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	2.0

Table of Contents

I. **Introduction** **2**

Acknowledgement 2

Problem Statement 2

Operating Environment 2

Intended Users 3

Assumptions and Limitations 3

Expected End Product 3

II. **Specification and Analysis** **3**

3.1 Completed Testing 3

3.2.1 Non-Functionals 4

3.2.2 Functional 4

3.3 Highlights of Proposed Solution 4

3.4 Standards 5

3.5 Proposed Design 5

3.6 Design Analysis 5

III. **4. Testing and Implementation** 7

4.1 Interface Specifications 7

4.2 Hardware/Software 7

Post-Processing 8

4.3 Process 8

4.4 Results 9

4.5 Pre-Processing Testing Process 9

IV. **5. Closing** 9

5.1 Summary of work so far 9

5.2 Goals 9

5.3 Plan of action 10

5.4 Technical Resources 10

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 tam by the parking division, was the inability of staff to locate parking in a timely fashion. This is resulting in time being wasted to both 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 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 each camera remains operational. 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 base 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.

Specification and Analysis

3.1 Completed Testing

The original plan proposed was to use sensors at each spot in a lot. We thought this idea would not only be expensive, but would be very complex in providing power in large lots, and create a system to connect all these devices without them getting damaged. We then decided to take a more software focused approach. Our plan will be implemented with cameras, an image detection system, and mobile application. The cameras will be in charge of capturing images. These will be passed into the

detection model for identification, and then this information will be transformed and sent to a database where we can display it on mobile devices in a way that the users can understand.

The heart of this solution will be the image processing. We need our system to correctly identify If there is a vehicle in a spot, and not be confused by cars around it. So far we have completed the design of our detection model, and have now begun training it. Training is the process of running the detection on many images to help it learn what we are looking for, motor vehicles in our situation. We can contentiously train the model so that over time it continues to get better.

3.2.1 Non-Functionals

We will have a few non-functional requirements for this project:

- A simple user interface.
- The design will stay constant across multiple devices/platforms.
- All development will avoid licensing issues.
- Our application will be secure.

3.2.2 Functional

The functional requirements for our project will be as follows:

- Accurately give the status of a parking lot with regard to the number of spots left.
- Have fall-back systems for damaged cameras or pre-processing units.
- Not to store any images that might have personal information permanently.

3.3 Highlights of Proposed Solution

Our proposed solution highlights include a low cost camera and preprocessing unit that can be easily installed into several parking lot locations, real-time updates to current parking spots, and is easily improved over time.

The low cost camera and preprocessing unit is beneficial to users and consumers of our system. This is because the users benefit from not worrying about destroying or damaging the sensors. The consumers of our system however get the most beneficial by reducing the overhead costs of each sensor install, maintenance, and damages. The weakness in this system however is finding a location to put the cameras such that they have the best point of view of the parking lot and can cover each parking spot evenly. Weather is going to play a big factor in this solution due to being able to see a car is important and detect it when it has snow covered by it.

Real-time updates will be beneficial to the users because they will be able to see in real time which parking lots are open and which parking spots are open in each lot in real-time. This is extremely useful when looking for parking spots on the fly and instead of searching the entire lot they can

easily look at the live application and see which spots are open and which are closed. This will also be beneficial to the consumers because our bounding box data that will be stored on the real-time database can be updated on the fly and thus be easily modified for weather and placement of camera positions.

The solution will also be able to be easily improved over time due to the nature that we can improve the machine learning model by adding training data, modifying layers in our model, and moving the model to a different form of processing. Our training data can be modified by adding relevant images to the data set and equalizing the images between vehicle and not vehicle. We can also remove invalid images or images that don't apply to the location that we are looking at. Modifying the layers in the model will be used to better how the model trains and predicts in certain situations and will be improved over the course of development. Lastly, moving the model to a different form of processing could include transferring our model to a network that can process bounding box information of vehicles specifically and referencing these bounding boxes on each parking space. Thus allowing it to be less process because it is processing one image instead of one for each parking spot.

3.4 Standards

We will be using two Google products in this solution, Tensorflow and Firebase. Tensorflow is an open source software library, and thus its use in our project will follow the Apache License. This allows us to use it for commercial use with modification. Firebase will follow the standard Google API terms of service. The code written for this project will follow IEE standards, and be well document for easy modification by future developers that might be tasked with expanding this solution.

