Speed estimation for vehicles using SVM classification and edge detection

Mathias Dierickx, Tim Ranson, Maarten Tindemans, and Bjorn Vandenbussche

Ghent University, Faculty of Engineering and Architecture
Valentin Vaerwyckweg 1, 9000, Ghent

Abstract. This is the abstract: The goal of this paper it to determine the maximum speed for a moving vehicle in traffic, using only camera images and information from the steering column. We construct a system that combines SVM classification and edge detection to foresee possible obstacles on the given trajectory of the car. The maximum speed is computed so that braking in time will avoid collision with those obstacles. TODO: hoe goed is dit systeem?

Keywords: SVM classification, Local Binary Patterns, edge detection

1 Introduction

This paper determines the maximum speed for a moving vehicle in traffic, using only camera images and information from the steering column. We construct a system that combines SVM classification and edge detection to foresee possible obstacles on the given trajectory. The maximum speed is computed so that braking in time will avoid collision with those obstacles.

In recent years, a lot of advanced electronics have been used to help automate driving. It is already common for some cars to have (semi-)automated parking [**?**] and adaptive cruise control [**?**]. Also, a lot of safety features are being developed to increase environmental awareness of cars: pedestrian detection [**?**], automated detection of speed limit signs [**?**], etc.

An additional feature that can be realized using the electronics that are already available in these cars, is automated maximum speed detection. This speed detection would simulate the braking distance towards the nearest obstacle and inform the user. There are various ways to achieve this. Commonly used sensors in adaptive cruise control are lidar and radar. The choice of sensor presents classic design trade-offs [**?**]. Lidar is less expensive to produce and easier to package but performs poorly in rain and snow. Another problem is that accumulations of mud, dust or snow on the car can block lidar beams. Radar-based systems, on the other hand, can "see" at least 150 meters ahead in fog or rain heavy enough to cut the driver's ability to see down to 10 meters or less.

Because of the problems with both these systems, this paper proposes the use of a detection system based on camera images. These images present some advantages over a system using lidar or radar. First of all cameras are way

cheaper to install. On top of that they see color and are able to gain very high resolution for more distant objects. There are several downsides to using cameras. They require light to work properly, which becomes a problem for example at night. Secondly computer vision systems using camera images can use a lot of CPU power. Nonetheless cameras provide a cheap and accessible platform for immediate applications in real-world scenarios. On top of that, cameras could be used in combination with lidar and radar to provide even better results.

1.1 Overview

Section 2 discusses some relevant background information mostly concerning Local Binary Patterns and SVM classification. Section 3 describes the methods used in order to achieve the best results. The results are then discussed in section 4. In the last section, an overview of the most important results is presented.

2 Background

2.1 Texture analysis using Local Binary Patterns

Local Binary Patterns (LBP) [**?**] is a non-parametric descriptor whose aim is to efficiently summarize the local structures of images. As a non-parametric method, LBP summarizes local structures of images efficiently by comparing each pixel with its neighboring pixels. The most important properties of LBP are its tolerance regarding monotonic illumination changes and its computational simplicity. LBP was originally proposed for texture analysis [**?**], and has proved a simple yet powerful approach to describe local structures.

The original LBP operator labels the pixels of an image with decimal numbers, called Local Binary Patterns or LBP codes, which encode the local structure around each pixel (see figure 1). Each pixel is then compared with its eight neighbors in a 3x3 neighborhood by subtracting the center pixel value. The resulting strictly negative values are encoded with 0 and the others with 1. A binary number is obtained by concatenating all these binary codes in a clockwise direction starting from the top-left one and its corresponding decimal value is used for labeling. The derived binary numbers are referred to as Local Binary Patterns or LBP codes.

One limitation of the basic LBP operator is that its small 3x3 neighborhood cannot capture dominant features with large scale structures. To deal with the texture at different scales, the operator was later generalized to use neighborhoods of different sizes [**?**]. The histogram of LBP labels can then be exploited as a texture descriptor.

