Machine Learning Solutions for an Autonomous Vehicle

Jagannath Aghav
College of Engineering Pune
jva.comp@coep.ac.in

Rinku Nemade
College of Engineering Pune
nemaderh13.it@coep.ac.in

Apoorva Nitsure
College of Engineering Pune
nitsuream13.it@coep.ac.in

***Abstract*— Autonomous vehicles use and produce enormous amount of data.The use and analysis of data for safety and reliability in such cars is a critical aspect. A complex decision based mechanism is vital to ensure safe navigation for autonomous vehicles in the uncertain environment. Pedestrian detection is necessary as human safety is of utmost importance in today's urban traffic conditions. Vehicle detection is paramount to avoid collisions leading to accidents. For this, autonomous cars must follow traffic rules and regulations to ensure smooth traffic. In this paper, we focus on detection of pedestrians, vehicles and road signs using image processing and machine learning. To ensure disciplined driving in the world of autonomous cars, speed control according to the terrain type needs to be specified. We have proposed lane detection and speed control algorithms as well.**

***Index Terms*— HOG(Histogram Oriented Gradient), SVM(Support Vector Machine), Haar cascade, NMS(Non Maximum Suppression), LIDAR(Laser Illuminated Detection And Ranging), IRC (Indian Road Congress), MLP(Multilayer Perceptron), RBF(Radial Basis Function)**

I. INTRODUCTION

Autonomous cars are now a reality. The need of the hour is to reliably and accurately detect obstacles and avoid collision. For instance, Uber has recently launched self driving cars as a pilot program in Pittsburgh. But still they are nowhere near full autonomy and face problems with identification of suddenly disappearing objects, bridges and dealing with other human drivers.

Active and passive are two approaches for obstacle detection. Active methods include use of sensors whereas passive methods use measurements of the scene e.g. camera images. Passive methods are advantageous as they work in most of the weather and lighting conditions, with a high resolution. Apart from that, cameras require less investment.

Obstacles with respect to a car generally include other cars, pedestrians, cyclist, road-signs, vegetation, animals, traffic signals, speed breakers. In this paper, we focus on three types of obstacles: cars, pedestrians, road signs.

Autonomous vehicles need to adjust their speed according to traffic, weather and road conditions. In this paper, we have discussed speed control for different terrains only. As terrain detection is more accurate using passive methods we have proposed a speed control algorithm which uses real time LIDAR readings.

Additionally we propose a lane detection algorithm using a connected component scheme.

II. PEDESTRIAN DETECTION

We suggest that pedestrian detection be done using HOG descriptors and a linear SVM model for classification.[15]

A. Histogram Oriented Gradient

HOG descriptors are used to extract features from the images. HOG descriptors carry out gradient computation, object binning and lastly extracting descriptors from blocks. This is an alternative to Haar Cascade classifiers i.e. Viola-Jones detectors[9]. HOG descriptors were originally invented for human detection and currently give the most accurate features from images. As haar cascade features are best for texture identification. HOG features are most suitable in the case of human identification from images. There are six steps involved in the process of training an object detector using HOG. The steps are as follows:

Algorithm 1 Training an object detector for HOG

1: Extraction of HOG descriptors from positive samples from the training data.
2: Extraction of descriptors from negative samples of training data which is generally greater than the positive samples in previous step.
3: Using above samples train SVM.
4: For images in the negative training set, use the sliding window technique by moving the window across the image and calculate the HOG descriptors.
5: Find the vector of features with the false positive area and get the classification probability.
6: Using the false positives in the previous step, carry out sorting according to confidence and retrain the classifier.
7: The trained classifier can be applied to the test data-set.

For accurate results, the image must be pre-processed and normalized to a certain size. The image is then stored in the form of residuals in a pyramidal structure. HOG descriptors can be obtained by using a sliding window technique. The window is chosen to select apt number of layers to get optimal balance between speed of computation and accuracy of pedestrian detection to avoid false positives and negatives. Generally sliding window of 4 x 4 is considered to be the optimal size. Bounding boxes must be drawn after detection. Using the above, multiple overlapping bounding boxes may be detected for a single person.

