Internet of Things (IoT)

Project-phase-2

IoT based Smart parking system

- N.Udhayanithi

-Parisutham Institute of Technology and Science

INTRODUCTION:

# We use python code to connect the raspberry pi with camera and also, going to do a python code for camera detecting a vehicle in parking slots. Here, we can check the empty space for park a vehicle. This code detects the movement or presence of an object in each slot. Using OpenCV-python we include those function used to detect the object and we include YoloV5 algorithm to run the code in python. Let's discuss about the works involves in to detect the object(vehicle(i.e)cars) in a parking slot at particular area we need.

ALOGORITHM FOR VEHICLE DETECTION:

# The vehicles are detected using the YOLOv3 object detection algorithm . this algorithm can improve the small object detection effect and solve the problem that the object is difficult to detect due to sharp change of the object scale .

PARKING SPACE DETECTION CAMERA:

# Parking Detection Camera is a smart parking solution that uses cutting-edge guesswork the system will notify you when someone leaves or enters a certain parking place, as well as how long they remain there. This technology is simple to operate and very dependable. The camera captures the parking image/video,

and the parking floor plan is formed based on it. As a result, the user enjoys continuous video streaming of the monitored parking space.

# Detect cameras are designed for indoor/outdoor use and are resistant to any heavy weather conditions**,** such as snow or rain and they can also work during the night.

HARDWARE SPECIFICATION:

- ✓ Size – around 25*24*9 mm
- ✓ Weight- 3 g
- ✓ Still resolution – 5 Megapixels
- ✓ Video modes – 1080p30,720p60 and 640 X 480P60/90

VEHICLE DETECTION BY CAMERA USING PYTHON CODE:

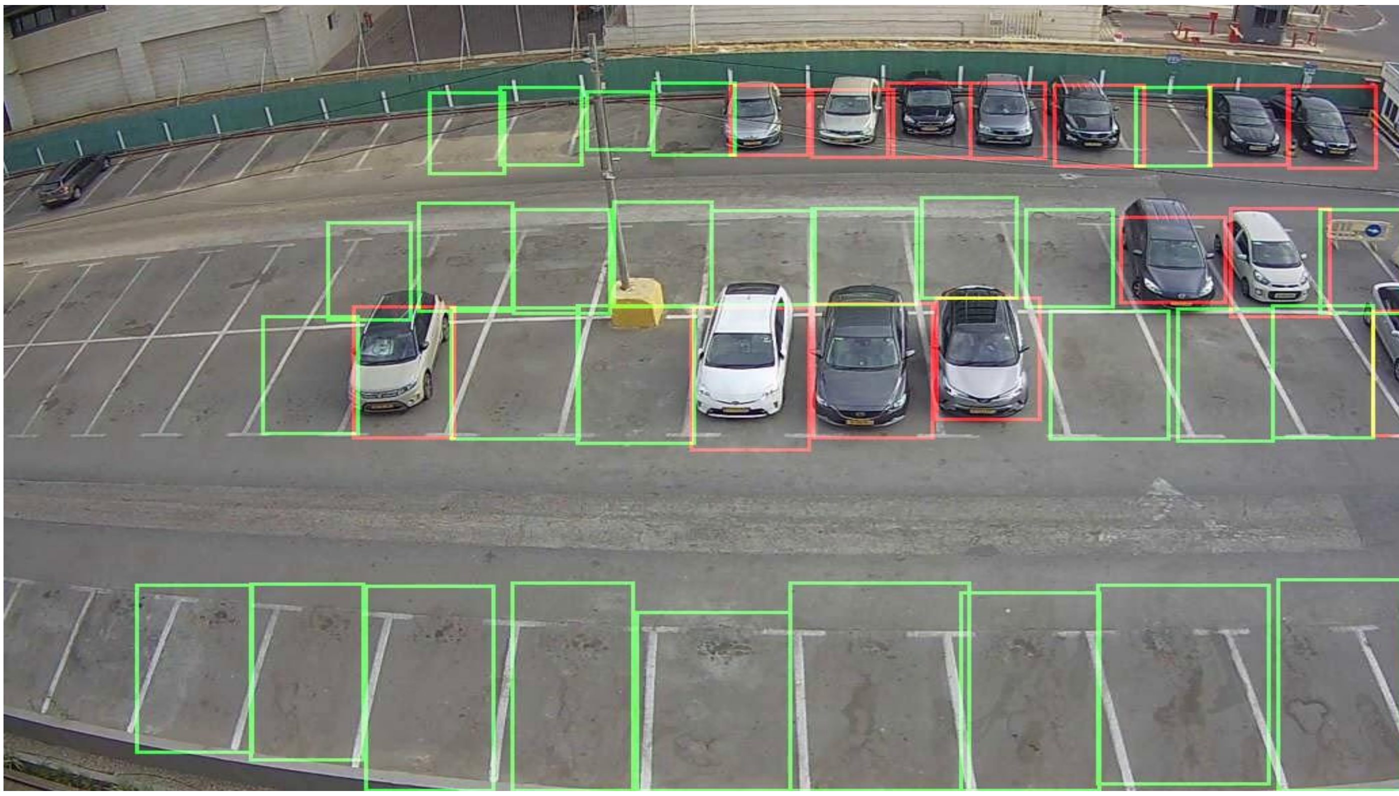
# Detecting vehicles in a smart parking system using a camera can be achieved with more advanced techniques like object detection using YOLO (You Only Look Once). Here's a Python code example using the opencv-python and darknet library for YOLO-based vehicle detection. We will need to install the required libraries and download the YOLO weights and configuration files.

PYTHON SOURCE CODE:

```
#Use the following Python code to perform vehicle detection
import cv2
import numpy as np

# Load YOLOv3 or YOLOv4 model
net = cv2.dnn.readNet('yolov3.weights', 'yolov3.cfg')

# Replace with your YOLO model files

# Load the COCO dataset class names
classes = []
with open('coco.names', 'r') as f:
    classes = f.read().strip().split('\n')

# Initialize the camera or video source (0 for webcam)
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Get the frame dimensions
    height, width, _ = frame.shape

    # Prepare the frame for YOLO detection
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True,
crop=False)
    net.setInput(blob)

    # Get output layer names
    layer_names = net.getUnconnectedOutLayersNames()

    # Run YOLO detection
    outs = net.forward(layer_names)

    # Initialize lists for detected objects' information
    class_ids = []
    confidences = []
    boxes = []

    # Analyze the output
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.5: # You can adjust the confidence threshold
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)

                # Rectangle coordinates
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)

                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)
    # Non-maximum suppression to remove duplicate detections
    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

    # Draw bounding boxes around vehicles
    for i in range(len(boxes)):
        if i in indexes:
            x, y, w, h = boxes[i]
            label = str(classes[class_ids[i]])
            confidence = confidences[i]
            color = (0, 255, 0) # Green
            cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
            cv2.putText(frame, label, (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

    # Display the frame with vehicle detection
    cv2.imshow('Vehicle Detection', frame)

    if cv2.waitKey(1) & 0xFF == 27:
# Press 'Esc' to exit
        break
cap.release()
cv2.destroyAllWindows()
```

Replace 'yolov3.weights', 'yolov3.cfg', and 'coco.names' with the paths to your downloaded YOLO model files and class names. This code captures video from a camera (you can change the source as needed) and detects vehicles using YOLOv3. You can adjust confidence thresholds and customize the appearance of detected vehicles.

EXAMPLE OUTPUT:



CONCLUSION:

Thus, we did a program to identify the empty slots in the parking area.