2.2 SVM classification

Support Vector Machines (SVM's) [**?**] are a learning method used for binary classification. The idea is to find a hyperplane which separates the d-dimensional

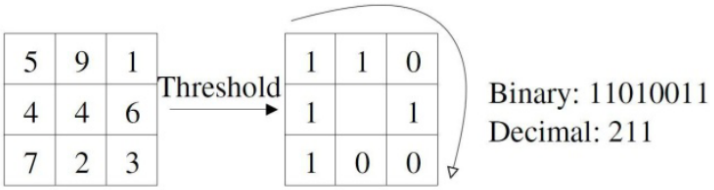


Fig. 1: An example of the basic LBP operator

data perfectly into its two classes. However, since example data is often not linearly separable, SVM's introduce the notion of a "kernel induced feature space" which casts the data into a higher dimensional space where the data is separable. Typically, casting into such a space would cause problems computationally, and with overfitting. The key insight used in SVM's is that the higher-dimensional space doesn't need to be dealt with directly, which eliminates the above concerns.

Basically, we want the hyperplane that maximizes the geometric distance to the closest data points (as shown in figure 2).

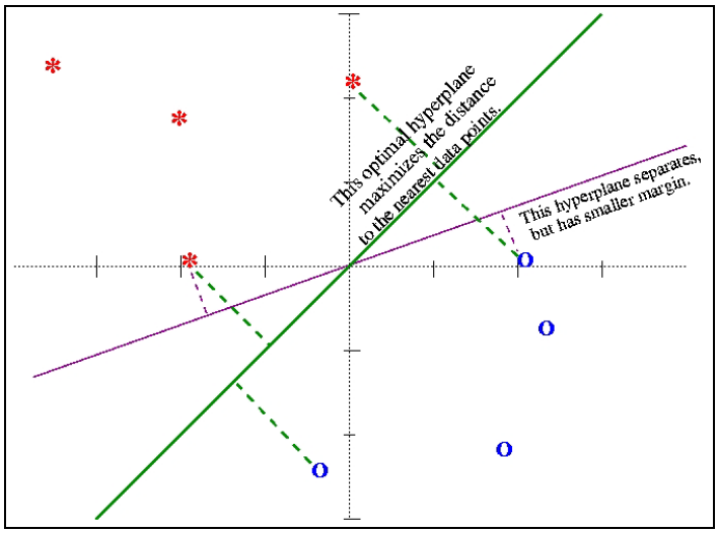


Fig. 2: Choosing the hyperplane that maximizes the margin.

Only the closest data points contribute to deciding the optimal hyperplane, which is why these are called the *support vectors*. They are the only ones needed in defining (and finding) the optimal hyperplane.

The equation for the hyperplane is subject to a constraint concerning the parameter C. One can think of C as a tunable parameter: higher C corresponds to more importance on classifying all the training data correctly, lower C results in a more "flexible" hyperplane that tries to minimize the margin error for each example.

Kernels The original equation for the hyperplane to linearly separate the data contains a linear kernel. [**?**] points out when the use of a simple linear kernel is appropriate: if the number of features is large, one may not need to map data to a higher dimensional space. That is, the nonlinear mapping does not improve the performance. Using the linear kernel is good enough, and one only searches for the parameter C.

We mentioned that SVM's can use a "kernel induced feature space" which casts the data into a higher dimensional space where the data is separable, in case the data is not linearly separable in its original space. Pending on the type of data and application of the SVM, there are several types of kernels which could be used. The Radial Basis Function (RBF) kernel is a often a reasonable first choice, because it can handle the case when the relation between class labels and attributes is non-linear, while only adding one more parameter γ to search for.

Parameters There are two parameters for an RBF kernel: C and γ. It is not known beforehand which C and γ are best for a given problem; consequently some kind of model selection (parameter search) must be done. The goal is to identify good (C, γ) so that the classifier can accurately predict unknown data (i.e. testing data).

One method for comparing results for a set of parameter values is cross-validation. In v-fold cross-validation, we first divide the training set into v subsets of equal size. Sequentially one subset is tested using the classifier trained on the remaining $v1$ subsets. Thus, each instance of the whole training set is predicted once so the cross-validation accuracy is the percentage of data which are correctly classified.

