Smart Traffic Signal Management on Detection of Emergency Vehicles Using IoT

[1]DEEPTHI P, [2]KILARI DHANUSH, [3]MADAN T,[4]MADHUSMITHA NAYAK
[5]JAGADEVI BAKKA

[1,2,3,4]Student, [5]Assistant Professor
Department of Computer Science
East Point College of Engineering and Technology
Bengaluru.

Abstract: **One of the unavoidable issues of contemporary societies is traffic, which results in unfavorable outcomes like impeding emergency vehicles' access to destinations, wasting time, and increasing the likelihood of accidents. The utilization of Internet of Things(IoT) in managing traffic signals can greatly enhance the flow of traffic and alleviate road congestion. A new method is suggested in thisarticle to identify emergency vehicles and control traffic signals accordingly. To control traffic lights more efficiently, the use of IoT approaches is implemented where sensors such as surveillance cameras capture real-time traffic information for the smart traffic signal management system. To guarantee that emergency vehicles reach their destination with minimal delays, the system is created to provide a clear path.**

Index terms: **Smart traffic signal management, Internet of Things (IoT), Emergency vehicles, Sensors.**

I. INTRODUCTION

Traffic flow can be improved, congestion reduced, and the safety of motorists and pedestrians on roads enhanced by smart traffic signal management systems. Efficient traffic management systems have become more crucial due to the increasing number of vehicles on the road.

Road congestion continues to be a significant problem for everyone, despite extensive research in the field of traffic management. The continuous and rapid increase in the number of vehicles exceeds the available traffic infrastructure, which is the main reason behind this. [1]. In every life-threatening situation, emergency vehicles have a crucial role to play. When a patient's condition is very serious, the percentage of patient deaths increases due to traffic jams, which can take up more than 20% of their lives in an ambulance [2]. Emergency vehicles are not being taken into consideration. The lack of crisis measures in traffic signal control frameworks leads to congested intersections and unnecessary financial losses [3]. To prevent accidents, the smart traffic light's main objective is to assign the right of way alternately. Smart traffic lights aim to reduce traffic delay and fuel consumption while increasing the capacity for passing cars and traffic flow, as well as prioritizing emergency vehicles over time [4].

Introducing a smart traffic signal management integrated with a traffic control system that can detect and give priority to emergency carscan reduce these problems. A model was constructed in this paper to identify cars and distinguish between emergency and regular vehicles. The signal changes upon recognition of the emergency vehicle. The traffic signals will eventually be changed to make way for emergency vehicles.

II. SYSTEM ARCHITECTURE

1. MODEL ARCHITECTURE

The figure 2.1 represents the architecture of Smart Traffic Signal Management. The model begins with the pre-processing were the system takes the image as input. The pre-processed image is then resized in form of a grid, the camera connected with the raspberry pi detects the vehicle is a emergency vehicle or not. If the image is a Emergency vehicle the model detects it with the help of camera and the signal changes from red to yellow and then green.

