Heaven's Light is Our Guide

Rajshahi University of Engineering & Technology

Project Name:	Automatic Car Parking System Using CCTV Footage.
Course Title:	Electronic Shop Practice
Course No:	EEE 3100

Submitted By		***Submitted to***
Name:	Animesh Sarkar Tusher	**Md. Nuhi-Alamin**
Roll:	1801077	**Lecturer**
Section:	B	Dept. of Electrical & Electronic Engineering.
Dept. :	*Electrical & Electronic Engineering .*	Rajshahi University of Engineering & Technology.

Heaven's Light is Our Guide

Rajshahi University of Engineering & Technology

Course Title :	Electronic Shop Practice		
Course Code :	EEE 3100		
Submitted By		***Submitted to***	
Name:	Animesh Sarkar Tusher	**Md. Nuhi-Alamin**	
Roll:	1801077	**Lecturer**	
Section:	B	Dept. of Electrical & Electronic Engineering.	
Dept. :	*Electrical & Electronic Engineering .*	Rajshahi University of Engineering & Technology.	

Exp. No	**Experiment Name**
1	Study of Preventive Maintenance.
2	Introduction to PCB's and Overview of Making Them.

Project Name Automatic Carparking System Using CCTV Footage.

Project Name: Automatic car parking system using CCTV footage.

Objectives:

(1) Stopping the usage of "**IR Sensor**" to automate parking garages is a particular aim of this initiative. In our project, no **IR sensors** are utilized.
(2) The goal of this project is to use image processing to create an intelligent parking system.

Project Component:

CCTV: CCTV (closed-circuit television) is a TV system in which signals are not publicly distributed but are monitored, primarily for surveillance and security purposes.
Closed-circuit television, also known as video surveillance, is the use of video cameras to transmit a signal to a specific place, on a limited set of monitors. This system allows the use of videos cameras to monitor the interior and exterior of a property, transmitting the signal to a monitor or set of monitors.

We use mobile phone camera as a CCTV camera by USB cable. It observes the parking slots frame by frame. The entering and exit points of the parking spot is also observed. The parking slots are defined by programming

Microcontroller: the Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button.

Memory: SRAM

Storage: Flash, EEPROM

Developer: Arduino

CPU: microchip AVR (8-bit)

RGB LED: An RGB LED is basically an LED package that can produce almost any color. RGB (red, green, and blue) refers to a system for representing the colors to be used. There are two kinds of RGB

LEDs: common cathode and common anode RGB LEDs. The common cathode has all the cathode of the LED connected together.

We use common cathode RGB LED. The blue light will active when any car trace place at exit point. If any car take place at the enter point but there is no slot available, the red light will active. If there any slot is available, green light will active.

Diffuser: An LED diffuser sheet is, typically, a component in screen backlights. Its main purpose is to evenly distribute light from LEDs on the edge of the screen, so that there are no bright spots near them. It is usually composed of many sheets of plastic in varying thicknesses and opacity or reflectivity.

LCD display: A liquid-crystal display (LCD) is a flat-panel display or other electronically modulated optical device that uses the light-modulating properties of liquid crystals combined with polarizers. Liquid crystals do not emit light directly, instead using a backlight or reflector to produce images in color or monochrome.LCD1602, or 1602 character-type liquid crystal display, is a kind of dot matrix module to show letters, numbers, and characters and so on. It's composed of 5x7 or 5x11 dot matrix positions, each position can display one character.

We use LCD Display 16x2 which model is LCD1602. Number of available slots are displayed in the LCD. If there is no slot available, it shows “no slot available”.

I2C module: I2C is a synchronous, multi slave, multi master packet switched, single-ended serial bus. Multiple chips can be connect to the same bus. I2C uses only two bidirectional open collector or open drain lines, Serial Data Line (SDA) and Serial Clock Line (SCL), pulled up with resistors.The *PCF8574 module* can be used to expand the digital I/O of an MCU using the *I2C* bus. The *modules* can be daisy-chained to increase I/O up to 64.

We use I2C module which model is PCF9574.

Servo motor: Micro Servo Motor SG90 is a tiny and lightweight server motor with high output power. Servo can rotate approximately 0 to180 degrees (90 in each direction), and works just like the standard kinds but smaller.

We use it as a door and the door is controlled by programming. When the door is open its position is at 0 degree and when it close the door it position is 100 degree. If there is any slot available, when a new car wants to enter the servo motor rotate 100 degree to zero degree opens the door and after entering the car the motor automatically close. And when there is no available slot, the door will remain closed. If any car take place at the exit point the door automatically open.

Buzzer: A buzzer or beeper is an audio signaling device. Which may be mechanical, electromechanical or piezoelectric (piezo for short). Typical uses of buzzers and beepers include alarm devices, timers, train and confirmation of user input such as a mouse click or keystroke.

We use a buzzer to make an audio signal when a car take place at the enter point but there is no slots are available.

PCB: A printed circuit board (PCB) is the board base for physically supporting and wiring the surface-mounted and socketed components in most electronics.

It is used to mechanically support and electrically connect electronic components using conductive pathways, tracks or signal traces etched from copper sheets laminated onto a non-conductive substrate. We use it to connect the devices.

Required Components

1. **CCTV**
2. **Microcontroller(Arduino Uno)**
3. **RGB LED (Common Cathod)**
4. **Diffuser**
5. **LCD display(LCD1602)**
6. **I2C Module (PCF8574)**
7. **Servo Motor (SG90)**
8. **Jumper Wire**
9. **Buzzer**
10. **PCB**

Name	Model	Price (tk)
CCTV	Use mobile phone camera	
Microcontroller (Arduino Uno)		738
RGB (Common Cathod)		15
LCD Display16x2	LCD1602	310
I2C Module	PCF8574	
Servo Motor	SG90	200
Jumper Wires (male to male, male to female , female to female)		30
Buzzer		12
Diffuser		
PCB		60
Foam Board		60
String		5
Glue		20
Toy Cars		170
Tripod		150

Total=1790 Tk

Block diagram:-

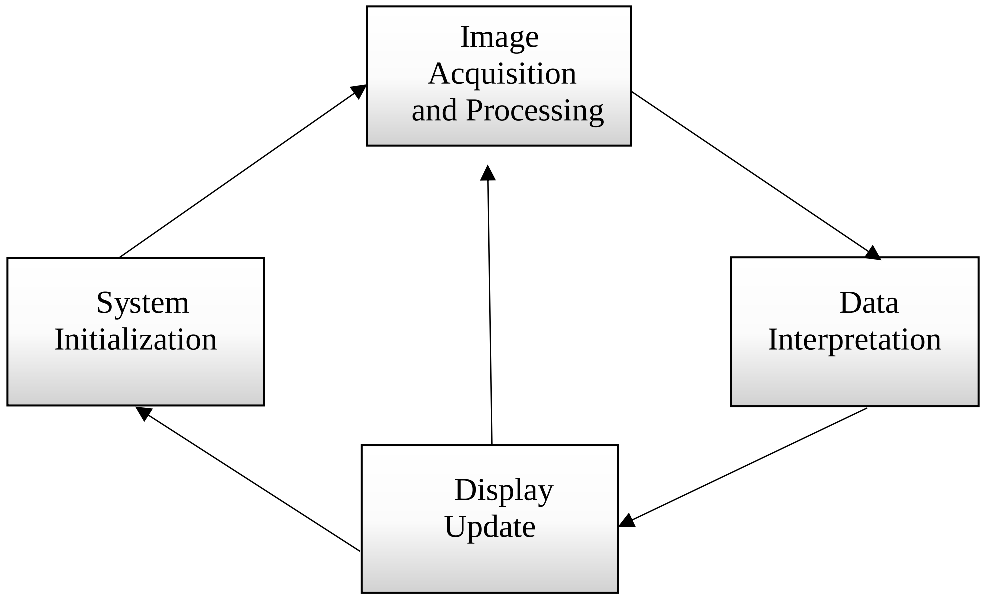


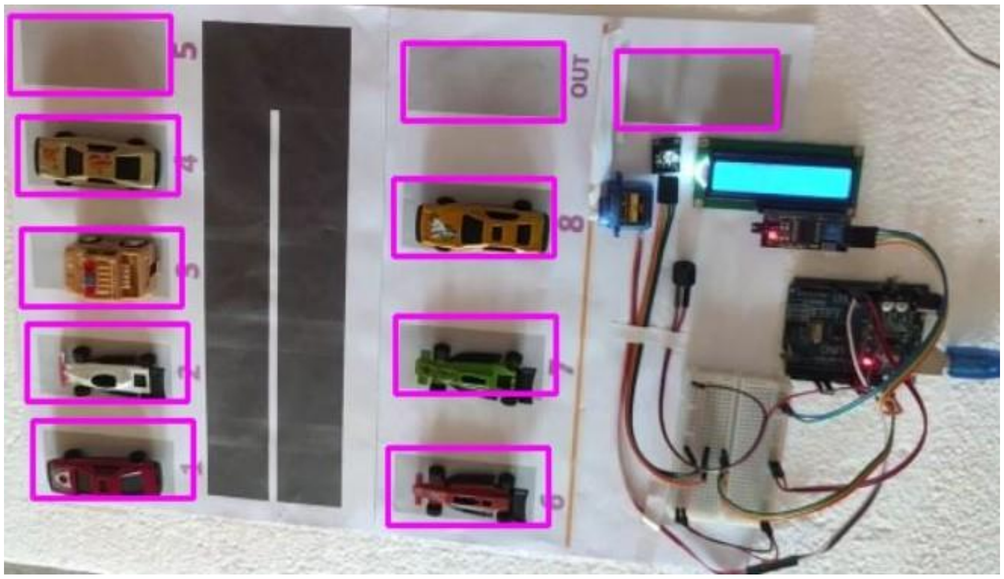
Figure1:-Block diagram of car parking system

Methodology

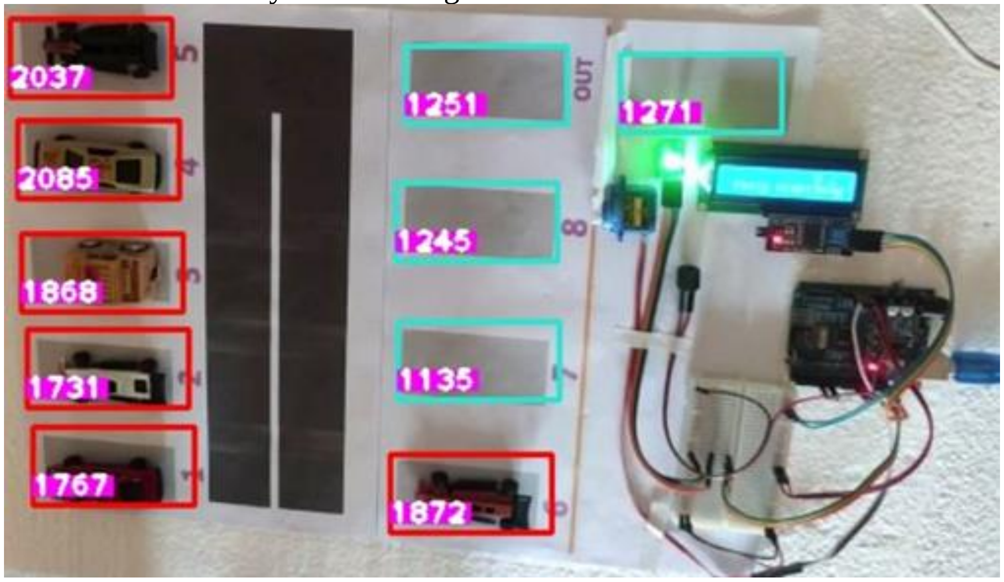
System Operation

The operation of the system involves four major sub-operations. These modules are system initialization, image acquisition and processing, data interpretation, and display update. The last three processes are then repeated as long as the system is active. The initialization of the parking guidance system takes place once, when the system is being set up for the first time or after a replacement of any of the systems module. During initialization, a refresh signal is sent from the node's controller to the image sensors in order to activate the process. Then image acquisition and processing takes place and starts working. Image processing can be discussed step by step:

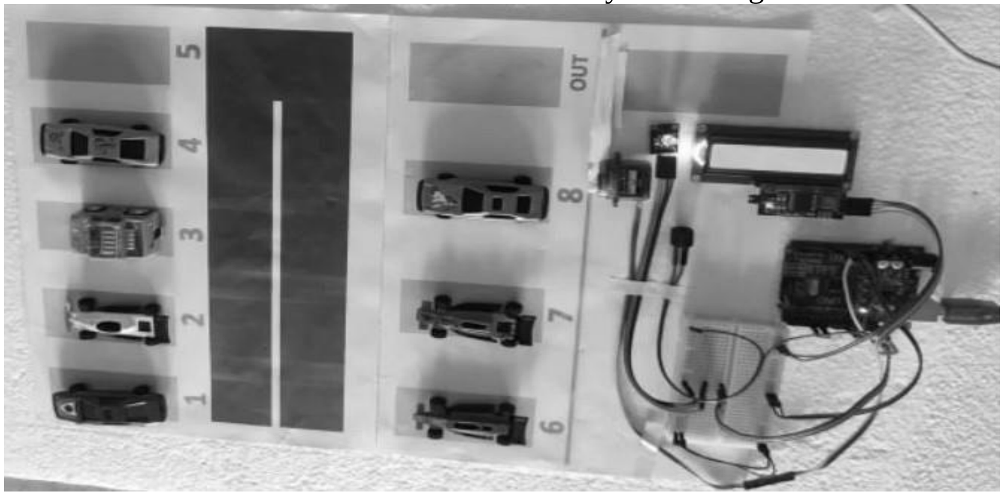
1. First of all we need to define all parking slots, entry & exist slots by detecting all of its height and weight.



