Mohammad Akhlaghi
Chief Executive Officer

Phone: 778-997-1717
Email: maa31@sfu.ca

January 23, 2012

Dr. Andrew Rawicz
School of Engineering Science
Simon Fraser University
Burnaby, British Columbia
V5A 1S6

Re: ENSC 440 Design Specifications for the Pvision Parking Convenience System

Dear Dr. Rawicz,

We are writing with regards to the functional specifications for a Parking Convenience System which we have attached for your ready reference. The objective of our project is to design a system which dynamically monitors the parking spots in a parking lot and informs drivers of the vacant and occupied spots in advance of them entering the site. The project functions on the principles of minimalistic design in order to develop an efficient and robust system which is user centered and financially attractive to our direct clients - owners and administrators of parking lots in malls, universities, hotels and other institutions.

The document provides the detailed functional specifications of our product by breaking it down into its three independent, yet linked, components. The overview provides a black-box representation of our system and is followed by the specifications which range from electrical and mechanical to durability and environmental requirements.

PVision is composed of five engineering students in the final year of their undergraduate degrees. The executive team consists of Mohammad Akhlaghi (CEO), Oshi Mathur (COO), Milad Haji Hassan (CTO), Noah Park (CFO) and YuJie Xu (CMO). We would be delighted to hear from you in case you would like to further discuss our proposal.

On behalf of the executive team,
Sincerely,

Mohammad Akhlaghi
Chief Executive Officer (CEO)
Pvision Electronics Limited

Enclosed: Functional Specification for Pvision Parking Convenience System

Design Specification
Pvision Parking Convenience System

Pvision Electronics

Project Team:

Mohammad Akhlaghi - CEO
Oshi Mathur - COO
Milad Hajihassan - CTO
Noah Park - CFO
YuJie Xu - CMO

Contact Person:

Mohammad Akhlaghi
Pvision-ensc440@sfu.ca
778-997-1717

Submitted to:

Andrew Rawicz (ENSC 440)
Steve Whitmore (ENSC 305)
School of Engineering Science
Simon Fraser University

Issued date:

March 5th, 2012
Revision 1.7

Executive Summary

In most major cities, almost every driver has experienced busy cruising for a vacant parking spot in a very crowded parking lot. Countless drivers have wasted their time and gas money while looking for free parking spaces. The Pvision smart parking system will help drivers identify vacant parking spaces in a timely manner before they enter the parking lot.

The Pvision parking guidance system can be easily installed and removed in any parking lot without any damages to the existing facilities. The system has three units which are integrated together to detect and display the current availability of the parking spaces to the users. The system include following three units: input module, embedded module, and output module.

This document will provide detailed specifications regarding the design and implementation method of the first and second prototype models of the smart parking system. From the conceptual design to final realization, all of the design requirements need to be identified and functionality of the smart parking system needs to be thoroughly discussed prior to the development. This system design specification document includes: mechanical design for camera mounting unit, algorithm development for the embedded system, database design, graphical processing unit (GPU), graphical user interface design (GUI) for the first and second prototype models design, and the system test plans in normal and extreme cases, are all described in detail.

The smart parking system will not only benefit the drivers by saving the time and gas spent in finding vacant parking spots, but also will provide the entire parking lot statistics to the parking lot manager. Through the graphical user interface of the system, the parking lot manager will find out the several parking lot statistics by simply clicking on the bar graphs option. Thus, they can more effectively manage their parking lots.

Table of Contents

List of Figures

List of Tables

Glossary

A/D	Analog-to-Digital
BMP	Bitmap image file format
C++	A general-purpose programming language
DC	Direct Current
GPU	Graphics Processing Unit
I/O	Input/Output
IP	Internet Protocol
LAN	Local Area Network
LCD	Liquid Crystal Display
LED	Light-Emitting Diode
OpenCV	Open source Computer Vision
POE	Power-over-Ethernet
RAM	Random Access Memory
RGB	Red, Green, and Blue
SRAM	Static Random Access Memory
USB	Universal Serial Bus

Introduction

The Pvision parking guidance system shows the current status of the parking lot to the drivers before entering the parking lot. The system has three main units: input module, embedded module, and output module. The input module consists of two basic modules: video capture module and video to image conversion module .The video capture module uses two high definition cameras to capture video of the parking lot, and the video to image conversion module is responsible for converting the video to image at rate of approximately five samples per second. The embedded module is core of the entire system. It uses the images generated in the input module to perform the vehicle detection procedure. There are three methods including color detection method, edge detection method, empty parking comparison method. The results obtained from the above three algorithms are analyzed together to provide final vehicle occupancy result in verification function. In the output module, the microcontroller and the LCD display are two main components. The microcontroller will convert the data provided by verification function to readable data for the LCD display. More detailed explanations will be provided in the following document.

The purpose of our smart parking system is to provide an efficient way of helping the drivers to find vacant parking spaces in a huge parking lot without spending more gas and time. To help parking lot managers get the real-time parking lot statistics by using the GUI provided by the system. It will let them monitor and manage the whole parking lot in more efficient way.

Scope and Intended Audience

Scope

This document outlines the entire design specification for the Pvision smart parking system. The design specifications includes all the detailed explanations for the three system modules: input module, embedded module, and output module. In each module, the implementation method has been described in detail.

The first and second prototype's design specifications will be fully described in the following document. The system test plan will describe testing plans in both normal and extreme cases.

Intended Audience

This document is intended to be used by the members of Pvision electronics. The design specification will provide a useful reference to the design engineers to ensure that all requirements have been met. By reading this document, members of Pvision electronics will be easily able to monitor the progress and aim to satisfy all of the requirements of the system during implementation phase.

System Specifications

The Pvision parking convenience systems will make use of its installed cameras to continually take quick pictures of the parking lot at regular intervals of seconds. The embedded system will use these images and employ a series of three algorithms to determine whether the parking spot is vacant or taken. This information will them be displayed on the display output system which will use a schematic of the parking lot layout to present the vacancy of the parking spots visually using green and red colour blocks. The display output consists of a graphics-processing unit (GPU) coupled with an Arduino (prototyping platform) in case of the first prototype. The final prototype will make use of a much larger LCD / LED screen to achieve the same purpose.

The system stats, activity and control will be available to the parking lot administrators - managers and ticket vendors - via the user interface (UI) linked to the embedded system. The cameras and output displays will communicate with the embedded system remotely.

There were many factors that we needed to take into consideration while designing our prototype.
Reliability: The model is very reliable in a sense that switching from small scale model to full scale would not require us to change the software behind it.

Durability: We knew from the beginning that the model was meant to move around and be tested in different lights and conditions. Therefore we designed it in a way that it won't break or get damaged easily.

Sustainability: To help the environment, in our model we made use of materials that can be reused or recycled. The wood can be reused in other projects and the toy cars will be donated to children in need.

Full functionality: The model should be a good test for all the different scenarios we might have. In order to avoid surprizes in real time testing, we made the model in a way to be able to test all the different functionalities that our system might offer before moving to the large scale.

Scalability: All the components in our model should have the ability to be scaled up.

Diversity: The model can be adjusted to perform any test we desire. In that, the cameras have the ability to move around and also elevate up and down.

Overview

Images captured form the cameras are continuously fed to the embedded module. Visual Studio then runs a series of process to analyze and determine the occupancy of designated parking spaces from the images. The source code consists of three components; RGB color comparison method, edge detection method, and empty picture comparison method. Each method has its pros and cons when it comes to determining the vehicle occupancy of each parking space. Thus, along with above three methods, block size control and standard deviation are further used to increase the accuracy of vehicle detection. Since the parking guidance system heavily involves image processing, the software programming aspect of project will be discussed in detail.

