[[1]](#footnote-1)

Car Identification in Videos

Tanvi Vidhate, 820851740, Meng Zhou, 813349139, Raghav Kishan Sunku Ravindranath, 820174908, Niranjan Hegde, 820625579, Shwetha Kisan shahane, 820899242

*Abstract*—With the increasing number of vehicles on campus, there arises a need to provide an automated system to provide students and staff with the information about the availability of parking space in a parking lot, thereby reducing the time taken to find empty parking spaces with minimal human intervention. In this project, we will be exploring options of solving the above stated problem using computer vision and machine learning techniques. We are using a video camera to record the vehicle movement in the parking lot. A set of videos are obtained from cameras set up at the entrances/exits of a parking lot. Using multiple approaches, we identify the number of cars entering/exiting the parking lot. These approaches include separating the data into training and testing data, feature extraction and machine learning techniques to predict the number of cars in a video. The results of the predictions are evaluated with the ground truth and the accuracy is obtained from the generated confusion matrix. To decrease the cost of production(implementation), an integrated vision-based system is an excellent choice.

*Index Terms*—Histogram of Oriented Gradients(HOG), Scale - Invariant Feature Transform(SVM), Support Vector Machine (SVM)

# INTRODUCTION

F

INDING empty parking spaces in a parking lot can be an aggravating experience. A system that can automatically identify empty parking spaces and guide users to it will save a lot of time, money and effort. Many solutions which employ sensors for this purpose have been suggested, but they are either too expensive to implement or have failed to be effective. This led people to look for other economical solutions. Computer vision based techniques have the potential to provide a cost-effective solution to this problem because they can use existing infrastructure such as security cameras to capture videos which can be processed to classify the state of the parking space.

Image processing techniques based on software component can be used on video frames to identify cars. This methodology does not require special hardware components. A detection and counting device can be created using a video camera and a normal computer.

Multiple Image processing methodologies such as blob analysis, background-foreground subtraction with supervised machine learning techniques using feature vectors are used to detect the vehicle in the video. These feature vectors are generated using different techniques to obtain better accuracy.

# Task Description

The process of object detection and counting is performed by following the below mentioned steps.

1. Preparation of data set and ground truth – cameras were fit at the entry and exits points of a parking lot to record the movement of cars from and to the parking lot.
2. The recorded videos were then annotated to generate the ground truth files which are being used for training, validation and testing machine learning models.
3. The ground truth data consisting of car Id, Xmin, Ymin, Xmax, Ymax, frame number, lost, and occluded is supplied to the machine learning models.
4. Multiple machine learning algorithms are applied on the training data which generates models.
5. These models are validated using the validation data set and tested using the testing data set.
6. The machine learning techniques used in this project for vehicle identification are Blob Analysis, Linear Support Vector Machine, RBF Support Vector Machine.
7. The above-mentioned algorithms are supplied with different feature vectors such as black & white ratio, Histogram of gradient, Scale-invariant feature transform.
8. The black & white ratio feature vector is generated by diving a frame of a video into blocks & cells. Then the ratio of count of black to white pixels are taken to form the feature vector.
9. The Histogram of Oriented gradient feature vector is generated using the HOG () function of MATLAB.

# MAJOR CHALLENGES

1. D

# EXPERIMENTS

VEHICLE DETECTION USING BLOB

## Dataset Description

The data set being used here is obtained from cameras fixed at the entry and exit points of parking lots. These cameras record videos of resolution 720 \* 1024 pixels with 60 frames per second. These videos are resized to increase the processing speed of algorithm. [1]

## Evaluation Metrics

To visualize the performance of this algorithm, the total count of identified vehicles for the test videos is compared with the ground truth data obtained from the annotations.

## Major Results

As described in the Evalautaion Metrics section the counts of cars obtained from the Blob analysis results which is then compared with the counts obtained from the ground truth. It is observed that the accuracy is about (90%) which is fairly accurate.