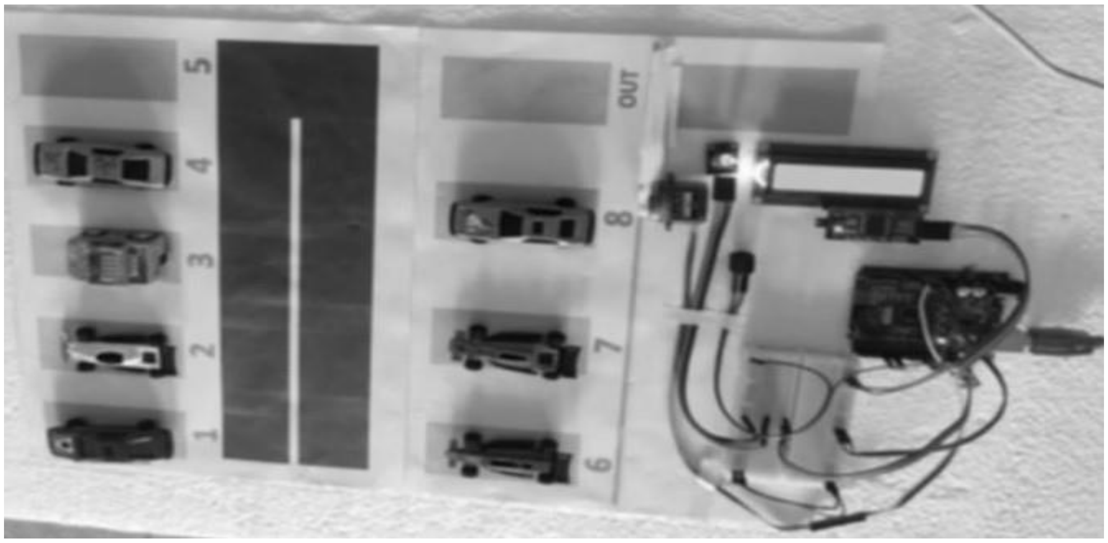
2. CCTV detects the slots pixel and then give a footage that is in BGR format. The empty slot's pixel and the pixel of the slots with cars are different. By image processing this difference is detected and the system working.



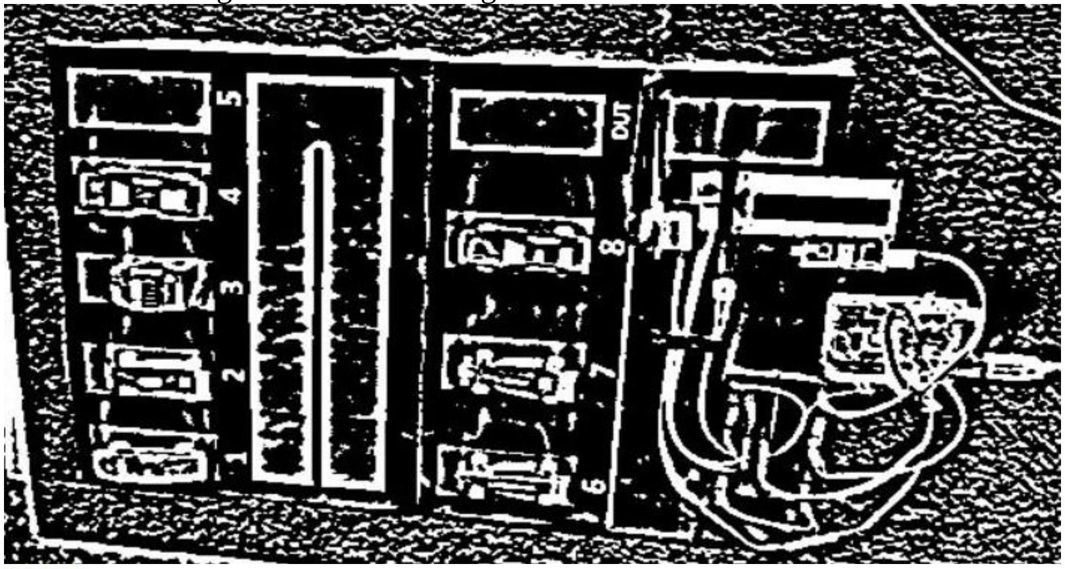
3. This BGR format is then converted into Gray scale image.

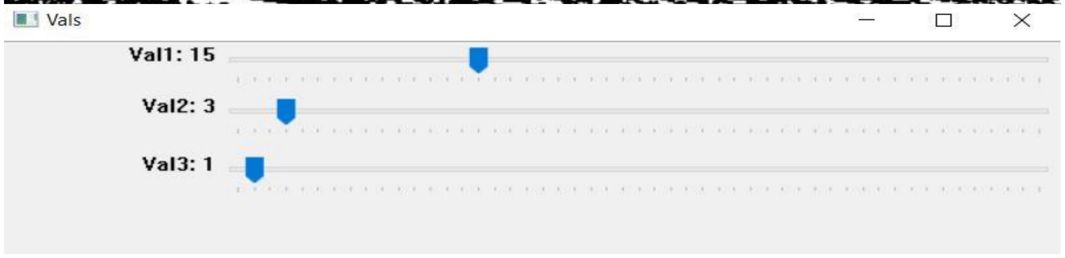


4. The Gray scale image is then converted into GaussianBlur format. This process is done with convolution theory and a 3*3 matrix is used here which all element's value is 1.



5. The image that is received from GaussianBlur is then transformed into Threshold image. Here Threshold value 15 & 3 are used by using trackbar. Adaptive Threshold Gaussian method is used for taking this Threshold image.



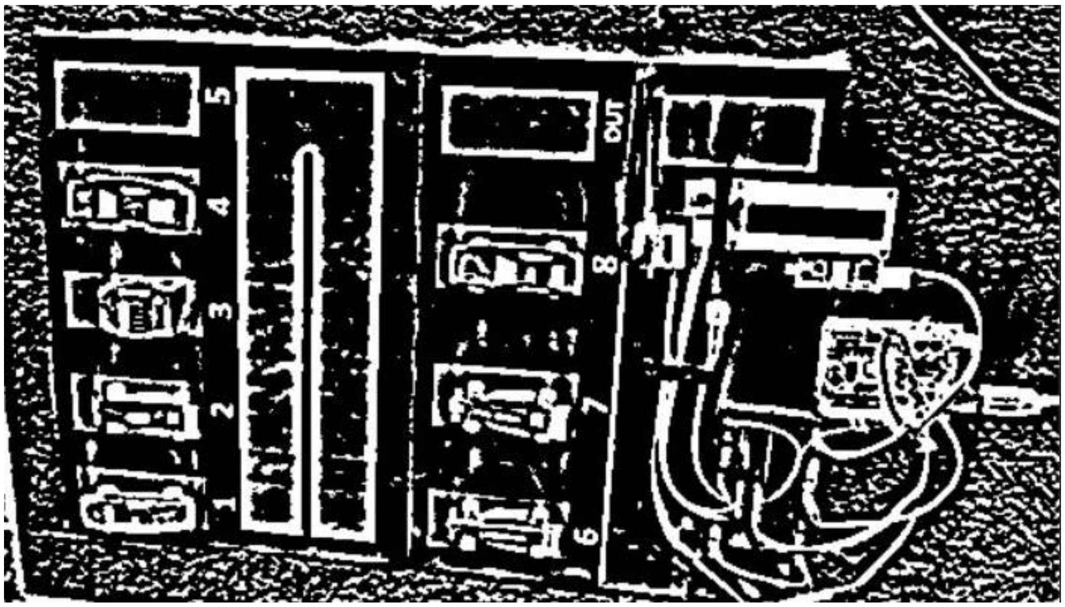


Val1: 15

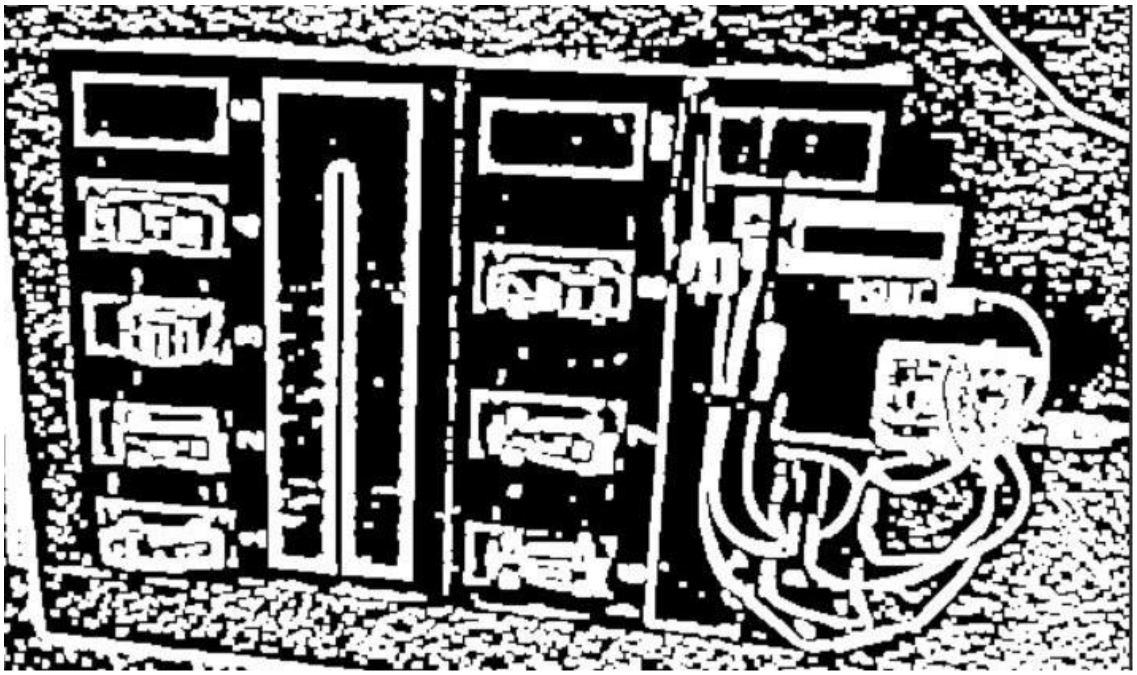
Val2: 3

Val3: 1

6. Threshold image is then converted into Median image.



7. The Median image is then dialted and for this purpose the Kernel Matrix (3*3) (with all elements are 1) is used.



The pixel of image is counted from this dialted image.

Description:

(1) Every parking place must now have a **CCTV** in the current day. This **CCTV** footage served as the basis for our project, which employed the "**Image Processing**" method. We will identify vacant parking spaces and spaces that are not empty using **CCTV** footage analysis. Depending on the outcome, it will let a car to enter a parking spot. The driver of a car may check the number of available slots via an LCD display that has been installed at the parking space's entrance before entering.

(2) When every parking space is occupied by cars and a car tries to enter, the parking gate won't open and a buzzer will ring.

(3) If there are any available slots, the gate will be opened and the LCD display will indicate the free slot numbers.

(4) If a vehicle wishes to depart, the parking gate must be opened.

(5) When there is available slots in parking space, the RGB will emit green light. If there is no slots available RGB will emit red light and if any vehicle wants to go out from parking space then RGB will emit blue light.