Easy BMP library

Easy BMP library is a C++ open source library designed to read, write, and modify uncompressed bitmap files. The library supports grey-scale image conversion, image rotation, and a range-of-pixel copy functions. However, the source code mainly uses the library to access RGB pixels for red, green and blue color values ranging from 0 to 256.

The coordinate system of a BMP file has its origin in the top left corner of the image as [0,0]. The (i, j)th pixel is i pixels from the left and j pixels from the top. The Easy BMP library indexes pixels with the common BMP file coordinate system as seen in Figure 1 below. [1]

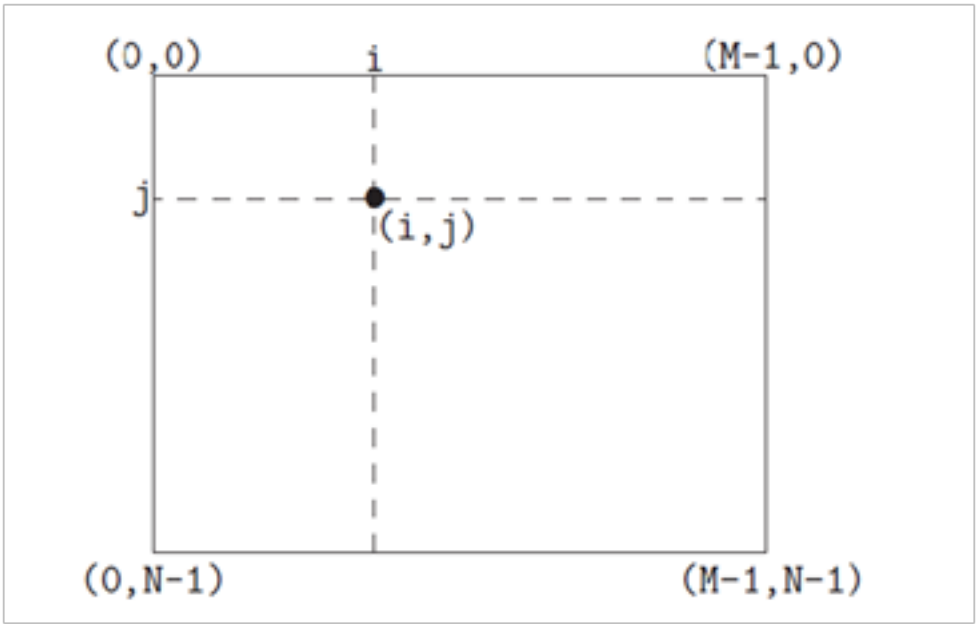


Figure 1: Coordinate system for an M £ N BMP image.

In order to access RGB value of a single pixel, following method is used.

```
RGBApixel Temp = image.GetPixel(xaxis,yaxis);
RGBvalue[0] = (int) Temp.Red;
```

The code for the color Red above is now saved in RGBvalue[0]. The RGB color code for the green and blue colors can be found by repeating the above method.

Easy BMP provides relatively simple libraries which do not require any installation except for adding 4 header files and a single C++ file to the Visual Studio project. Header files define specific data structures and various BMP utilities. The C++ file defines specific methods and handle exceptions.

Block size control

In most images, it is impossible to find an even distribution of color when we zoom into the image to the point where all the individual pixels are visible.

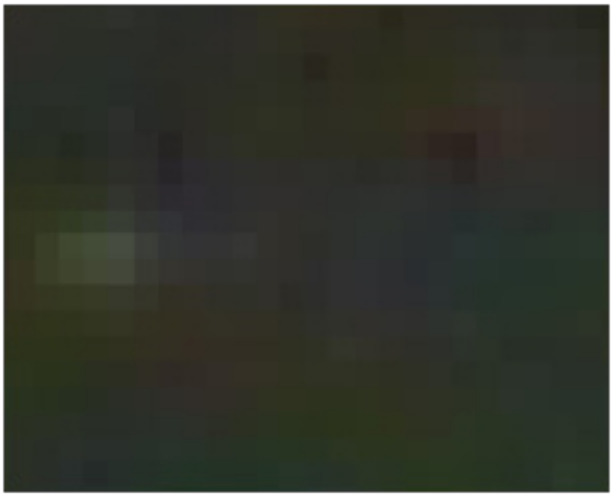


Figure 2. zoom in image

Figure 2 above is a sample picture taken from indoor model parking lot. When we zoom into the image of the surface mimicking the asphalt. While it is dark grey, the zoomed in picture displays a range of colours spread out unevenly. When the color code is analyzed, the RGB color of brightest pixel and that of the darkest pixel can have a significant difference in color values. Table 1 below shows the same.

	Red	Green	Blue
Brightest	70	76	64
Darkest	44	35	30

Table 1: Values of RGB colours from the brightest and darkest pixels in the sample input image

The difference in the RGB values is often big enough to affect the result of vehicle detection, when it is conducted using this method. In order to eliminate this problem, the block-pixel analysis method is integrated into the existing source code. For Figure 2, the average of value of the pixels' RGB colour is calculated. Thus, in our sample, we get Red as 55, Green as 59, and Blue as 58 which turns out to be the around the middle value of their respective brightest and darkest pixels.

While the block size RGB color analysis improves the accuracy of color detection, it also significantly increases the processing time. As the size of block gets bigger, the number of RGB color comparisons and consequently the processing power required by CPU, increases. Table 2 below shows examples of processing time for different sized blocks.

Block size	# of pixel access & calculation	Required time for processing
3*3	9*35 =315	~4.8 sec
9*9	81*35 = 2835	~6.7 sec
15*15	225*35 = 7875	~7.1 sec

Table 2: Processing time for different sized pixel blocks using block-pixel analysis

The increase in the number of pixel accessed does not linearly increase the required image processing time. This is because the time it takes for the operating system to load the images to and from the RAM increases significantly each time a bigger set of pixels is accessed. For our algorithm, it is important to find the optimizing block size that accurately determines the true color of surface while minimizing the delay.

Standard Deviation

As explained above, when the code runs the block-pixel control algorithm, the noise from the camera often interferes with finding the true color of the parking lot surface. In order to further increase the accuracy, standard deviation is used to exclude pixels that show a big difference in colors from their surroundings. When block pixel access begins, the source code stores the RGB values of sample pixels and finds the mean and standard deviation. Equation (1) below is used for finding standard deviation (σ) of a set of values, where X stands for value of single sample,

$$\sigma = \sqrt{\mathrm{E}[(X - \mu)^2]} \quad (1)$$

When mean (μ) and the standard deviation (σ) are calculated, the source code excludes all of the values that are not within one standard deviation of X. Then the average is recalculated to find the more precise surface color.

RGB color comparison

The RGB color comparison method is designed to automatically adjust to the weather and lighting changes. The method first takes a few sample points on the road to find the expected value of the pixels from the image of an empty parking space. The average RGB values of these sample points are then saved. An example of these sample points is shown in Figure 3 as the five white dots.

Figure 3: Side view of the prototype

In order to further increase the accuracy, additional sample points in front of each parking space are accessed. As these extra sample points are placed closer to each specified parking space, more accurate predictions can be made. The average of the first set of sample points (white dots) and the second set of points (yellow) are both used in calculations.

	Sample1 Average (white)	Extra Sample2 near parking space (yellow)	Actual Color found in empty space
Red	72	64	71
Green	71	79	85
Blue	61	69	66

Table 3: Example showing the average RGB values obtained from the test points on the road

It must be noted that the results may be affected by the reflection of light off of the surface. As shown in the table above, the extra samples near parking spaces capture the color closest to the actual color found in empty space.