## Analysis

**Blob Detection techniques are aimed at identifying regions in an image based on certain properties such as such as brightness and color changes in the image. In general, a blob is a region in which some properties are constant.**

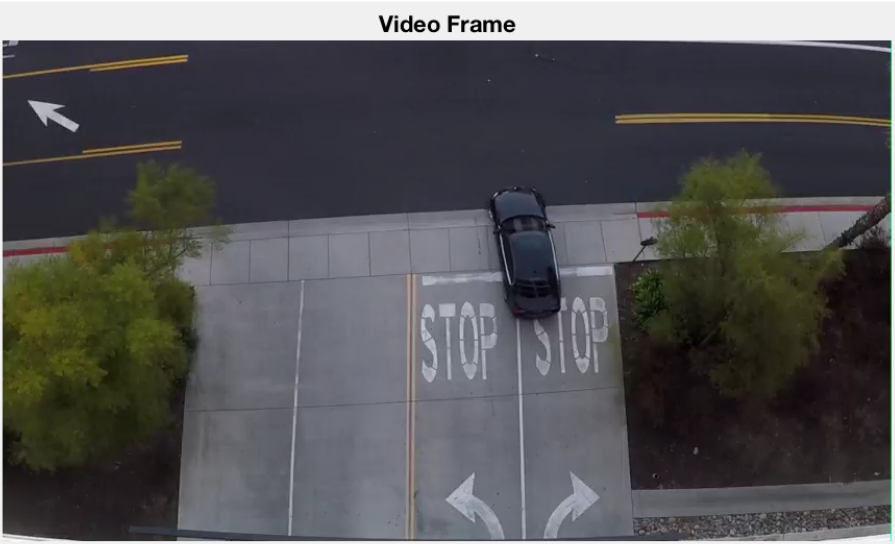
**The purpose of BLOB extraction is to isolate** the blobs (objects) in a binary image. A blob consists of a group of connected pixels. Whether or not two pixels are connected is defined by the connectivity, that is, which pixels are neighbors and which are not.

An advantage of this method is that it includes high flexibility and good performance. The disadvantage is it requires a clear background-foreground subtraction.

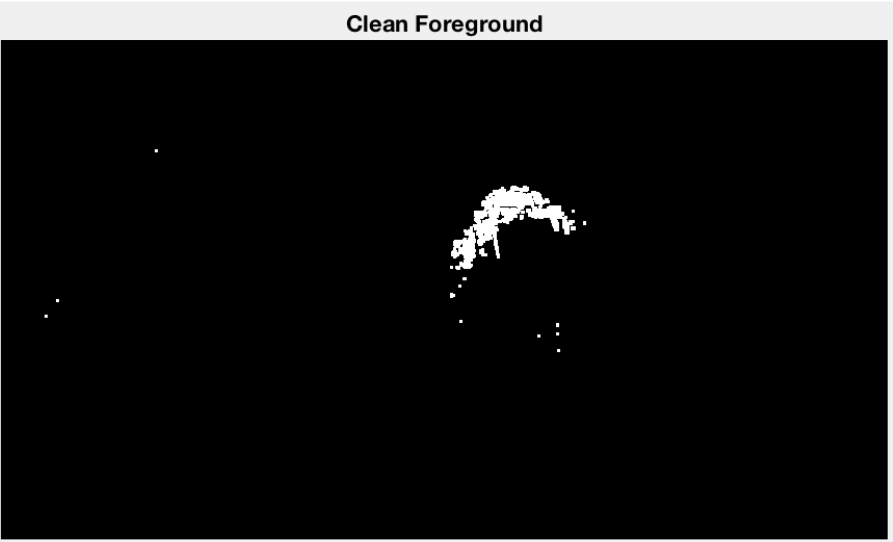
A foreground detector is created using the matlab function called vision.ForegroundDetector () which is supplied with model name – NumGaussians and the number of initial frames of the video to be utilized.