3 Method

In order to determine the maximum speed of a vehicle, two image processing techniques are used: SVM classification and edge detection. To combine the best of both techniques, two strategies are discussed in section 3.7.

For the implementation of our method, we use OpenCV 3.1 [**?**] and C++. Four different datasets were used to either train or test the solution.

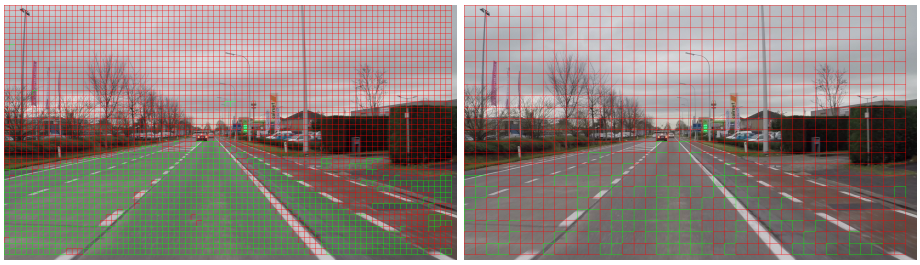
3.1 Parameters for SVM classification

The goal of this classifier is to determine what pixels are part of the road and which are not. We will describe multiple tunable parameters in order to get

optimal classification for road detection. These parameters determine both the length and values of the feature vectors, which are then fed to the SVM classifier. We compare classification results varying these parameters for the feature vector in section 4.1. To train our SVM classifier, we use a function of the OpenCV SVM library which selects the optimal parameters using cross-validation.

Block size Each frame is split into multiple blocks of predetermined size. We chose to vary the block sizes between 8x8px, 16x16px and 32x32px. On the one hand, larger block sizes would too easily contain multiple structures of different classes, resulting in poorer classification. For example a block on the edge of the road containing part of the street and grass. On the other hand, smaller block sizes would omit too many details to be descriptive enough to use for classification.

The block size affects the level of detail taken into account while calculating the feature vector, as the values for the feature vector operate within one block size. Some examples of different block sizes can be found in figure 3.



(a) Block size 16x16px (b) Block size 32x32px

Fig. 3: SVM prediction using only LBP values and including road marks. Green blocks were predicted to be road, red to be non-road.

LBP With LBP we try to characterize the structure of roads within a block. The histogram of LBP values within that block can be added to the feature vector in order to improve classification. These histograms are calculated per channel (e.g. 3 channels for RGB) and are added consecutively to the feature vector.

Color Because working with LBP values only doesn't take color intensity into account [**?**], histogram values of color intensity per block and channel are added to the feature vector.

3.2 Kernels for SVM classification

Judging from the OpenCV documentation[1] there are two SVM kernels fit for a two-class classification problem. A linear kernel is faster, but the RBF kernel will outperform the linear kernel in case the data is not linearly separable. We tested and compared the results using each kernel to decide which is the best fit for our classification problem.

3.3 Training the SVM classifier

For each combination of the previously mentioned parameters, the SVM classifier must be trained. For a training set (a subset of the datasets available) each frame is divided into blocks. Each block must be manually labeled with the expected value for classification. For practical reasons we put a margin between each block in the frames of the training set and only used the i-th frame of each dataset ($i = 0, 10, ..., 40$). Figure 4 shows a visualization of pre-labeled blocks in a frame. The classifier is trained using the label and calculated feature vector for each block in each frame of the training set.

Fig. 4: Visualization of a pre-labelled frame of a training set. Block size 32x32px, excluding road marks

3.4 Road marks

The training dataset was labeled twice, once considering blocks containing road marks as part of the road class and once as part of the non-road class. Initially road marks were labeled as non-road, in order to get a good classification of road (i.e. different types of asphalt). But as these road marks aren't really considered an obstacle for a car driver, ideally they should be classified as road. This makes up for a more difficult classification problem. To quantify this effect we apply classification both with and without road marks.