Edge detection

The RGB colour comparison algorithm returns inaccurate results when lighting is uneven in different spots in the parking lot. The edge detection method can significantly reduce the effects of such lighting changes. A library called OpenCV supports the edge detection image conversion. The original picture and edge-detection version of it are shown in Figures 4 and 5 on the next page.

Figure 4: Original image

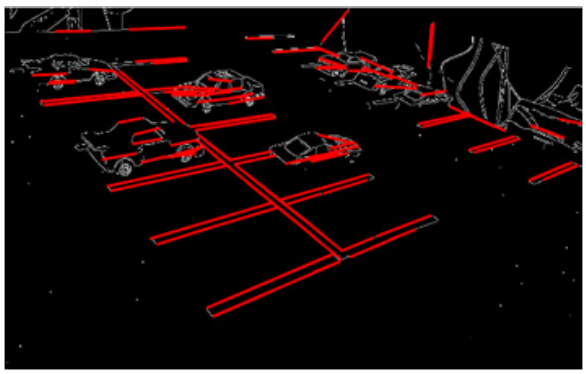


Figure 5: Edge detection image

When cars are located in each parking space, their color difference to the ground and the shape of the vehicle forms the edges. These edges are shown as red lines and white dots in the edge detection image. OpenCV supports simple configuration options to change the sensitivity of edge detection.

Canny (image,edge image,threshold_high,threshold_low,aperture_size)

The threshold_high, threshold_low in the code above determine the sensitivity of the edge detection algorithm. While processing the image conversion code above, it is assumed that important edges are continuous curves. When it detects a sudden change in pixel color, the algorithm analyzes the orientation of the edge to determine the direction of the curve.

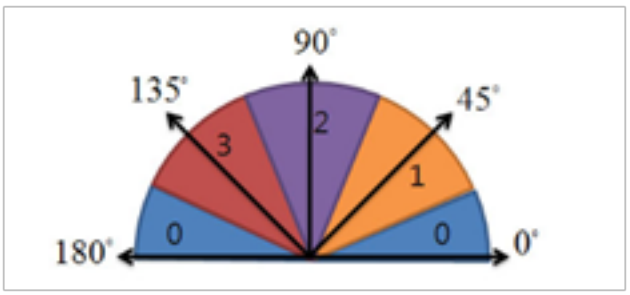
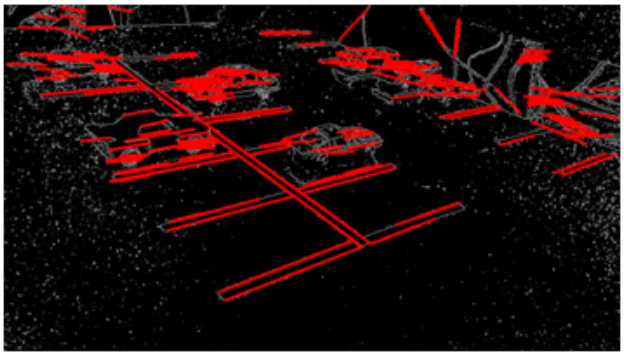
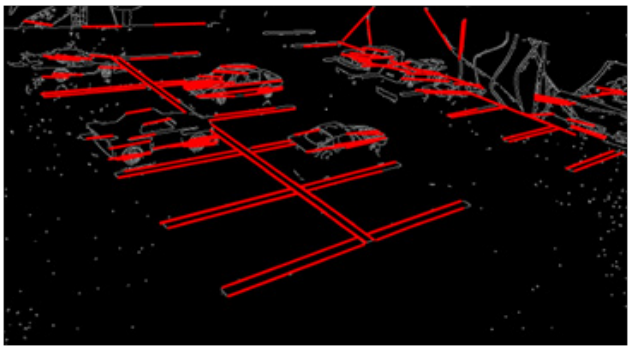


Figure 6: Directional reference for determining the offset as required by the canny algorithm

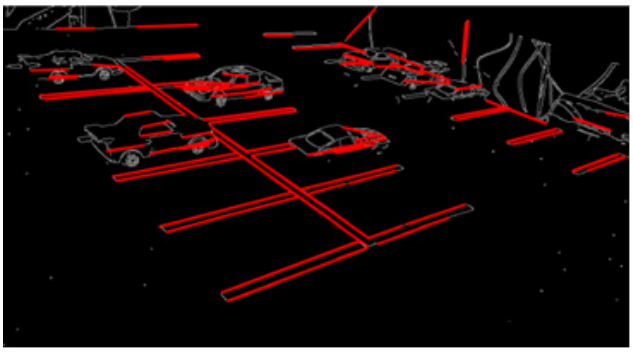
If the orientation offset indicates the direction 0, the canny algorithm compares the value to the pixels on the immediate left and right. If the offset is 1, it looks for pixel in (-1,-1), (+1,+1). An offset of 3 means it will look for a pixel in (+1,-1), (-1,+1). Similarly, an offset of 2 would compare to the pixels at the top and bottom of the given pixel. The canny algorithm first applies a large threshold, which marks out the genuine edges using the above directional information in Figure 6 above. Next, the lower threshold is used to trace the faint sections of the edges. The following images in Figure 7 are the edge-detection converted images with different threshold settings.



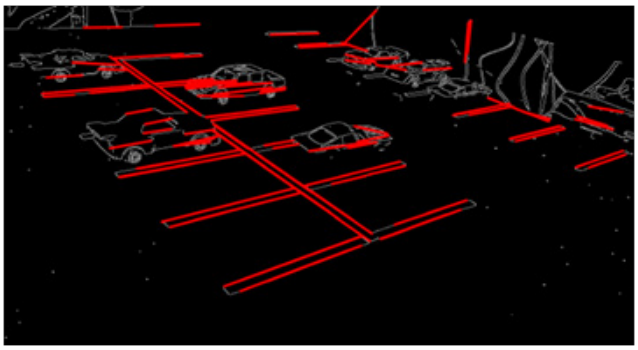
threshold_high:50 threshold_low:50



threshold_high:100 threshold_low:50



threshold_high:150 threshold_low:50



threshold_high:150 threshold_low:100

Figure 7: Edge detection algorithm setting

As shown in last page's picture, a high threshold of 150 returns the most accurate edge-detection images while a low threshold does not play a big role in the edge detection algorithm.

When the edges-detection images are successfully obtained, the source code is designed to take the new image to do further image processing. After defining a big sized rectangular block as the middle of the parking spot and counting the number of red and white pixels within the block. When they amount to a number larger than the tolerance level we set, it indicates an occupied parking space.

Empty-picture comparison

Although in real situations, regularly taking an all-encompassing picture of an empty parking lot is unrealistic, the empty-picture comparison method algorithm guarantees close to 100% accuracy for our first prototype. Assuming the camera mounting location and the lighting do not change in our model as well as most of indoor facilities, simply comparing an empty parking lot picture to that of an occupied parking lot using the RGB color code can return reliable vehicle detection result. However, when significant lighting changes occur, it will be liable to interpret empty spaces as occupied. Furthermore, as the operating system continuously needs to load the images back and forth to compare the color values, this algorithm require the longest time for processing. This can be visualized in Figure 8 below.

