**TRAFFIC MANAGEMENT FOR EMERGENCY VEHICLE DETECTION USING OPEN CV**

A PROJECT REPORT

submitted

*in the partial fulfillment of the requirements for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE AND ENGINEERING**

by

**Peddi.P.M. Laasya Rao 18B81A05L3**

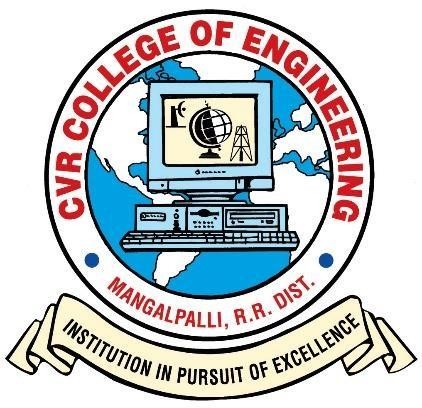
**Tummala Lekha 18B81A05K8**

**Rasamalla Sushmitha 18B81A05N7**

Under the guidance of

Ms.D.A.Rachana

Professor : CSE



CVR COLLEGE OF ENGINEERING

#### (An Autonomous Institution)

#### Department of Electronics and Instrumentation Engineering

Vastunagar, Mangalpalli (V), Ibrahimpatnam (M), R.R. District, Telangana 501510

 CVR COLLEGE OF ENGINEERING

(An Autonomous Institution)

ACCREDITED BY NBA, NAAC 'A' GRADE

(Approved by AICTE & Government of Telangana and Affiliated to INTU Hyderabad]

Vastunagar, Mangalpalli (V) Ibrahimpatnam (M), R.R. District.

Web: http://www.cvr.ac.in Email: info@cvr.ac.in

**CERTIFICATE**

This is to certify that the project entitled “TRAFFIC MANAGEMENT FOR EMERGENCY VEHICLE DETECTION USING OPEN CV” being submitted by

Peddi. P. M. Laasya Rao 18B81A05L3

Tummala Lekha 18B81A05K8

Rasamalla Sushmitha 18B81A05N7

in partial fulfillment for the award of Bachelor of Technology in Computer Science and Engineeringto the CVR College of Engineering, is a record of bona fide work carried out by them under my guidance and supervision during the year 2020-2021.

The results embodied in this project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Signature of the project guide, Signature of the HOD

**Ms. D. A. Rachana**  **Dr. A. Vani Vathsala**

Asst. Professor, CSE HOD, CSE

CVR College of Engineering CVR College of Engineering

External Examiner

# ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mentioning of the people whose constant guidance and encouragement made it possible.

We express our utmost gratitude to **Dr. K.S. Nayanathara**, Principal of our college, for constant encouragement and motivation throughout the academic.

We derive great pleasure in expressing our sincere gratitude to our Professor and Head of the Department **Ms.Vani vatsala** for her timely suggestions which helped us to complete this work successfully. We express our sincere thanks to our internal guide **Ms.D.A.Rachana ,** Professor CSE Dept, for giving us support, kind attention and valuable guidance throughout the project

Tummala Lekha ,18B81A05K8

Peddi. P. M. Laasya Rao,18B81A05L3

Rasamalla Sushmitha,18B81A05N7

`

i

# ABSTRACT

Traffic control systems play an important role to manage traffic congestion on the road especially during peak hours and peak seasons. One of the main challenges is to control the traffic when there are emergency cases at traffic light intersections, especially peak hours. This could affect the route for emergency vehicles such as ambulances, fire brigade and police cars to reach their destination. Due to the increase of traffic congestion during peak hours and peak seasons in Malaysia, there is a need for further evaluation of traffic control techniques. This paper reviewed and consolidated information on the different types of the existing traffic control system for road traffic management such as OpenCV and image processing. This paper analysed and compared the design, benefits and limitations of each technique. Through the reviews, this paper recommends the best traffic control technique for emergency vehicles that offers low price, low maintenance and can be used in various areas of applications.

ii

|  |  |  |  |
| --- | --- | --- | --- |
| **Table of Contents** | | | |
|  | | | Page No. |
|  |  | List of tables |  |
|  |  | List of Figures |  |
|  |  | Symbols |  |
|  |  | Abbreviations |  |
| **Chapters** |  |  |  |
| 1 |  | **Introduction** |  |
|  | 1.1 | Introduction |  |
|  | 1.2 | Problem statement and objective of the project |  |
|  | 1.3 | Existing System |  |
|  | 1.4 | Proposed System |  |
|  | 1.5 | Literature survey |  |
|  | 1.6 | Organization of the report |  |
| 2 |  | **Software & Hardware specifications** |  |
|  | 2.1 | Hardware requirements |  |
|  | 2.2 | Software requirements |  |
| 3 |  | **Design and Planning** |  |
|  | 3.1 | System Architecture |  |
|  | 3.2 | Use Case Diagram |  |
|  | 3.3 | Class Diagram |  |
|  | 3.4 | Sequence Diagram |  |
|  | 3.5 | Activity Diagram |  |
|  | 3.6 | Input Design and Output Design |  |
| 4 |  | **Implementation Details** |  |
|  | 4.1 | Sample Code |  |
|  | 4.2 | System study |  |
|  | 4.3 | System Testing |  |
|  | 4.4 | Project methods |  |
| 5 |  | **Result** |  |
| 6 |  | **Conclusion** |  |
| 7 |  | **References:** |  |
| 8 |  | **Appendix:** |  |
|  |  |  |  |

# LIST OF FIGURES

[Figure 1.1 Curve on the effect of Social Distancing 2](#_bookmark6)

[Figure 1.2 Importance of Social Distancing 4](#_bookmark8)

[Figure 2. 1 Arduino Uno 9](#_bookmark16)

[Figure 2.2 Arduino UNO Microcontroller 11](#_bookmark18)

[Figure 2.3 Ultrasonic Sensor Schematic Diagram 15](#_bookmark20)

[Figure 2.4 Ultrasonic Sensor 16](#_bookmark21)

[Figure 2.5 Timing Diagram 17](#_bookmark22)