[1] http://docs.opencv.org/3.1.0/

3.5 Removing details

Detecting edges is very straightforward using an edge detection algorithm like Canny Edge Detection [**?**]. However, there might be a lot of edges on the road itself (think of road markings and cobbles). The edges on the road, considered as noise, and the edges of the road are not unambiguously separable. The width of the patterns that cause the edges on the road is rather small. Using morphological image processing operations, it is possible to remove these small patterns. In figure 5b, a smoothed cobbled road is shown.

(a) Original frame (b) Eroded cobbled road

Fig. 5: Perform eroding to remove the edges on the road.

Eroding [**?**] is a morphology operator to make the objects on the foreground, which are the brightest, smaller. Figure 7b shows an example of a thinned white line. When the kernel is large enough, white lines can be filtered out completely. The same technique is used to smooth the surface of cobbled roads. A cobblestone consist of a bright center, surrounded by darker joints. By eroding the cobbled road, the erode function will minimize the center of the cobble, which will cause enlarged joints. With a large kernel, the joints will spread until they overlap. The best results were achieved with an ellipsoidal kernel of 100 x 30 px.

(a) Original frame (b) Bigger joints after eroding

Fig. 6: The effect of eroding cobbled roads.

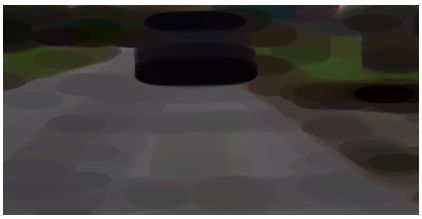
(a) Original frame (b) Smaller lines after eroding

Fig. 7: The effect of eroding road markings.

Unfortunately, the erode function will cause the darker objects to expand. The original size of the objects has been modified, as seen in 8b. This will falsify the eventual edge detection. In this example, the observed car is reported too close. In order to restore the original measures, dilation is executed. This is the opposite morphology operator of eroding. The combination of eroding and dilation is called opening. When dilation is executed with the same kernel size

of the erode function, the original size of the dark objects will decrease to their original size, as seen in 8c. Note that, after eroding, some details of the shape of the objects are lost. In order to provide a safety margin, the dilate function is executed with a slightly smaller kernel.

(a) Original frame



(b) Dark objects are bigger after eroding

(c) Restore original sizes with dilation

Fig. 8: Perform eroding and dilation to remove details.

3.6 Canny Edge Detection

To detect the edges of the road, the Canny Edge Detection algorithm is used. Figure 9 shows an example.

Fig. 9: Canny edge detection

This algorithm requires a minimum and a maximum threshold value, which were determined experimentally. The best results were achieved with a maximum threshold that is twice the minimum threshold. The minimum threshold is

determined by the required minimum sensitivity. When the algorithm is configured too sensitive, some imperfections on the road will be detected as edges. On the other hand, a certain level of sensitivity is necessary to detect road edges that are less clear. For example, a transition from road to dirt has a very small distinction of color and brightness, as seen in 10b. The minimum threshold should be high enough to detect this transition as edge.

(a) Original frame (b) Eroded road

Fig. 10: Unclear transition from road to non-road.

3.7 Combination of classification and edge detection

In order to calculate the maximum speed a vehicle can have in a particular frame, we use a combination of road classification and edge detection. Like that, we can detect where it is safe to drive. For each frame in the dataset, a mask is given that shows the projected trajectory of the car in that frame. They are used to derive the maximum speed to be able to brake safely without hitting an object. When an object is detected at a certain point on the mask, the intensity at that point gives the maximum safe driving speed. The edges of the edge detection are visualized as blue lines on figure 11. The intersections are shown in green. The closest intersection to the car is shown in red. At this point the maximum speed is derived from the intensity of this pixel.

In order to calculate the edges, two different strategies were tested:

1. A region is considered to be road only if it's identified as road by the road classification and there are no edges. Both classification and edge detection should indicate a pixel as road in order to be used for safe driving.
2. A region is considered to be road if there are no edges. When edges are detected, road classification is used to decide what part of the image is road.