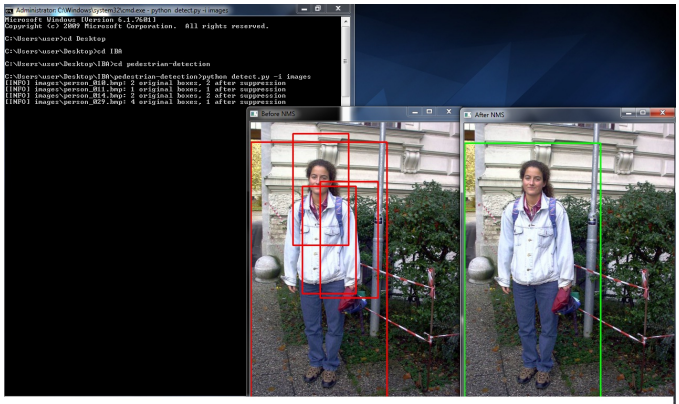

Fig. 1. **Pedestrian Detection:** Output of code to detect pedestrian using HOG before NMS with multiple bounding boxes and after NMS with final bounding box.

B. Non Maximum Suppression

For handling multiple and bounding boxes which overlap, non-maximum suppression should be used. It can ignore small overlapping boxes and therefore return only the larger ones. It involves calculation of the overlap ratio. If this ratio is greater than the threshold, then it signifies that the bounding boxes overlap sufficiently and as a result the current bounding box can be suppressed. Non-maximum suppression creates finalized bounding boxes on the images.

C. Implementation

We have implemented the above algorithm using Python and the OpenCv library. OpenCv includes built-in methods with pre-trained HOG and linear SVM model for classification.

Input: set of images given as input to cv2 [16] [17] [18] [19]

```
# HOG descriptor initialization
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.
  HOGDescriptor_getDefaultPeopleDetector())

# detect people in the image
(rects, weights) =
    hog.detectMultiScale(image,
    winStride=(4, 4),
     padding=(8, 8), scale=1.05)

#NMS
rects = np.array([[x, y, x + w, y + h] for
    (x, y, w, h) in rects])
nms = non_max_suppression(rects, probs=None,
    overlapThresh=0.65)

# draw the final bounding boxes
for (x1, y1, x2, y2) in nms:
  cv2.rectangle(image, (x1, y1), (x2, y2),
      (0, 255, 0), 1.2)
```

III. Vehicle Detection

For vehicle detection, various features like shadow, corner, edges, color, texture can be used. But shadows are dependent on weather condition, corner features can be corrupted by noise, finding exact edges and color is difficult as they are relative features. This makes texture an important feature for vehicle detection. Vehicle detection can be performed using Haar Cascade as textures are accurately identifiable using Haar features and cascade classification. Haar classification utilizes a technique similar to tree structure. Firstly there is a phase of training and a cascade of rejection is formed. This indicates that a classifier which is strong is created from a classifier which is weak in which the classification in at least more than half of the cases is right. It uses a non-acceptance cascade where the final classifier consists of a cascade of many simple classifiers and a region of interest must pass all the stages of the cascade for success. This generates Haar like features.Haar cascades are chosen here because they are useful to detect texture.Orientation is not a necessary piece of information to spot cars or other vehichles in an image.Though orientation cannot be handled in Haar classifiers, since they focus on textures and can be computed with relatively low speed as compared to HOG descriptors they become the most suitable choice for vehichle detection and can be used with cascade classifiers.

A. Haar like features

The concept Haar like features was developed by Voila and Jones, which are similar to Haar wavelets, that are not continuous. The adjacent rectangular regions at a particular location are taken and calculations of the total intensities of pixel in every region and the difference between the sums are considered in Haar like features. The differences are used to categorize subsections of an image. In the object detection framework, a window of the specified size is moved over the image and for every subsection the Haar like feature is computed. The difference thus obtained can be compared with a pre-learned threshold in which there is clear separation between objects and non-objects. Haar like features are a weak classifier and therefore a great number of them are essential for adequate accuracy. Thus to form a strong learner the features must be organized in a classifier cascade. The simplest Haar like feature is rectangular which can be described as the difference of the sum of pixels of regions at any position and scale in the original image in the rectangle. There are also tilted Haar like features.