PCB Layout:-

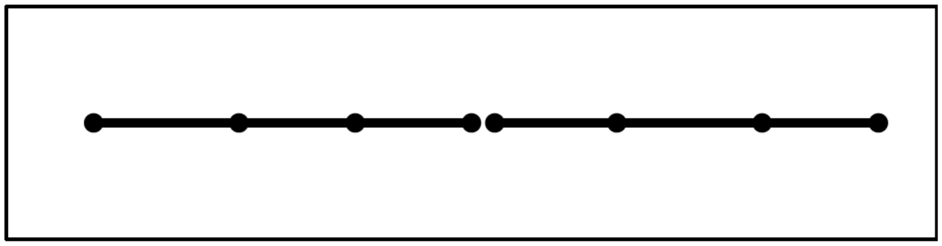


Figure 2:PCB layout for car parking system

Project picture:-

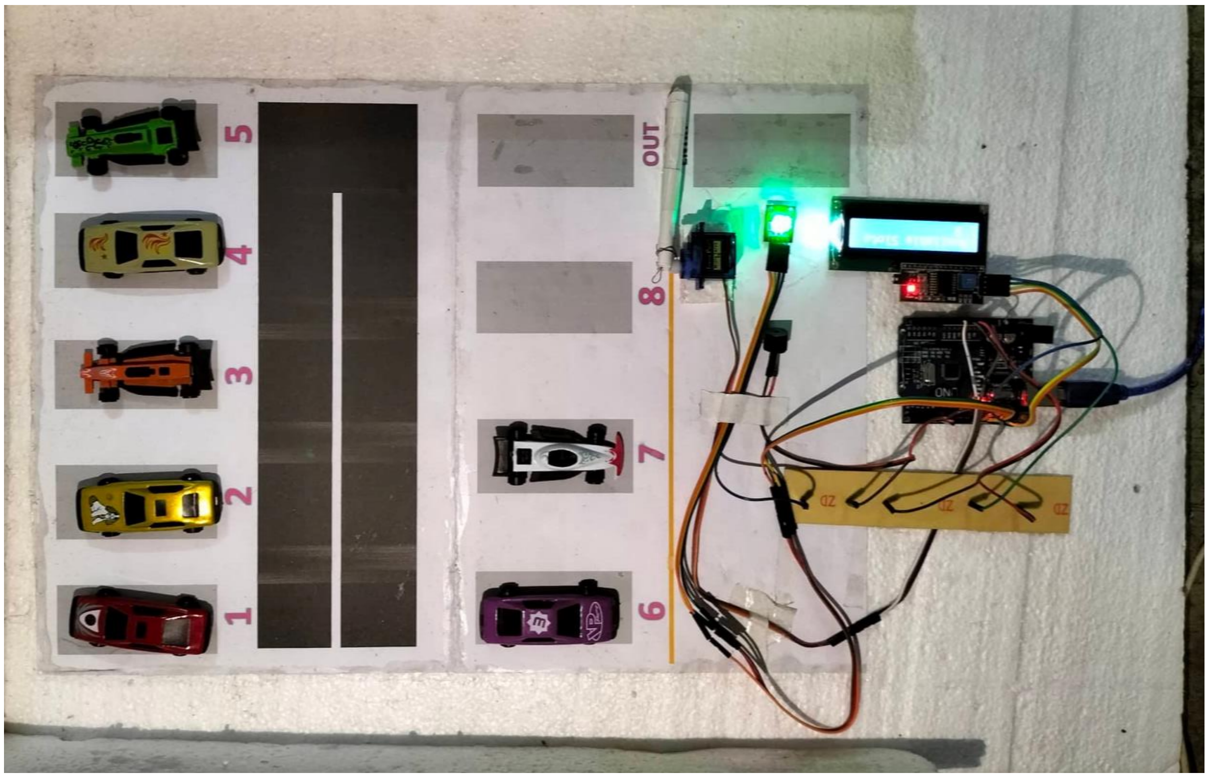


Figure 3:-Representation of an automatic car parking system

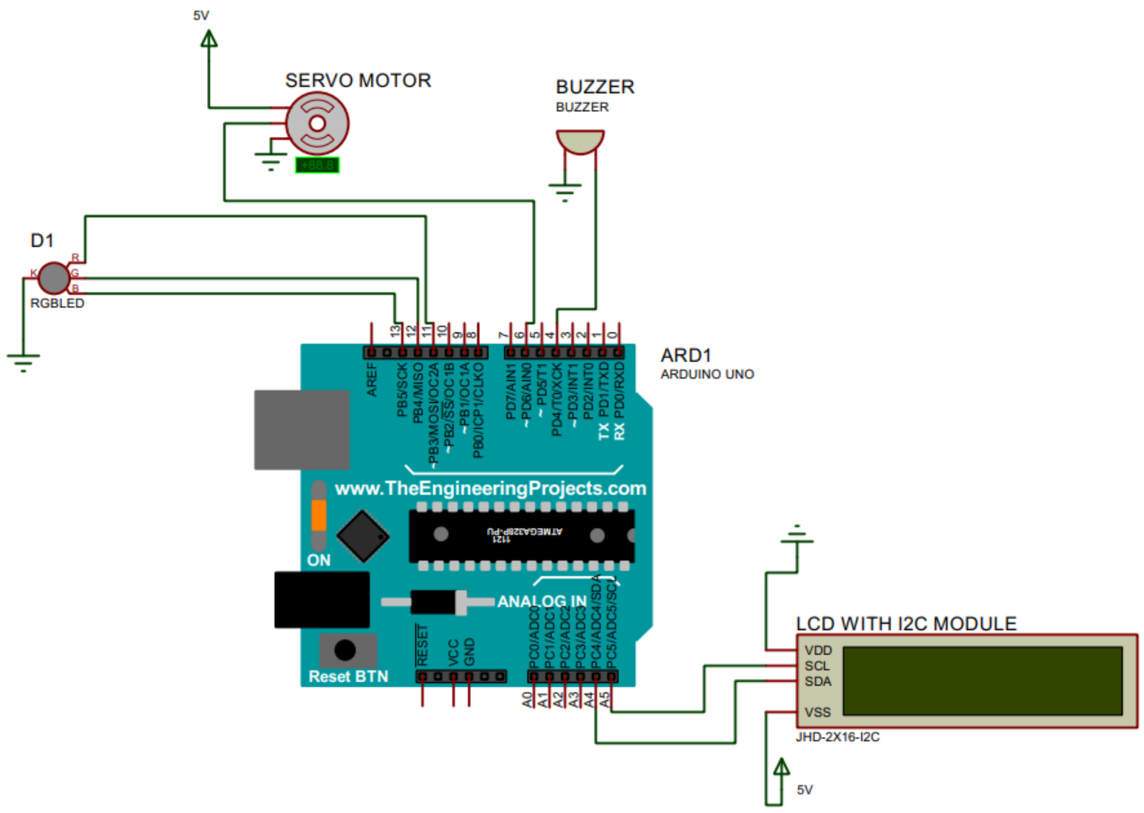


Figure 4:Connection Diagram of Car parking system

Conclusion :

The image processing system accurately detected the presence of cars in parking slots. The filled image approach performed better than the use of dilated edge images to determine vacancy of parking slots. The pixel value of a vacant slot is compared with a car-parked slot and that's how the system find out whether a slot is parked or vacant. LCD display shows the vacant slots number. If the parking slots are filled and a new car is trying to get into, then a buzzer beeps. The project met a few drawbacks. Overcoming the limitations could result it in an accomplished project. As the camera angle changes, so does the position of the parking slots. Therefore the position of slots is to be reassigned, while the project initiates. Besides CCTV footage gives more accurate pixel value than the phone camera. Adequate financial support can make it a fruitful project which will ease our problem in finding parking slots in busy areas like Residential area, office and other important places. From the outcome of project, it can easily be deduced that the proposed image processing-based car parking project is a viable option for parking space vacancy management.

Reference:-

- Kommey, Benjamin, Ernest O. Addo, and Andrew S. Agbemenu. "A smart image processing-based system for parking space vacancy management." *International Journal of Computer Applications* 182.5 (2018): 1-6.

  (https://www.researchgate.net/publication/326439400_A_Smart_Image_Processingbased_System_for_Parking_Space_Vacancy_Management)

- Kaarthik, K., A. Sridevi, and C. Vivek. "Image processing based intelligent parking system." *2017 IEEE International Conference on Electrical, Instrumentation and Communication Engineering (ICEICE)*. IEEE, 2017.
  (https://www.researchgate.net/publication/321816743_Image_processing_based_intelligent_parking_system)