For the first strategy, the edges of the area classified as road are computed. These get combined with the edges from the edge detection. The closest intersection of any edge with the given mask of the image is used to calculate the maximum speed of the vehicle.

In the second strategy, the edges are first calculated using edge detection. Each edge is then checked for whether it lies on the road or not using road

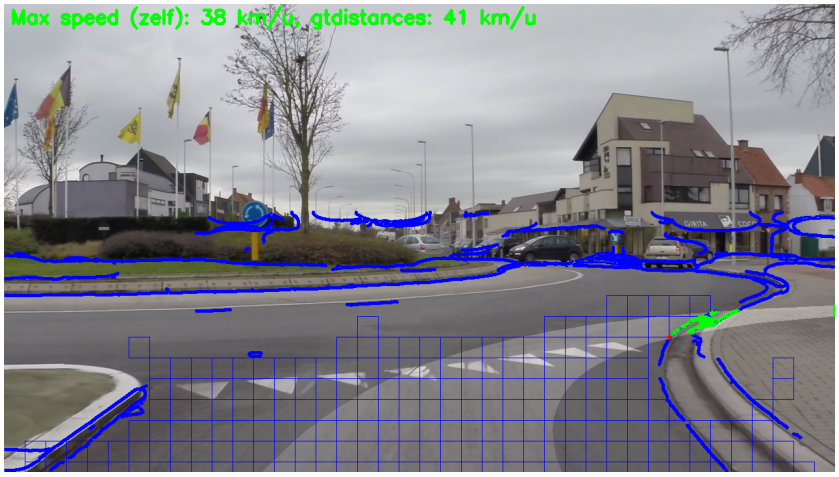


Fig. 11: Visualization of a processed frame showing both classification and edge detection

classification. For safety reasons there is an additional check. The two blocks above, left and right of an edge should also be considered as non-road. When an edge lies on a part of the frame that is not classified as road, it is added to a collection. The intersections of the edges in the collection and the masks are then computed. Using the mask and the location of the intersection closest to the car, the maximum driving speed is derived.

4 Results

4.1 SVM classification

We iterated through several parameter values (as discussed in 3.1) to find optimal classification for road detection. For each combination of parameters we train and test on different combinations of datasets. For each combination we calculate the F1-scores [**?**]. These scores are used to compare the classification results (higher F-scores indicate better classification).

Different combinations of training sets were selected of the four given datasets (training sets used are each time shown on the x-axis on the graphs). The trained classifier is then tested on the remaining datasets.

Kernels As previously mentioned in section 3.2, there are two SVM kernels we can use for this classification problem. On average, we tend to get a nearly 10% increase in F-scores using an RBF kernel instead of a linear kernel. It is safe to say the RBF kernel outperforms the linear kernel for this classification problem in all of our test scenarios.

Block size In general, block sizes 16x16px and 32x32px tend to show the better results. As an example we present the F-scores for using color feature vectors with road marks in figure 12. Other combinations follow the same trend, although 32x32px sometimes performs slightly better. This is why we conclude the block size of 32x32px to be a more reliable choice.