B. Cascade Classifier

The cascade classifier involves two major operations: training and detection.

- Training data involves samples labeled as positive or negative.
- Negative samples can be taken from arbitrary images.
- After training the suitable data-set a xml file is formed.
- Cascade classification includes various stages having weak learners.

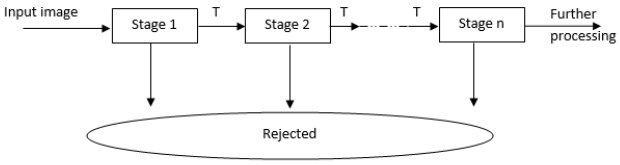

Fig. 2. Stages in Cascade Classifier.

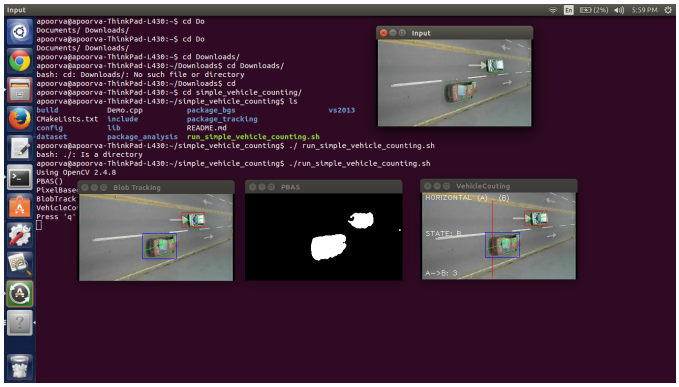

Fig. 3. Vehicle Detection using Haar cascade

- At each stage, the region of the current window is marked either positive or negative.
- A negative result implies the classification is complete for the region and the window is shifted to the next location. A positive result thereby causes the same region to move the next stage.
- A positive outcome results after passing through all the stages of classifier.

The results thus obtained may be in either of the four categories which are true positive, false positive, true negative, false negative. For accurate results, each stage must have low false negatives but a high false positive rate may be acceptable as even if the m-th stage considers a non-object to be a true object, the mistake may be rectified in the (m+1)-th stage.

C. Implementation

Input: xml file trained using sample images

```
cascade_src = 'cars.xml'

cap = cv2.VideoCapture(video_src)
car_cascade =
    cv2.CascadeClassifier(cascade_src)
ret, img = cap.read()
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cars = car_cascade.detectMultiScale(gray,
    1.1, 1)
for (x,y,w,h) in cars:
  cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
```

IV. Road Sign Detection and Recognition

Violation of traffic rules causes a havoc and life threatening accidents. Intelligent systems will ensure safe driving in turn saving time and money. Navigation will also become fast, robust and easier.Detection of road signs and recognition using neural networks is the most preferred choice. Detection may be based on color, shape or both. The system we propose includes the following steps:

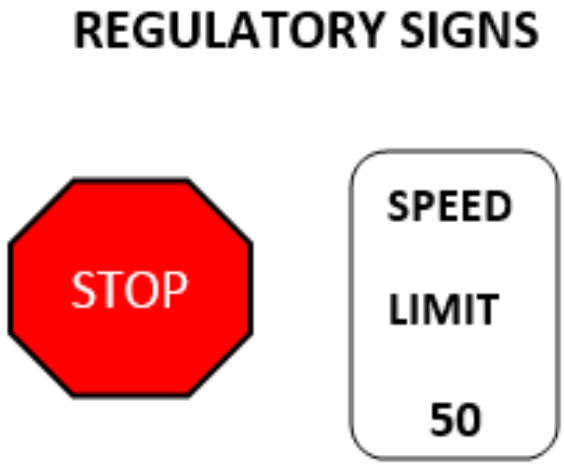

Fig. 4. Regulatory Signs

- Image Acquisition and Pre-processing
- Feature Extraction
- Recognition

A. Attributes of Road Signs

The road signs are designed in such a way that they are easy to recognize and distinguish. The signs have fixed shapes and certain fixed color schemes.

