BVRIT HYDERABAD College of Engineering for Women

Department of Information Technology

VEHICLE DETECTION USING YOLO ALGORITHM

Under the Guidance of

Guide Name:Mr.B.Srinivasulu

Designation:Assistant Professor

Team- 15

K.Sanjana Reddy(19WH1A1219)

M.Deepthi Sharvani (19WH1A1220)

R.Rajani (19WH1A1241)

K.Aarthi (19WH1A1258)

Contents

VISHNU
UNIVERSAL LEARNING

Abstract

- Vehicle detection in real-time is a challenging and important task. The existing real-time vehicle detection lacks accuracy and speed.

- Real-time systems must detect and locate vehicles during criminal activities like theft of vehicle and road traffic violations with high accuracy. We use YOLO(You Only Look Once Algorithm) to detect vehicles effectively in real-time.

VISHNU
UNIVERSAL LEARNING

Introduction

- Vehicle detection in real-time is mostly helpful in detect and locate the vehicles using theft of vehicles or in violating the traffic rules which should be one with higher frequency for which we use YOLO algorithm to detect vehicles.

- The network training includes vehicle-based classes datasets such as car, bus, and truck etc. Little YOLO-SPP network detects the vehicle in real-time with high accuracy regardless of video frame and weather conditions.

VISHNU
UNIVERSAL LEARNING

Problem Definition

- The aim is to build an automatic system that can accurately localise and track any vehicles that appear in aerial video frames.

- A background subtraction technique used to detect and extract features for vehicles in complex road scenes in traffic surveillance

VISHNU
UNIVERSAL LEARNING

Literature Survey

S.No	Title of the paper	Journal with Author	Analysis
1	Vehicle Detection and Tracking using YOLO and DeepSORT	Muhammad Azhad bin Zuraimi,Fadhlan Hafizhelmi Kamaru Zaman(2021)	To detect traffic management we have several traitional methods where y use the movement of a vehicle to distinct it from static background image.But later CNN is introduced which involves less pre-processing.
2	Object Detection Based on YOLO Network	Chengji Liu ,Yufan Tao,Jiawei Liang,Kai Li1 ,Yihang Chen(2018)	YOLO is the most popular framework for detecting objects which direct extracts candidate boxes from images and objects are detected through the entire image features.

Proposed System

- Real-time systems must detect and locate vehicles during criminal activites like theft of vehicles, traffic violation with high accuracy.
- The existing real-time vehicle detection lacks accuracy and speed.
- A YOLO network is proposed to detect vehicles effectively in real-time, which increases the speed and accuracy and vehicle detection.

VISHNU
UNIVERSAL LEARNING

Project Modules

- Drawing detection boxes
  - Extracting bounding box coordinates
  - Draw a bounding box and label on the frame
- Box in previous frames
  - Identifying if the current box was present in previous frames
- Vehicle count
  - Gives number of vehicles in the frame.