ARDUINO CODE

```
#include <Servo.h>

#include <Wire.h>

#include <Firmata.h>

#include<LiquidCrystal_I2C.h>


#define I2C_WRITE                   B00000000

#define I2C_READ                    B00001000

#define I2C_READ_CONTINUOUSLY       B00010000

#define I2C_STOP_READING            B00011000

#define I2C_READ_WRITE_MODE_MASK    B00011000

#define I2C_10BIT_ADDRESS_MODE_MASK B00100000

#define I2C_END_TX_MASK             B01000000

#define I2C_STOP_TX                 1

#define I2C_RESTART_TX              0

#define I2C_MAX_QUERIES             8

#define I2C_REGISTER_NOT_SPECIFIED  -1


// the minimum interval for sampling analog input

#define MINIMUM_SAMPLING_INTERVAL   1



/*==============================================================================

  GLOBAL VARIABLES

==============================================================================*/


#ifdef FIRMATA_SERIAL_FEATURE
```

```
SerialFirmata serialFeature;
#endif

/* analog inputs */
int analogInputsToReport = 0; // bitwise array to store pin reporting

/* digital input ports */
byte reportPINs[TOTAL_PORTS];       // 1 = report this port, 0 = silence
byte previousPINs[TOTAL_PORTS];     // previous 8 bits sent

/* pins configuration */
byte portConfigInputs[TOTAL_PORTS]; // each bit: 1 = pin in INPUT, 0 = anything else

/* timer variables */
unsigned long currentMillis;        // store the current value from millis()
unsigned long previousMillis;       // for comparison with currentMillis
unsigned int samplingInterval = 19; // how often to run the main loop (in ms)

/* i2c data */
struct i2c_device_info {
  byte addr;
  int reg;
  byte bytes;
  byte stopTX;
};

/* for i2c read continuous more */
i2c_device_info query[I2C_MAX_QUERIES];
```

```
byte i2cRxData[64];
boolean isI2CEnabled = false;
signed char queryIndex = -1;
// default delay time between i2c read request and Wire.requestFrom()
unsigned int i2cReadDelayTime = 0;

Servo servos[MAX_SERVOS];
byte servoPinMap[TOTAL_PINS];
byte detachedServos[MAX_SERVOS];
byte detachedServoCount = 0;
byte servoCount = 0;

boolean isResetting = false;

// Forward declare a few functions to avoid compiler errors with older versions
// of the Arduino IDE.
void setPinModeCallback(byte, int);
void reportAnalogCallback(byte analogPin, int value);
void sysexCallback(byte, byte, byte*);

//For lcd
LiquidCrystal_I2C lcd(0x27, 16, 2);
int lastLine = 1;


/* utility functions */
void wireWrite(byte data)
```

```
{
#if ARDUINO >= 100
  Wire.write((byte)data);
#else
  Wire.send(data);
#endif
}

byte wireRead(void)
{
#if ARDUINO >= 100
  return Wire.read();
#else
  return Wire.receive();
#endif
}

/*==============================================================================
   FUNCTIONS

==============================================================================*/

void attachServo(byte pin, int minPulse, int maxPulse)
{
  if (servoCount < MAX_SERVOS) {
    // reuse indexes of detached servos until all have been reallocated
    if (detachedServoCount > 0) {
      servoPinMap[pin] = detachedServos[detachedServoCount - 1];
      if (detachedServoCount > 0) detachedServoCount--;
    } else {
      servoPinMap[pin] = servoCount;
      servoCount++;
    }
    if (minPulse > 0 && maxPulse > 0) {
      servos[servoPinMap[pin]].attach(PIN_TO_DIGITAL(pin), minPulse, maxPulse);
    } else {
      servos[servoPinMap[pin]].attach(PIN_TO_DIGITAL(pin));
    }
  } else {
    Firmata.sendString("Max servos attached");
  }
}

void detachServo(byte pin)
{
  servos[servoPinMap[pin]].detach();
  // if we're detaching the last servo, decrement the count
  // otherwise store the index of the detached servo
  if (servoPinMap[pin] == servoCount && servoCount > 0) {
    servoCount--;
  } else if (servoCount > 0) {
    // keep track of detached servos because we want to reuse their indexes
    // before incrementing the count of attached servos
    detachedServoCount++;
    detachedServos[detachedServoCount - 1] = servoPinMap[pin];
  }

  servoPinMap[pin] = 255;
}

void enableI2CPins()
{
  byte i;
  // is there a faster way to do this? would probaby require importing
  // Arduino.h to get SCL and SDA pins
  for (i = 0; i < TOTAL_PINS; i++) {
    if (IS_PIN_I2C(i)) {
      // mark pins as i2c so they are ignore in non i2c data requests
      setPinModeCallback(i, PIN_MODE_I2C);
    }
  }

  isI2CEnabled = true;

  Wire.begin();
}

/* disable the i2c pins so they can be used for other functions */
void disableI2CPins() {
  isI2CEnabled = false;
  // disable read continuous mode for all devices
  queryIndex = -1;
}
```

```
void readAndReportData(byte address, int theRegister, byte numBytes, byte stopTX) {
  // allow I2C requests that don't require a register read
  // for example, some devices using an interrupt pin to signify new data available
  // do not always require the register read so upon interrupt you call Wire.requestFrom()
  if (theRegister != I2C_REGISTER_NOT_SPECIFIED) {
    Wire.beginTransmission(address);
    wireWrite((byte)theRegister);
    Wire.endTransmission(stopTX); // default = true
    // do not set a value of 0
    if (i2cReadDelayTime > 0) {
      // delay is necessary for some devices such as WiiNunchuck
      delayMicroseconds(i2cReadDelayTime);
    }
  } else {
    theRegister = 0;  // fill the register with a dummy value
  }

  Wire.requestFrom(address, numBytes);  // all bytes are returned in requestFrom

  // check to be sure correct number of bytes were returned by slave
  if (numBytes < Wire.available()) {
    Firmata.sendString("I2C: Too many bytes received");
  } else if (numBytes > Wire.available()) {
    Firmata.sendString("I2C: Too few bytes received");
  }

  i2cRxData[0] = address;
  i2cRxData[1] = theRegister;

  for (int i = 0; i < numBytes && Wire.available(); i++) {
    i2cRxData[2 + i] = wireRead();
  }

  // send slave address, register and received bytes
  Firmata.sendSysex(SYSEX_I2C_REPLY, numBytes + 2, i2cRxData);
}

void outputPort(byte portNumber, byte portValue, byte forceSend)
{
  // pins not configured as INPUT are cleared to zeros
  portValue = portValue & portConfigInputs[portNumber];
  // only send if the value is different than previously sent
  if (forceSend || previousPINs[portNumber] != portValue) {
    Firmata.sendDigitalPort(portNumber, portValue);
    previousPINs[portNumber] = portValue;
  }
}

/* ----------------------------------------------------------------------------
   check all the active digital inputs for change of state, then add any events
   to the Serial output queue using Serial.print() */
void checkDigitalInputs(void)
{
  /* Using non-looping code allows constants to be given to readPort().
     The compiler will apply substantial optimizations if the inputs
    to readPort() are compile-time constants. */
  if (TOTAL_PORTS > 0 && reportPINs[0]) outputPort(0, readPort(0, portConfigInputs[0]),
false);
  if (TOTAL_PORTS > 1 && reportPINs[1]) outputPort(1, readPort(1, portConfigInputs[1]),
false);
  if (TOTAL_PORTS > 2 && reportPINs[2]) outputPort(2, readPort(2, portConfigInputs[2]),
false);
  if (TOTAL_PORTS > 3 && reportPINs[3]) outputPort(3, readPort(3, portConfigInputs[3]),
false);
  if (TOTAL_PORTS > 4 && reportPINs[4]) outputPort(4, readPort(4, portConfigInputs[4]),
false);
  if (TOTAL_PORTS > 5 && reportPINs[5]) outputPort(5, readPort(5, portConfigInputs[5]),
false);
  if (TOTAL_PORTS > 6 && reportPINs[6]) outputPort(6, readPort(6, portConfigInputs[6]),
false);
  if (TOTAL_PORTS > 7 && reportPINs[7]) outputPort(7, readPort(7, portConfigInputs[7]),
false);
  if (TOTAL_PORTS > 8 && reportPINs[8]) outputPort(8, readPort(8, portConfigInputs[8]),
false);
  if (TOTAL_PORTS > 9 && reportPINs[9]) outputPort(9, readPort(9, portConfigInputs[9]),
false);
  if (TOTAL_PORTS > 10 && reportPINs[10]) outputPort(10, readPort(10,
portConfigInputs[10]), false);
  if (TOTAL_PORTS > 11 && reportPINs[11]) outputPort(11, readPort(11,
portConfigInputs[11]), false);
  if (TOTAL_PORTS > 12 && reportPINs[12]) outputPort(12, readPort(12,
portConfigInputs[12]), false);
  if (TOTAL_PORTS > 13 && reportPINs[13]) outputPort(13, readPort(13,
portConfigInputs[13]), false);
  if (TOTAL_PORTS > 14 && reportPINs[14]) outputPort(14, readPort(14,
portConfigInputs[14]), false);
  if (TOTAL_PORTS > 15 && reportPINs[15]) outputPort(15, readPort(15,
portConfigInputs[15]), false);
}
```

```
// -----------------------------------------------------------------------------
/* sets the pin mode to the correct state and sets the relevant bits in the
   two bit-arrays that track Digital I/O and PWM status
*/
void setPinModeCallback(byte pin, int mode)
{
  if (Firmata.getPinMode(pin) == PIN_MODE_IGNORE)
    return;

  if (Firmata.getPinMode(pin) == PIN_MODE_I2C && isI2CEnabled && mode !=
PIN_MODE_I2C) {
    // disable i2c so pins can be used for other functions
    // the following if statements should reconfigure the pins properly
    disableI2CPins();
  }
  if (IS_PIN_DIGITAL(pin) && mode != PIN_MODE_SERVO) {
    if (servoPinMap[pin] < MAX_SERVOS && servos[servoPinMap[pin]].attached()) {
      detachServo(pin);
    }
  }
  if (IS_PIN_ANALOG(pin)) {
    reportAnalogCallback(PIN_TO_ANALOG(pin), mode == PIN_MODE_ANALOG ? 1 : 0); //
turn on/off reporting
  }
  if (IS_PIN_DIGITAL(pin)) {
    if (mode == INPUT || mode == PIN_MODE_PULLUP) {
      portConfigInputs[pin / 8] |= (1 << (pin & 7));
    } else {
      portConfigInputs[pin / 8] &= ~(1 << (pin & 7));
    }
  }
  Firmata.setPinState(pin, 0);
  switch (mode) {
    case PIN_MODE_ANALOG:
      if (IS_PIN_ANALOG(pin)) {
        if (IS_PIN_DIGITAL(pin)) {
          pinMode(PIN_TO_DIGITAL(pin), INPUT);    // disable output driver
#if ARDUINO <= 100
          // deprecated since Arduino 1.0.1 - TODO: drop support in Firmata 2.6
          digitalWrite(PIN_TO_DIGITAL(pin), LOW); // disable internal pull-ups
#endif
        }
        Firmata.setPinMode(pin, PIN_MODE_ANALOG);
      }
      break;
    case INPUT:
      if (IS_PIN_DIGITAL(pin)) {
        pinMode(PIN_TO_DIGITAL(pin), INPUT);    // disable output driver
#if ARDUINO <= 100
        // deprecated since Arduino 1.0.1 - TODO: drop support in Firmata 2.6
        digitalWrite(PIN_TO_DIGITAL(pin), LOW); // disable internal pull-ups
#endif
        Firmata.setPinMode(pin, INPUT);
      }
      break;
    case PIN_MODE_PULLUP:
```

```
  if (IS_PIN_DIGITAL(pin)) {
    pinMode(PIN_TO_DIGITAL(pin), INPUT_PULLUP);
    Firmata.setPinMode(pin, PIN_MODE_PULLUP);
    Firmata.setPinState(pin, 1);
  }
  break;
case OUTPUT:
  if (IS_PIN_DIGITAL(pin)) {
    if (Firmata.getPinMode(pin) == PIN_MODE_PWM) {
      // Disable PWM if pin mode was previously set to PWM.
      digitalWrite(PIN_TO_DIGITAL(pin), LOW);
    }
    pinMode(PIN_TO_DIGITAL(pin), OUTPUT);
    Firmata.setPinMode(pin, OUTPUT);
  }
  break;
case PIN_MODE_PWM:
  if (IS_PIN_PWM(pin)) {
    pinMode(PIN_TO_PWM(pin), OUTPUT);
    analogWrite(PIN_TO_PWM(pin), 0);
    Firmata.setPinMode(pin, PIN_MODE_PWM);
  }
  break;
case PIN_MODE_SERVO:
  if (IS_PIN_DIGITAL(pin)) {
    Firmata.setPinMode(pin, PIN_MODE_SERVO);
    if (servoPinMap[pin] == 255 || !servos[servoPinMap[pin]].attached()) {
      // pass -1 for min and max pulse values to use default values set
```

```
          // by Servo library
          attachServo(pin, -1, -1);
        }
      }
      break;
    case PIN_MODE_I2C:
      if (IS_PIN_I2C(pin)) {
        // mark the pin as i2c
        // the user must call I2C_CONFIG to enable I2C for a device
        Firmata.setPinMode(pin, PIN_MODE_I2C);
      }
      break;
    case PIN_MODE_SERIAL:
#ifdef FIRMATA_SERIAL_FEATURE
      serialFeature.handlePinMode(pin, PIN_MODE_SERIAL);
#endif
      break;
    default:
      Firmata.sendString("Unknown pin mode"); // TODO: put error msgs in EEPROM
  }
  // TODO: save status to EEPROM here, if changed
}

/*
   Sets the value of an individual pin. Useful if you want to set a pin value but
   are not tracking the digital port state.
   Can only be used on pins configured as OUTPUT.
   Cannot be used to enable pull-ups on Digital INPUT pins.
```

```
*/
void setPinValueCallback(byte pin, int value)
{
  if (pin < TOTAL_PINS && IS_PIN_DIGITAL(pin)) {
    if (Firmata.getPinMode(pin) == OUTPUT) {
      Firmata.setPinState(pin, value);
      digitalWrite(PIN_TO_DIGITAL(pin), value);
    }
  }
}

void analogWriteCallback(byte pin, int value)
{
  if (pin < TOTAL_PINS) {
    switch (Firmata.getPinMode(pin)) {
      case PIN_MODE_SERVO:
        if (IS_PIN_DIGITAL(pin))
          servos[servoPinMap[pin]].write(value);
        Firmata.setPinState(pin, value);
        break;
      case PIN_MODE_PWM:
        if (IS_PIN_PWM(pin))
          analogWrite(PIN_TO_PWM(pin), value);
        Firmata.setPinState(pin, value);
        break;
    }
  }
}
```

```
void digitalWriteCallback(byte port, int value)
{
  byte pin, lastPin, pinValue, mask = 1, pinWriteMask = 0;

  if (port < TOTAL_PORTS) {
    // create a mask of the pins on this port that are writable.
    lastPin = port * 8 + 8;
    if (lastPin > TOTAL_PINS) lastPin = TOTAL_PINS;
    for (pin = port * 8; pin < lastPin; pin++) {
      // do not disturb non-digital pins (eg, Rx & Tx)
      if (IS_PIN_DIGITAL(pin)) {
        // do not touch pins in PWM, ANALOG, SERVO or other modes
        if (Firmata.getPinMode(pin) == OUTPUT || Firmata.getPinMode(pin) == INPUT) {
          pinValue = ((byte)value & mask) ? 1 : 0;
          if (Firmata.getPinMode(pin) == OUTPUT) {
            pinWriteMask |= mask;
          } else if (Firmata.getPinMode(pin) == INPUT && pinValue == 1 &&
Firmata.getPinState(pin) != 1) {
            // only handle INPUT here for backwards compatibility
#if ARDUINO > 100
            pinMode(pin, INPUT_PULLUP);
#else
            // only write to the INPUT pin to enable pullups if Arduino v1.0.0 or earlier
            pinWriteMask |= mask;
#endif
          }
          Firmata.setPinState(pin, pinValue);
        }
      }
      mask = mask << 1;
    }
    writePort(port, (byte)value, pinWriteMask);
  }
}



// -----------------------------------------------------------------------------
/* sets bits in a bit array (int) to toggle the reporting of the analogIns
*/
//void FirmataClass::setAnalogPinReporting(byte pin, byte state) {
//}
void reportAnalogCallback(byte analogPin, int value)
{
  if (analogPin < TOTAL_ANALOG_PINS) {
    if (value == 0) {
      analogInputsToReport = analogInputsToReport & ~ (1 << analogPin);
    } else {
      analogInputsToReport = analogInputsToReport | (1 << analogPin);
      // prevent during system reset or all analog pin values will be reported
      // which may report noise for unconnected analog pins
      if (!isResetting) {
        // Send pin value immediately. This is helpful when connected via
        // ethernet, wi-fi or bluetooth so pin states can be known upon
        // reconnecting.
        Firmata.sendAnalog(analogPin, analogRead(analogPin));
      }
    }
  }
  // TODO: save status to EEPROM here, if changed
}

void reportDigitalCallback(byte port, int value)
{
  if (port < TOTAL_PORTS) {
    reportPINs[port] = (byte)value;
    // Send port value immediately. This is helpful when connected via
    // ethernet, wi-fi or bluetooth so pin states can be known upon
    // reconnecting.
    if (value) outputPort(port, readPort(port, portConfigInputs[port]), true);
  }
  // do not disable analog reporting on these 8 pins, to allow some
  // pins used for digital, others analog.  Instead, allow both types
  // of reporting to be enabled, but check if the pin is configured
  // as analog when sampling the analog inputs.  Likewise, while
  // scanning digital pins, portConfigInputs will mask off values from any
  // pins configured as analog
}

/*==============================================================================
   SYSEX-BASED commands

==============================================================================*/
```

```
void sysexCallback(byte command, byte argc, byte *argv)
{
  byte mode;
  byte stopTX;
  byte slaveAddress;
  byte data;
  int slaveRegister;
  unsigned int delayTime;

  switch (command) {
    case I2C_REQUEST:
      mode = argv[1] & I2C_READ_WRITE_MODE_MASK;
      if (argv[1] & I2C_10BIT_ADDRESS_MODE_MASK) {
        Firmata.sendString("10-bit addressing not supported");
        return;
      }
      else {
        slaveAddress = argv[0];
      }

      // need to invert the logic here since 0 will be default for client
      // libraries that have not updated to add support for restart tx
      if (argv[1] & I2C_END_TX_MASK) {
        stopTX = I2C_RESTART_TX;
      }
      else {
        stopTX = I2C_STOP_TX; // default
      }
```

```
switch (mode) {
  case I2C_WRITE:
    Wire.beginTransmission(slaveAddress);
    for (byte i = 2; i < argc; i += 2) {
      data = argv[i] + (argv[i + 1] << 7);
      wireWrite(data);
    }
    Wire.endTransmission();
    delayMicroseconds(70);
    break;
  case I2C_READ:
    if (argc == 6) {
      // a slave register is specified
      slaveRegister = argv[2] + (argv[3] << 7);
      data = argv[4] + (argv[5] << 7);  // bytes to read
    }
    else {
      // a slave register is NOT specified
      slaveRegister = I2C_REGISTER_NOT_SPECIFIED;
      data = argv[2] + (argv[3] << 7);  // bytes to read
    }
    readAndReportData(slaveAddress, (int)slaveRegister, data, stopTX);
    break;
  case I2C_READ_CONTINUOUSLY:
    if ((queryIndex + 1) >= I2C_MAX_QUERIES) {
      // too many queries, just ignore
      Firmata.sendString("too many queries");
    break;
  }
  if (argc == 6) {
    // a slave register is specified
    slaveRegister = argv[2] + (argv[3] << 7);
    data = argv[4] + (argv[5] << 7);  // bytes to read
  }
  else {
    // a slave register is NOT specified
    slaveRegister = (int)I2C_REGISTER_NOT_SPECIFIED;
    data = argv[2] + (argv[3] << 7);  // bytes to read
  }
  queryIndex++;
  query[queryIndex].addr = slaveAddress;
  query[queryIndex].reg = slaveRegister;
  query[queryIndex].bytes = data;
  query[queryIndex].stopTX = stopTX;
  break;
case I2C_STOP_READING:
  byte queryIndexToSkip;
  // if read continuous mode is enabled for only 1 i2c device, disable
  // read continuous reporting for that device
  if (queryIndex <= 0) {
    queryIndex = -1;
  } else {
    queryIndexToSkip = 0;
    // if read continuous mode is enabled for multiple devices,
    // determine which device to stop reading and remove it's data from
```

```
        // the array, shifiting other array data to fill the space
        for (byte i = 0; i < queryIndex + 1; i++) {
          if (query[i].addr == slaveAddress) {
            queryIndexToSkip = i;
            break;
          }
        }

        for (byte i = queryIndexToSkip; i < queryIndex + 1; i++) {
          if (i < I2C_MAX_QUERIES) {
            query[i].addr = query[i + 1].addr;
            query[i].reg = query[i + 1].reg;
            query[i].bytes = query[i + 1].bytes;
            query[i].stopTX = query[i + 1].stopTX;
          }
        }
        queryIndex--;
      }
      break;
    default:
      break;
  }
  break;
case I2C_CONFIG:
  delayTime = (argv[0] + (argv[1] << 7));

  if (argc > 1 && delayTime > 0) {
    i2cReadDelayTime = delayTime;
  }

  if (!isI2CEnabled) {
    enableI2CPins();
  }

  break;
case SERVO_CONFIG:
  if (argc > 4) {
    // these vars are here for clarity, they'll optimized away by the compiler
    byte pin = argv[0];
    int minPulse = argv[1] + (argv[2] << 7);
    int maxPulse = argv[3] + (argv[4] << 7);

    if (IS_PIN_DIGITAL(pin)) {
      if (servoPinMap[pin] < MAX_SERVOS && servos[servoPinMap[pin]].attached()) {
        detachServo(pin);
      }
      attachServo(pin, minPulse, maxPulse);
      setPinModeCallback(pin, PIN_MODE_SERVO);
    }
  }
  break;
case SAMPLING_INTERVAL:
  if (argc > 1) {
    samplingInterval = argv[0] + (argv[1] << 7);
    if (samplingInterval < MINIMUM_SAMPLING_INTERVAL) {
      samplingInterval = MINIMUM_SAMPLING_INTERVAL;
    }
  } else {
    //Firmata.sendString("Not enough data");
  }
  break;
case EXTENDED_ANALOG:
  if (argc > 1) {
    int val = argv[1];
    if (argc > 2) val |= (argv[2] << 7);
    if (argc > 3) val |= (argv[3] << 14);
    analogWriteCallback(argv[0], val);
  }
  break;
case CAPABILITY_QUERY:
  Firmata.write(START_SYSEX);
  Firmata.write(CAPABILITY_RESPONSE);
  for (byte pin = 0; pin < TOTAL_PINS; pin++) {
    if (IS_PIN_DIGITAL(pin)) {
      Firmata.write((byte)INPUT);
      Firmata.write(1);
      Firmata.write((byte)PIN_MODE_PULLUP);
      Firmata.write(1);
      Firmata.write((byte)OUTPUT);
      Firmata.write(1);
    }
    if (IS_PIN_ANALOG(pin)) {
      Firmata.write(PIN_MODE_ANALOG);
      Firmata.write(10); // 10 = 10-bit resolution
```

```
        }
        if (IS_PIN_PWM(pin)) {
          Firmata.write(PIN_MODE_PWM);
          Firmata.write(DEFAULT_PWM_RESOLUTION);
        }
        if (IS_PIN_DIGITAL(pin)) {
          Firmata.write(PIN_MODE_SERVO);
          Firmata.write(14);
        }
        if (IS_PIN_I2C(pin)) {
          Firmata.write(PIN_MODE_I2C);
          Firmata.write(1);  // TODO: could assign a number to map to SCL or SDA
        }
#ifdef FIRMATA_SERIAL_FEATURE
        serialFeature.handleCapability(pin);
#endif
        Firmata.write(127);
      }
      Firmata.write(END_SYSEX);
      break;
    case PIN_STATE_QUERY:
      if (argc > 0) {
        byte pin = argv[0];
        Firmata.write(START_SYSEX);
        Firmata.write(PIN_STATE_RESPONSE);
        Firmata.write(pin);
        if (pin < TOTAL_PINS) {
          Firmata.write(Firmata.getPinMode(pin));
          Firmata.write((byte)Firmata.getPinState(pin) & 0x7F);
          if (Firmata.getPinState(pin) & 0xFF80) Firmata.write((byte)(Firmata.getPinState(pin) >>
7) & 0x7F);
          if (Firmata.getPinState(pin) & 0xC000) Firmata.write((byte)(Firmata.getPinState(pin) >>
14) & 0x7F);
        }
        Firmata.write(END_SYSEX);
      }
      break;
    case ANALOG_MAPPING_QUERY:
      Firmata.write(START_SYSEX);
      Firmata.write(ANALOG_MAPPING_RESPONSE);
      for (byte pin = 0; pin < TOTAL_PINS; pin++) {
        Firmata.write(IS_PIN_ANALOG(pin) ? PIN_TO_ANALOG(pin) : 127);
      }
      Firmata.write(END_SYSEX);
      break;

    case SERIAL_MESSAGE:
#ifdef FIRMATA_SERIAL_FEATURE
      serialFeature.handleSysex(command, argc, argv);
#endif
      break;
  }
}

/*==============================================================
===========
   SETUP()
```

```
==============================================================================
=======*/

void systemResetCallback()
{
  isResetting = true;

  // initialize a defalt state
  // TODO: option to load config from EEPROM instead of default

#ifdef FIRMATA_SERIAL_FEATURE
  serialFeature.reset();
#endif

  if (isI2CEnabled) {
    disableI2CPins();
  }

  for (byte i = 0; i < TOTAL_PORTS; i++) {
    reportPINs[i] = false;    // by default, reporting off
    portConfigInputs[i] = 0;  // until activated
    previousPINs[i] = 0;
  }

  for (byte i = 0; i < TOTAL_PINS; i++) {
    // pins with analog capability default to analog input
    // otherwise, pins default to digital output
    if (IS_PIN_ANALOG(i)) {
      // turns off pullup, configures everything
      setPinModeCallback(i, PIN_MODE_ANALOG);
    } else if (IS_PIN_DIGITAL(i)) {
      // sets the output to 0, configures portConfigInputs
      setPinModeCallback(i, OUTPUT);
    }

    servoPinMap[i] = 255;
  }
  // by default, do not report any analog inputs
  analogInputsToReport = 0;

  detachedServoCount = 0;
  servoCount = 0;

  /* send digital inputs to set the initial state on the host computer,
     since once in the loop(), this firmware will only send on change */
  /*
    TODO: this can never execute, since no pins default to digital input
        but it will be needed when/if we support EEPROM stored config
    for (byte i=0; i < TOTAL_PORTS; i++) {
    outputPort(i, readPort(i, portConfigInputs[i]), true);
    }
  */
  isResetting = false;
}
//void function for lcd
void stringDataCallback(char *stringData) {
  if ( lastLine ) {
    lastLine = 0;
    lcd.clear();
  } else {
    lastLine = 1;
    lcd.setCursor(0, 1);
  }
  lcd.print(stringData);
}

void setup()
{
  // for lcd
  lcd.init();
  lcd.backlight();
  Firmata.setFirmwareVersion( FIRMATA_MAJOR_VERSION,
FIRMATA_MINOR_VERSION );
  Firmata.attach( STRING_DATA, stringDataCallback);

  Firmata.setFirmwareVersion(FIRMATA_FIRMWARE_MAJOR_VERSION,
FIRMATA_FIRMWARE_MINOR_VERSION);

  Firmata.attach(ANALOG_MESSAGE, analogWriteCallback);
  Firmata.attach(DIGITAL_MESSAGE, digitalWriteCallback);
  Firmata.attach(REPORT_ANALOG, reportAnalogCallback);
  Firmata.attach(REPORT_DIGITAL, reportDigitalCallback);
  Firmata.attach(SET_PIN_MODE, setPinModeCallback);
  Firmata.attach(SET_DIGITAL_PIN_VALUE, setPinValueCallback);
  Firmata.attach(START_SYSEX, sysexCallback);
  Firmata.attach(SYSTEM_RESET, systemResetCallback);

  // to use a port other than Serial, such as Serial1 on an Arduino Leonardo or Mega,
  // Call begin(baud) on the alternate serial port and pass it to Firmata to begin like this:
  // Serial1.begin(57600);
  // Firmata.begin(Serial1);
  // However do not do this if you are using SERIAL_MESSAGE

  Firmata.begin(57600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for ATmega32u4-based boards and Arduino 101
  }

  systemResetCallback();  // reset to default config
}

/*==============================================================================
 * LOOP()
 *============================================================================*/
void loop()
{
  byte pin, analogPin;

  /* DIGITALREAD - as fast as possible, check for changes and output them to the
   * FTDI buffer using Serial.print()  */
  checkDigitalInputs();
```

```
/* STREAMREAD - processing incoming messagse as soon as possible, while still
   checking digital inputs.  */
while (Firmata.available()) {
  Firmata.processInput();
}
// TODO - ensure that Stream buffer doesn't go over 60 bytes

currentMillis = millis();
if (currentMillis - previousMillis > samplingInterval) {
  previousMillis += samplingInterval;
  /* ANALOGREAD - do all analogReads() at the configured sampling interval */
  for (pin = 0; pin < TOTAL_PINS; pin++) {
    if (IS_PIN_ANALOG(pin) && Firmata.getPinMode(pin) == PIN_MODE_ANALOG) {
      analogPin = PIN_TO_ANALOG(pin);
      if (analogInputsToReport & (1 << analogPin)) {
        Firmata.sendAnalog(analogPin, analogRead(analogPin));
      }
    }
  }
  // report i2c data for all device with read continuous mode enabled
  if (queryIndex > -1) {
    for (byte i = 0; i < queryIndex + 1; i++) {
      readAndReportData(query[i].addr, query[i].reg, query[i].bytes, query[i].stopTX);
    }
  }
}
```

```
#ifdef FIRMATA_SERIAL_FEATURE
  serialFeature.update();
#endif
}
```