[Figure 2.6 Pin Description 18](#_bookmark24)

[Figure 2. 7 Buzzer 19](#_bookmark25)

[Figure 2. 8 Male to Female jumper Wires 20](#_bookmark26)

[Figure 2. 9 9V Battery 21](#_bookmark27)

[Figure 2. 10 PP(Power Pack) battery series from left to right: PP1, PP3, PP4, PP6, PP7,](#_bookmark28) [PP8, PP9, PP10, PP11. 22](#_bookmark28)

[Figure 2. 11 LED Symbol 23](#_bookmark29)

[Figure 2. 12 LED 25](#_bookmark31)

[Figure 3. 1 Arduino IDE 27](#_bookmark35)

[Figure 3. 2 Arduino IDE Logo 28](#_bookmark36)

[Figure 4. 1 Block Diagram 31](#_bookmark42)

[Figure 4. 2 Schematic Diagram 33](#_bookmark45)

[Figure 4. 3 Flow Chart 34](#_bookmark48)

# CHAPTER 1 INTRODUCTION

## 1.1 INTRODUCTION

Traffic light controllers play a significant role in maintaining smooth traffic flows in city environments. The sequences and durations of traffic light signals are two key factors that must be considered when designing a traffic light controller. In many countries, most traffic light controllers feature fixed sequences and durations of light signals, which do not consider dynamically changing traffic environments. Such fixed traffic light control methods are only suitable for stable and regular traffic, and not for dynamic traffic situations. Therefore, traditional traffic light controllers are one of the main factors contributing to severe road congestion in urban areas. In addition, facilitating and prioritising the transit of emergency vehicles in urban areas comprises an important safety issue. However, in traditional traffic light controllers, light sequences are determined without considering the presence of emergency vehicles. Therefore, emergency vehicles such as ambulances, police cars, and fire engines must wait at intersections, which increases their delays and leads to the loss of lives and property. Improving emergency response times is extremely critical, particularly for fire and health-related incidents. However, when the number of vehicles increases at an intersection, this not only increases the response times of emergency vehicles but also increases the likelihood of accidents occurring when emergency vehicles enter intersections at high speed. For example, in Ireland, an average of 700 fatalities are recorded every year because of late ambulance responses . The National Highway Traffic Safety Administration in the US released a report regarding accidents that involved emergency vehicles such as fire trucks, ambulances, and police cars. The agency studied the number of ambulance accidents over the 20-year period of 1992–2011 and estimated that an average of 1,500 accidents involving ambulances occurred each year, wherein 33 people were killed and 2,600 people were estimated to be injured. According to the report, fire engine accidents are the second leading cause of death for firefighters. There were roughly 31,600 accidents involving fire vehicles over a 10-year period in which 630 firefighters were killed. In addition, it is reported that there are approximately 300 fatalities in the US each year during police pursuits, where 30% of the fatalities comprise people not involved in a pursuit. Therefore, reducing emergency response times by minutes or even seconds is crucial in emergency situations. An intelligent traffic management system is mandatory for effectively avoiding emergency vehicle accidents at intersections by presenting green and red signals to emergency vehicles and non-emergency vehicles, respectively, based on an intelligent priority algorithm.

**1.2 PROBLEM STATEMENT:**

Automatic traffic monitoring and surveillance are important for road usage

and management. Traffic parameter estimation has been an active research area for the

development of Intelligent Transportation Systems (ITS). Various sensors have been

employed to estimate traffic parameters for updating traffic information.

It is well recognized that vision-based camera system are more versatile for

traffic parameter estimation. In addition to quantitative description of road congestion, image

measurement can provide quantitative description of traffic flow.

**OBJECTIVE OF THE PROJECT:**

To provide clear way to the ambulance whenever it enters into the range of camera

and to control the signals by measuring the density of traffic thereby avoiding the wastage of

time and saving the lives of human being.

**A picture containing text, car, outdoor, road

Description automatically generated**

**1.3 Existing System:**

Being responsible citizens, we have to understand that there are people’s lives in the ambulance are at stake. Showing a little kindness and responsibility towards the traffic rules while giving way to an ambulance can actually save a life. It’s important for motorists to take into consideration that hearing a siren gives the sign to move to a halt on the left side, open the right lane and no vehicle should move until the ambulance passes.

**Disadvantages:**

To create awareness among other motorists against  tailgating the ambulance. There are multiple options in the market but they are not cost effective and user friendly. Moreover the accuracy of detecting the vehicle is very poor.

## A picture containing text, outdoor, tree, car Description automatically generated

## 1.4 Proposed System:

In this paper we are proposing a model which uses existing python libraries to detect the captions on the emergency vehicles and blows a buzzer indicating there is an emergency vehicle behind. So that emergency vehicles gets the way.

**Advantages:**

Proposed solution has many advantages like it is easy to implement and cost effective at the same time.

## A picture containing road, car, outdoor, street Description automatically generated

## Literature Servey:

**1.Title: Arrival time based traffic signal optimization for intelligent transportation systems**

**Author/s:V. Paruchuri, S. Chellappan, and R. B. Lenin**

**Abstract:** Road Transportation is a crucial component of today’s society, which drives several facets of our lives. The goal of intelligent transportation systems (ITS) is to improve the effectiveness, efficiency, and safety of the transportation system. Traffic signals are an elementary component of all road transportation systems. In order to maximize the productivity of a city, traffic signals must be able to efficiently control the flow of vehicles. Traditionally, current traffic signal optimization is based on traffic arrival rates, either estimated or forecasted. In this paper, we illustrate that arrival time based solutions can outperform arrival rate based approaches. To the best of our knowledge, this is the first work that exploits arrival times of vehicles to improve traffic signal efficiency in order to reduce stopped delays and fuel consumptions, thus in turn reducing greenhouse gases and emissions. We show that arrival time knowledge can be utilized in realizing drastic gains in sparse load scenarios and significant gains in moderate load scenarios. The performance improvement translates to reducing stopped delays by over 40,000 hours daily and in reducing fuel consumption by over 650 gallons/signal/day.