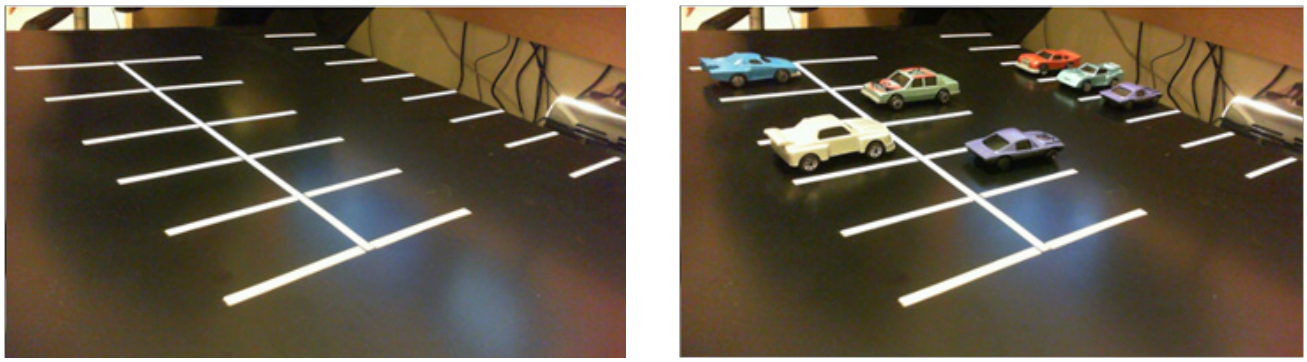


Figure 8: Empty-occupied picture comparison

Final Vehicle determination Module

For every frame of image from the camera, the above 3 algorithms return total of 6 vehicle detection results. For the first indoor prototype, the result from the empty picture comparison will be made to weigh in more than two other methods owing to its higher accuracy. For the second outdoor prototype, the results from color RGB algorithm and edge detection algorithm will gain greater consideration since lighting will play a crucial role outdoors. Also, depending on the camera placement, there will be blind spots created by other vehicles located closer to the camera. In these parking spaces, the vehicle determination module will eliminate the results obtained from blind spot images when determining the final results.

Additionally, when the reference RGB color from RGB color comparison algorithm indicates that red, green, blue RGB code values of reference points are all above 230, it is safe to assume that the ground is covered with snow. In this case, the final vehicle determination module shows the results of edge-detection method as final output. When there is snow in the ground, RGB color comparison method and picture compare method become useless as it's hard to retrieve the reference color and image.

Database

In the Pvision smart parking system, a graphical user interface will be provided to the parking lot managers to help them manage their parking lots more efficiently. When the parking lot manager clicks on a button on the user interface called statistics and rates, user interface will provide the overall statistics for the parking lot including the number of total spaces, the number of available spaces, the number of occupied spaces and the time that designated parking spot has been occupied or not occupied. Thus, in order to implement this design specification, we built a database by using the C++ programming language to achieve this requirement.

There are four main arrays built and included in the database: time array, occupation status array, total occupation time array and total free time array.

For the time array, we defined some functions to calculate the time for a designated spot from occupied to not occupied since the last car came in or left.

For the occupation status array, we wrote a boolean function to implement this array. If the boolean value returned by the function is true, so the current status of the designated spot is occupied. Otherwise, the boolean value returned is false, the parking spot is not occupied.

The total occupation time array, means the time for a car has been parked in a specified parking spot from the starting of the system to the current time. We used a loop to keep adding the occupied time for every car comes into the designated spot till the current time.

The total free time array has the same structure as the total occupation time array, which stands for the time that designated spot has been unoccupied from the starting of the system to the current time.

With the above four statistics showing to the parking lot managers, they will have better knowing the current status of the whole parking lot. Thus, they can monitor their parking lots in more efficient ways.

The output display lets the user interact with the system. The Display shows the basic layout of the parking lot and the status of each parking space at any instant of time. The purpose of the display is providing the user a visual picture of the database. This visual picture helps the users to have easy access to information about each parking space. The output display hardware includes two main parts: the Smart GPU and the Arduino Uno board. These two components work together to display the result of image processing on the output display screen. Arduino Uno board includes a microcontroller which is programmed to read the data and to control the SmartGPU. The SmartGPU is an embedded board with a touch color LCD.

Arduino Uno Board

The Arduino Uno is a microcontroller board based on the ATmega328. The microcontroller is programmed using C++ to read the data generated by the image processing stored in the database and convert them to the data which can be represented on the SmartGPU. The microcontroller is programmed using C++ in Microsoft Visual Studio environment to read the data stored in a text file in a timely fashion and convert the parallel data to serial data readable by the SmartGPU. The microcontroller reads the text file which contains a Boolean value for each parking space and represents zeros with green rectangles and ones with red rectangles on SmartGPU.

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328)
SRAM	2 KB (ATmega328)
Clock Speed	16 MHz

Table 4: Summary of Microcontroller specifications

The code uploaded to the microcontroller is synchronized with the image processing code to ensure the output data is sent to SmartGPU as timely efficient as possible.The Arduino Uno includes 14 digital input/output pins which 6 of them are used for outputs. The microcontroller is simply powered by a computer using a USB cable. It can also be powered using an AC-to-DC adapter. A summary of Arduino Uno board specifications is represented in table 1.[8]

The Smart GPU

The Smart GPU LCD is a touch controlled screen that displays the layout of the parking, the location of unoccupied parking spaces, the number of free spaces, and the time for occupied parking spaces. The Smart GPU displays the logo of the Pvision electronics as default in the standby mode when the system is not being used by users. The layout of the parking is displayed on the screen in the active mode when user touches the screen as it is shown in Figure 9 for a parking lot with the capacity of fifteen parking spaces.

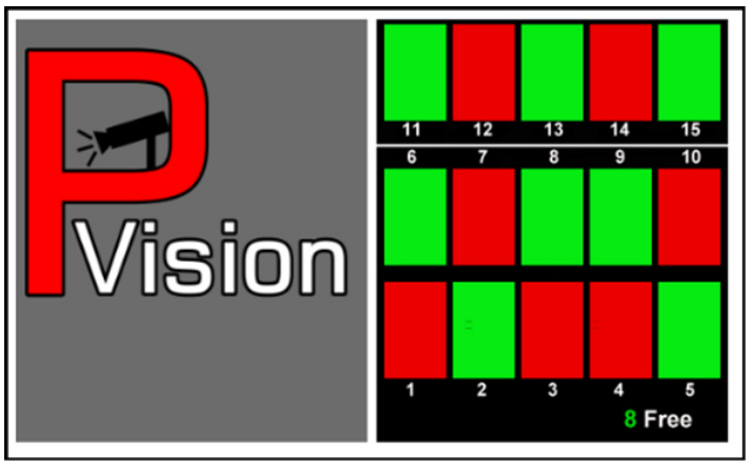


Figure 9: The LCD display in the standby mode on the left and the in the active mode on the right

The color of each parking space is the indication of its availability at each instant of time. The red color indicates the occupied spaces and the green color indicates the unoccupied spaces. The number of free parking spaces is displayed on bottom corner of the screen. The time for which that a space has been occupied is displayed when the user touches the occupied parking spaces.

The screen displays goes back to standby mode after ten seconds if there is no interaction between system and the user and the system is not being used. The standby mode is optional and can be excluded from the system depending on the parking lot where the system is being used. The standby mode can be used to provide other information such as temperature or advertisements.

Item	Standard Value
Display Type	240*(R G B)*320 Dots
LCD Type	a-si TFT, Positive,Transmissive Type
Screen Size(inch)	2.4"(Diagonal)
Mechanical Dimensions	54 mm (W) x 69.5 mm (L) x 7 mm(H)
Viewing Direction	12 O'clock
Color Configuration	R.G.B. vertical stripe
Backlight Type	White LED B/L
LCD Panel Active Area	36.72 mm (W) x 48.96 (L) mm
Outline Dimension	42.32 mm (W) x 59.03 (L) mm
System power supply Voltage	2.7 V – 3.4 V
Input Voltage	-0.3 V – 3.7 V
Operating Temperature	-20 – +70 Degree C
Power Supply Current	60 mA

Table 5: Summary of SmartGPU specifications [4]

The smart GPU and the Arduino board work together to output the result the LCD screen. SmartGPU is connected using input voltage pins, digital output pins, and reset pin to the Arduino Uno board.