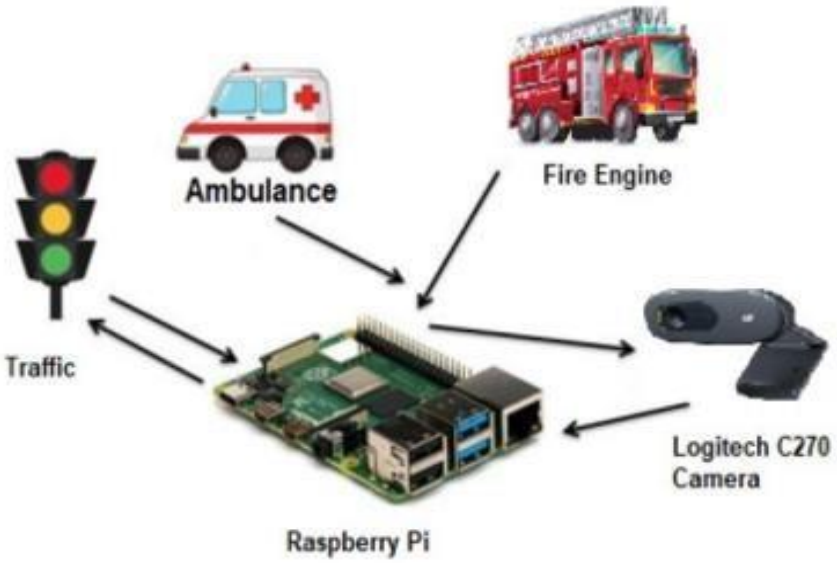

Figure 2.1 Model Architecture

2. BLOCK DIAGRAM

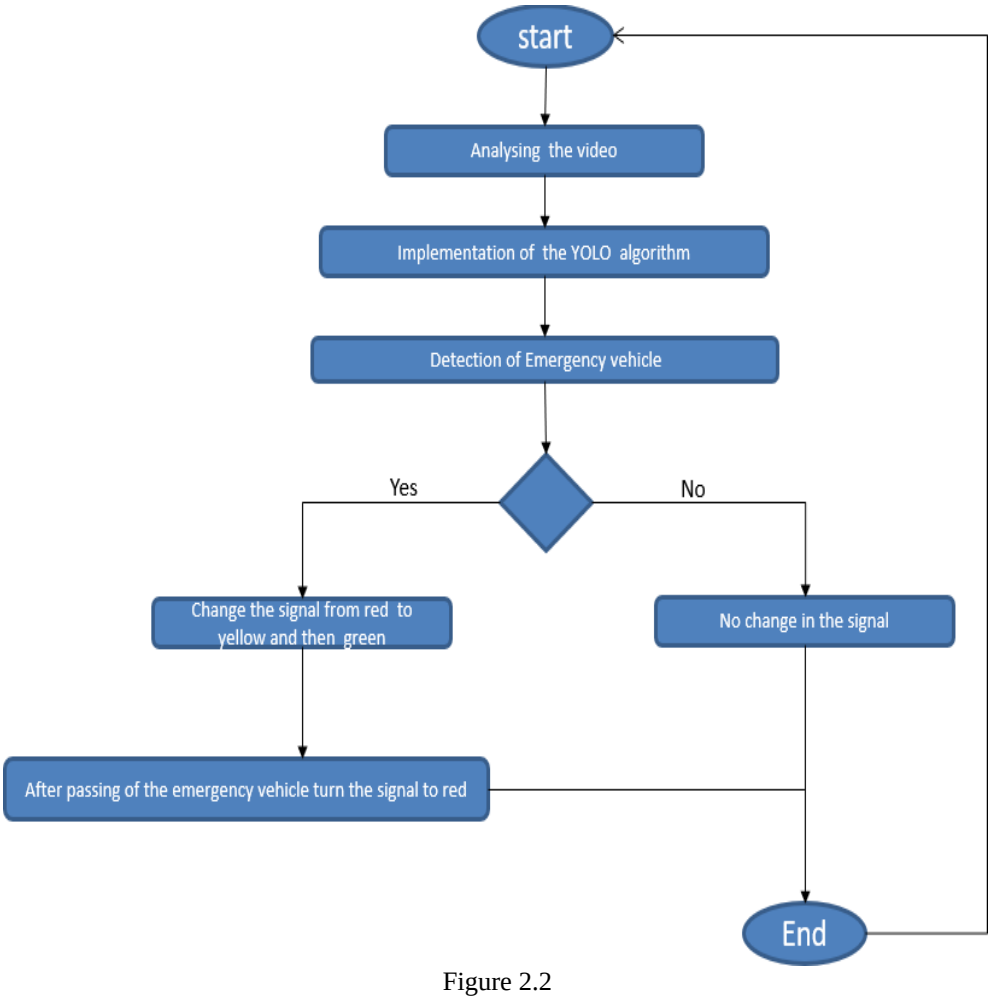

Figure 2.2

The figure 2.2 represents the block diagram of the Smart Traffic Signal Management. This begins with the analysing of theimage and then the image is implemented in the YOLO and the object is present in the grid. It detects whether the image isemergency vehicle. If it is an emergency vehicle the signal changes or else the signal change will be as usual.