- Warning Sign: It is triangular in shape and has a backframe which is white and a border which is red. These are used to help drivers avoid hazards.
- Compulsory Sign: These are circular in shape and have a white backframe and a border red in color. These signs are instructive in nature e.g.speed limits, stop signs, right turn forbidden and so on.
- Regulatory Sign: These are associated with traffic flow and generally are represented with a blue colored circle having a white border.
- Information Sign: These signs convey helpline or emergency contacts like a hospital is nearby or recreational information like food joints and so on. They are represented with a white rectangle having a blue border.

B. Image Acquisition and Pre-processing

High definition videos are recorded using digital video cameras. Image extraction is carried out to get frames at a particular rate which may involve sampling. The raw images might appear blurred due to various lighting and weather conditions. The raw images are therefore pre-processed. The steps are:

- Convert RGB to Gray scale: Image conversion from R-G-B to gray makes use of weighted average of red, green and blue colors is used as only some colors like blue, black, white, red are used in traffic signs.
  Gray scale value = 0.29989 * Red + 0.5870 * Green + 0.1140 * Blue
- Normalization: For reducing the processing time, enough information is acquired through normalization.

$$Rednew(p,q) = \frac{Red(p,q)}{Red(p,q) + Green(p,q) + Blue(p,q)}$$

Rednew(p,q) is the normalized red pixel and similar processes are carried out for green and blue pixels.

C. Feature Extraction

Specific features are extracted which serve the purpose of the input to recognition phase. Features[6] that we propose:

Feature 1: Height-width relation for road signs.

$$feature1 = \frac{height}{width}$$

Feature 2: Number of white pixels The total pixels are calculated by the formula

$$T = height * width$$

$$P = \left(\frac{W}{T}\right)$$

where P = proportion of white pixels, W = All white pixels in the image, T = Total pixels

Feature 3: Average distance between black pixels and central point

$$Formula : \left(\frac{1}{totalpixels}\right) \sum_x \sum_y \left(\left((x-i)^2\right)\left((y-j)^2\right)\right)^{\frac{1}{2}}$$

D. Recognition

To classify the image according to the features into the categories mentioned above neural networks are apt. Advantages are that input image manipulation is not required and classification is largely dependent on the weight of the network. Artificial neural networks may include MLP, RBF or other networks. Here, input information travels in only one direction through input nodes and after that nodes in layers that are hidden to the o/p nodes. For classification of traffic signs, three layer feed forward neural network can be used.

V. Lane Detection and Speed Control

Autonomous cars make use of GPS for locating the destination entered by a passenger. A typical car must be able to detects obstacles, signals, lanes (in range of 2 meters) and drive at an average speed of 40 kmph. Therefore it should select an alternative path in case of traffic and obstacles as well as be capable of taking decisions as per weather conditions like reducing speed during rains, rocky terrains and so on. Here we focus on lane detection and adaptive speed control. Cameras are used to get images for lane detection. These images are then converted to grayscale images using RGB values of each pixel. Connected components labeling scans an image from the left-hand side to right-hand side of the picture and the opposite as well, then makes groups of the pixels to form components based on connectivity of pixels. Intensity of pixels is used to determine their connectivity. The unwanted region is subtracted from the image. For subtracting the unwanted region, lane width and intensity is compared with connected components.

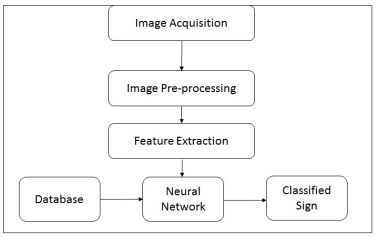

Fig. 5. Methodology for Traffic Sign Detection

A. Lane Detection

Lane detection is necessary for orientation of automated vehicles. This will increase safety during traffic conditions. Lane detection is difficult in presence of other vehicles on lane, along curved roads, obstacles like shadow. For lane detection three cameras can be used, one on front side of car and other two on sides of car. The processor will then form a combined image of surrounding area using images from these cameras.

