Statistical classification of drivable roadways

Maurice Frank,[*] *Ruprecht-Karls-Universität Heidelberg*
Supervisor: ***Miguel Bautista,*** *Heidelberg Collaboratory for Image Processing*

29 February 2016

1 Introduction

As automated driving assistance and fully automated car systems get more relevant and prevalent, intelligent object classification in the surroundings of roadways becomes increasingly important. Therefore road detection, which means that we visually classify the area in front of a car as drivable or not-drivable, is one of the basics task that has to be solved. In recent years multiple methods were proposed with different focus on precision, time efficiency or flexibility. This final report about the practicum I did at the Heidelberg Collaboratory for Image Processing describes such a method. Using multiple structural features two classifiers are trained from training data. These trained models are used to segment frames of videos. Training and predicting is done on videos from a camera that is fixed at the windscreen of a car. The general difficulty with this approach is that roadways are very diverse and the conditions under which the video is taken are changing constantly. A road can be a highway with clearly defined roadside or a gravel path without markings and varying shapes and the video can be recorded under a strong shadow casting sun or through rain. Therefore a method has to be more flexible successful then with for example simple supervised topological structuring.

2 Algorithm

The implemented method consists of four steps. First the given training data, which consists of video frames with labeled parts, is used to build a set of feature vectors using two descriptors. This set is the basis for the training of a random forest and a linear classifier. The models then classify the frames by being applied on a sliding window resulting in a heat map of drivable roadway. In a fourth step the generated heat map gets refined using a series of morphological operations.

[*]E-Mail: Maurice.Frank@stud.uni-heidelberg.de

2.1 Training data

The training data consists of two parts. First a series of eleven videos that show posed car rides with obstacles and interfering actors. The frames are present as compressed grayscale images with a size of $960 \cdot 540$px. Second for each, except the first, video, a hand labeled set of positive and negative rectangle image parts. These boxes are overlapping and of varying size. As we want feature vectors of the same size the boxes are resized to a standard width of 51px. Then the part of the corresponding frame the box bounds is gathered and the features are computed. Combined with the label this makes one testing instance.

2.2 Used features

Two features are computed in the process of training and predicting. To describe the structural information found in the testing data a histogram of oriented gradients[1] (HOG) is used. With this calculation the

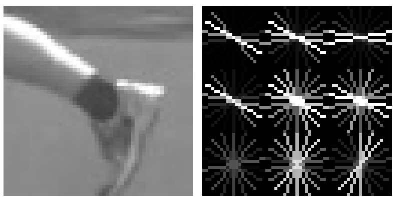


Figure 1: 51px wide extract and its HOG

image is divided into overlapping blocks and on these blocks a 1-D kernel is applied vertically and horizontally. Then the pixels in the blocks are weighted and so vote for the intensity of one of nine directions of the gradients.

Furthermore we want to use the similarity of intensity distribution in positive samples and therefore add a color histogram to the feature vector. For that, pixels of the same intensity are counted and the counts are ordered by the intensity.
So for a box we have 9 HOG cells and with the standard of 9 orientations we get $9 * 31 = 279$ dimensions

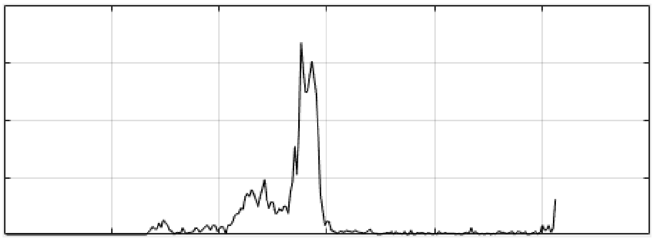


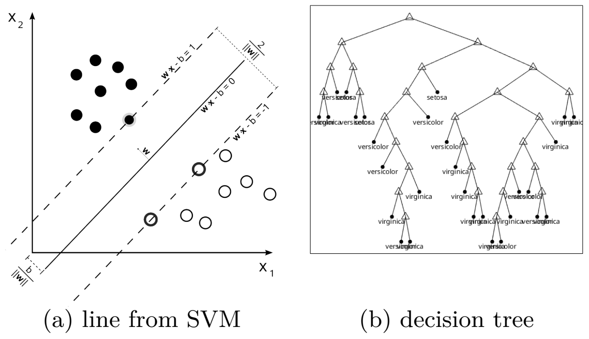
Figure 2: The histogram for the image in Figure 1

for the first feature. The color histogram has 256 dimensions. We then have $279 + 256 = 535$ dimensional feature vector.