**2.Traffic Management for Emergency Vehicle Priority Based on Visual Sensing**

**Author/s:K. Nellore and G. Hancke**

**Abstract:** Vehicular traffic is endlessly increasing everywhere in the world and can cause terrible traffic congestion at intersections. Most of the traffic lights today feature a fixed green light sequence, therefore the green light sequence is determined without taking the presence of the emergency vehicles into account. Therefore, emergency vehicles such as ambulances, police cars, fire engines, etc. stuck in a traffic jam and delayed in reaching their destination can lead to loss of property and valuable lives. This paper presents an approach to schedule emergency vehicles in traffic. The approach combines the measurement of the distance between the emergency vehicle and an intersection using visual sensing methods, vehicle counting and time sensitive alert transmission within the sensor network.

**3.Reducing emergency services response time in smart cities: An advanced adaptive and fuzzy approach**

**Author/s:**Soufiene Djahel; Nicolas Smith; Shen Wang; John Murphy

**Abstract**: Nowadays, the unprecedented increase in road traffic congestion has led to severe consequences on individuals, economy and environment, especially in urban areas in most of big cities worldwide. The most critical among the above consequences is the delay of emergency vehicles, such as ambulances and police cars, leading to increased deaths on roads and substantial financial losses. To alleviate the impact of this problem, we design an advanced adaptive traffic control system that enables faster emergency services response in smart cities while maintaining a minimal increase in congestion level around the route of the emergency vehicle. This can be achieved with a Traffic Management System (TMS) capable of implementing changes to the road network's control and driving policies following an appropriate and well-tuned adaptation strategy. This latter is determined based on the severity of the emergency situation and current traffic conditions estimated using a fuzzy logic-based scheme. The obtained simulation results, using a set of typical road networks, have demonstrated the effectiveness of our approach in terms of the significant reduction of emergency vehicles' response time and the negligible disruption caused to the non-emergency vehicles travelling on the same road network.

**4.A Survey on Urban Traffic Management System Using Wireless Sensor Networks**

**Author/s:K. Nellore and G. P. Hancke**

**Abstract:** Nowadays, the number of vehicles has increased exponentially, but the bedrock capacities of roads and transportation systems have not developed in an equivalent way to efficiently cope with the number of vehicles traveling on them. Due to this, road jamming and traffic correlated pollution have increased with the associated adverse societal and financial effect on different markets worldwide. A static control system may block emergency vehicles due to traffic jams. Wireless Sensor networks (WSNs) have gained increasing attention in traffic detection and avoiding road congestion. WSNs are very trendy due to their faster transfer of information, easy installation, less maintenance, compactness and for being less expensive compared to other network options. There has been significant research on Traffic Management Systems using WSNs to avoid congestion, ensure priority for emergency vehicles and cut the Average Waiting Time (AWT) of vehicles at intersections. In recent decades, researchers have started to monitor real-time traffic using WSNs, RFIDs, ZigBee, VANETs, Bluetooth devices, cameras and infrared signals. This paper presents a survey of current urban traffic management schemes for priority-based signalling, and reducing congestion and the AWT of vehicles. The main objective of this survey is to provide a taxonomy of different traffic management schemes used for avoiding congestion. Existing urban traffic management schemes for the avoidance of congestion and providing priority to emergency vehicles are considered and set the foundation for further research.

**5.An efficient distributed localisation algorithm for wireless sensor networks: Based on smart reference selection method**

**Author/s:A. M. Abu-Mahfouz and G. P. Hancke**

**Abstract:** Determining the location of nodes is a key part of wireless sensor networks (WSNs). Many WSN applications require knowledge of nodes' locations to perform their functions successfully. Several localisation algorithms rely on using all or most of the available references to enhance their performance. However, to implement an efficient localisation algorithm for WSNs one should reconsider this assumption. This paper introduces an efficient localisation algorithm that is based on a novel smart reference-selection method. This method chooses only those references that would increase the overall localisation accuracy, and it also minimises the number of iterations needed to refine the accuracy of the estimated position. Simulation results confirm that, compared to existing approaches, the proposed reference selection technique and associated localisation algorithm achieves both reliable and accurate position estimate using a minimal number of references. This decreases the computational burden of gathering and analysing location data from the high number of references previously believed to be necessary.

## 

**1.6 ORGANISATION OF THE REPORT**

**1.6.1 INTRODUCTION**

This section includes the overall view of the project i.e. the basic problem definition and the general overview of the problem which describes the problem in layman terms. It also specifies the software used and the proposed solution strategy.

**1.6.2 SOFTWARE REQUIREMENTS SPECIFICATION**

This section includes the Software and hardware requirements for the smooth running of the application.

**1.6.3 DESIGN & PLANNING**

This section consists of the Software Development Life Cycle model. It also contains technical diagrams like the Data Flow Diagram and the Entity Relationship diagram.

**1.6.4 IMPLEMENTATION DETAILS**

This section describes the different technologies used for the entire development process of the Front-end as well as the Back-end development of the application.

**1.6.5 RESULTS AND DISCUSSION**

This section has screenshots of all the implementation i.e. user interface and their description.

**1.6.6 SUMMARY AND CONCLUSION**

This section has screenshots of all the implementation i.e, user interface and their description.