Encasement of the Output Display

First Prototype

The first prototype will make use of a GPU and an Arduino board to display the parking layout on the mini LCD (Figure 10). The first prototype will involve a model parking lot simulated by a hard board parking lot and toy cars. The testing for this will take place indoors at room temperature and low humidity as compared to the outdoor parking lots. As such, the output display will require minimal protection in the form of a plastic enclosure measuring 55 mm (H) x 70.5 mm (L) x 8 mm(W), which has slots for all the port and the LCD screen.

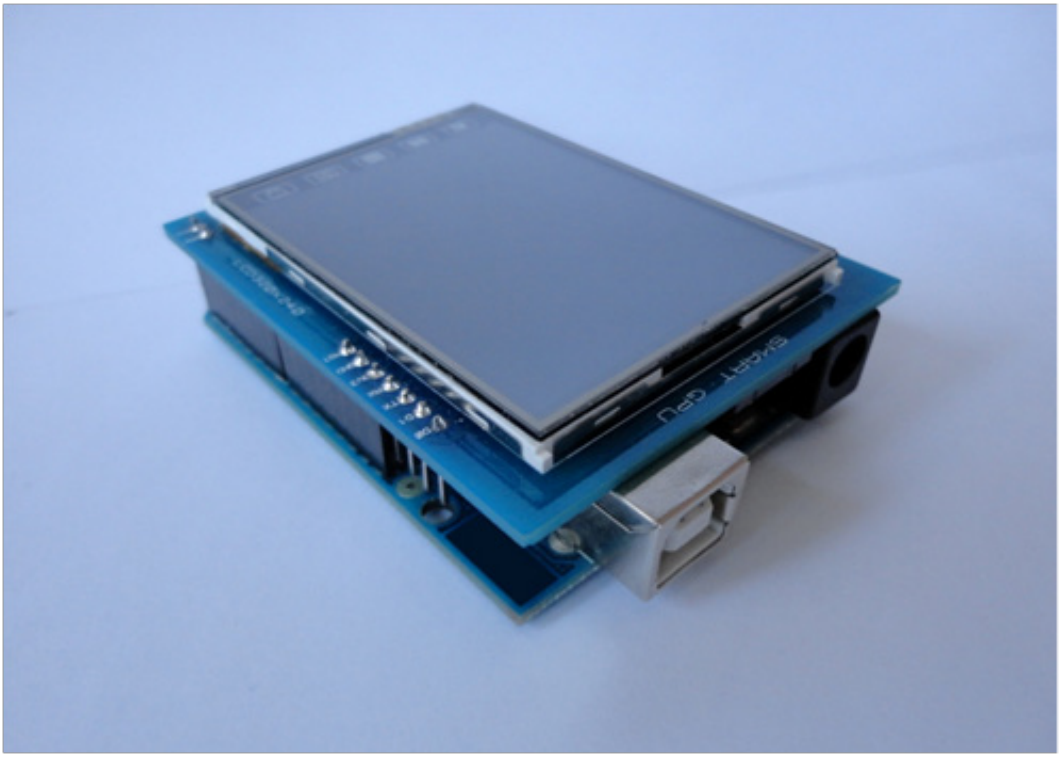


Figure 10: GPU with LCD screen mounted on an Arduino prototyping board

Second Prototype

The second prototype will see the use of a much larger LCD screen, which will need protection from dust, moisture, sun light, outside temperatures and wind. Just like the P12 Outdoor LED (Figure 11) manufactured by Shenzhen ERALED Optoelectrnics Co., Ltd, the second prototype will need two levels of protection. Firstly, it will need a first level protection to prevent the entry of dust and other particles into the system. The second barrier will have to keep the system intact from moisture.

Figure 11: P12 Outdoor LED, Shenzhen ERALED Optoelectrnics Co., Ltd,

For our first prototype we down-sized an actual parking lot with a factor of 1/100 and used toy cars to test our model. The prototype consists of three main parts, the table (representing the parking surface), the camera mount (representing the pole in the parking lot) and the web cameras (representing the IP cameras).

Table

Table is the foundation of the model. It represents the layout of the parking lot. In order to fully represent an actual parking lot, two features were taken into consideration. Firstly each space was measured to match the full scale parking lot with a fixed scaling down factor and they were equally aligned. Secondly, we included 4 handicap parking spaces in the model, each being wider than the normal spaces.

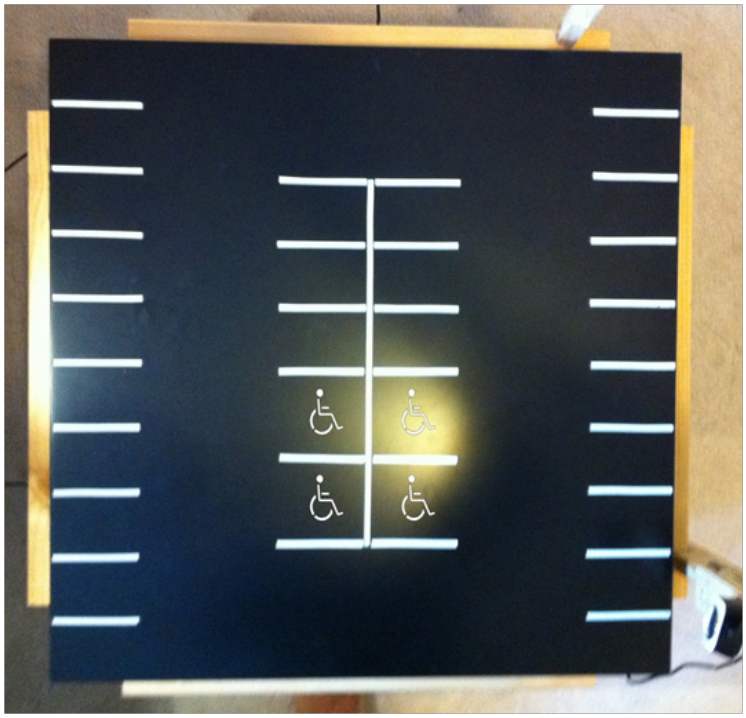


Figure 12: Top view of the first prototype showing the layout of the parking space.

These features will help us later to write the most reliable software which works with any parking lot no matter what the layout is. The surface of the table was later painted with non-reflective paint to avoid reflection off of the surface which could be interpreted incorrectly by the embedded system based on the camera image of that reflection.

Each side of the table has a wing that is designed to hold the camera mounts. The wings go all around the table so that the cameras can be places anywhere around the table; hence the model can test any possible scenario.

Figure 13: Side view of the first prototype showing the wings before getting painted

Figure 14: Side view of the first prototype showing the wings after paint

Camera Mount

The camera mount must have three main functionalities. First it must be able to hold the camera in a stable position above the table. In order to make this happen, a piece of wood carefully cut to the base shape of the camera is firmly screwed to the main mount using a latch. This part is oriented to face the parking model so that the cameras can capture accurate images.