III. IMPLEMENTATION

This shows the code that is implemented to identify the emergency vehicles and the signal changes that happen when an emergency vehicle is detected with time lap.

```
import cv2
import numpy as np

name = 'okam.jpg'
template = cv2.imread(name,0)
face_w, face_h = template.shape[::-1]

cv2.namedWindow('image')

cap = cv2.VideoCapture(0)

threshold = 1
ret = True

while ret :
    ret, img = cap.read()

    #flip the image  ! optional
    img = cv2.flip(img,1)

    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    res = cv2.matchTemplate(img_gray,template,cv2.TM_CCOEFF_NORMED)

    if len(res):
        location = np.where( res >= threshold)
        for i in zip(*location[::-1]):
            #puting  rectangle on recognized erea
            cv2.rectangle(img, pt, (pt[0] + face_w, pt[1] + face_h), (0,0,255), 2)

    cv2.imshow('image',img)
```

```
# import the necessary packages
import argparse
import numpy as np
import time
import cv2
# construct the argument parser and parse the arguments
##ap = argparse.ArgumentParser()
##ap.add_argument("-i", "--image", type=str, required=True,
##  help="path to input image where we'll apply template matching")
##ap.add_argument("-t", "--template", type=str, required=True,
##  help="path to template image")
##args = vars(ap.parse_args())
### load the input image and template image from disk, then display
### them on our screen

cap = cv2.VideoCapture(0)
threshold = 1
ret = True
template = cv2.imread("ok.jpg")

template1 = cv2.imread("fra.jpg")
while True:
    time.sleep(0.4)
    ret, image = cap.read()

    print("[INFO] loading images...")
    #image = cv2.imread(args["image"])

    cv2.imshow("Image", image)
    cv2.imshow("Template", template)
```

```
# convert both the image and template to grayscale
imageGray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
templateGray = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)

template1Gray = cv2.cvtColor(template1, cv2.COLOR_BGR2GRAY)
# perform template matching
print("[INFO] performing template matching...")
result = cv2.matchTemplate(imageGray, templateGray,
        cv2.TM_CCOEFF_NORMED)
(minVal, maxVal, minLoc, maxLoc) = cv2.minMaxLoc(result)

result1 = cv2.matchTemplate(imageGray, template1Gray,
        cv2.TM_CCOEFF_NORMED)
(minVal1, maxVal1, minLoc1, maxLoc1) = cv2.minMaxLoc(result1)
# determine the starting and ending (x, y)-coordinates of the
# bounding box
(startX, startY) = maxLoc

(startX1, startY1) = maxLoc1
print(maxVal)
print(maxVal1)
if maxVal > 0.4:
    print("Ambu detcted")

if maxVal1 > 0.27:
    print("FE Detected")
endX = startX + template.shape[1]
endY = startY + template.shape[0]

endX1 = startX + template1.shape[1]
endY1 = startY + template1.shape[0]
```

```
# draw the bounding box on the image
cv2.rectangle(image, (startX, startY), (endX, endY), (255, 0, 0), 3)
cv2.rectangle(image, (startX1, startY1), (endX1, endY1), (255, 0, 0), 5)
# show the output image
cv2.imshow("Output", image)
k = cv2.waitKey(5) & 0xFF
if k == 27:
    break
cv2.destroyAllWindows()
```

Figure 3: Code for Detection of Emergency Vehicles

IV. METHODOLOGY

1. **Collecting image data:** The Raspberry Pi is connected to a camera to capture images of the road or intersection where the emergency vehicle detection system will be installed. To give input to the model, it is used.
2. **Pre-processing the data:** The data is pre-processed by adjusting for lighting conditions and enhancing features relevant to detecting emergency vehicles, as well as processing the model efficiently**.**
3. **Object detection:** To identify potential emergency vehicles in the scene, object detection algorithms analyze the pre- processed images. Object classification is commonly achieved through the use of machine learning models, such as convolutional neural networks (CNNs).
4. **Train a model**: Train an object detection model on the dataset to detect vehicles and emergency vehicles.
5. **Deploy the Model:** Once the model is trained, deploy it in the raspberry pi to detect emergency vehicles. Label the dataset and distinguish between cars, buses, trucks, and emergency vehicles.
6. **Tracking and alerting**: The system can track the movement of an emergency vehicle through the scene and change the signal from red to yellow and then green for alerting.

V. RESULTS

In this study, the model is constructed for detecting the emergency vehicle and the signal is changed when the emergency vehicle is detected in the system.