**CHAPTER 2**

**HARDWARE AND SOFTWARE REQUIREMENTS SPECIFICATION**

**HARDWARE REQUIREMENTS:**

Processor : intel i5,

Ram : 8 GB

Hard Disk : 1 TB

**SOFTWARE REQUIREMENTS:**

Programming language : Python 3.7

IDE : Pycharm

**CHAPTER 3: DESIGN AND PLANNING**

**3.1 System Architecture:**

**Diagram

Description automatically generated**

**3.2 USE CASE DIAGRAM:**

A picture containing diagram

Description automatically generated

**3.3 CLASS DIAGRAM:**

Text

Description automatically generated

**3.4 SEQUENCE DIAGRAM:**

A picture containing text

Description automatically generated

**3.5 ACTIVITY DIAGRAM**

Graphical user interface, text

Description automatically generated

**3.6 INPUT DESIGN AND OUTPUT DESIGN**

**INPUT DESIGN**

**The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:**

**➢ What data should be given as input?**

**➢ How the data should be arranged or coded?**

**➢ The dialog to guide the operating personnel in providing input.**

**➢ Methods for preparing input validations and steps to follow when error occur.**

**OBJECTIVES**

**1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.**

**2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.**

**3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow**

**OUTPUT DESIGN**

**A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.**

**1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.**

**2. Select methods for presenting information.**

**3. Create document, report, or other formats that contain information produced by the system.**

**The output form of an information system should accomplish one or more of the following objectives.**

**❖ Convey information about past activities, current status or projections of the**

**❖ Future.**

**❖ Signal important events, opportunities, problems, or warnings.**

**❖ Trigger an action.**

**❖ confirm action.**

**CHAPTER 4 : IMPLEMENTATION DETAILS**

**4.1 SAMPLE CODE:**

**Main.py:**

from flask import Flask, render\_template, request,session,send\_from\_directory,Response

import os

import cv2

import pytesseract

pytesseract.pytesseract.tesseract\_cmd = r'C:/Program Files/Tesseract-OCR/tesseract.exe'

from threading import Thread

from text\_to\_speech import speak

from MyRecordVideo import recordVideo

from alarm import alarm

import numpy as np

from time import sleep

app = Flask(\_name\_)

APP\_ROOT = os.path.dirname(os.path.abspath(\_file\_))

cameraNumber = 0

def pega\_centro(x, y, w, h):

x1 = int(w / 2)

y1 = int(h / 2)

cx = x + x1

cy = y + y1

return cx,cy

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/fimage')

def fimage():

return render\_template('fimage.html')

@app.route('/fvideo')

def fvideo():

return render\_template('fvideo.html')

@app.route('/flive')

def flive():

return render\_template('flive.html')

@app.route('/detectimage')

def detectimage():

return render\_template('detectImage.html')

@app.route('/detectVideo')

def detectVideo():

return render\_template('detectVideo.html')

@app.route('/detectLive')

def detectLive():

return render\_template('detectLive.html')

@app.route('/record')

def record():

return render\_template('record.html')

@app.route('/record1')

def record1():

recordVideo()

return render\_template('umsg.html', msg='Operation Completed', color='text-success')

@app.route('/VehicleCount')

def VehicleCount():

return render\_template('VehicleCount.html')

@app.route('/fimage1',methods=['POST'])

def fimage1():

target = os.path.join(APP\_ROOT, 'uploads/')

destination = ''

filename = ''

for upload in request.files.getlist("file"):

filename = upload.filename

destination = "/".join([target, filename])

upload.save(destination)

path = 'uploads/'+filename

frame = cv2.imread(path)

text = pytesseract.image\_to\_string(frame)

Thread(target=say\_text, args=(text,)).start()

return render\_template('fimage1.html',filename=filename,emotion=text)

@app.route('/detectImage1',methods=['POST'])

def detectImage1():

al = alarm()

target = os.path.join(APP\_ROOT, 'uploads/')

destination = ''

filename = ''

for upload in request.files.getlist("file"):

filename = upload.filename

destination = "/".join([target, filename])

upload.save(destination)

path = 'uploads/'+filename

frame = cv2.imread(path)

frame2 = cv2.flip(frame, 1)

text = pytesseract.image\_to\_string(frame)

text2 = pytesseract.image\_to\_string(frame2)

print(text2)

print(text)

text= text.upper()

display = ""

if "POLICE" in text:

display="POLICE VEHICLE DETECTED"

al.detectAlarm()

if "FIRE" in text:

display = display +"FIRE ENGINE VEHICLE DETECTED"

al.detectAlarm()

if "AMBULANCE" in text:

display = display +"FIRE ENGINE VEHICLE DETECTED"

al.detectAlarm()

if "AMBULANCE" in text2:

display = display + "AMBULANCE VEHICLE DETECTED"

al.detectAlarm()

if len(display) == 0:

display="NO EMERGENCY VEHICLE DETECTED"

font = cv2.FONT\_HERSHEY\_SIMPLEX

org = (50, 50)

fontScale = .3

color = (255, 0, 0)

thickness = 1

image = cv2.putText(frame, display, org, font,

fontScale, color, thickness, cv2.LINE\_AA)

cv2.imshow("hai", image)

cv2.waitKey(0)

Thread(target=say\_text, args=(text,)).start()

return render\_template('detectImage1.html',filename=filename,emotion=text,display=display)

@app.route('/fvideo1',methods=['POST'])

def fvideo1():

target = os.path.join(APP\_ROOT, 'uploads/')

destination = ''

filename = ''

for upload in request.files.getlist("file"):

filename = upload.filename

destination = "/".join([target, filename])

upload.save(destination)

print('uploads/' + filename)

cam = cv2.VideoCapture('uploads/' + filename)

cv2.namedWindow("Character Recognition")

img\_counter = 0

capturedImage = None

while True:

ret, frame = cam.read()

if not ret:

print("failed to grab frame")

break

cv2.imshow("Character Recognition", frame)

k = cv2.waitKey(1)

if k % 256 == 27:

# ESC pressed

print("Escape hit, closing...")

break

elif k % 256 == 32:

# SPACE pressed

img\_name = "opencv\_frame\_{}.png".format(img\_counter)

cv2.imwrite('uploads/'+img\_name, frame)

print("{} written!".format(img\_name))

img\_counter += 1

capturedImage = img\_name

break

cam.release()

cv2.destroyAllWindows()

if capturedImage==None:

return render\_template('umsg.html', msg='Image not Captured', color='bg-success')

img = cv2.imread('uploads/' + capturedImage)

text = pytesseract.image\_to\_string(img)

Thread(target=say\_text, args=(text,)).start()

return render\_template('fimage1.html', filename=capturedImage, emotion=text)