Algorithm 2 Lane Detection Algorithm

```
Take RGB image
Convert RGB image to gray-scale image
for each pixel p in image do
    if intensity of the pixel p is same as neighbor then
        Assign its label to pixel p
    else
        assign different label to pixel p
Assign label to connected components
Find width of connected components
if Width of connected component does not match with width of lane then
    Subtract the component
```

- Conversion of image from RGB to gray
- Labeling Connected components :

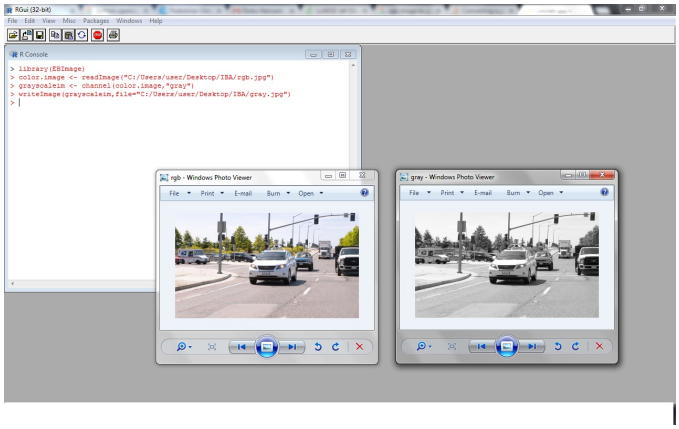

Fig. 6. RGB to Gray-scale conversion in R

Pixel components that are connected have similar values of intensity or difference is minor. Thus this involves scanning image from left to right and right to left. Pixels close to each other and having same intensity value belong to connected component. In a gray scale image set of intensity (say V) will take on a range of values. The operator scans the image by going row-wise till it approaches a point X (where X is the pixel to be given a label). Intensity of pixel p is calculated and then following steps are carried out: If all nearby pixels (4) have differing intensity values, X is given a new label, otherwise if just one has intensity close to p, give its label to X, but if greater than one of the neighboring pixels have intensity close to X, assign one of the labels to p. After scanning, all equivalent labels are collected in a class and a new label is given. Finally, each label in image is replaced by the unique label of the class.

- Component width calculation:
  Width of each connected component is calculated. For finding width, algorithm searches horizontally pixel by pixel. Next, for each component we consider the number of pixels . Then we get total sum. Finally, connected component of maximum width and highest number of pixels is extracted in feature extraction stage.
- Subtracting regions of no use :
  In this step, all labeled connected components are searched and unnecessary regions are subtracted. Hence, we work with the regions that are on the road. Unwanted regions are also found on road. To subtract those regions algorithm makes use of, lane width and lane intensity. If any region has more width than the lane in the image or if intensity value of pixel is far from white color then subtract these pixels because road lanes are white.

B. Speed Control

To make the car to track the entire lap at desired speed irrespective of the curvature and according to the terrain of the track, adaptive speed control algorithm is proposed. Considering the wheel has a constant speed , we can find it by knowing the rotation speed of the car. This is indicated by current speed(CR). The desired speed is calculated by taking readings from the LIDAR[15]. This is indicated by the speed variable. Depending on the LIDAR readings (X) , it is possible to predict the terrain. Thus for various terrains the optimum speed varies. Hence comparing X to some pre defined value will help guess the terrain and accordingly set the optimum speed equal to s1/s2/s3/s4. Recommended speed by IRC of different types vehicles on different type of terrains:

Type	Plain	Rolling	Hilly	Steep
4-Wheeler	100-80	80-65	50-40	40-30
2-Wheeler	50-40	40-35	25-20	25-20

Algorithm 3 Speed Control Algorithm

```
Get LIDAR readings(X)
ks = 0
if X <= Y then
    Speed = s1
else if X <= W then
    Speed = s2
else if X <= Z then
    Speed = s3
Speed = s4
loop:
    Get current speed(CR)
    if Speed > CR then
        while ks < 1 do
            Increment ks with a constant period
            CR = ks * (CR - Speed)
    else if Speed < CR then
        while ks >= -1 do
            Decrement ks with a constant period
            CR = ks * (CR - Speed)
    else if CR == Speed then
        Go to step 10
```

