Aim: To Perform Detecting and Recognizing Objects/Text

Objective: Object Detection and recognition techniques HOG descriptor The Scale issues the location issue Non-maximum (or non-maxima) suppression vector machine people detection

Theory:

Object detection and recognition Techniques :-

Object recognition in computer vision identifies, locates, and classifies objects in images and real-life scenarios. It utilizes techniques like image recognition, localization, detection, and segmentation. By analyzing an object's features and applying a classifier, such as SVM or Decision Trees, it predicts the object's category. Object recognition algorithms are commonly implemented using frameworks like Darknet, written in C, Cuda, or Python.

HOG descriptors :-

HOG, a feature descriptor, is part of the same family as SIFT, SURF, and ORB, providing crucial information for feature matching, object detection, and recognition. HOG is primarily employed for object detection, notably as a people detector, popularized by Navneet Dalal and Bill Triggs in their 2005 paper "Histograms of Oriented Gradients for Human Detection." It ingeniously divides an image into cells and calculates gradients within each cell, forming a histogram representation of pixel intensity changes in specific directions.

The Location issue :-

A HOG-based object detector deals with location and scale variations by using a sliding window technique. The window slides with small steps across the image, potentially framing an object multiple times. To avoid reporting multiple locations for the same object, we prioritize the detection with the highest confidence score when multiple detections overlap. This ensures that we only select the most confident and accurate detection within a set of overlapping results.

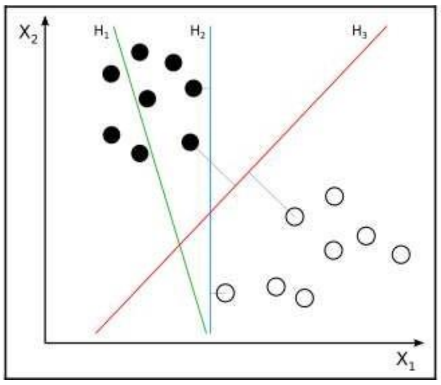
Non-maximum(or Non-maxima)Suppression:-

A typical Non-Maximum Suppression (NMS) implementation involves:

1. Creating an image pyramid to handle variations in scale.
2. Scanning each pyramid level using the sliding window approach for object detection. Detected windows above a set confidence threshold are converted to the original image scale and added to a list.
3. Sorting the list based on descending confidence scores to prioritize the best detections.
4. Applying NMS: For each window, remove subsequent windows with significant overlap.
5. An additional filtering method is eliminating subwindows, which are entirely contained within larger windows. This is achieved by comparing window coordinates. NMS and subwindow suppression can be combined optionally for more refined results.

Support vector machines

Given labeled training data, an SVM learns to classify the same kind of data by finding an optimal hyperplane, which, in plain English, is the plane that divides differently labeled data by the largest possible margin



Hyperplane H1 (shown as a green line) does not divide the two classes (the black dots versus the white dots). Hyperplanes H2 (shown as a blue line) and H3 (shown as a red line) both divide the classes; however, only hyperplane H3 divides the classes by a maximal margin.

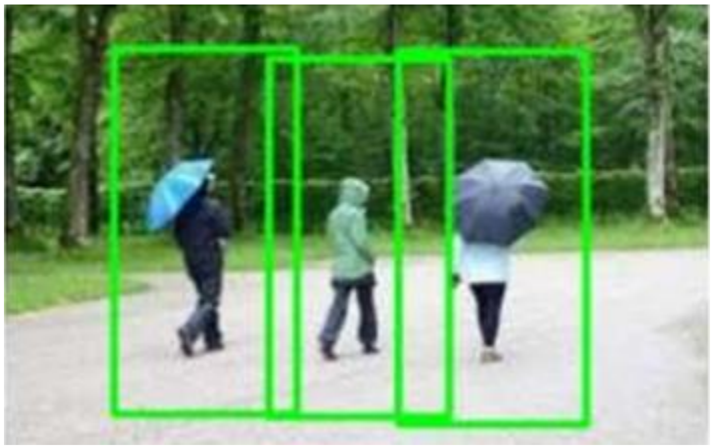
Code :-

```
import cv2
import pytesseract
import numpy as np
import matplotlib.pyplot as plt
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract'
font_scale = 1.5
font = cv2.FONT_HERSHEY_PLAIN
cap = cv2.VideoCapture(1)
 #cap.set(cv2.CAP_PROP_FPS, 170)
#Check if the webcam is opened correctly
if not cap.isOpened():
    cap = cv2.VideoCapture(0)
if not cap.isOpened():
    raise IOError("Cannot open webcam")
cntr = 0;
while True:
    ret ,frame = cap.read()
    cntr= cntr+1;
    if ((cntr%20)==0):
        imgH, imgw,_ = frame.shape
        #eye_cascade = cv2.CascadeClassifier(cv2.data. haarcascades + +"haarcascade_eye.xml")
        x1,y1,w1,h1= 0,0,imgH, imgw
        imgchar = pytesseract.image_to_string(frame)
        imgboxes = pytesseract.image_to_boxes (frame)
        for boxes in imgboxes.splitlines():
            boxes = boxes.split(' ')
            x,y,w,h= int(boxes[1]), int (boxes[2]), int (boxes[3]), int (boxes[4])
            cv2.rectangle(frame, (x,imgH-y), (w, imgH-h), (0,0,255), 3)
    cv2.putText(frame, imgchar, (x1+ int(w1/50),y1+ int(h1/50)), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
(255,0,0), 2)
    font = cv2.FONT_HERSHEY_SIMPLEX
    cv2.imshow('text detection tutorial',frame)
    if cv2.waitKey(2) & 0xff ==ord('q'):
        break
cap.release()
cv2.destroyAllwindows()
```

Output :-

Original Image:-

Processed Image:-



Conclusion: -

The Histogram of Oriented Gradients (HOG) is a widely used technique in the field of computer vision and image processing. It serves as a feature descriptor that helps capture the shape and appearance of objects within images by analyzing the distribution of edge orientations. To implement the HOG method for car detection, the process involves calculating the magnitude and orientation of gradients for each pixel in an image. This image is then partitioned into smaller cells. A Support Vector Machine (SVM) is used in conjunction with the HOG algorithm to create a program that can determine whether a given image contains a car or not.