@app.route('/detectVideo1',methods=['POST'])

def detectVideo1():

al = alarm()

target = os.path.join(APP\_ROOT, 'uploads/')

destination = ''

filename = ''

for upload in request.files.getlist("file"):

filename = upload.filename

destination = "/".join([target, filename])

upload.save(destination)

print('uploads/' + filename)

cam = cv2.VideoCapture('uploads/' + filename)

cv2.namedWindow("Character Recognition")

img\_counter = 0

capturedImage = None

largura\_min = 80 # Largura minima do retangulo

altura\_min = 80 # Altura minima do retangulo

offset = 6 # Erro permitido entre pixel

pos\_linha = 550 # Posição da linha de contagem

delay = 60 # FPS do vídeo

detec = []

carros = 0

subtracao = cv2.bgsegm.createBackgroundSubtractorMOG()

while True:

ret, frame = cam.read()

if not ret:

print("failed to grab frame")

break

tempo = float(1 / delay)

sleep(tempo)

grey = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

blur = cv2.GaussianBlur(grey, (3, 3), 5)

img\_sub = subtracao.apply(blur)

dilat = cv2.dilate(img\_sub, np.ones((5, 5)))

kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (5, 5))

dilatada = cv2.morphologyEx(dilat, cv2.MORPH\_CLOSE, kernel)

dilatada = cv2.morphologyEx(dilatada, cv2.MORPH\_CLOSE, kernel)

contorno, h = cv2.findContours(dilatada, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

cv2.line(frame, (25, pos\_linha), (1200, pos\_linha), (255, 127, 0), 3)

for (i, c) in enumerate(contorno):

(x, y, w, h) = cv2.boundingRect(c)

validar\_contorno = (w >= largura\_min) and (h >= altura\_min)

if not validar\_contorno:

continue

cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

centro = pega\_centro(x, y, w, h)

detec.append(centro)

cv2.circle(frame, centro, 4, (0, 0, 255), -1)

for (x, y) in detec:

if y < (pos\_linha + offset) and y > (pos\_linha - offset):

carros += 1

# cv2.line(frame, (25, pos\_linha), (1200, pos\_linha), (0, 127, 255), 3)

detec.remove((x, y))

# print("car is detected : " + str(carros))

frame2 = cv2.flip(frame, 1)

text = pytesseract.image\_to\_string(frame)

text2 = pytesseract.image\_to\_string(frame2)

print(text2)

print(text)

text = text.upper()

display = ""

if "POLICE" in text:

display = "POLICE VEHICLE DETECTED"

al.detectAlarm()

if "FIRE" in text:

display = display + "FIRE ENGINE VEHICLE DETECTED"

al.detectAlarm()

if "AMBULANCE" in text:

display = display + "FIRE ENGINE VEHICLE DETECTED"

al.detectAlarm()

if "AMBULANCE" in text2:

display = display + "AMBULANCE VEHICLE DETECTED "

al.detectAlarm()

if "FIRE" in text2:

display = display + "FIRE ENGINE VEHICLE DETECTED "

al.detectAlarm()

if len(display) == 0:

display = "NO EMERGENCY VEHICLE DETECTED"

font = cv2.FONT\_HERSHEY\_SIMPLEX

org = (50, 50)

fontScale = 1

color = (255, 0, 0)

thickness = 2

image = cv2.putText(frame, display, org, font,

fontScale, color, thickness, cv2.LINE\_AA)

cv2.imshow("hai", image)

k = cv2.waitKey(1)

if k % 256 == 27:

# ESC pressed

print("Escape hit, closing...")

break

elif k % 256 == 32:

# SPACE pressed

img\_name = "opencv\_frame\_{}.png".format(img\_counter)

cv2.imwrite('uploads/' + img\_name, frame)

print("{} written!".format(img\_name))

img\_counter += 1

capturedImage = img\_name

break

cam.release()

cv2.destroyAllWindows()

if capturedImage == None:

return render\_template('umsg.html', msg='Image not Captured', color='bg-success')

img = cv2.imread('uploads/' + capturedImage)

text = pytesseract.image\_to\_string(img)

Thread(target=say\_text, args=(text,)).start()

return render\_template('fimage1.html', filename=capturedImage, emotion=text)

@app.route('/VehicleCount1',methods=['POST'])

def VehicleCount1():

target = os.path.join(APP\_ROOT, 'uploads/')

destination = ''

filename = ''

for upload in request.files.getlist("file"):

filename = upload.filename

destination = "/".join([target, filename])

upload.save(destination)

print('uploads/' + filename)

cam = cv2.VideoCapture('uploads/' + filename)

cv2.namedWindow("Character Recognition")

img\_counter = 0

capturedImage = None

largura\_min = 80 # Largura minima do retangulo

altura\_min = 80 # Altura minima do retangulo

offset = 6 # Erro permitido entre pixel

pos\_linha = 550 # Posição da linha de contagem

delay = 60 # FPS do vídeo

detec = []

carros = 0

subtracao = cv2.bgsegm.createBackgroundSubtractorMOG()

while True:

ret, frame1 = cam.read()

tempo = float(1 / delay)

sleep(tempo)

grey = cv2.cvtColor(frame1, cv2.COLOR\_BGR2GRAY)

blur = cv2.GaussianBlur(grey, (3, 3), 5)

img\_sub = subtracao.apply(blur)

dilat = cv2.dilate(img\_sub, np.ones((5, 5)))

kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (5, 5))

dilatada = cv2.morphologyEx(dilat, cv2.MORPH\_CLOSE, kernel)

dilatada = cv2.morphologyEx(dilatada, cv2.MORPH\_CLOSE, kernel)

contorno, h = cv2.findContours(dilatada, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

cv2.line(frame1, (25, pos\_linha), (1200, pos\_linha), (255, 127, 0), 3)

for (i, c) in enumerate(contorno):

(x, y, w, h) = cv2.boundingRect(c)

validar\_contorno = (w >= largura\_min) and (h >= altura\_min)

if not validar\_contorno:

continue

cv2.rectangle(frame1, (x, y), (x + w, y + h), (0, 255, 0), 2)

centro = pega\_centro(x, y, w, h)

detec.append(centro)

cv2.circle(frame1, centro, 4, (0, 0, 255), -1)

for (x, y) in detec:

if y < (pos\_linha + offset) and y > (pos\_linha - offset):

carros += 1

# cv2.line(frame1, (25, pos\_linha), (1200, pos\_linha), (0, 127, 255), 3)

detec.remove((x, y))