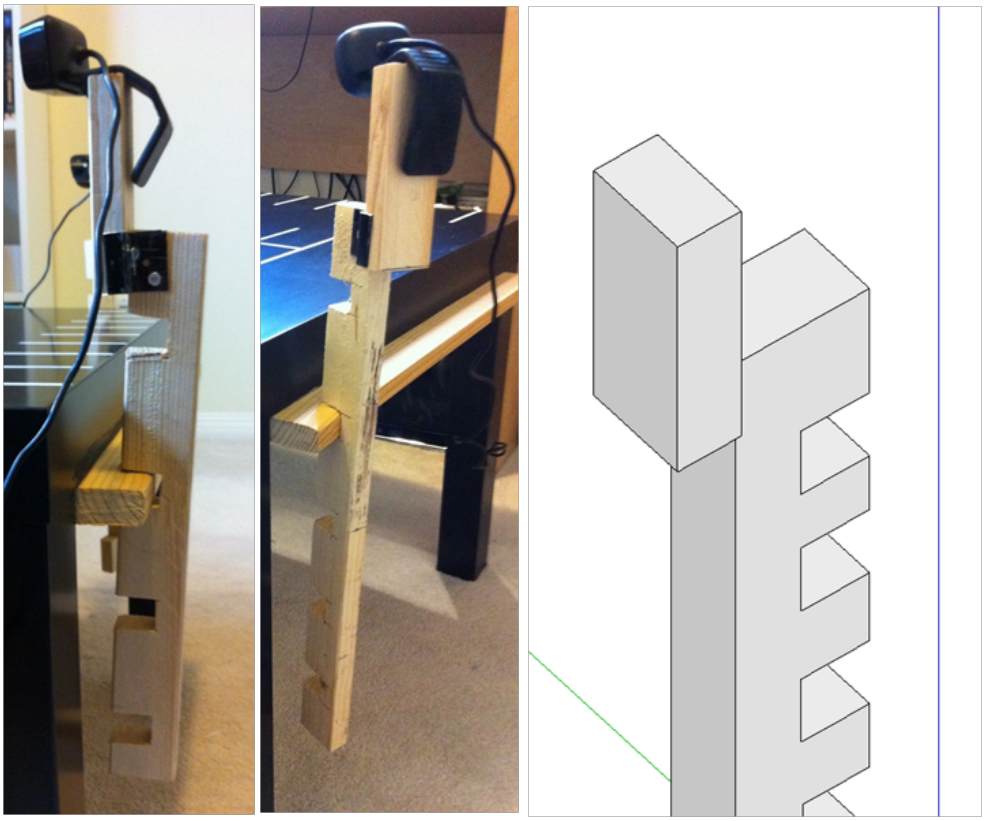


Figure 15: Camera mount showing the holder of the camera

The second functionality of the camera mount is that it can be placed anywhere around the table. Due to the placement of wings around the table, our mount can firmly fit into place in any desired location around the table. This will give us the ability to test our model in many different angles.

Last but not least, the third functionality is that the mount can be elevated to 4 (and 5 in one of the mounts) different heights. As you can see in Figure 5, there are specific spaces designed on the mount that can be used to elevate the mount higher or lower. This will give us the ability to test our model in different elevations and to see which height would give us the best results.

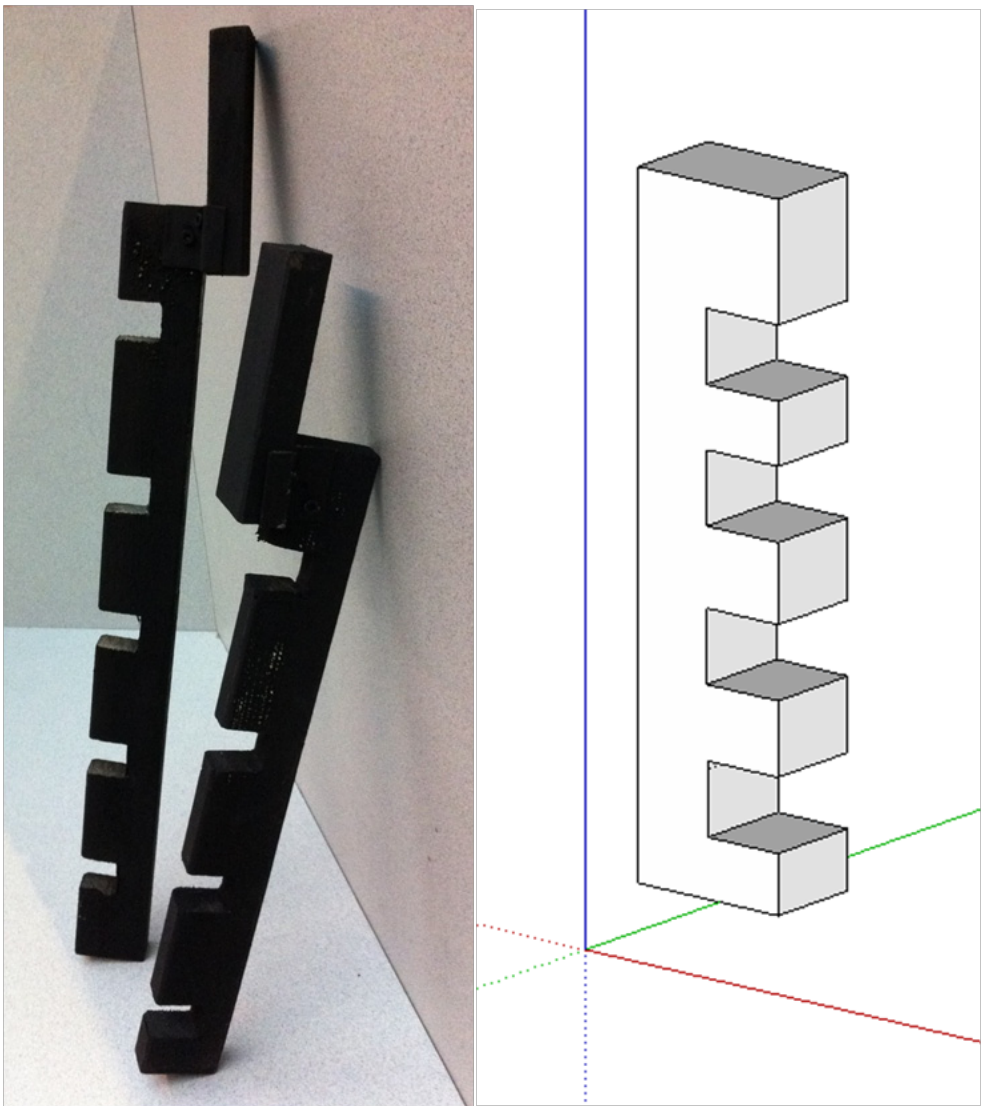


Figure 16: Two camera mounts showing the design for different elevations

The carved spaces in the wood will fit into the wing around our table. Using these mounts we have unlimited number of possibilities to test our system leaving no scenario undone.

Cameras

Two high definition cameras are used in our model. The resolutions for our cameras are 720p. The software will take images every few seconds of each camera and send it to our embedded software to perform the image processing on them. As shown in Figure 6, the two cameras are mounted on two sides of the model and work as a master and slave. In that, there is one main camera and the second one works as a support to the main camera.