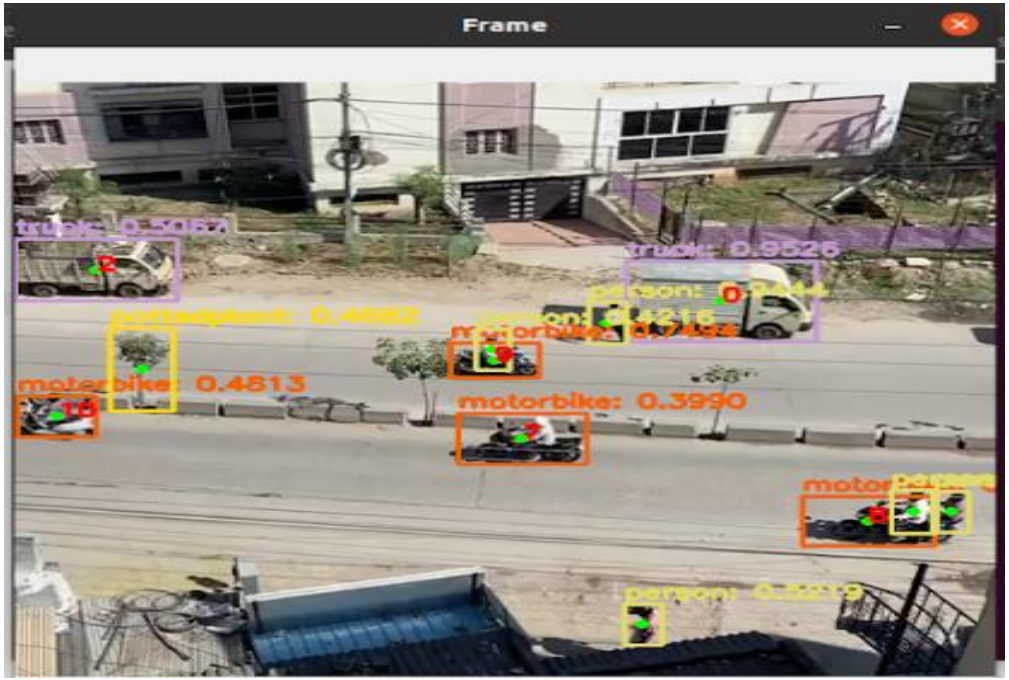
Implementation and Results

```
def drawDetectionBoxes(idxs, boxes, classIDs, confidences, frame):
        if len(idxs) > 0:
                for i in idxs.flatten():
                        # extract the bounding box coordinates
                        (x, y) = (boxes[i][0], boxes[i][1])
                        (w, h) = (boxes[i][2], boxes[i][3])

                        # draw a bounding box rectangle and label on the frame
                        color = [int(c) for c in COLORS[classIDs[i]]]
                        cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
                        text = "{}: {:.4f}".format(LABELS[classIDs[i]],
                                confidences[i])
                        cv2.putText(frame, text, (x, y - 5),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
                        cv2.circle(frame, (x + (w//2), y+ (h//2)), 2, (0, 0xFF, 0),
thickness=2)
```

Implementation and Results

```
def boxInPreviousFrames(previous_frame_detections, current_box, current_detections):
        centerX, centerY, width, height = current_box
        dist = np.inf
        for i in range(FRAMES_BEFORE_CURRENT):
                coordinate_list = list(previous_frame_detections[i].keys())
                if len(coordinate_list) == 0: # When there are no detections in the previous
frame
                        continue
                # Finding the distance to the closest point and the index
                temp_dist, index = spatial.KDTree(coordinate_list).query([(centerX, centerY)])
                if (temp_dist < dist):
                        dist = temp_dist
                        frame_num = i
                        coord = coordinate_list[index[0]]

        if (dist > (max(width, height)/2)):
                return False
        current_detections[(centerX, centerY)] = previous_frame_detections[frame_num][coord]
        return True
```