REFERENCE

Image processing based intelligent parking system

Conference Paper · April 2017

DOI: 10.1109/ICEICE.2017.8191876

Annathurai Sridevi
M.Kumarasamy College of Engineering
14 PUBLICATIONS **79** CITATIONS

Image Processing Based Intelligent Parking System

Kaarthik.K[1], Sridevi.A[2], Vivek.C[3]

Department of Electronics and Communication Engineering, M.Kumarasamy College of Engineering(Autonomous), Thalavapalayam, Karur-639113.

[1]kaarthikk.ece@mkce.ac.in, [3]vivekc.ece@mkce.ac.in

Abstract— **India is one the Country with High dense population. Due this high population Transportation and Parking of the Vehicles is the major issue faced by the Peoples. This Paper is to Provide a Intelligent Parking System through Image Processing. In this Systematic Approach the Image Processing Technique can be used to Identify the free empty Parking area to Park our Vehicles. In the Proposed process the Parking area can be marked with certain specific number and an sensor and with the help of these sensor the empty space can be identified to park the vehicle. The Image processing Display consists of the seven segments of display in real time. In addition to the Display an Audio system have been interfaced in order to provide Oral information of the parking system. The Seven segment display can be used to identify the empty parking area with specific numbers. The specific numbers can be displayed inorder to park the vehicle at vacant position without any struggle. The proposed process can be implemented in software platform with the help of Image processing technique and Hardware implementation can be done by interfacing with the Arduino Uno.**

Keywords— **Intelligent Parking, Image Processing, Vehicle, Arduino uno**

I. Introduction

In Olden days people use the public mode of transportation as Bus and Train for moving from one place to another. But due to Globalization the people move from rural area to urban areas for employment and other needs the Individual transportation have been improved a lot for their ease. Owing to these increase in vehicle the parking become very complicated and the people can park their on the either sides of the roads results in heavy traffic. At present there is no systematic approach in parking system. the manual control can be implemented in some areas but it is no so sound to satisfy the present number of vehicles. the number of vehicle was more than the number of parking areas[1- 4]. The administration have implemented many techniques to ensure the effortlessness of traffic at car parking zones.

At Present most the people not aware of the empty space at the parking areas. In proposed idea the camera can be used to sense the empty space through video image detection. The image can be captured and they can be allowed to Image segmentation and edge detection through boundaries with canny operator method. The moving car is to be parked at specific zone, at first the parking area has to be identified at the zone then the parking space has to be identified and check whether there is empty space is available or not. In The parking area the Image Processing Technique have been Implemented that will undergone the Image Segmentation and Edge Detection in addition to that a Counter is also Interfaced to count the Number of Entries/ Exit and have an note on that. In the Paper [5, 7] work attempts on the variation between the moving image and the Stationary Image on the Basis of Variance in Brightness of the Image. The Existing process[6,12] attempts on taking Time Differential Image. The Problem Occurs is when the object is moving at high speed it is a tedious job to take snap on that image. The Present work attempts on Identifying the Parking lots by the respective number and the Sensor present at that lot. The Sensors are placed at the parking area if the signal from transmitter will not received by the sensor then the sensor is sensed and result shows that the Parking lot is filled, so the driver will prefer the rest of the parking lots.

II. System Module

The Proposed module attempts on the Image Processing Technique through the MATLAB as a Software platform[8]. The projected mechanism involves five step module to perform the operation. The processing steps can be shown in the block diagram as shown in Fig. 1. as follows.

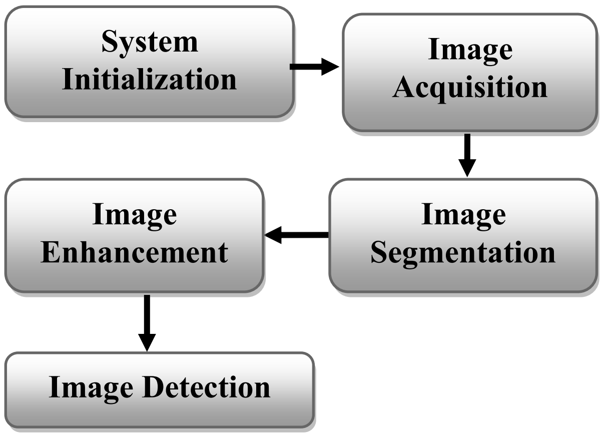


Fig. 1 Block Diagram of System Module

A. System Initialization

In the System Initialization Process the manual drawing procedure will be put into practice. In the manual Drawing process the image can be drawn with Park slot number which will be helpful in identifying the Empty parking area. The main objective of this process is to Identify the Empty Parking area without any manual interruption[9, 10]. The Diagram drawn should be visible, clear, easy to understand, complete information about the parking slot and it should be sufficient during the process of Initialization. The Sensor and the

Camera should be stationary during the initialization plan of the system architecture. Thus the detected image from the camera can be undergone for the further image processing techniques as shown in Fig. 2.

