Smart Traffic Management

* Here’s a Python code for vehicle detection from an image employs computer vision and machine learning techniques to identify and locate vehicles within a given picture.
* This code utilizes pre-trained models, such as YOLO (You Only Look Once) or Haar cascades, to recognize patterns and shapes characteristic of vehicles.
* By analyzing the image data, the Python script can accurately detect vehicles, which is a fundamental component in various applications, including traffic management, surveillance, and autonomous driving systems.
* Vehicle detection in images showcases the power of Python in the field of computer vision and contributes to enhancing safety and efficiency in transportation and security domains.

Python code:

import cv2

from darkflow.net.build import  TFNet

import matplotlib.pyplot as plt

import os

options={

   'model':'./cfg/yolo.cfg',        #specifying the path of model

   'load':'./bin/yolov2.weights',   #weights

   'threshold':0.3                  #minimum confidence factor to create a box, greater than 0.3 good

}

tfnet=TFNet(options)

inputPath = os.getcwd() + "/test\_images/"

outputPath = os.getcwd() + "/output\_images/"

*def* detectVehicles(*filename*):

   global tfnet, inputPath, outputPath

   img=cv2.imread(inputPath+filename,cv2.IMREAD\_COLOR)

   # img=cv2.cvtColor(img,cv2.COLOR\_BGR2RGB)

   result=tfnet.return\_predict(img)

   # print(result)

   for vehicle in result:

      label=vehicle['label']   #extracting label

      if(label=="car" or label=="bus" or label=="bike" or label=="truck" or label=="rickshaw"):    # drawing box and writing label

         top\_left=(vehicle['topleft']['x'],vehicle['topleft']['y'])

         bottom\_right=(vehicle['bottomright']['x'],vehicle['bottomright']['y'])

         img=cv2.rectangle(img,top\_left,bottom\_right,(0,255,0),3)    #green box of width 5

         img=cv2.putText(img,label,top\_left,cv2.FONT\_HERSHEY\_COMPLEX,0.5,(0,0,0),1)   #image, label, position, font, font scale, colour: black, line width

   outputFilename = outputPath + "output\_" +filename

   cv2.imwrite(outputFilename,img)

   print('Output image stored at:', outputFilename)

   # plt.imshow(img)

   # plt.show()

   # return result

for filename in os.listdir(inputPath):

   if(filename.endswith(".png") or filename.endswith(".jpg") or filename.endswith(".jpeg")):

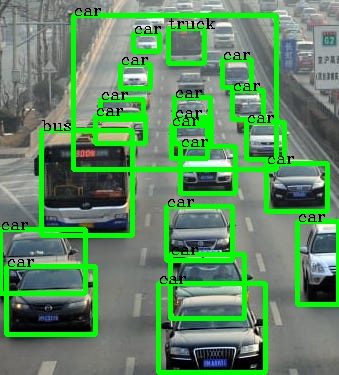
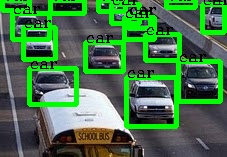
      detectVehicles(filename)

print("Done!")

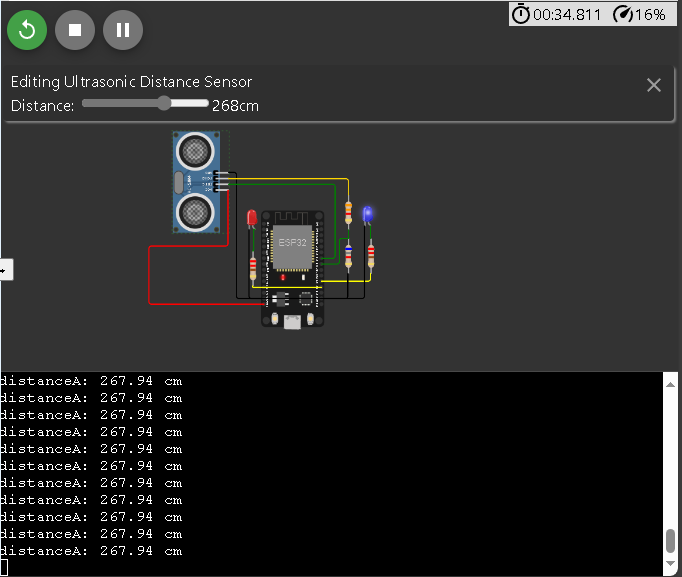
Input Image:

Output Image:

Simulation :



Code:

#define TRIG\_PIN 18 // ESP32 pin GIOP23 connected to Ultrasonic Sensor's TRIG

pin

#define ECHO\_PIN 5 // ESP32 pin GIOP22 connected to Ultrasonic Sensor's ECHO pin

#define LED 2

#define LED2 4

float duration\_us, distance\_cm;

void setup() {

  // begin serial port

**Serial**.begin (9600);

  // configure the trigger pin to output mode

  pinMode(TRIG\_PIN, OUTPUT);

  // configure the echo pin to input mode

  pinMode(ECHO\_PIN, INPUT);

  pinMode(LED, OUTPUT);

  pinMode(LED2, OUTPUT);

}

void loop() {

  // generate 10-microsecond pulse to TRIG pin

  digitalWrite(TRIG\_PIN, HIGH);

  delayMicroseconds(10);

  digitalWrite(TRIG\_PIN, LOW);

  // measure duration of pulse from ECHO pin

  duration\_us = pulseIn(ECHO\_PIN, HIGH);

  // calculate the distance

  distance\_cm = 0.017 \* duration\_us;

  // ligt up led if distance under limit

  if (distance\_cm < 30) {digitalWrite(LED, HIGH);

  }else{digitalWrite(LED, LOW);

  }

  // print the value to Serial Monitor

**Serial**.print("distanceA: ");

**Serial**.print(distance\_cm);

**Serial**.println(" cm      ");

  Blink(LED2);

  delay(500);

}

void Blink(int x){

  digitalWrite(x, HIGH);

  delay(10);

  digitalWrite(x, LOW);

}

Here’s the simulation for vehicle detection using an Ultrasonic sensor. Which detects the distance and changes the LED accordingly.