non-functional requirements that we have defined for our project

Project Tracking Procedures

For the code development on this project we will be suing Github's issue tracking system. It is a very good system putting code and problems close together for the user to clearly see. It also allows for labeling and the creation of milestones for the project. Any other issues are discussed in a weekly report or at team meetings. Issues are to be brought up early so the whole group can discuss and see where the project needs to go.

Statement of Work

1. **Tensorflow Model**
    a. **Objective: Build a model to successfully detect cars using the open source machine learning API Tensorflow**
    b. **Approach: Using Tensorflow, feed a model a large base of images with cars. The images will contain different makes and models of cars, along with different weather conditions. This model will drive the object detection when the IP Cameras take a snapshot of a parking lot**
    c. **Expected Result: A model that is able to detect cars with a 70% confidence threshold**
2. **Set Up Infrastructure**
    a. **Objective: Have IP Cameras in place at a parking lot that can communicate with a pre-processing unit**
    b. **Approach: Position the IP Cameras such that they have an optimal view of the parking lot or portion of the parking lot. The IP Cameras also have to be in a position that allows them to communicate with a pre-processing unit. The pre-processing unit will either be placed where it is able to connect to the IP Cameras or where it is able to have constant power. Ideally the pre-processing unit will be plugged into a power source.**
    c. **Expected Result: IP Cameras having an adequate line of sight to a parking lot. Additionally, IP Cameras and pre-processing unit positioned such that they are able to communicate with each other.**
3. **Setup Model on Server for Training**
    a. **Objective: Train Tensorflow model with images from IP Camera**
    b. **Approach: Once the model is complete and the infrastructure is setup, being training in the real world setting to ensure everything is working as expected. IP Cameras should send images to the pre-processing unit. Then the pre-processing unit gets the image ready for the server. The pre-processing unit then sends the image to the server for the model to calculate if a car is in a spot or not**
    c. **Expected Result: Successful connectivity between infrastructure and server allowing the model to calculate if a spot is open or not**

4. **Web, iOS, Android Development**
    a. **Objective: Applications that will allow users to check the availability of spots in a parking lot of their choosing. Supported platforms are: Web, iOS, and Android.**
    b. **Approach: Create an application that is consistent between all platforms. The application will allow users to login and save their favorite parking lots. Users that are logged in or not logged in will be able to user the same basic functionality, which is searching for a parking lot on a map to check for open spots.**
    c. **Expected Result: Applications on the three major platforms delivering a consistent and fluid experience.**
5. **Work through Bugs, Optimize, Polish**
    a. **Objective: Refine the system**
    b. **Approach: Continue testing and fine-tuning the model to ensure that car detection becomes optimal, regardless of the make and model of the car and weather conditions. Ensure that a car taking spot updates to all application platforms within seconds.**
    c. **Expected Result: An overall fast and reliable service that gives users real time updates of parking spot availability for a parking lot of their choosing.**

Estimated Resources

1. Personnel Effort Requirements

Task	Projected Effort
Tensorflow Model	The Tensorflow model should start processing images wi cars to begin building a strong foundation for the object detection to work effectively. This should be done as soon possible. This will be done all the way up to when the infrastructure is set up.
Set up Infrastructure	Setting up infrastructure will possibly be the most crucial. Ideal locations for the IP cameras will need to searched fo The IP cameras will need to be placed in as close to bird's-eye-view as possible. Additionally the IP cameras n to have reliable connection with the pre-processing unit. This task could potentially take some trial and error to fin the right viewpoints that also support good connection
Setup model on server for training	This task will be the first time all pieces are put together. infrastructure needs to be able to get a clear picture of the parking lot and send the processed image to the model to make a prediction. The model needs to be able to detect i car is in a parking spot or not. Assuming that the infrastructure is successfully set up and the model is ready then this task should seamless barring any unforeseen err
Web, iOS, Android development	This task is the front-end of the project. Our team has experience in all 3 platforms, so creating a consistent and reliable application across all platforms should be a smoo process. All of the back-end and infrastructure needs to properly work in order for the applications to serve its purpose though.