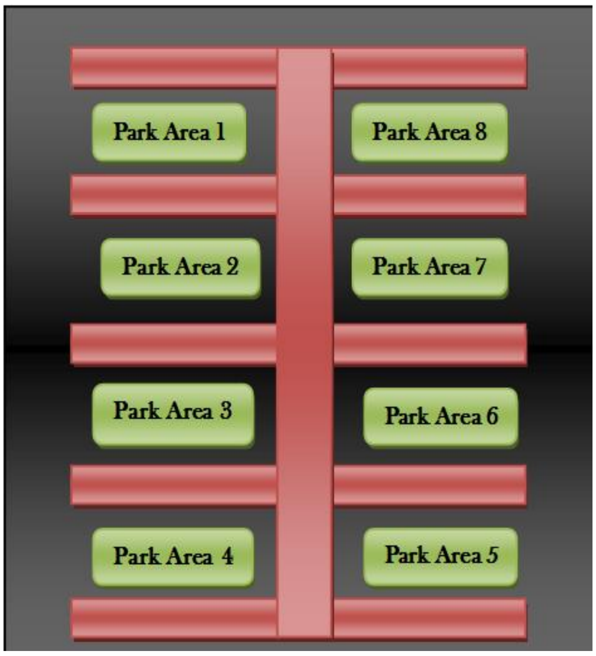


Fig. 2 System Initialization

B. Image Acquisition

Once the System Initialization module gets completed it can be allowed to the next processing module called as the image acquisition module in the Image Processing Techniques. In the Image Acquisition module the Images can been captured from the parking area through the Camera. The acquired images for the processing can be captured with High definition cameras present at the parking area[11-13]. The Images can be captured by the Camera by top view and side view of the Parking slot inorder to sense the Incoming input image consider the image as shown in Figure. 3.

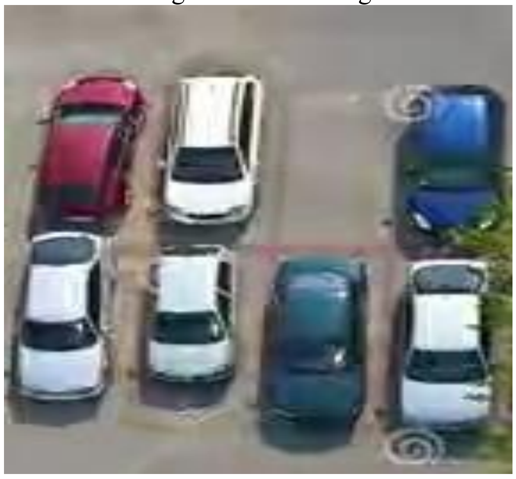


Fig. 3 Image Captured from Camera

C. Image Segmentation

Image Segmentation can be a Next type of module present at the Image Processing Steps. The Image Segmentation can be a Major part of the Image Processing technique which can be used to identify and analyze the image at a glance [4]. The process involved in the Image segmentation can be show in the block diagram as shown in figure. 4,

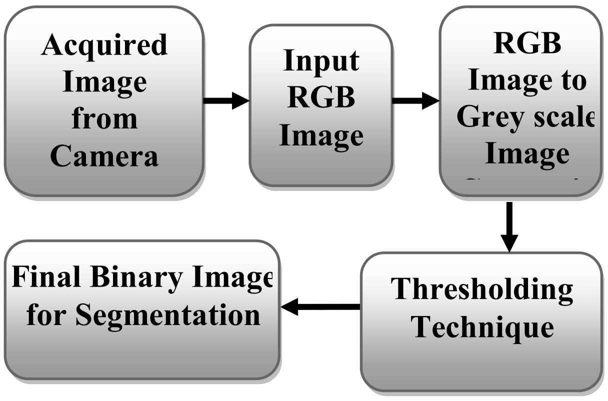

Acquired Image from Camera
Input RGB Image
RGB Image to Grey scale Image
Thresholding Technique
Final Binary Image for Segmentation

Fig. 4 Flow chart of Image Segmentation Process

The Image Segmentation process will provides the each and every part of the Image. The visual characteristics are obtained by the considering the number of pixels present in that captured image. Thus the Obtained image after segmentation process will be a better quality output result. The obtained set of Pixel can collectively provide the entire image. Thus the Empty parking area in a parking slot can be recognized by the outline, Edge, Boundary, Object, etc.,.[8] The process involved Image Segmentation is Clustering will partition the Image into number of clusters. The Clustering process can be preferred on the source of assortment of Manually or through Random Selection process as Shown in Figure. 5.

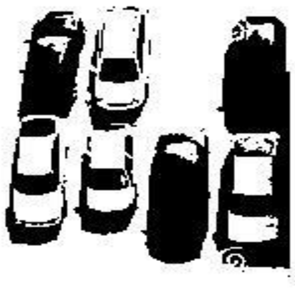


Fig. 5 Binary Image for Segmentation

D. Image Enhancement

The Image Enhancement module the binary Image obtained from the Image segmentation module is been considered. In this Process the Image has be Enhanced to remove the unwanted noise obtained during the Binary Image Conversion. They can be used to Trace the outline of the Detected Image. The Digital camera will take the Images from various locations with some noise.[9] The Obtained noise can be removed with the help of a technique called Morphology. The Morphology can be a special Technique which is used to neglect the Imperfection obtained during the Image Segmentation[1,7]. The Morphological mechanism undergone the following process named as Dilation, Erosion, Opening and Closing Process and among those four process the Opening and Closing Morphological process are the most commonly used Process for the noise removal. The Opening process is to remove the tiny objects present in the Segmented Image and the Closing process is to remove the unwanted and tiny holes present at the Segmentation process. The main role of the Morphological mechanism is to provide the exact Edge and shape of the image without any Distortion[6]. In the proposed mechanism the exact boundary of the image has to be used to detect the empty parking space is to be traced. The rest of the process named Dilation and Erosion is used in this stage inorder to increase or decrease the pixel range of the Output Image after enhancement. The Dilation is used to improve the Pixel range to the outline boundary of an image. The erosion is an another process which will removes the unwanted pixels on the boundaries[3,18]. In the proposed process if the input pixel value of an binary image is equal to '0' then the output pixel is to be '0'. Thus the obtained image of Figure. 6 will shows the Enhanced for processing [14,17].

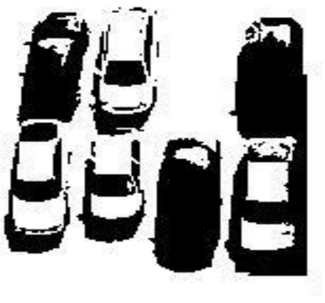


Figure 6. Image after Noise Reduction

E. Image Detection Module

The Image Detection Module is Implemented only if the Exact Edge and outline Boundaries of an Image is obtained by the Image Enhancement Module. In this process the parameters of Area and Perimeter has to be considered inorder to obtain the exact shape of an Image[5]. The exact shape of an Image is necessary to provide information to the drivers to park their vehicles in the empty parking area. The Shape of an Image can be obtained using the below given expression[9,18,21].

Shape = (4×pi×area) / (perimeter^2)

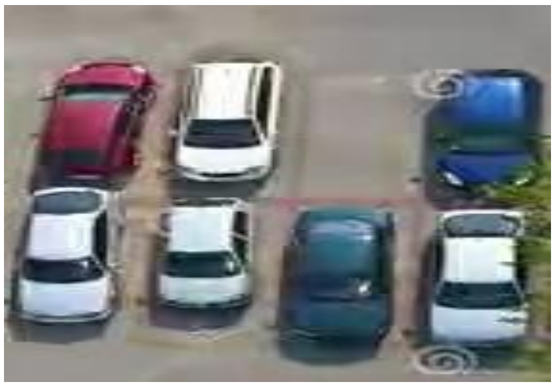


Figure 7. Detected Parking Slot

III. Results and Discussion

The parking of a vehicle through the image processing can be highly efficient and more accurate without any manual interruption. In the proposed architecture will shows a parking area with 8 slots of parking area. Based on the threshold value the Empty parking area can be displayed through an Camera preview display unit. The display can be indicated through the LED display. In addition to the LED display an Audio system have been Interfaced to the System design. Thus the Vehicle is sensed by the Sensor the availability of Parking area and appropriate Parking slot can be informed to the driver through audio announcement. Thus the proposed architecture will be very helpful in park the vehicle in the parking area without any distortion and which results in the time and parking area consumption can be reduced.

IV. Conclusion

If the proposed architecture is Implemented in the day to day life the parking of vehicle can be made easier. The Process of identifying the parking area and the number of empty parking slots can be determined with the help of an Image processing technique. The parking slots can be easily identified and the shape of that particular slot can be determined which results in occurrence of parking the vehicle in safe area within short span of time without any delay. The Proposed design is implemented for a small area of 8 Parking slots in a particular region. But the proposed idea can be extended for all around the city by providing additional information through the GPS module which would be helpful to identify the parking area in a particular zone through GSM with an Mobile application and the status of that parking area that is the availability of the empty parking slot has also to be considered in future through the Mobile Application.

References

[1] Mr. Basavaraju S R, "Automatic Smart Parking System using Internet of Things(IOT)", International Journal of Scientific and Research Publications, vol. 5,no. 12, December 2015

[2] Amit U.Godge, Vijay D. Chuadhari, Kantilal P. Rane, "Automatic Car Parking Using Image Processing", IPASJ International Journal of Electronics & Communication (IIJEC), vol. 2, no. 8, August 2014

[3] J. K. Suhr, H. G. Jung, "Sensor fusion-based vacant parking slot detection and tracking", IEEE Trans. on Intell. Transp. Syst., vol. 15, no. l, pp. 21-36, 2014.

[4] P. Xu, "Ultrasonic-based detection system design for intelligent parking spaces", Inform. & Commun, vol. 143, pp. 76-77, 2014.

[5] S. Bitam, A. Mellouk, "ITS-cloud: Cloud computing for intelligent transportation system", Proc. IEEE Global Commun. Conf., pp. 2054-2059, 2012.

[6] Atmadja Wiedjaja, "Parking guidance system based on real time operating system [J]", International Conference on Industrial Automation Information and Communications Technology, pp. 5-8, 2014.

[7] M. Anitha, K. Kaarthik, Analysis of nutrient requirement of crops using its leaf, Journal of Chemical and Pharmaceutical Sciences, Special Issue 8, December 2016, pp. 99 - 103.

[8] R. Yusnita, Fariza Norbaya, and Norazwinawati Basharuddin, "Intelligent Parking Space Detection System Based on Image Processing", International Journal of Innovation, Management and Technology, vol. 3, no. 3, June 2012

[9] Z. Bin, J. Dalin, W. Fang, and W. Tingting,"A design of parking space detector based on video image", The Ninth International Conference On Electronic Measurement &Instruments ICEMI, 2009.

[10] K. Ueda, I. Horiba, K. Ikeda, H. Onodera, and S. Ozawa, "An algorithm for detecting parking cars by the use of picture processing", IEICE Trans. Information and Systems,vol.J74-D-I1. IO, pp. 1379-1389,1991.

[11] T. Hasegawa and S. Ozawa, "Counting cars by tracking of moving object in the outdoor parking," Vehicle Navigation and Information Systems Conference, pp 63-68, 1994.

[12] E. Maeda and K. Ishii, "Evaluation of normalized principal component features in object detection," IEICE Trans. Information and Systems. vol.J75-D-11(3),pp 520-529, 1992

[13] M. Yachida, M. Asada, and S. Tsuji, "Automatic Analysis of Moving Image" IEEE Trans. Pattern Anal and Mach.Intell., vol.PAM1-3(1) pp.12-19.

[14] Ms .Sayanti Banerjee, Ms. Pallavi Choudekar and Prof. M. K. Muju. "Real time car parking system using image processing," IEEE proc, pp. 99-103, 2011.

[15] S. Arora, J. Acharya, A. Verma, Prasanta, and K. Panigrahi. "Multilevel thresholding for image segmentation through a fast statistical recursive algorithm," Elsevier 2007,Pattern recognition letters 29, pp. 119-125, 2008.

[16] L.Wang and J. Bai., "Threshold Selection by Clustering Grey Levels of Boundary", Elsevier Science. Pattern recognition letters 24. pp. 1983–1999.

[17] Thiang, Andre Teguh Guntoro, and Resmana Lim, "Type of vehicle recognition using template matching method," International conference on Electrical, Electronics, Communication, Information, March 7-8, 2001.

[18] K. Kaarthik, P. Yuvarani, "Implementation of Distributed Operating System for industrial process automation using embedded technology", Journal of Chemical and Pharmaceutical Sciences, Special Issue 8, December 2016.

[19] S. Saleh Al-Amri, N. V. Kalyankar, and Khamitkar S, "Image segmentation by using thershold techniques," Journal of Computing, vol. 2, no. 5. , ISSN 2151-9617, MAY 2010.

[20] Per Christian Henden, "Exercise in computer: a comparison of thresholding methods," NTNU–20, Nov 2004.

[21] M Rama Bai, Dr V Venkata Krishna, and J SreeDevi, "A new morphological approach for noise removal edge detection," IJCSI International Journal of Computer Science Issues, vol.7, no. 6, Nov 2010.

[22] S. Singh, "Morphological image processing," A Seminar Report, University Of Science and Technology Kochi, October 2010.

A Smart Image Processing-based System for Parking Space Vacancy Management

Benjamin Kommey
Kwame Nkrumah University Of Science and Technology
51 PUBLICATIONS **38** CITATIONS
SEE PROFILE

Andrew Selasi Agbemenu
Kwame Nkrumah University Of Science and Technology
19 PUBLICATIONS **30** CITATIONS
SEE PROFILE

International Journal of Computer Applications (0975 - 8887)
Volume 182 - No.5, July 2018

A Smart Image Processing-based System for Parking Space Vacancy Management

Benjamin Kommey
Department of Computer Engineering
KNUST
Kumasi, Ghana

Ernest O. Addo
Department of Computer Engineering
KNUST
Kumasi, Ghana

Andrew S. Agbemenu
Department of Computer Engineering
KNUST
Kumasi, Ghana

ABSTRACT

Drivers often encounter problems associated with locating empty parking slots in parking areas. This paper presents a smart parking lot management system which operates using image processing. An image processing algorithm is used to detect empty parking areas from aerial images of the parking space. The algorithm processes the image, extracts occupancy information concerning spots, and their positions thereof. The system also reports if individual parking spots are occupied or otherwise. Occupancy information is made available to newly arriving drivers by projecting it unto large displays positioned at vantage points near the vicinity. The smart parking lot management system reduces the stress and time wastage associated with car parking and makes management of such areas less costly.

