Parking Lot Controller System

Nathan Potter – Old Dominion University

Research Sponsors – Dr. Ayman Elmesalami & Dr. Soad Ibrahim

Introduction

This system is designed to accomplish two objectives:

1. Determine if a parking lot has open spaces or if they are all occupied
2. Illuminate a LED that represents the parking lot status, red for full and green if not

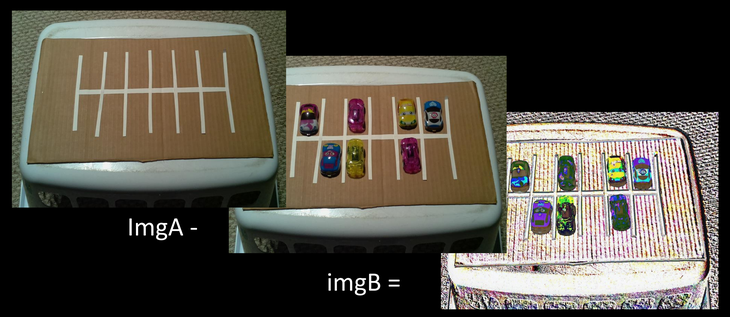
This system achieves these objectives through the exploration of image processing techniques that allow the manipulation and the extraction of quantitative information.

Image Processing

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be an image or characteristics/features associated with that image.

This system utilizes a simple form of image processing known as image arithmetic. The operation within image arithmetic used to achieve the system's objectives is pixel subtraction.

The pixel subtraction operation takes two images as input and produces as output, a third image whose pixel values are simply those of the first image minus the corresponding pixel values from the second image.



ImgA -

imgB =

imgC

The System

Constraints

The system was constrained to a model parking lot and toy vehicles to mock a real parking lot scenario. The access to a public/private parking lot and life-sized vehicles was unavailable.

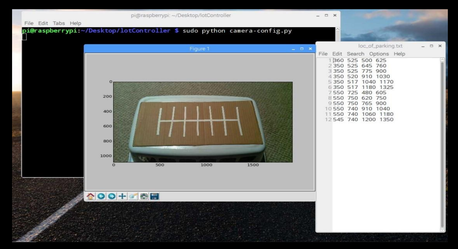
Hardware

- Raspberry Pi
- Raspberry Pi capable camera
- Breadboard
- Red & Green LED lights
- Resistor & Jumper wires

System Configuration

Prior to the main program, is the user configuring the controller. This initial step consists of two parts:

1. Capture a reference image (empty parking lot)
2. Input the coordinate sets that correlate to each parking space



Assumptions

Before the system runs the main program, assume that:

- Fixed camera position
- Consistent lighting
- Corresponding pixel values from reference to live image will not be identical

Algorithm

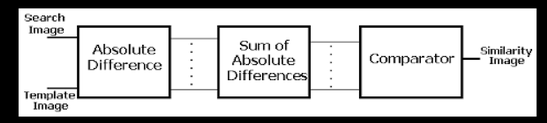
The algorithm for this system as previously noted, starts with a configuration of the controller, establishing a reference image and coordinate sets that represent each parking space. The coordinate sets will be inputted into a text file which will be read in later applied to the live image.

Preceding the configuration, the main program will proceed to execute. The algorithm can be described in the following steps:

1. Capture a "live" image of the parking lot
2. Open the reference and live image
3. Covert/ensure both images to RGB mode
4. Remove the red and blue color channels from both images, leaving just the green channel
5. Read-in the coordinates previously recorded
6. Slice each parking space for each image
7. Compare each "live" parking space to the corresponding "reference" space
8. Calculate the "Sum of Absolute Differences" for each space comparison
9. Compare each SAD result to a preprogramed threshold, to determine if the space is occupied or not
10. Calculate the number of occupied spaces in the live image
11. Red LED if number of occupied = total spaces, green LED if not
12. Repeat

Sum of Absolute Differences

In digital image processing, the sum of absolute differences (SAD) is **a measure of the similarity between image blocks**. It is calculated by taking the absolute difference between each pixel in the original, reference image, block and the corresponding pixel in the block being used for comparison (ie. live image).



This system utilizes SAD to quantitatively describe between the reference parking spaces and the live parking spaces. In conjunction, the SAD is compared to a heuristic threshold to derive at the final determination, whether the space is filled or not.