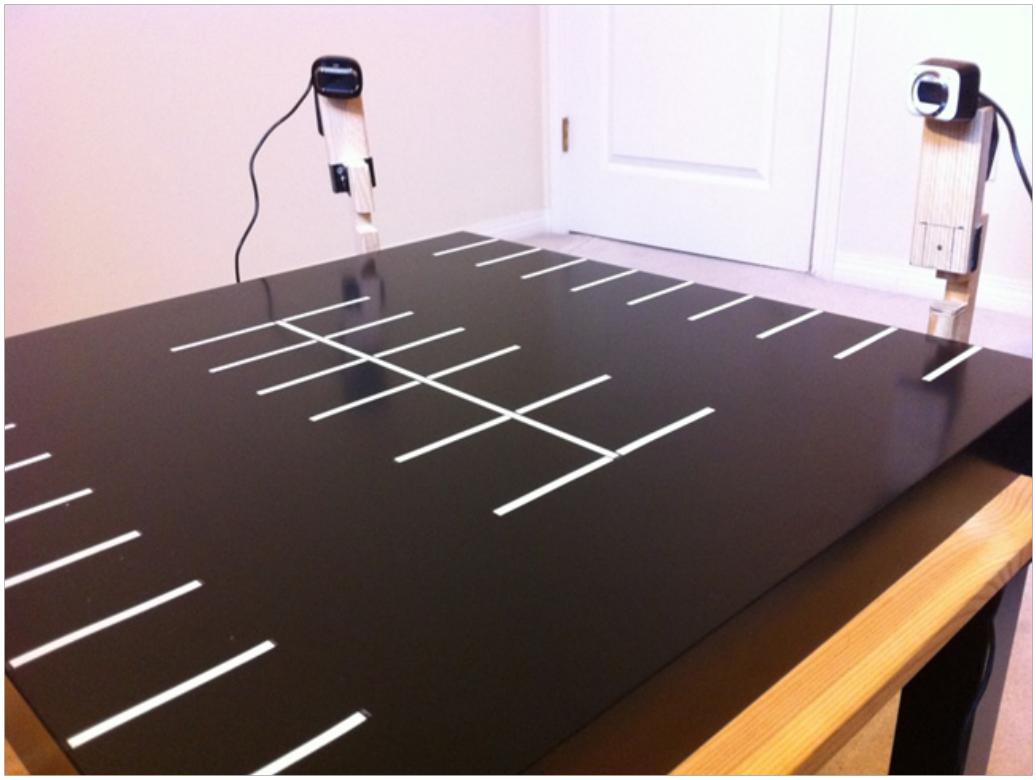


Figure 17: Two HD web cameras facing the model from different angles

In the large scale prototype, these cameras are going to be replaced with IP cameras connected to a POE (Power Over Ethernet) Switch. The IP cameras have similar resolutions and do not require us to change anything in the software transferred from the first prototype.

Second Prototype Demo Location

The second prototype of the system is tested in an actual parking lot located in Simon Fraser University Burnaby campus. The parking lot is a permit parking lot next to the Madge Hogarth residence building in west mall center. The system covers up to fifteen parking spaces using two IP cameras each one installed on a pole. The other option is installing one of the cameras on top of the Madge Hogarth residence building if the installation permission is granted. The installation of one of the IP cameras on the top of the building can provide pictures with better angles to the system. The other camera is installed on the opposite side of the parking lot to cover opposite angles. More parking spaces can be covered by installation of additional cameras.

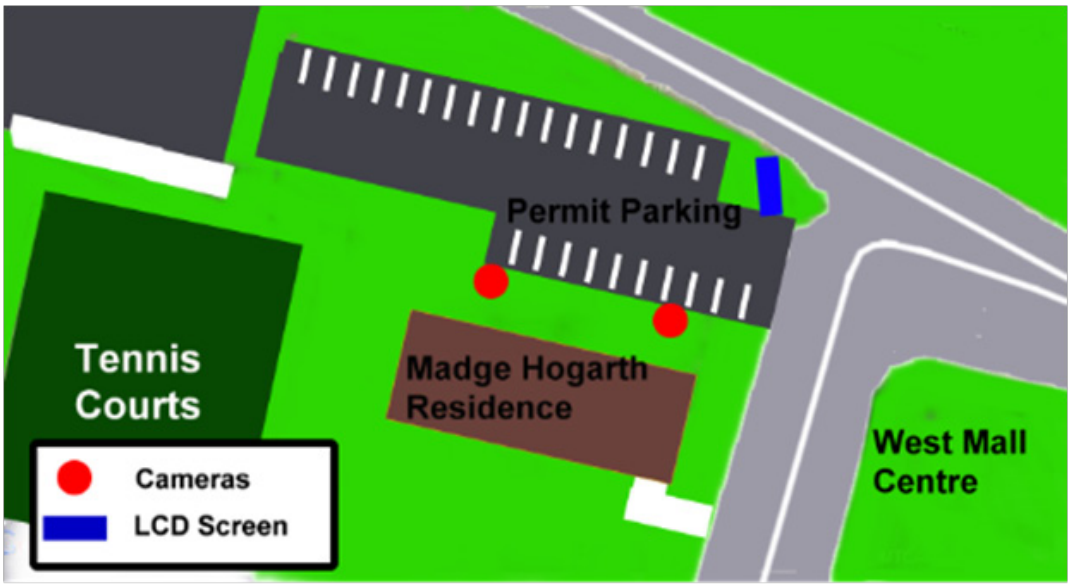


Figure 18: The possible location for installation of IP cameras

The cameras are installed in a way to cover the parking lot as much as possible and not be blocked by the trees. However, the trees may still block some portion of the view and this problem can be resolved by adjusting the program to ignore the trees. The possible locations of the IP cameras and the LCD screen are represented in Figure 18.The sample input images captured by the two IP cameras are represented in Figure 18 which are used for image processing.

Here are images taken at the specified location

Figure 19: The possible images captured by IP cameras

Since the system relies heavily on image processing, environmental factors play a great role in affecting the accuracy of the results. The reliability and durability of each component of system in normal and extreme environment require extensive testing. Below is the list of the factors that are likely to influence the image processing as well as the testing plans that aim to eliminate or minimize the system failure and incorrect results.

Weather / Lighting

Although the vehicle detection algorithms are designed to work in variable lighting and weather conditions, it is impossible for them to generate correct results under extreme conditions of the same. The system testing will be conducted on days that are sunny, overcast, with light rain, with heavy rain, and with snow to ensure that the algorithm does not output an empty parking space as occupied or vice versa. Unfortunately, 2 IP cameras that are mounted in the real parking lot do not have night vision functionality. Therefore the lighting factors testing will be conducted from during the daytime only (approximately 7AM ~ 6PM).

Camera Position

Wide angle cameras are ideal for capturing large a number of cars in a specified parking lot. However, as the cameras installed for the 2nd prototype are regular cameras that can only capture a limited field of view, it is important that the cameras are mounted in a location where which captures most number of parking spaces. Thus, the height at which the cameras are mounted is pivotal. The higher the cameras located, the clearer and wider view of parking spaces secured. The cameras will be mounted in various heights such as 3m, 5m, 7m on a pole or on a nearby building. Two cameras will be mounted in various positions around the parking lot. The angle at which the two cameras are positioned is also very important. The cameras will be mounted side by side, at 90 degree, and on opposite sides of the parking lot.

Camera

Although the cameras for 2nd prototype are waterproof, it is important to ensure that further testing guarantees the durability of the cameras. The cameras will be left outside for several days along with the LAN cables that are connect cameras and switch.

LCD display / microcontroller

The microcontroller and display need to make sure that input from the embedded unit can be displayed in a timely and error-free manner. Visibility tests are required for the items that will appear on-screen. If the clarity of vehicle occupancy indication on-screen is low, drivers will not leverage the convenience of our parking guidance system. We will conduct visibility testing with drivers who come to park their cars in the designated parking lot. We will install the display on left/right side of entrance of parking lot while changing the distance of the display from the driver. Drivers in different types of vehicle will be invited to participate in our testing. Through these series of testing, the Pvision team will determine the best LCD display position. Packaging needs to be tested to make sure that the screen inside the package is well protected from weather and theft. The system will be left out in the parking lot for several days in sunny and rainy weather to make sure the it can withstand various weather conditions.