General Terms

Parking Space Vacancy Management, Image Processing

Keywords

Edge Detection, Morphological Dilation,WiFi, ROI, FOV

1. INTRODUCTION

In recent times, household vehicles have increased rapidly with population growth and economic development. Meanwhile, spaces required to park these cars in public places has rather been in short supply and are becoming very costly too [14, 6, 12]. This rising imbalance is very evident in major attraction areas that witness the influx of large numbers of people visiting in motor vehicles. Finding parking spaces in such places during events is challenging. Vehicle owners often undergo the extremely frustrating process of driving through the entire area in an unguaranteed search of available parking spots. Productive working hours are lost, fuel is expended, and carbon emissions from conventional engine vehicles increases. Management of these areas often resort to simply providing more parking spaces in order to check this problem. This effort however, compounds the existing problem as locating vacant spots become increasingly difficult when the lot is more than half filled. It is therefore quite clear that proper management of available spaces is the most suitable solution. Contrary to the conventional use of wardens, smart systems that provide readily available spot vacancy information are needed to effectively satisfy these parking demands.

In order to assist drivers in this regard, some parking guidance systems that attempt to notify the drivers of vacant spots have been designed and installed in various parking lots. A very common system features sensors and light indicators at each parking space. The indicator which is lit by default is positioned such that it can be viewed from an appreciable distance and turns off only when a vehicle occupies the spot. Drivers are able to spot vacant spaces from a distant without having to move around the whole parking lot. There are numerous variants of this basic system proposed in recent years. These systems mainly vary based on the vehicular detection mechanism employed. The methods used ranged from the use of radio frequency identification (RFID) technology [4, 2] or infrared sensors [11] to ultrasonic sensors [7]. Among these is an RFID based smart parking system prototype developed by [8] which implements an automated process of vehicular checking in and checking out for parking lots using RFID reader devices. In another system proposed by [13], the effect of the Earth's magnetic field on magnetic sensors is harnessed to determine the vacancy status of parking spaces. Although these mechanisms achieve the basic intended function of vehicular detection, their operations are often marred by environmental factors such as changes in weather conditions and/or the presence of electromagnetic interferences. Upon detection of a vehicle in the allotted space, the system require a way to inform drivers of the occupancy status of that space. In an attempt to do this, some systems implement vehicular ad hoc networks (VANETs) with the assumption that the vehicles are capable of transmitting and receiving information to and fro sensors within a certain range [15, 3, 9]. The drawback of VANET systems is that they require that vehicles have a dedicated tamper-resistant communication device: a feature that older vehicles lack. Without modification to these vehicles, the operation of VANET system is flawed. There is therefore the need for a parking guidance system that works well with all types of vehicle without modication irrespective of their types and natures. An additional factor of concern for VANET systems is the network infrastructure employed for the vehicle detection sensors. Traditional systems such as [13] have often used wired infrastructure however recent developments have suggested the use of wireless sensor networks. In these networks, all detection sensors are able to wirelessly communicate with a central controller that has information about all nodes. Sensors can be repositioned effortlessly without the need for costly reinstallations. [5, 10] used ZigBee networks in their system implementations. In the work done by [3], WiFi was employed in the broadcast vacancy reports in the infrastructure. This paper presents the design and implementation of a smart system which uses image processing methods for vehicular detection coupled with wireless network technology for monitoring parking areas.

2. PROPOSED SYSTEM MODEL

2.1 Architecture of System

The smart parking system proposed in this paper consists of three main components. These are the parking detection nodes with incorporated WiFi access points (APs) deployed within each major section of the parking facility, a wireless local area network (WLAN) integrated local base station, and a notification system for information delivery. The architecture of the proposed system is shown in Figure 1.

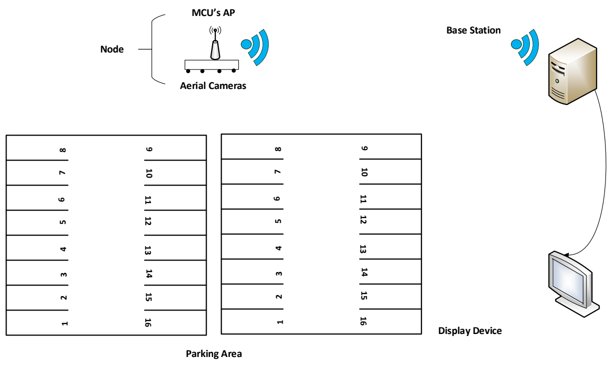


Fig. 1. Architecture of proposed parking management system.

The parking area is logically demarcated into major sections for easy deployment of parking detection nodes. Each major section of the area is equipped with a detection node. A node consists of wide field view cameras for taking snapshots of the assigned parking minor section at regular intervals and IEEE 802.11 b/g/n compatible microcontroller unit (MCU) for communication between the node and the base station. As shown in Figure 2, a node is affixed at a higher altitude close to the perimeter of its assigned parking section such that the node overlooks the area and the cameras are able to capture their individual regions of interest (ROIs) or minor sections in their field of view (FOV).
Each node's control unit is built around the TI CC3200 microcontroller which operates on an industry-standard ARM Cortex-M4 core running at 80 MHz. The cameras are connected to the MCU via the chip's fast parallel camera peripheral interface. The image sensor interface receives captured images from the camera immediately they are taken. The controller forwards the collected data to the base station over the setup local area network. For larger parking area, repeaters could be employed to enhance wireless communication between the nodes and the central station by regenerating signals transmitted over the network before they becomes too weak or corrupted. The base station receives the data from the nodes, processes them, retrieves vacancy information from the images, and updates the occupancy status of the parking lot and its vacancy thereof on the display positioned at vantage points.

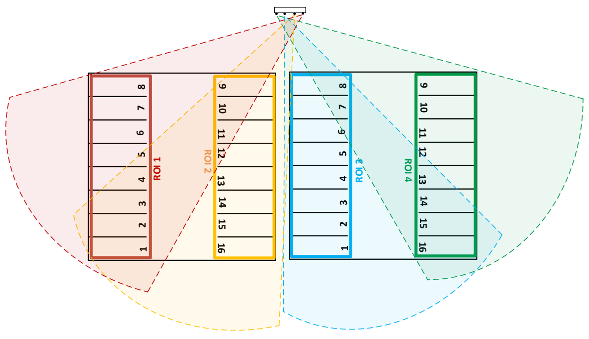


Fig. 2. Camera positions with their respective FOVs and assigned ROIs.

2.2 System Operation

The operation of the system involved four major sub-operations as shown in Figure 3.

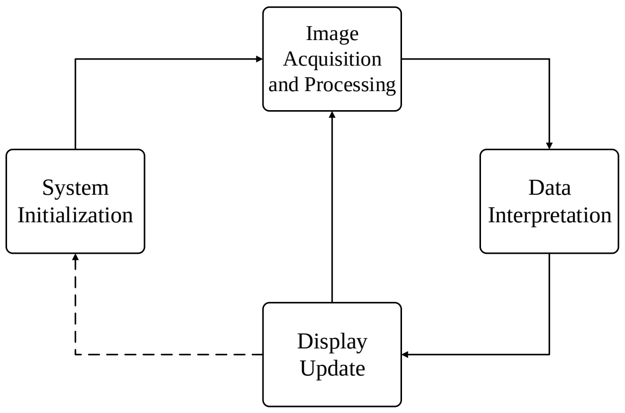


Fig. 3. Operational modules of proposed system.

These modules are system initialization, image acquisition and processing, data interpretation, and display update. The last three processes were then repeated as long as the system was active. The initialization of the parking guidance system takes place once, when the system is being set up for the first time or after a replacement of any of the systems module (shown by dashed arrow). During initialization, a refresh signal is sent from the node's controller to the image sensors in order to activate the process. The aerial cameras capture images of their assigned parking lot section whiles the area is empty. The images are transferred from the cameras to the MCU via the fast parallel image sensor interface. The MCU packages the received images into TCP packets and transmit the data from its configured on-chip Websocket (RFC 6455) HTTP server over WiFi to the base stations Websocket client application. The images are extracted from the received packets and are then processed using the image processing algorithm described in a later section of this paper. Coordinates of the marked individual parking slots and their boundaries are obtained and stored for reference during subsequent image processing of images. These coordinates could be used as references as long as positions and orientations of the nodes are fixed. Replacement of faulty cameras, change of positions of existing devices or installation of new ones would require re-initialization of the system thereafter. Figures 4, 5, 6, 7, and 8 illustrate the processing of the initial frame. At the base station, the received images undergo a preprocessing stage which involves the reduction of the cost of computing the image information [1]. The coloured image is grayscaled using the Luminosity method which calculates each resultant gray image pixel, I_{gray} as a weighted sum of the three linear-intensity RGB values. I_{gray} is related to its respective colour image RGB values: I_{red}, I_{green}, and I_{blue} by equation 1.

$$I_{gray} = 0.213(I_{red}) + 0.715(I_{green}) + 0.072(I_{blue}) \tag{1}$$

Figure 4 shows the image of an empty minor section of a test parking area.

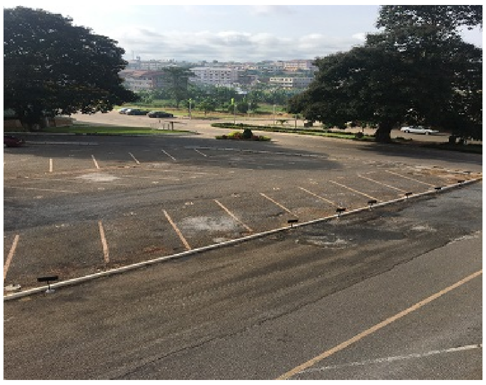


Fig. 4. An empty minor section of a test parking space.

The grayscaled image is then linearly normalized by contrast stretching to adjust image intensity values and compensate for poor image contrast which may occur due to glare. The stretched image is smoothed to remove noise and enhance accuracy of object detection and then binarized using Otsu's global thresholding method which chooses the threshold to minimize the intraclass variance of the thresholded black and white pixels. The resulting blurred and binarized image is shown in Figure 5.

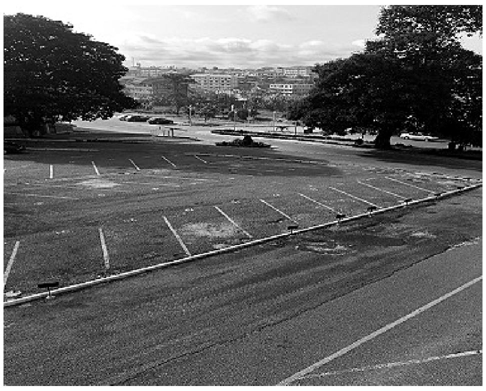


Fig. 5. Grayed, linearly normalized, and binarized image of empty minor section.

Canny edge detection is performed to identify the edges of elements in the binarized image. The edge detection process uses the first derivative of the Gaussian to approximately optimize the product of signal-to-noise ratio and localization. The Canny edge detector first smoothens the binarized image with a Gaussian filter with kernel size (σ) of 5 to reduce noise and unwanted details and textures. The gradient is then computed using the Sobel-Feldman gradient operator. Non-maximum suppression is performed on the combined result of the Sobel-Feldman convolutions to thin out edges. Double thresholding using thresholds, T_1= 0.6 and T_2= 0.85 was done to ensure less noise and fewer false edges. Finally edge tracking by hysteresis was performed to determine which weak edges were actual edges. Figure 6 shows the image obtained from applying the canny edge detection algorithm to the image in Figure 5.

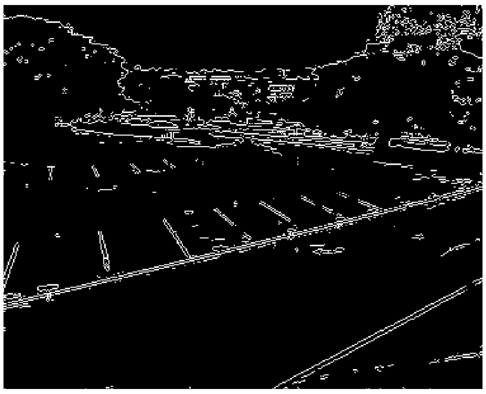


Fig. 6. Strong edges identified in binarized test image using the Canny edge detection algorithm.