# print("car is detected : " + str(carros))

cv2.putText(frame1, "VEHICLE COUNT : " + str(carros), (450, 70), cv2.FONT\_HERSHEY\_SIMPLEX, 2, (0, 0, 255), 5)

cv2.imshow("Video Original", frame1)

cv2.imshow("Detectar", dilatada)

k = cv2.waitKey(1)

if k % 256 == 27:

# ESC pressed

print("Escape hit, closing...")

break

elif k % 256 == 32:

# SPACE pressed

img\_name = "opencv\_frame\_{}.png".format(img\_counter)

cv2.imwrite('uploads/' + img\_name, frame1)

print("{} written!".format(img\_name))

img\_counter += 1

capturedImage = img\_name

break

cam.release()

cv2.destroyAllWindows()

if capturedImage == None:

return render\_template('umsg.html', msg='Image not Captured', color='bg-success')

img = cv2.imread('uploads/' + capturedImage)

text = pytesseract.image\_to\_string(img)

Thread(target=say\_text, args=(text,)).start()

return render\_template('fimage1.html', filename=capturedImage, emotion=text)

@app.route('/flive1')

def flive1():

cam = cv2.VideoCapture(cameraNumber)

cv2.namedWindow("Character Recognition")

img\_counter = 0

capturedImage = None

while True:

ret, frame = cam.read()

if not ret:

print("failed to grab frame")

break

cv2.imshow("Character Recognition", frame)

k = cv2.waitKey(1)

if k % 256 == 27:

# ESC pressed

print("Escape hit, closing...")

break

elif k % 256 == 32:

# SPACE pressed

img\_name = "opencv\_frame\_{}.png".format(img\_counter)

cv2.imwrite('uploads/'+img\_name, frame)

print("{} written!".format(img\_name))

img\_counter += 1

capturedImage = img\_name

break

cam.release()

cv2.destroyAllWindows()

if capturedImage==None:

return render\_template('umsg.html', msg='Process Completed', color='bg-success')

img = cv2.imread('uploads/' + capturedImage)

text = pytesseract.image\_to\_string(img)

Thread(target=say\_text, args=(text,)).start()

return render\_template('fimage1.html', filename=capturedImage, emotion=text)

@app.route('/detectLive1')

def detectLive1():

al = alarm()

cam = cv2.VideoCapture(cameraNumber)

# cv2.namedWindow("Character Recognition")

img\_counter = 0

capturedImage = None

largura\_min = 80 # Largura minima do retangulo

altura\_min = 80 # Altura minima do retangulo

offset = 6 # Erro permitido entre pixel

pos\_linha = 550 # Posição da linha de contagem

delay = 60 # FPS do vídeo

detec = []

carros = 0

subtracao = cv2.bgsegm.createBackgroundSubtractorMOG()

while True:

ret, frame = cam.read()

if not ret:

print("failed to grab frame")

break

tempo = float(1 / delay)

sleep(tempo)

grey = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

blur = cv2.GaussianBlur(grey, (3, 3), 5)

img\_sub = subtracao.apply(blur)

dilat = cv2.dilate(img\_sub, np.ones((5, 5)))

kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (5, 5))

dilatada = cv2.morphologyEx(dilat, cv2.MORPH\_CLOSE, kernel)

dilatada = cv2.morphologyEx(dilatada, cv2.MORPH\_CLOSE, kernel)

contorno, h = cv2.findContours(dilatada, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

cv2.line(frame, (25, pos\_linha), (1200, pos\_linha), (255, 127, 0), 3)

for (i, c) in enumerate(contorno):

(x, y, w, h) = cv2.boundingRect(c)

validar\_contorno = (w >= largura\_min) and (h >= altura\_min)

if not validar\_contorno:

continue

cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

centro = pega\_centro(x, y, w, h)

detec.append(centro)

cv2.circle(frame, centro, 4, (0, 0, 255), -1)

for (x, y) in detec:

if y < (pos\_linha + offset) and y > (pos\_linha - offset):

carros += 1

cv2.line(frame, (25, pos\_linha), (1200, pos\_linha), (0, 127, 255), 3)

detec.remove((x, y))

# print("car is detected : " + str(carros))

frame2 = cv2.flip(frame, 1)

text = pytesseract.image\_to\_string(frame)

text2 = pytesseract.image\_to\_string(frame2)

print(text2)

print(text)

text = text.upper()

display = ""

if "POLICE" in text:

display = "POLICE VEHICLE DETECTED"

al.detectAlarm()

if "FIRE" in text:

display = display + "FIRE ENGINE VEHICLE DETECTED"

al.detectAlarm()

if "AMBULANCE" in text:

display = display + "FIRE ENGINE VEHICLE DETECTED"

al.detectAlarm()

if "AMBULANCE" in text2:

display = display + "AMBULANCE VEHICLE DETECTED "

al.detectAlarm()

if len(display) == 0:

display = "NO EMERGENCY VEHICLE DETECTED"

font = cv2.FONT\_HERSHEY\_SIMPLEX

org = (50, 50)

fontScale = 1

color = (255, 0, 0)

thickness = 2

image = cv2.putText(frame, display, org, font,

fontScale, color, thickness, cv2.LINE\_AA)

cv2.imshow("hai", image)

k = cv2.waitKey(1)

if k % 256 == 27:

# ESC pressed

print("Escape hit, closing...")

break

elif k % 256 == 32:

# SPACE pressed

img\_name = "opencv\_frame\_{}.png".format(img\_counter)

cv2.imwrite('uploads/'+img\_name, frame)

print("{} written!".format(img\_name))

img\_counter += 1

capturedImage = img\_name

break

cam.release()

cv2.destroyAllWindows()

if capturedImage==None:

return render\_template('umsg.html', msg='Process Completed', color='bg-success')

img = cv2.imread('uploads/' + capturedImage)

text = pytesseract.image\_to\_string(img)

Thread(target=say\_text, args=(text,)).start()

return render\_template('fimage1.html', filename=capturedImage, emotion=text)