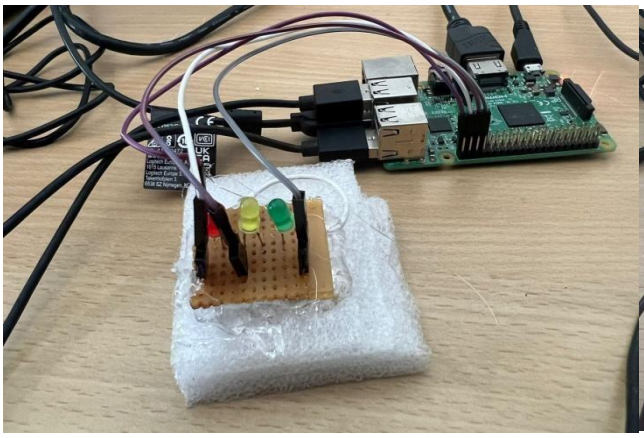

Figure 4.1.1 Model setup of the signal

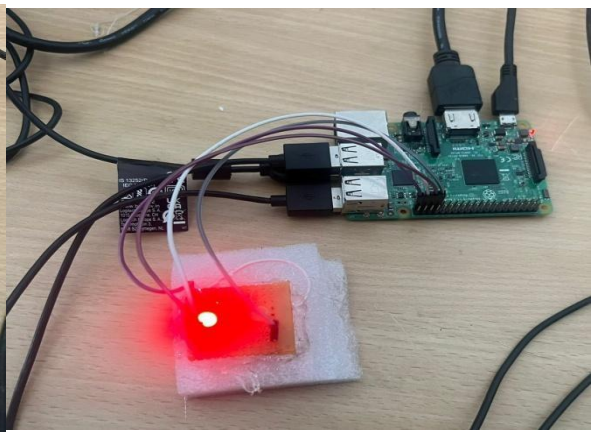

Figure 4.1.2 No Detection of Emergency vehicle

Figure 4.1.3 Basic change of the Signal

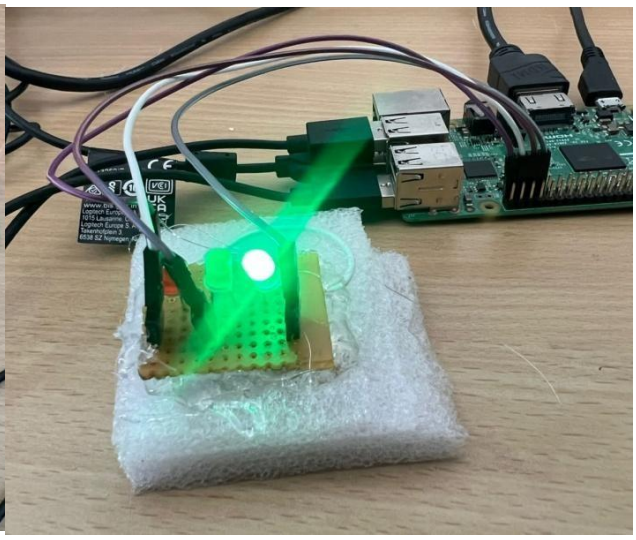

Figure 4.1.4 Detection of Emergency vehicle

VI. CONCLUSION

Recognizing Emergency vehicles and making way for them to travel swiftly and smoothly was more pronounced in importance with the higher number of vehicles during peak hours. This project analyzes the techniques and technologies employed in smart traffic signal management to decrease traffic congestion and identify emergency vehicles.

Our plan for future work is to enhance the camera by using a high-quality one with a wide field of view. Additionally, we aim to increase the system's reliability by incorporating audio detectors that can identify the sound of emergency vehicles.

REFERENCES:

1. Asma AitOuallane, “A Overview of Road Traffic Management Solutions based on IOT and AI,” Workshop of Innovation andTechnologies (IWIT 2021).
2. Shuvendu Roy, “Emergency Vehicle Detection on Heavy Traffic Road: 2019 International Conference on ECCE,7-9 February,2019
3. Moolchand Sharma et al 2021 IOP conf.Ser.:Mater. Sci. Eng. 1022012122.
4. H. M. Almawgani International Journal of Industrial Electronics Engineering, ISSN(P)-2347-6982,Apr-2018, http://ijeee.org.in.
5. Dia, H. (2002). An agent-based approach to modelling driver route choice behavior under the influence of real-time information. Transportation Research Part C: Emerging Technologies, 10(56), pp. 331–349. Doi:10.1016/s0968-090x(02)00025-6.
6. GAO, F., & WANG, M. (2010). Route Choice Behavior Model with Guidance Information. Journal of Transportation SystemsEngineering and Information Technology, 10(6), pp. 64–69. Doi:10.1016/s1570-6672(09)60072-6