A four stage morphological dilation was performed on the resulting image for better and clearer outer borders. Four different flat linear structuring elements were applied in the morphological operations. The shapes and sizes of the structuring elements (SE_n) were:

$$SE_1 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$SE_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$SE_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$SE_4 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

The outputs of the complete dilation procedure are shown in Figure 7. A flood-fill operation is performed on background pixels of the binary image to fill holes in detected objects as shown in Figure 8. Each hole is a set of background pixels that cannot be reached by filling in the background from the edge of the image.
With the initialization of the system, the ROIs in Figures 7 and 8 are identified, marked out as shown in Figures 9 and 10 respectively. Properties of the marked regions such as coordinates of element edges and mean pixel count are stored. Figure 11 shows the

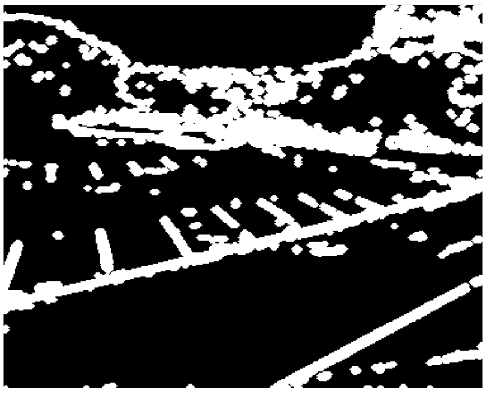


Fig. 7. Resulting edge image after fourth dilation

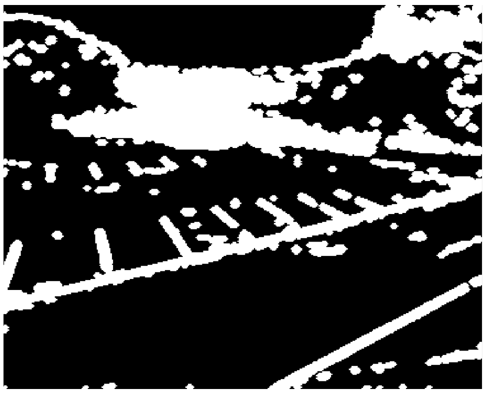


Fig. 8. Dilated edge image with holes filled

measured mean pixel value per identified spot in Figures 9 and 10 for the empty parking space during initialization.

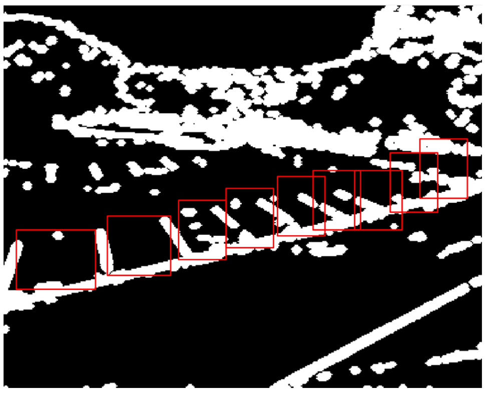


Fig. 9. Identified and marked park spaces in ROI of Figure 7

During subsequent operations of the system, parking areas in the filled binarized images are marked with bounding-boxed regions based on the coordinates obtained from the initialization process. Pixel count in each region is determined and compared against the stored initial values. Parking spots with cars would have extra edges within the marked regions which increases the mean intensity within the regions. A substantial increase in a region's pixel count infers occupancy of the corresponding spot. This occupancy information is then sent from the base station to the display devices for use by incoming drivers.

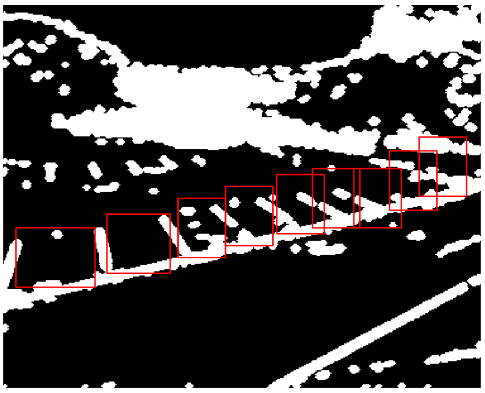


Fig. 10. Identified and marked park spaces in ROI of Figure 8

Fig. 11. A graph showing the mean intensity per empty parking region from Figures 9 and 10

3. TESTING, RESULTS AND DISCUSSION

The proposed parking space vacancy management system was tested to determine its ability to accurately extract vacancy information from captured images of the parking space. A test image was obtained using an 8MP aerially positioned camera. The image used is shown in Figure 12. The image was grayed, stretched, smoothed, and binarized by using the image processing techniques described earlier. Figure 13 shows the grayed, smoothed and binarized test image.

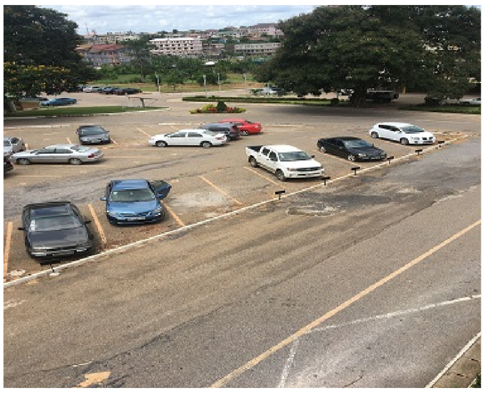


Fig. 12. Test image showing parking space minor section with cars in some spots.

The Canny edge detection algorithm was applied to identify strong edge elements in the binarized image. The obtained edge image is shown in Figure 14. The edge image was then dilated with the four stage morphological procedure using the four structural element objects stated in the previous section (Figure 15). Holes in the dilated test image were then filled. Figure 16 shows the result of the background pixel flood-filling procedure.

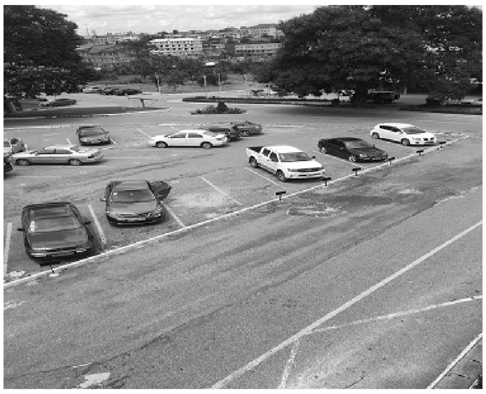


Fig. 13. Grayed, contrast stretched, and binarized test image

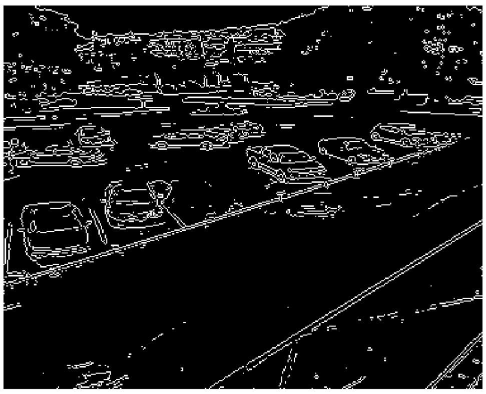


Fig. 14. Edge test image obtained using the Canny algorithm.

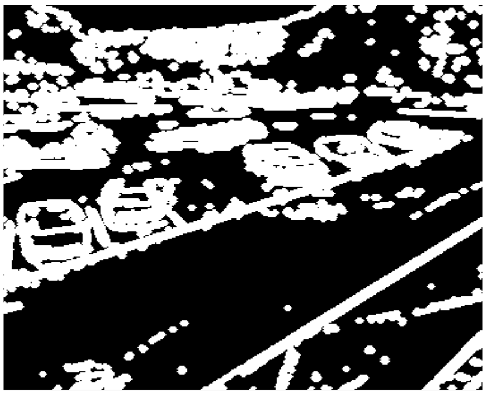


Fig. 15. Completely dilated test edge image

The parking spots in Figures 15 and 16 were all marked as shown in Figures 17 and 18 and their individual bounded pixel measurements were taken. The graph in Figure 19 shows the pixel counts for each park spots in both images.
Graphical data in Figure 19 shows a substantial increase in the mean pixel count in regions which had cars occupying park spots when compared to the data in Figure 11's graph. In spot vacancy determination, the average percentage increase in intensity recorded for occupied spots when using the filled images (Figures 10 and 18) was 166.9% with a mean absolute deviation (MAD) of 96.4.

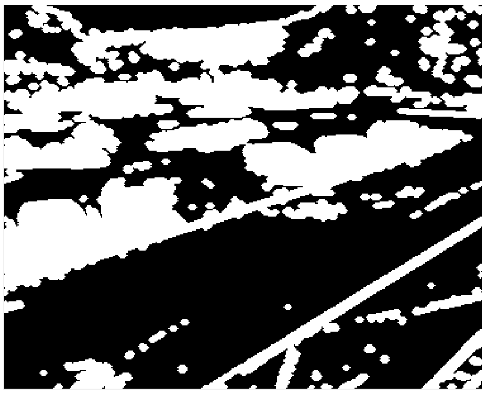


Fig. 16. Dilated edge test image with holes in detected objects filled

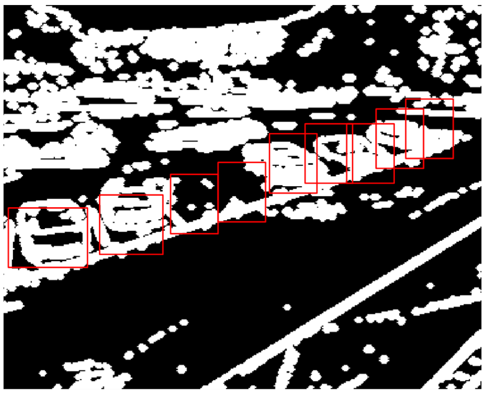


Fig. 17. Marked park spaces in dilated test edge image

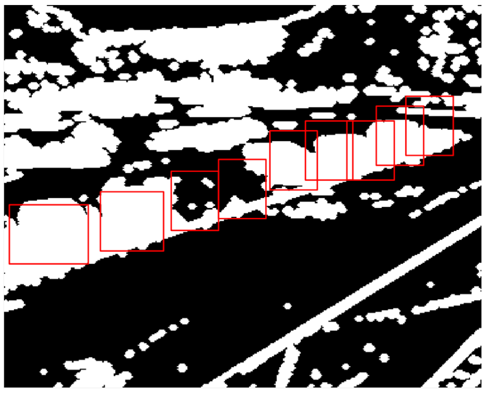


Fig. 18. Marked park slots in filled edge image

The dilated image approach recorded a mean percentage increase in pixel count was 133.7% with an MAD of 86.5. It was observed that the use of filled edge images provided greater intensity in occupied parking regions and hence provided a much more solid basis for declaring occupancy especially in areas which lie in the middle of the camera's FOV as compared to the use of only the dilated edge images. However, due to the position and height of the camera used, the filled image approach faired averagely in detecting vacant spots (spots 6 and 8) located between occupied ones at the extreme

Fig. 19. A graph showing the mean intensity per parking region from Figures 17 and 18

right side of the sensor's FOV. In these instances, the dilated edge image approach exhibited good success.

4. CONCLUSION

The proposed system accurately detected the presence of cars in parking slots. The filled image approach performed better than the use of dilated edge images to determine vacancy of parking spots. The two approaches could be fused into one system. The camera position could be adjusted to improve the performance. From the above set of test results, it has been shown that the proposed image-processing-based system is a viable option for parking space vacancy management. Other technologies such as automatic number plate recognition and traffic light control could be conjoined with this system to form holistic intelligent transport systems.

5. REFERENCES

[1] A. S. Agbemenu, J. Yankey, and E. O. Addo. An automatic number plate recognition system using opencv and tesseract ocr engine. *International Journal of Computer Applications*, 180:1–5, 2018.

[2] J. P. Benson, T. O'Donovan, P. O'Sullivan, U. Roedig, and C. Sreenan. Car-park management using wireless sensor networks. In *31st IEEE Conf. Local Computer Networks*, pages 588–595, 2006.

[3] M. Caliskan, D. Graupner, and M. Mauve. Decentralized discovery of free parking places. In *VANET 06: 3rd international workshop on Vehicular ad hoc networks*, pages 30–39, 2006.

[4] S. C. Hanche, P. Munot, P. Bagal, K. Sonawane, and P. Pooja. Automated vehicle parking system using rfid. *International Journal on Research and Development*, 1:89–92, 2013.

[5] Y. Hirakata, A. Nakamura, K. Ohno, and M. Itami. Navigation system using zigbee wireless sensor network for parking. In *12th International Conference on ITS Telecommunications*, pages 605–609, 2012.

[6] T. Huang, Z. Zeng, and C. Li. Design and implementation of a prototype smart parking (spark) system using wireless sensor networks. In *International Conference on Neural Information Processing*, pages 506–515, 2012.

[7] A. Kianpisheh, N. Mustaffa, P. Limtrairut, and P. Keikhosrokiani. Smart parking system (sps) architecture using ultrasonic detector. *International Journal of Software Engineering and Its Applications*, 6:55–58, 2012.

[8] Z. Pala and N. Inanc. Implementation of rfid technology in parking lot access control system. In *12th International Conference on ITS Telecommunications*, pages 1–3, 2007.

[9] R. Panayappan, J. M. Trivedi, A. Studer, and A. Perrig. Vanet-based approach for parking space availability. In *VANET 07: Fourth ACM International Workshop on Vehicular Ad-Hoc Network*, pages 75–76, 2007.

[10] S. Srikanth, P. Pramod, K. Dileep, S. Tapas, M. U. Patil, and C. B. N. Sarat. Design and implementation of a prototype smart parking (spark) system using wireless sensor networks. In *International Conference on Advanced Information Networking and Applications Workshops*, pages 401–406, 2009.

[11] S. Srikurinji, U. Prema, S. Sathya, and P. Manivannan. Smart parking system architecture using infrared detector. *International Journal of Advanced Information and Communication Technology*, 2:1113–1116, 2016.

[12] D. Teodorovic and P. Lucic. Intelligent parking systems. *European Journal of Operational Research*, 175:16661681, 2006.

[13] J. Wolff, T. Heuer, H. Gao, M. Weinmann, S. Voit, and U. Hartmann. Parking monitor system based on magnetic eld sensors. In *IEEE Conference on Intelligent Transportation Systems*, pages 1275–1279, 2006.

[14] G. Yan, S. Olariu, M. C. Weigle, and M. Abuelela. Smart-parking: A secure and intelligent parking system using notice. In *11th International IEEE Conference on Intelligent Transportation Systems*, pages 569–574, 2008.

[15] H. Zhao, L. Lu, C. Song, and Y. Wui. Ipark: Location-aware-based intelligent parking guidance over infrastructure-less vanets. *International Journal of Distributed Sensor Networks*, 2012:1–2, 2012.