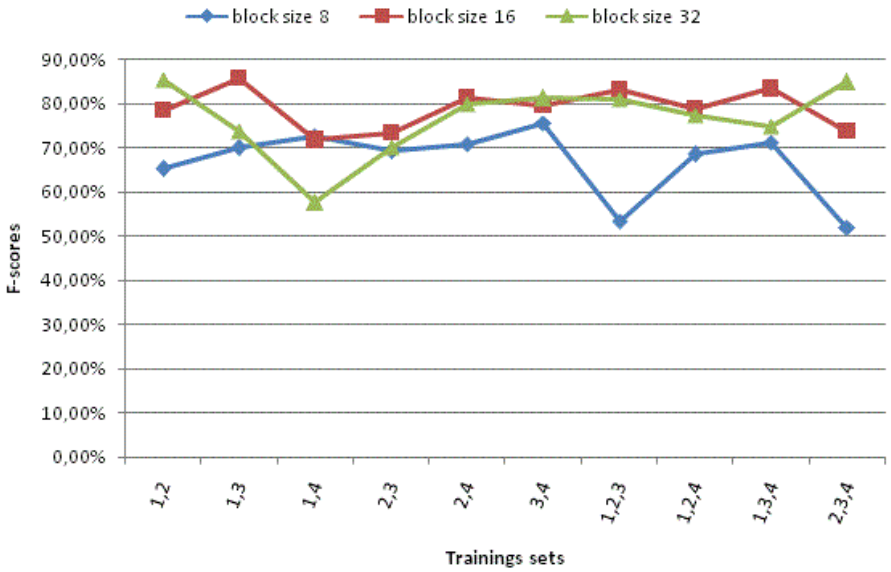


Fig. 12: F-scores for color with road marks using RBF kernel

Feature vector values The use of LBP represents no significant gains in order to get a viable classifier for road detection. If we compare results (block size 32x32px) of feature vectors on color only to the combination of LBP and color, performance remains roughly the same. Therefore, in the case of our classification problem, we do not add LBP to the feature vector.

The best results are found when using color histogram values for the feature vector. This implies the combinations of color values are sufficient to distinguish the road from other surroundings. A possible explanation for this phenomenon is that when asphalt and road marks are considered to be road, color is a much more consistent factor to weigh in to the classification.

TODO grafiek wo/wi RM

Road marks Including road marks as part of the road gets better results for the classification problem. For our application this is more practical, because road marks should not be considered to be obstacles.

Overall we can conclude that using only color with road marks shows the optimal results for our classification problem.

TODO figure aanpassen

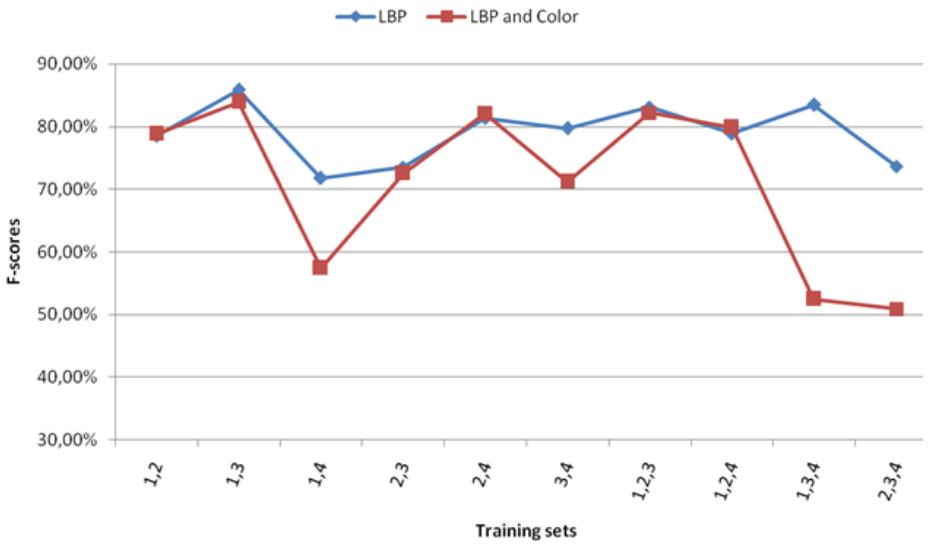


Fig. 13: Comparision of F-scores for 16x16px block size with road marks using RBF kernel

Considering the results of the different scenarios, we can conclude that training on the fourth dataset results in poorer classification overall. This is because another type of road is used in this dataset, conflicting with the more common road type present in the other datasets. This distorts the classifier increases misclassifications.

TODO check if list not f***ed up.

In retrospect, the optimal parameters for our classification problem are:

- RBF kernel
- Block size 32x32px
- Using only color histogram values for the feature vectors
- Including road marks as part of the road

4.2 Driving results

Driving results get evaluated using a script that calculates the distance a vehicle can drive in one minute, based on the given speed by our system. This script includes a "ground truth" for each frame in each dataset. If our speeds exceed this ground truth, the vehicle crashes and is not able to continue driving.