The foreground detector created here is then subjected to the function strel () to reduce the noise in the generated foreground. This filtered foreground detector is then applied on every frame of the video to obtain the foreground of the frame. The foreground which is obtained is then subjected to a BLOB analysis function created which is shown below.

blobAnalysis = Vision.BlobAnalysis ('BoundingBoxOutputPort', true, ... 'AreaOutputPort', false, 'CentroidOutputPort', false, ... 'MinimumBlobArea', 2500, 'MaximumBlobArea', 100000);



Extracted video Frame



A clean foreground obtained after background foreground subtraction.



A car being identified and counted using a foreground detector and Blob Analysis.

VEHICLE DETECTION USING LINEAR SVM AND RBF SVM ALONG WITH SIFT – SCALE-INVARIANT FEATURE TRANSFORMATION

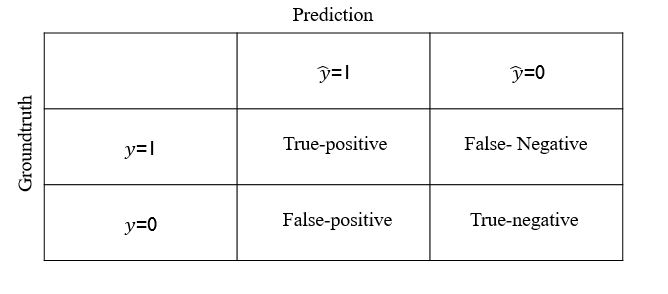
## Dataset Description.

Using cameras at certain points at the entrance and exit points of a parking lot, videos of vehicles entering and exiting the parking lot have been recorded. Few videos from these recording have been annotated to generate the ground truth. [1]

The above data set is separated into training, validation and testing data set. The videos are recoded with the resolution of 720 \* 1024 pixels with 60 frames per second.

1. Evaluation Metrics

To visualize the performance of an algorithm, typically a supervised learning a confusion matrix is used. Also, known as error matrix, each column of the confusion matrix signifies an instance of a predicted class and each row signifies an instance of the actual class.



The above table is an example of a confusion matrix. If the prediction and ground truth are equal, then it is either True-positive or True negative based on the classification labels. If the prediction is not equal to the ground truth, then it is either False-positive or False-Negative based on the classification labels.

The value obtained for the confusion matrix for our algorithm is as follows:

Linear SVM:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Predictions | | | | | |
| Ground Truth |  |  |  |  |  |
|  | 2074 | 6 | 0 | 0 |
|  | 42 | 1232 | 1 | 0 |
|  | 1 | 27 | 112 | 0 |
|  | 0 | 0 | 0 | 39 |

Accuracy = 97.82%

(3457/3534)

RBF SVM:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Predictions | | | | | |
| Ground Truth |  |  |  |  |  |
|  | 2075 | 5 | 0 | 0 |
|  | 34 | 1238 | 3 | 0 |
|  | 1 | 8 | 131 | 0 |
|  | 0 | 0 | 0 | 39 |

Accuracy = 98.56%

(3483/3534)

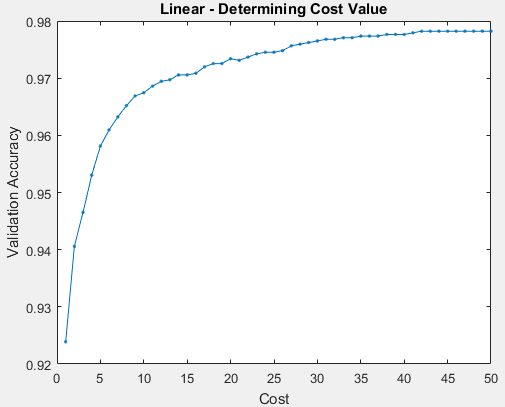
1. *Major Results*

The results obtained from Linear SVM and HOG as the feature set is as follows.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Predictions | | | | | |
| Ground Truth |  |  |  |  |  |
|  | 2127 | 6 | 0 | 0 |
|  | 47 | 1159 | 2 | 0 |
|  | 2 | 26 | 113 | 0 |
|  | 0 | 0 | 0 | 51 |

Accuracy = 97.65%

(3450/3533)