Algorithm reliability Testing

Before system integration, the source code will be continuously run for 5 hours while memory overflow and compliers are being closely monitored. Processing time will be measured for different quality of image files and system environments. The amount of time it takes for a specific parking space to be occupied or vacated will be measured to make sure that system does not slow down after a certain period of time.

Normal Case 1: A car enters the parking lot

User Input: User drives in.

Conditions: In a relatively sunny weather day, a white civic drives into the parking lot. The driver in the car identifies an empty parking space on LCD screen and moves into that parking space. The car is parked directly in the middle of the parking space where two cameras both have clear visibilities. Driver walks out of the parked car.
Expected Observation: The driver successfully identifies an empty parking space on the LCD display. S/he manages to quickly drive to the empty parking space and park the car. Within 15 seconds after the parking, the current status of the parking space turns from empty to occupied. The parking manager accesses the user interface (UI) and is able to observe this change.

Normal Case 2: driving out from the parking lot

User Input: User drives out of the parking lot

Conditions: In a light rainy weather day, an owner of black SUV walks to his vehicle in the parking lot. The driver then gets in to car seat and drives the car out of the parking lot.
Expected Observation: Within 15 seconds of the black SUV leaving the parking lot, the status of the parking space turns from occupied to empty.

Normal Case 3: A car enters a fully occupied parking lot

User Input: User drives in.

Conditions: On a relatively sunny day at 5pm, a blue Mercedes drives into the parking lot. The driver in the car identifies no empty parking space on LCD screen however still drives into the parking space, and waits for other car to drive out. After a few minutes of waiting, one of the parking spaces becomes available and the owner parks his car in the middle of the parking space where only one camera has clear visibilities.
Expected Observation: The driver finds no empty parking space on the LCD display. However, when s/he sees the new empty space, s/he quickly drives into the empty parking space and park his/her car. Since the parking space is immediately occupied after identified as free space, the current status of the parking space is still set to “occupied”.

Extreme Case 1: A car enters the parking lot in snowing weather

User Input: User drives in.

Conditions: The weather is extremely bad. It has been snowing all day and the ground in covered with snow. Then a white civic drives into the parking lot. The car is parked directly in the middle of the parking space; however the driver is not sure if he parked in a right space since the lines in the ground are not visible. Both of the cameras have clear visibilities to the parking space except the ground is covered with snow.
Expected Observation: The final determination module automatically detects that there is snow on the ground as it analyzes the reference ground color. Then the final vehicle detection module stops ignoring results from RGB color comparison method and picture compare method. It will directly output the edge detection algorithm as the final algorithm

Extreme Case 2: A motor cycle enters the parking lot

User Input: A motor cycle enters the parking lot

Conditions: In a clear weather. A person on motor cycle drives into the parking lot. The cyclist checks the current parking lot status on the LCD screen. The motorcycle is parked in the middle of the parking space. Only one of the cameras have clear visibilities of the parking space.
Expected Observation: If the motorcycle is located in the middle of the parking space and the motorcycle covers the exact spot which the algorithm make reference, all of three algorithms will successfully detect the occupancy. However, if small motorcycles are parked close to one side of the parking space, three algorithms will all fail to detect the occupancy.

Company Profile

Mohammad Akhlaghi
Chief Executive Officer (CEO)

Mohammad Akhlaghi is a fourth year electronics engineering student at Simon Fraser University and he is also working towards his minor in Business. A team-focused, task-oriented, and results-driven engineering student with a strong business background. He has gained a lot of technical knowledge through two coop work terms in the industry and research. Experienced in project management, technical support, and working in multi-disciplinary teams. Puts a strong emphasis on effective communication, organizational sustainability, and meeting client and market needs.

Noah Park
Chief Financial Officer (CFO)

Noah Park is a fifth year computer engineering student at Simon Fraser University. He has gained significant practical experience through co-op work terms at Siemens and DongWon where he worked as a software engineer for designing and developing software for analyzing test data. These coop experiences exposed him to the business side and organizations of a large corporation as well. As a student, he has gained wide-range of knowledge in programming languages such as C/C++ and Java. He is also familiar with using VHDL to design custom circuits. With his extensive experience in programming, his contribution to the team will be focused in the area of software engineering.

Milad Hajihassan
Chief Technology Officer (CTO)

Milad is a fourth year systems engineering student at Simon Fraser University. His areas of interest include software development and graphic designing. He has taken courses in microelectronics, robotics, and real-time and embedded systems during past few years of his studies. He has experience working with microcontrollers as well as implementing games using C and Linux which he gained from the previous engineering projects.

Oshi Mathur
Chief Operating Officer (COO)

Oshi is in his fourth year of my engineering degree with a concentration in Biomedical Engineering. Thanks to the engineering courses offered at SFU, he finds himself adept at the basics of circuit design, programming, and problem solving. A two-semester Co-op work term at Active Network Ltd. gave him great background in working as part of a big team while focusing on the technical back-end of software solutions that run the city halls, recreation centers and universities of all major North American Cities. Liaising with clients and different departments both within and outside a company are things that his position taught him best. Through plethora of extra-curricular employment in SFU's university Life, he has had the chance of working with a diverse population of people wile gaining leadership skills.

YuJie Xu
Chief Marketing Officer (CMO)

YuJie Xu is a fourth year electronics engineering student at Simon Fraser University. Through one year co-op work term at the Montior King Ltd, he obtained plenty of practical experiences on hardware assembling. He has finished the programming languages, like C or C++ and real-time and embedded systems in his third year studies. With his co-op work experience and course studies, he will be responsible for both hardware and software design in our company.

Conclusion

The design specification provides a detailed description and specification of the entire system including The three modules of the smart parking system. The document also discusses the system test plan in number of cases. These test plans will significantly increase the reliability of the overall system. The implementation method for each module is explained in detail. The prototype which satisfies all of above specifications will be demonstrated during demo in April 2012.

References

[1] Macklin, Paul, "EasyBMP Cross – Platform Windows BMP Library: Home", EasyBMP, 2006. [Online]. Project http://easybmp.sourceforge.net/
[Retrieved: 29, February 2012]

[2] Kuntz, Noah, "OpenCV Tutorial 5", 2009. [Online].
http://dasl.mem.drexel.edu/~noahKuntz/openCVTut5.html
[Retrieved: 3, March 2012]

[3] http://en.wikipedia.org/wiki/Canny_edge_detector
[Retrieved: 24, February 2012]

[4] http://arduino.cc/playground/SmartGPU/SmartGPU
[Retrieved: 5, March 2012]

[5] www.era-led.com/leddisplay_product/p10_outdoor_1R1G1B_LED_display.html
[Retrieved: 1, March 2012]

[6] http://eraled.centerblog.net/397-outdoor-led-display-protection-grade-ip-analytical
[Retrieved: 21, February 2012]

[7] Arduino Tutorial Learn Electronics using Arduino [Online] http://www.ladyada.net/learn/arduino/
[Accessed: March 2012].

[8] Arduino specification [Online] http://arduino.cc/en/Main/arduinoBoardUno
[Accessed: March 2012].

[9] Agasio Agasio A612-POE Outdoor Power Over Ethernet IP Camera with IR-Cut Filter
http://agasio.com/poe-cameras/agasio-a612-poe-outdoor-power-over-ethernet-ip-camera-ir-cut.html
[Accessed: March 2012]

[10] Axis communications Network technologies Local area network and Ethernet
http://www.axis.com/products/video/about_networkvideo/ip_networks.htm
[Accessed: March 2012]