3.5 Proposed Design

Our design has three main sections, Pre-Processing, Post-Processing, and Mobile delivery. Pre-Processing consists of the cameras and computer units. We will deploy cameras and pre processing units around the lots to capture images and process the data for the image recognition. We will develop the scripts and tests needed to verify the communication between the camera and processing units, as well as the programs to format the images into a final version. Once the images have successfully been retrieved and formatted, they will be sent to a remote server with large computational power.

Post-Processing is the actual scanning of the images to determine whether a car is parked in a particular spot or not. A model is being constantly trained to better identify the vehicles. The Tensorflow software library will be used in the model. The post processing will send information to Google's Firebase database to be stored for the applications to retrieve.

After a change in values, the user will be able to digest the information from their mobile devices. The applications we develop will be available on Android and iOS. The design of these apps will make it easy for the user to tell if a lot is a good choice for parking or not. A main factor in our

design is ease of access, since the users may be tempted to use this app while driving a motor vehicle.

3.6 Design Analysis

What we have worked on prototyping so far is the machine learning model and it's training of data. We have been able to load in the images from the network using a dataset website called imagenet by which we can download images and other useful data from it. We then process these images removing invalid ones and formatting them into a grayscale, resize, and histogram image which is then saved to disk. From here we began the model creation and added the layers that we thought were enough to determine a good learning rate. Then we loaded the images and the labels into a train dataset by which we use to train our model. Currently we are having issues with the loading of the images recognized by the training model and also a hardware limitation on out of memory error. We are going to continue with our current design and some things we are trying to work in is weather analysis and placement of camera for our predictions. On the pre-processing side we have prototyped the heartbeat of cameras and gathering of data from our real-time database to use to split the image into our images used to predict car locations. We decided we wanted to go with a command prompt system in our pre-processing device such that we could initiate subsystems and destroy subsystems as necessary.

On our user end, we have a prototyped iOS application that can grab data from the real-time database and display it in a list format. This will be used when we have data we can supply from our backend post-processing predictor. An issue we have here is how we are going to display the data we gather from the real-time database to the use in a map format. It is unclear to us at the moment on how we intend to go about solving a interactive iOS, Android, and Web map fragment that will allow the user to zoom and select different parking spots.

4. Testing and Implementation

4.1 Interface Specifications

We will be doing is with Gitlab in order to set up integration tests and regression test to run on our codebase. This will allow us to have end to end and specific tests run once something is pushed to our master branch. This will allow us to have great coverage of our codebase and to constantly ensure that our master branch is not broken.

4.2 Hardware/Software

During the testing phase we initially using a logitech HD Pro c920 webcam. This is a 1080p webcam that will provide us great resolution to capture the images of the parking lot we need. The pre and post processing will be handle by a group members PC which contains a Nvidia Geforce GTX 1070 GPU. Having a GPU is very important because it significantly increases the speed of our image predictions and model training.

When it comes to prototyping in the actual lot we will be using IP camera and Raspberry Pi. As of now we are not 100% sure exactly what camera we are going to use, but right now we have

proposed Arlo Security System cameras. We have come to this conclusion because they have met our criteria for durability, battery life, resolution and price. This criteria is very important because these cameras need the ability to run without a lot of human interaction, as well as survive the Iowa climate. The use of the Raspberry Pi allows to have a lightweight solution to handle the splitting of images and syncing of cameras

Post-Processing

For post-processing, we need to be able to train and test the accuracy of our Tensorflow model we created for classifying images as having a vehicle or not having a vehicle. In order to test and train our model we need a system to load the model onto. This system will be used to run sessions, which pass data sets of of around 6,000 images, vehicles and not vehicles, through our model in order to train it to recognize if an image has a car or does not have a car in it. In this system we would like to have a substantial GPU in order to the decrease time it would take to train our model. GPU's are designed to handle the matrix and vector operations that are required in order to train a neural network, and are significantly faster than using CPU's.

As of now we have a couple options of what system we will use to train and test the accuracy of our model:

Option 1: Potentially the ETG at Iowa State University, will provide us with a computer containing a Nvidia GeForce GTX 480. We will have sudo access and would be able to ssh into it to load and train our model with our data sets.