`

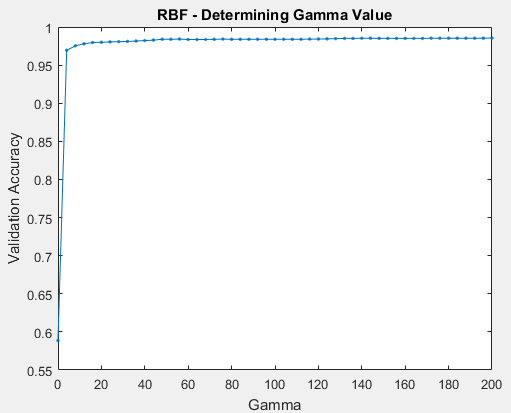
Best cost = 42

The results obtained from Linear SVM and HOG as the feature set is as follows.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Predictions | | | | | |
| Ground Truth |  |  |  |  |  |
|  | 2130 | 3 | 0 | 0 |
|  | 35 | 1171 | 2 | 0 |
|  | 1 | 7 | 133 | 0 |
|  | 0 | 0 | 0 | 51 |

Accuracy = 98.64%

(3485/3533)



Best gamma = 200

1. *Analysis*

To make comparisons available, SIFT algorithm is applied on each frame to generate a feature vector. The implementation of the extraction of SIFT is made up of four major steps:

1. Load the video and the ground truth annotations (mat) file.
2. Create a video data matrix of size num\_frames x (hog\_length+1). The last column is reserved for class labels (0, 1, 2, 3 cars and so forth).
3. Extract SIFT features for each frame with a cell size of 128x128 and a block size of 4x4. The sizes are ambiguous. The reason for the cell size and block size to be the specific values above is that it would not generate a result that is either too large or too small.
4. Iterate over the annotations matrix, looking for the corresponding frame numbers both in its cells and in the video data matrix. For every same frame encountered, one is added to the class label.

After obtaining SIFT features of the video data from the video and the class labels from the annotations matrix, the data is trained with an SVM classifier. LIBSVM [2] 3.2.2 for Linear and RBF SVM, is used for the implementation of both training and testing.

To do the training and testing, the video data was randomly split into a training dataset and a testing dataset with a ratio of 80%-to-20%.

VEHICLE DETECTION USING SVM WITH SIFT – SCALE-INVARIANT FEATURE TRANSFORMATION

1. *Dataset Description.*

The data set being used here is obtained from cameras fixed at the entry and exit points of parking lots. These cameras record videos of resolution 720 \* 1024 pixels with 60 frames per second. These videos are resized to increase the processing speed of algorithm. [1]

1. *Evaluation Metrics*

The evaluation metrics for the techniques is very similar to that used in the Linear and RBF SVM. A confusion matrix will be used to display the efficiency and results of the algorithm.

1. Major Results.
2. Analysis

VEHICLE DETECTION USING SVM WITH BLACK TO WHITE RATIO OF BLOCKS AS FEATURE SETS

1. Dataset Description

Using cameras at certain points at the entrance and exit points of a parking lot, videos of vehicles entering and exiting the parking lot have been recorded. Few videos from these recording have been annotated to generate the ground truth. [1]

The above data set is separated into training, validation and testing data set. The videos are recoded with the resolution of 720 \* 1024 pixels with 60 frames per second.

The frames obtained from the video undergo background foreground subtraction to obtain the foreground. On the foreground frames, cells/bins are created. For each of these cells/bins the black to white ratio is calculated. The black to white ratio to

1. *Evaluation Metrics*
2. *Major Results*
3. *Analysis*

# Conclusion AND FUTURE WORKS

## Using various techniques, we obtained accuracies as shown above from which it can be inferred that vehicle identification using SVM with different feature vectors generates better accuracy when compared to the BLOB analysis.

References

1. <https://drive.google.com/drive/folders/0B4muPJH7ZVoVaFJ2ellzV3hrWG8>
2. <https://www.csie.ntu.edu.tw/~cjlin/libsvm/>

1. [↑](#footnote-ref-1)