Backend development	**This task controls the information from the images receiv by the pre-processing unit and giving it to the model to make a prediction. From their store the prediction in Firebase for applications to update in real time.**
Work through bugs, optimize and polish	**Everything works at this point. All that is left is to fine tun everything and make sure there aren't any mysterious edg cases that breaks the system.**

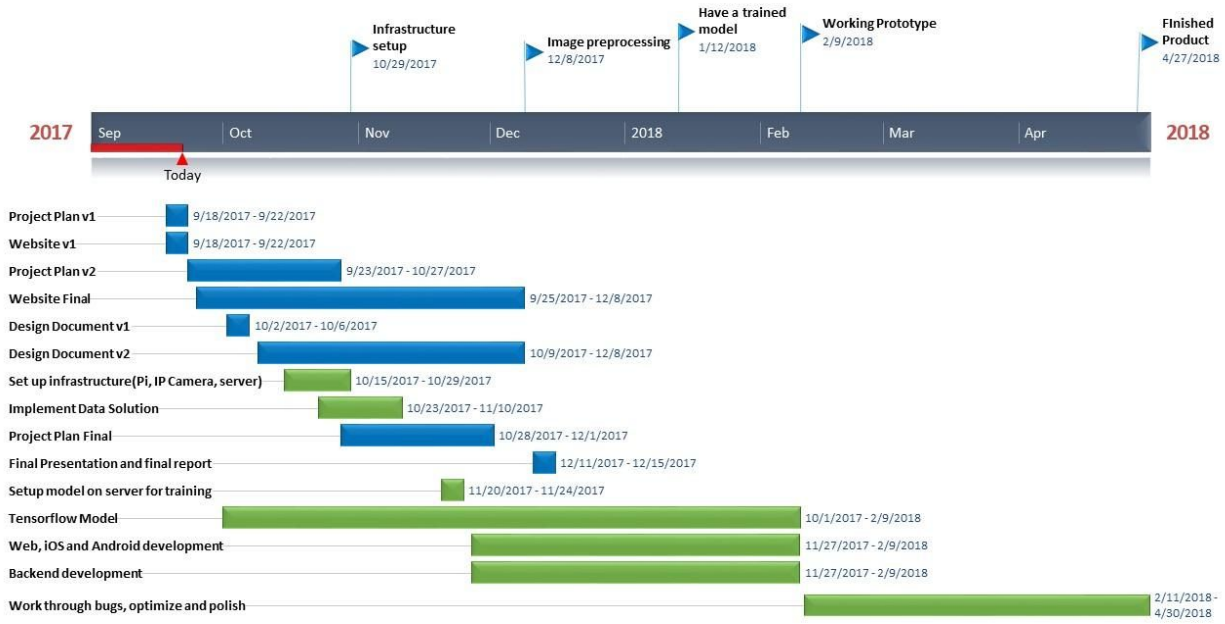
2. Other Resource Requirements

The only physical materials needed are IP cameras and pre-processing units. Other non-physical resources include a server for back-end and web services, Firebase application platform(if approved), and development of web, iOS, and Android applications.

3. Financial Requirements

Currently our project requires the use of an IP camera and pre-processing device. Tentatively, the devices our team is looking to use are Arlo security cameras and a Raspberry Pi 3. At the time of this writing, a set of 2 Arlo cameras with mounts cost $265.95 on Amazon. A Raspberry Pi 3 normally costs $35 for each unit. 2 Arlo cameras and a Raspberry Pi 3 should suffice for building and testing the infrastructure of our proposed approach, which would be a grand total of about $300.95. Assuming the devices and solution work, then more cameras would need to be bought in order to see an entire parking lot. A parking lot may need anywhere from several to tens of cameras. Assuming we put up 4 cameras to cover a medium/small parking lot with one Raspberry Pi 3 then the cost would be around $566.90.

Project Timeline



First Semester

In the first semester, outside of the documentation, website and final project presentation that must be completed, we are going to have a lot of infrastructure,backend and frontend development. First we are going to identify what equipment we will need in order to implement our architecture. After we receive our equipment, we will work with the parking division and our client to identify what lots we can use to set up our this infrastructure. This will consists of positioning cameras to give us optimal angles, establishing connection between cameras and pre-processing device as well as between our pre-processing device and server. We will also begin developing and training our Tensorflow model. We will set up our data solution(Firebase) to handle the data that we will need and begin Android, iOS Web and backend development. By the end of the semester our main goal is to have our infrastructure in place and have pre-processing for images completed. Having this done in the first semester is important because we will need to train our Tensorflow model with images of cars in parking spots during winter climate.

Second Semester

In the second semester we will ensure that we gain enough images in order to train our model with images in winter climate. Early in the semester we would like to have our trained model and tweak it as necessary. After that we would like to have a working prototype and then we will go into real world end-to-end testing. Having the prototype early will give us plenty of time to test in all weather

and identify bugs, edge cases and possible failures. We will fix any issues that we have found as well as optimize in order to fulfill our functional requirements.