@app.route('/uploads/<filename>')

def send\_image(filename):

return send\_from\_directory("uploads", filename)

@app.route('/images/<filename>')

def send\_image2(filename):

return send\_from\_directory("images", filename)

def say\_text(text):

speak(text)

if \_name\_ == '\_main\_':

app.run(debug=True)

pass

**alaram.py:**

from threading import Thread

import playsound

class alarm:

def sound\_alarm(self, path):

playsound.playsound(path)

def detectAlarm(self):

args = {'alarm': 'alarm.wav'}

t = Thread(target=self.sound\_alarm,args=(args["alarm"],))

t.deamon = True

t.start()

**Count.py:**

import cv2

import numpy as np

from time import sleep

largura\_min=80 #Largura minima do retangulo

altura\_min=80 #Altura minima do retangulo

offset=6 #Erro permitido entre pixel

pos\_linha=550 #Posição da linha de contagem

delay= 60 #FPS do vídeo

detec = []

carros= 0

def pega\_centro(x, y, w, h):

x1 = int(w / 2)

y1 = int(h / 2)

cx = x + x1

cy = y + y1

return cx,cy

cap = cv2.VideoCapture("./videos/video.mp4")

# subtracao = cv2.bgsegm.createBackgroundSubtractorMOG()

subtracao = cv2.bgsegm.createBackgroundSubtractorMOG()

while True:

ret , frame1 = cap.read()

tempo = float(1/delay)

sleep(tempo)

grey = cv2.cvtColor(frame1,cv2.COLOR\_BGR2GRAY)

blur = cv2.GaussianBlur(grey,(3,3),5)

img\_sub = subtracao.apply(blur)

dilat = cv2.dilate(img\_sub,np.ones((5,5)))

kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (5, 5))

dilatada = cv2.morphologyEx (dilat, cv2. MORPH\_CLOSE , kernel)

dilatada = cv2.morphologyEx (dilatada, cv2. MORPH\_CLOSE , kernel)

contorno,h=cv2.findContours(dilatada,cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_SIMPLE)

cv2.line(frame1, (25, pos\_linha), (1200, pos\_linha), (255,127,0), 3)

for(i,c) in enumerate(contorno):

(x,y,w,h) = cv2.boundingRect(c)

validar\_contorno = (w >= largura\_min) and (h >= altura\_min)

if not validar\_contorno:

continue

cv2.rectangle(frame1,(x,y),(x+w,y+h),(0,255,0),2)

centro = pega\_centro(x, y, w, h)

detec.append(centro)

cv2.circle(frame1, centro, 4, (0, 0,255), -1)

for (x,y) in detec:

if y<(pos\_linha+offset) and y>(pos\_linha-offset):

carros+=1

cv2.line(frame1, (25, pos\_linha), (1200, pos\_linha), (0,127,255), 3)

detec.remove((x,y))

print("car is detected : "+str(carros))

cv2.putText(frame1, "VEHICLE COUNT : "+str(carros), (450, 70), cv2.FONT\_HERSHEY\_SIMPLEX, 2, (0, 0, 255),5)

cv2.imshow("Video Original" , frame1)

cv2.imshow("Detectar",dilatada)

if cv2.waitKey(1) == 27:

break

cv2.destroyAllWindows()

cap.release()

**Character Recognition.py:**

import pytesseract

import cv2

from text\_to\_speech import speak

image = cv2.imread('sampleImages/4.PNG')

pytesseract.pytesseract.tesseract\_cmd = r'C:/Program Files/Tesseract-OCR/tesseract.exe'

text = pytesseract.image\_to\_string(image)

print(text)

speak(text)

**Record Video.py:**

import numpy as np

import cv2

import datetime

def recordVideo():

cap = cv2.VideoCapture(0)

fourcc = cv2.VideoWriter\_fourcc(\*'XVID')

r = datetime.datetime.utcnow().strftime('%Y%m%d%H%M%S%f')[:-3]

fpath = "videos/" + r + '.avi'

out = cv2.VideoWriter(fpath, fourcc, 20.0, (640, 480))

while (True):

ret, frame = cap.read()

hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

cv2.imshow('Video Recording', frame)

out.write(frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

out.release()

cv2.destroyAllWindows()

**4.2 SYSTEM STUDY**

**FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### **TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

### **4.3 SYSTEM TESTING**

Software testing is a process of executing a program or application with the intent

of finding the software bugs. It can also be stated as the process of validating and

verifying that a software program or application or product: Meets the business

and technical requirements that guided its design and development

* TESTING OBJECTIVES
* To ensure that during operation the system will perform as per specification.
* To make sure that system meets the user requirements during operation.
* To make sure that during the operation, incorrect input, processing and output will be detected.
* To see that when correct inputs are fed to the system the outputs are correct
* To verify that the controls incorporated in the same system as intended.
* Testing is a process of executing a program with the intent of finding an error.
* A good test case is one that has a high probability of finding an

undiscovered error.

The software developed has been tested successfully using the following

testing strategies and any errors that are encountered are corrected and again the

part of the program or the procedure or function is put to testing until all the errors

are removed.

Note that the result of the system testing will prove that the system is

working correctly. It will give confidence to system designer, users of the system,

prevent frustration during implementation process etc.,

**VALIDATION:**

From various types of testing, we need very few methods of testing for

database validation. Those are listed below.

Interfaces consistency should be validated in order to guarantee that

applications have a stable structure for data access. This testing is required for the

mobile store management system application to ensure we have a properly

constructed database and are concentrated properly.

Data availability and authorization tests are similar to interface

consistency tests, but are more focused on who can get data from the database

than how the data should be retrieved. This testing is also required for the

application in order to assure data connectivity.