Option 2: Members of our team have custom PC's with either Nvidia GeForce GTX 1080's or 1070's. We could use these PC's in order to train our model either at night or when they are away at class.

Option 3: We have signed up to use Google Cloud TPU's to run sessions of our Tensorflow model. These Cloud TPU's are specifically made to run machine learning models and provide up to 180 teraflops of performance, which would significantly speed up the process of training our model.

4.3 Process

In order to test our functional requirement of object detection having above a 70% confidence threshold we will test our trained model on images received from pre-processing, see the confidence value our model outputs and make tweaks to our model as necessary. This cyclical process will ensure that the model we use accurately determines if the image contains a car or not, meeting the needs of our project.

Outside of testing the main parts of the system, we have implemented a process to ensure that we have a codebase ridden of bugs or poorly written code. We have implemented this process through the use of merge requests. Before contributors can merge the feature they are working into our main project, it needs to be reviewed and approved by two other people on our team. These approvers usually consists of people working on the same aspect of the project, or a lead. The role of the approver is to look over the code to ensure it meets coding standards, as well as run tests/edge cases on the code to ensure it does what it is meant to. By implementing this process we should have full thorough coverage of our codebase, and limit the number of bug fixes needed throughout the creation of this project.

4.4 Results

Pre-Processing
For our pre-processing setup we are utilizing an IP camera and a Raspberry Pi and we need to be able to test a few key components on this setup. To be able to test if the Raspberry Pi is in constant contact with both Firebase and the chain of cameras that it is responsible for. Also we need to be able to test whether or not the Pi is splitting the images received from each paired camera correctly. To test the persistent connection, each Pi will have a script that runs on it to constantly check for a connection to each camera as well as to the Firebase data solution. If at anytime these connections fail then the development team will be alerted with the ID of the Pi on which the failure and the problem that has occurred. Lastly to check that the splitting of the images has occurred correctly will initially be just a visual check. We will dictate where the software is to split an image, then check the result to see if it matches up to what we had desired.

4.5 Pre-Processing Testing Process

To ensure that all of these actions are taking place as intended on the front end we have already stated that an initial visual appraisal of non-code based results are the first check. Since the pre-processing implementations revolve solely around software changes (without custom device interfaces) the main method of testing was mentioned above as utilizing merge requests. No critical aspects of the system will be updated without being reviewed by at least two developers beforehand. For the automated aspects of the pre-processing system ('heartbeat' and Firebase connection) we are planning to implement a system that will warn the developers of any issues as they occur. These issues could be loss of power, network connectivity issues, or camera connection failure, and we plan to have built in automated tests running constantly to catch these issues before they cause potentially catastrophic issues. If a thread on the pre-processing system responsible for constantly running these checks in the background fails, the team will be notified instantly of which error has occurred. This will allow for succinct fixes of the pre-processing system to avoid failure of the product stemming from the pre-processing system.

5. Closing

5.1 Summary of work so far

At this point in time our team has made much progress and we are continuing to come up with new ideas and implementations. We have looked at many options for cameras to determine which is best, and we have also considered different at mount points for these cameras. Our image detection software model has been created and is currently being trained to detect cars more efficiently and accurately. Heartbeats have been established between pre-processing devices to ensure uptime and connectivity.

5.2 Goals

Our biggest goal currently is to have our Tensorflow model successfully trained and accurately reporting parking availability to us. This is the backbone of our project and everything else relies on

the success of this model. Smaller goals include coming up with a mobile solution to display different parking maps within the same application, and also to come up with a reliable log in system to save user data across sessions. We would also like to have an IP camera and Raspberry Pi purchased and configured for our project. Once we have a mount point established and a plan in place we want to test all of our prototype designs together (camera, pre-processing, post-processing, mobile app).

5.3 Plan of action

First we must decide which post-processing power we would like to use. While Google's Cloud TPU may be the fastest option, it will be hard to always have access to it and run anything we would like within it. On the other hand using our own devices is easy but that is obviously not a long term solution if this project is to be expanded down the road. Our team will continue to enhance the mobile and web applications to ensure they are ready when needed. We will continue to meet with all of our project advisors to take their advice and apply it towards our goals. We are happy with our current progress and our plan to reach our end product.