In section 3.7 two strategies were presented. These use a combination of both classification and edge detection. First, the evaluation was run separately on these two methods. The results are shown in figure 14. The first bar of the histogram shows for every dataset the performance of road classification. Because of a crash in the first dataset there can only be traveled 317 meters (out of the total 1350 meters in this dataset). The second bar shows the distance traveled only relying on edge detection for each dataset. There are no crashes in the four datasets so we can say these results are way better than the ones only using classification. This makes us believe the combination strategy that relies more on edge detection (strategy 2) will be the better one.

The first way of combining the two methods is very safe and thus results in no crashes over the four datasets. A region is considered to be road only if it's identified as road by the road classification and if there are no edges. Both classification and edge detection should indicate a pixel as road in order to be used for safe driving. The performance of this combination is shown in the third bar of figure 14. As said, this strategy is very safe, giving up a lot of traveled distance in favor of safety.

The second strategy for combination also results in zero crashes over the four datasets. This strategy tries to focus on edge detection. A region is considered road if there are no edges. When there are edges, road classification is used to decide whether it's road or not. This strategy gave us the best results, as shown in figure 14 as the fourth bar.

5 Conclusion

In this paper we combined SVM classification with edge detection in order to calculate the maximum speed a vehicle can drive, considering possible obstacles on the given trajectory.

The SVM classification shows optimal results when using color histogram values for the feature vector. These feature vectors are calculated for each block of 32x32px in an image. These results tend to get better when we include road marks as part of the road in the training data. This is also optimal for use in our application as we don't want to detect road marks as obstacles.

To perform edge detection, the frame is first treated with morphological processing operations to remove insignificant details. In our application we used Canny Edge Detection to detect the side of the road and possible obstacles.

In order to combine the SVM classification with the edge detection, two strategies were proposed. The first strategy simply intersects the SVM classification results and edges to detect the nearest obstacle in the trajectory. This method show to be the safe option, by often falsely detecting an obstacle. Results show that the best strategy is to first detect all the edges and then classify these edges as being part of the road or not, using the SVM classifier. Only edges which are not considered part of the road remain. The maximum speed is then calculated based on the nearest edge in the trajectory of the car.

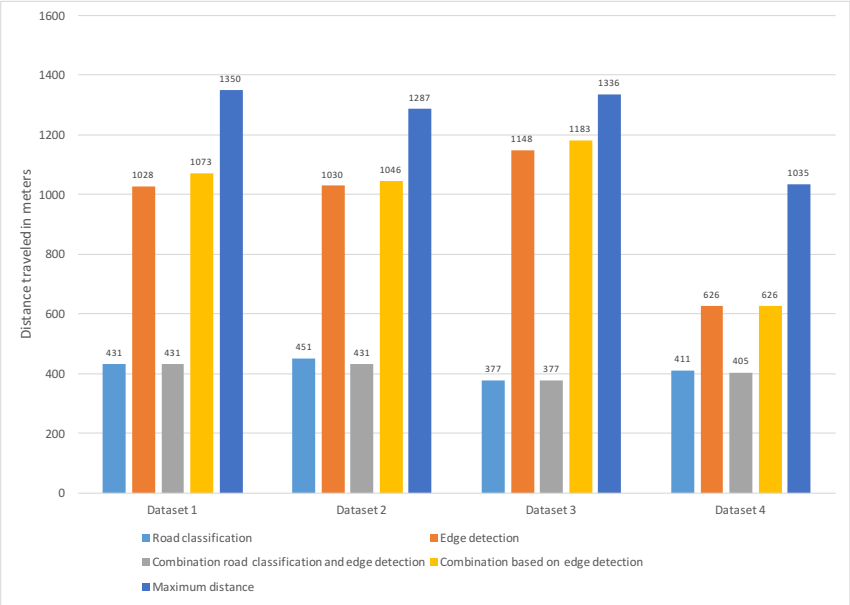


Fig. 14: Results of road classification, edge detection and combinations.