**TEST CASES**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl.no** | **Test case**  **Name** | **Test Procedure** | **Pre Condition** | **Post Condition** | **Status** |
| 1 | User login case | Is user provided true username and password | Redirect to next screen | Redirect to next screen | Success |
| 2 | Invalid username and password case | Is username and password wrong then | Please enter correct username and password | Please enter correct username and password | Fail |
| 3 | Add Contact Case | If null enter in mandatory fields | Please fill out the field | Please fill out the field | Fail |
|  | If the data entered is correct | Added successfully | If the data entered is correct | If the data entered is correct | Success |
| 4 | View Contact screen  case | Is represented page available | Successfully view that screen | Successfully view that screen | Success |
| 5 | Contact add | If null enter in mandatory fields | Contact added successful | Contact sent successful | Success |
| 6 | login case | Is user provided true username and password | Redirect to next screen | Redirect to next screen | success |
| 7 | Invalid username and password case | Is username and password wrong then | Please enter correct username and password | Please enter correct username and password | Fail |

**4.4 FUTURE ENHANCEMENTS:**

**CHAPTER 5**

**RESULT:**

* Traffic congestion can be solved.
* Emergency vehicles can reach the destination earliest.
* Traffic density is continuously monitor by video processing and converted into frames.
* The frames are analyzed by various techniques of image processing they are segmented

for distinguish between ambulance and other vehicles.

* Traffic signals continuously glow to green as long as emergency vehicle is passed

through the traffic and it is allowed to reach its destination.

A picture containing text, screenshot, electronics, display

Description automatically generated

Graphical user interface, website

Description automatically generated

A picture containing text, wall

Description automatically generated

A picture containing text, transport, van, car

Description automatically generated

A picture containing text, way, scene, road

Description automatically generated

**.**

**CHAPTER 6 : CONCLUSION**

This paper proposes an efficient library for emergency vehicle detection based Open CV, an open source computer vision library in combination with Pytesseract for character recognition. This method has the following characteristics: it is  very simple computation, and easy to implement. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others

By this project the problem of traffic can be easily sorted out: the timing of each signal

can be automatically adjusted according to density of traffic which is real time operation. It will

also clear the path for the ambulance, fire brigade in emergency cases and also it will help to

public in taking decisions for reaching their destination in time using auto-routing method. It

shows that it can reduce the traffic congestion and avoids the time being wasted by a green

light on an empty road. It is also more consistent in detecting vehicle presence because it uses

actual traffic images. It visualizes the reality so it functions much better than those systems that

rely on the detection of the vehicles metal content. Overall, the system is good but it still needs

improvement to achieve a hundred percent accuracy.

**CHAPTER 7**

**REFERENCE :**

[1] V. Paruchuri, S. Chellappan, and R. B. Lenin, ‘‘Arrival time based traffic signal optimization for intelligent transportation systems,’’ in Proc. AINA, Barcelona, Spain, Mar. 2013, pp. 703–709.

[2] K. Nellore and G. Hancke, ‘‘Traffic management for emergency vehicle priority based on visual sensing,’’ Sensors, vol. 16, no. 11, p. 1892, Oct. 2016.

[3] S. Djahel, N. Smith, S. Wang, and J. Murphy, ‘‘Reducing emergency services response time in smart cities: An advanced adaptive and fuzzy approach,’’ in Proc. ISC2, Guadalajara, Mexico, Oct. 2015, pp. 1–8.

[4] I. Arnold, Statistics on Emergency Vehicle Accidents in the United States. Accessed: Apr. 15, 2018. [Online]. Available: https://www. arnolditkin.com/personal-injury-blog/2015/november/statistics-onemergency-vehicle-accidents-in-the/

[5] K. Nellore and G. P. Hancke, ‘‘A survey on urban traffic management system using wireless sensor networks,’’ Sensors, vol. 16, no. 2, p. 157, Jan. 2016.

[6] R. Sundar, S. Hebbar, and V. Golla, ‘‘Implementing intelligent traffic control system for congestion control, ambulance clearance, and stolen vehicle detection,’’ IEEE Sensors J., vol. 15, no. 2, pp. 1109–1113, Feb. 2015.

[7] E. Sireesha and D. Rakesh, ‘‘Intelligent traffic light system to prioritized emergency purpose vehicles based on wireless sensor network,’’ Int. J. Comput. Appl., vol. 40, no. 12, pp. 24–27, 2012.

[8] K. R. Shruthi, and K. Vinodha, ‘‘Priority based traffic lights controller using wireless sensor networks,’’ Int. J. Electron. Signal Syst, vol. 1, no. 4, pp. 58–61, 2012.

[9] R. Hussian, S. Sharma, V. Sharma, and S. Sharma, ‘‘WSN applications: Automated intelligent traffic control system using sensors,’’ Int. J. Soft Comput. Eng, vol. 3, bo. 3, pp. 77–81, 2013.

[10] N. Kapileswar and M. S. Babu, ‘‘Automatic traffic monitoring system using lane centre edges,’’ IOSR-JEE, vol. 2, pp. 1–8, Aug. 2012.

[11] F. Shaikh and M. B. Chandak, ‘‘An approach towards traffic management system using density calculation and emergency vehicle alert,’’ IOSR J. Comput. Sci, vol. 4, no. 5, pp. 24–27, 2014. [12] R. Bharadwaj, J. Deepak, M. Baranitharan, and V. V. Vaidehi, ‘‘Efficient dynamic traffic control system using wireless sensor networks,’’ in Proc. IEEE ICRTIT, Chennai, India, Jul. 2013, pp. 668–673.

[13] B. Silva and G. P. Hancke, ‘‘IR-UWB-based non-line-of-sight identification in harsh environments: Principles and challenges,’’ IEEE Trans. Ind. Inf., vol. 12, no. 3, pp. 1188–1195, Jun. 2016.

14] A. M. Abu-Mahfouz and G. P. Hancke, ‘‘An efficient distributed localisation algorithm for wireless sensor networks: Based on smart referenceselection method,’’ Int. J. Sens. Netw, vol. 13, no. 2, pp. 94–111, 2013.

[15] N. Al-Ostath, F. Selityn, Z. Al-Roudhan, and M. El-Abd, ‘‘Implementation of an emergency vehicle to traffic lights communication system,’’ in Proc. NTMS, Paris, France, Jul. 2015, pp. 1–5. [16] A. K. Mittal and D. Bhandari, ‘‘A novel approach to implement green wave system and detection of stolen vehicles,’’ in Proc. IACC, G

**APPENDIX-1**