Implementation and Results

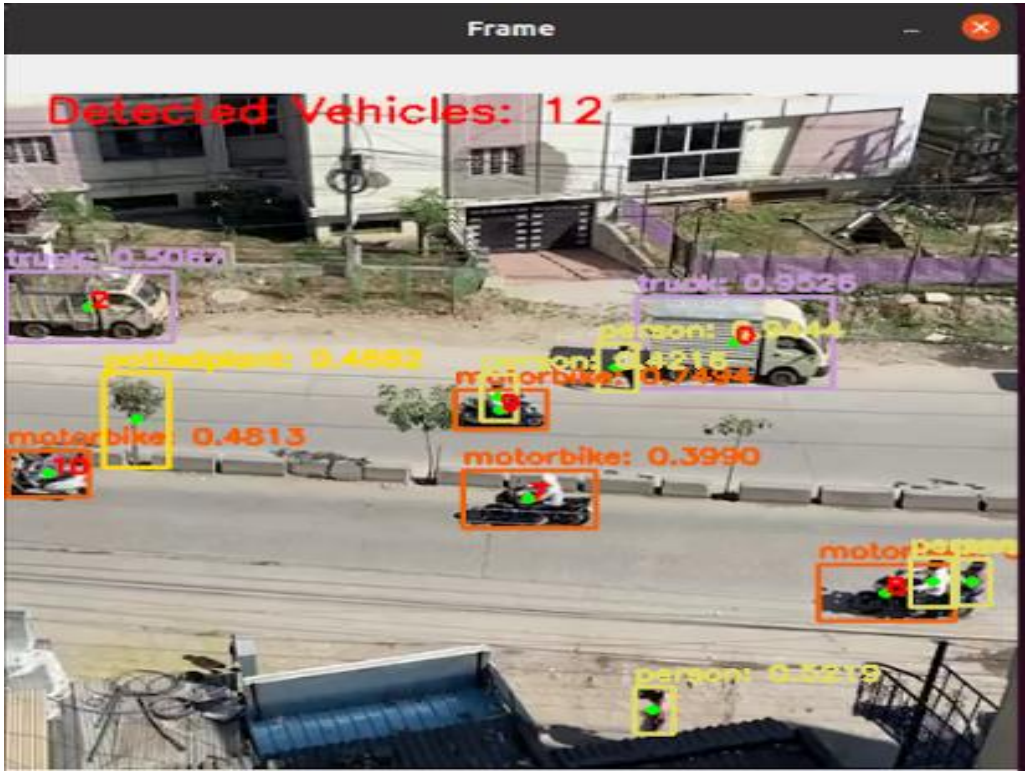


VISHNU
UNIVERSAL LEARNING

Implementation and Results

```
def count_vehicles(idxs, boxes, classIDs, vehicle_count, previous_frame_detections, frame):
        current_detections = {}
        if len(idxs) > 0:
                for i in idxs.flatten():
                        (x, y) = (boxes[i][0], boxes[i][1])
                        (w, h) = (boxes[i][2], boxes[i][3])

                        centerX = x + (w//2)
                        centerY = y+ (h//2)
                        if (LABELS[classIDs[i]] in list_of_vehicles):
                                current_detections[(centerX, centerY)] = vehicle_count
                                if (not boxInPreviousFrames(previous_frame_detections, (centerX, centerY, w, h), current_detections)):
                                        vehicle_count += 1

                                ID = current_detections.get((centerX, centerY))

                                if (list(current_detections.values()).count(ID) > 1):
                                        current_detections[(centerX, centerY)] = vehicle_count
                                        vehicle_count += 1

                                cv2.putText(frame, str(ID), (centerX, centerY),\
                                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, [0,0,255], 2)

        return vehicle_count, current_detections
```

Implementation and Results



VISHNU
UNIVERSAL LEARNING

Performance Measures

- We have used YOLO algorithm to detect vehicles.
- We calculate Confidence score, that represents how likely the box contains an object and how accurate is the boundary box . It ranges between 0-1.

VISHNU
UNIVERSAL LEARNING

Conclusion

When the vehicles are in the frame, they are being detected and also being counted. This can be combined with surveillance cameras and can be used in real-time, to monitor the vehicles passing by, and their count can be helpful to set the time of traffic dynamically.

References

1. Bin Zuraimi, M. A., & Kamaru Zaman, F. H. “Vehicle Detection and Tracking using YOLO and DeepSORT” IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE) 2021. doi:10.1109/iscaie51753.2021.9431784.
2. Maity, M., Banerjee, S., & Sinha Chaudhuri, S.”Faster R-CNN and YOLO based Vehicle detection: A Survey. 2021” 5th International Conference on Computing Methodologies and Communication (ICCMC) 2021. doi:10.1109/iccmc51019.2021.9418274.
3. Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, Bo Ma ,“A Review of Yolo Algorithm Developments”The 8th International Conference on Information Technology and Quantitative Management(2021)