VI. Conclusion

In this paper, we have discussed various obstacles with respect to an autonomous car and proposed solutions for their detection by using passive methods.For identification of pedestrians we have used HOG descriptors as orientation information is necessary to detect edges as compared to recognition of the class of an object.We tested our algorithm on a dataset of more than 100 images getting high accuracy. For vehicle detection we suggest use of cascade classifiers, as texture plays an important role in identification of vehicles.We have implemented the algorithm and achieved satisfactory accuracy.For road sign detection we have identified different features and used neural networks for recognition. Additionally we have proposed a solution for lane detection and for deciding the speed of the car according to the terrain. Speed control is better using active methods as image processing for identification of the type of terrains is less accurate and difficult. Hence, we have proposed

a simple algorithm using LIDAR readings. We propound lane detection using passive methods. It is a straightforward algorithm and has less time complexity as only a change in the intensity of pixels in grayscale images is used to detect lanes.

REFERENCES

[1] A Pedestrian Detector Using Histograms of Oriented Gradients and a Support Vector Machine Classifier. M. Bertozzi ; A. Broggi ; M. Del Rose ; M. Felisa ; A. Rakotomamonjy ; F. Suard. ITSC 2007. IEEE

[2] Traffic Sign Classification based on Neural Network for Advance Driver Assistance System, Aini HUSSAIN, Mahammad A. HANNAN, Tan J. PIN, Salina A. SAMAD, Safat B. WALI, Department of Electrical, Electronic and Systems Engineering, University Kebangsaan Malaysia,2015

[3] Machine Learning For Object Recognition and Scene Analysis, S. Moscatelli and Y. Kodratoff, 1994

[4] Object Recognition as Machine Translation-I: Learning a Lexicon for a Fixed Image Vocabulary K. Barnard, J.F.G.deFreitas,P. Duygulu, D.A. Forsyth Computer Science Division, U.C. Berkeley, CA Department of Computer Science, University of British Columbia, Vancouver, 2002

[5] Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting, Leon Bottou NEC Labs America, Princeton, Fu Jie Huang, Yann LeCun, 2004

[6] A Real Time Road Sign Recognition using Neural Network Mohammad Badrul Alam Miah Dept. of Information and Communication Technology Mawlana Bhashani Science and Technology University Santosh, Tangail-1902, Bangladesh

[7] Betke, Margrit, Esin Haritaoglu, and Larry S. Davis. "Real-time multiple vehicle detection and tracking from a moving vehicle." Machine vision and applications 12.2 (2000): 69-83.

[8] Rapid object detection using a boosted cascade of simple features, P. Viola, M. Jones, 2001

[9] Lienhart, Rainer, and Jochen Maydt. "An extended set of haar-like features for rapid object detection." Image Processing. 2002. Proceedings. 2002 International Conference on. Vol. 1. IEEE, 2002.

[10] Gavrila, Dariu M., and Vasanth Philomin. "Real-time object detection for" smart" vehicles." Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on. Vol. 1. IEEE, 1999.

[11] Andreasson, Henrik, Andr Treptow, and Tom Duckett. "Self-localization in non-stationary environments using omni-directional vision." Robotics and Autonomous Systems 55.7 (2007): 541-551.

[12] Sun, Zehang, et al. "A real-time precrash vehicle detection system." Applications of Computer Vision, 2002.(WACV 2002). Proceedings. Sixth IEEE Workshop on. IEEE, 2002.

[13] Han, Feng, et al. "A two-stage approach to people and vehicle detection with hog-based svm." Performance Metrics for Intelligent Systems 2006 Workshop. 2006.

[14] Varghese, Jaycil Z., and Randy G. Boone. "Overview of Autonomous Vehicle Sensors and Systems." Tech. rep. Proceedings of the 2015 International Conference on Operations Excellence and Service Engineering Orlando, Florida, USA, 2015.

[15] http://www.pyimagesearch.com/2015/11/09/pedestrian-detection-opencv/

[16] http://pascal.inrialpes.fr/data/human/

[17] http://www.cs.utexas.edu/ grauman/courses/spring2008/datasets.htm

[18] http://vision.stanford.edu/Datasets/40actions.html

[19] Convergence and Hybrid Information Technology: 5th International Conference, ICHIT 2011, Daejeon, Korea, September 22-24, 2011