2.3 Classifiers

I tested multiple classifiers for this task and this report describes the results with the linear classifier `LibLinear`[2] and with a random forest[3]. In a earlier stage I tried to use a not linear support vector machine (SVM) with a RBF kernel. Due to the technical limitations especially in memory it was not possible to train a model with enough data to make it usable. With change to a linear classifier I also got huge speed improvements. A linear classifier tries to find a hyperplane in the 535 dimensional feature space, the training points life in, that separates two classes. `LibLinear` is used with a L2-regularized SVM classifier as the solver. In that method the solver tries to maximize the distance between the training points and the hyperplane. A simple example of a separated set is in Figure 3(a). The cost parameter was set at 100 through training and evaluating at a range of values. This parameter tells the classifier how many misclassifications it can accept while training so a higher cost becomes a hyperplane with smaller margins.

For the random forest the implementation from `MATLAB` is used. A random forest is a special decision tree. Decision trees classify an instance by looking at every deeper node at a feature value so that lastly all leafs are either associated with the one class or the other. A small tree is shown in Figure 3(b). This is done for all trees in the forests and the most chosen label wins. In training the tree is build by choosing a random part of the feature vector ($\sqrt{length(instance)}$) and making a split based on the best splitter. By splitting, trees are built and in the end the trees are combined to the forest which is the



(a) line from SVM (b) decision tree

Figure 3: Example for the classification methods

classifier. The number of trees is set at 30 after testing at different values.

2.4 Computing segmentation

The general idea to get predictions for a frame is to slide a window of varying size over the image and sum up the predicted classes. After testing I have decided for window sizes of 50, 60 and 70px. So for each of this sizes we slide the window from the top-left to the bottom-right while pushing the window by the half width after each step which makes the same sized windows overlapping. In each step the bounded image part is resized to the 51px and the HOG and histogram is calculated. After a size pass the resulting feature vectors are collected in a matrix and then passed to the predict function of the given model. For the 3 chosen sizes we then get $325 + 480 + 703 = 1508$ instances for a frame from the testing videos. To generate the heat map we now just increment the values in the belonging window of original size if the label was positive. In a last step the heat map is normalized to the intensity range of $[0, 1]$.

2.5 Morphology

The heat map from the predict function is not the practical result we aimed at. As one can see for example in the predicted heat map in Figure 5(b) we get separated misclassified regions in the sky or in the houses. Also as we add up the predicted labels for the sliding windows the heat map of course is not binary but has a intensity range. Furthermore the idea with this method is to narrow the wide-scrawled predictions by using information from a edge detector. Shown in

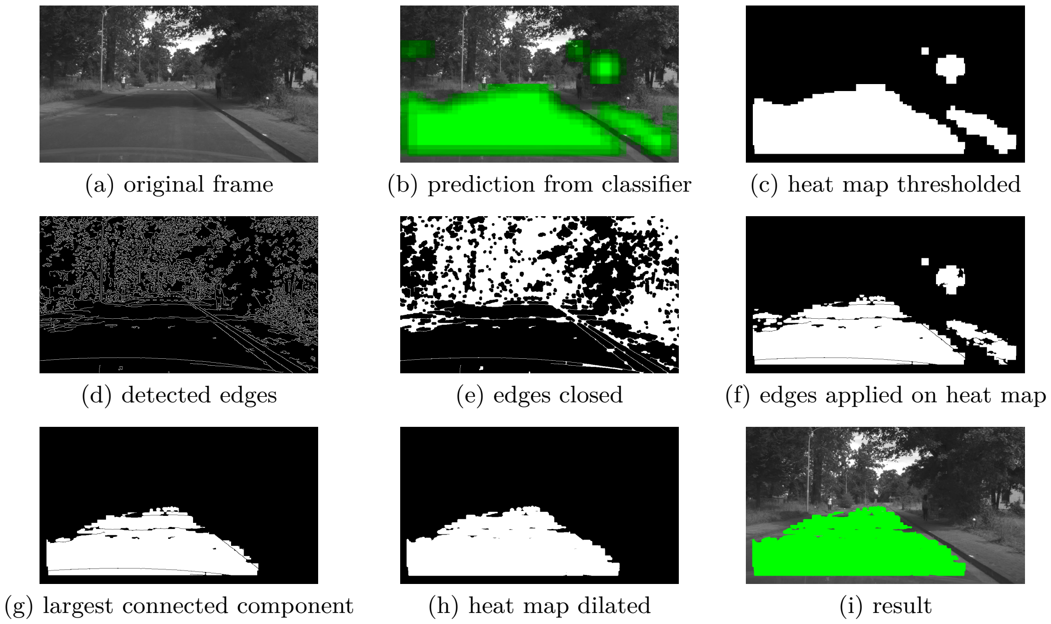

(a) original frame (b) prediction from classifier (c) heat map thresholded
(d) detected edges (e) edges closed (f) edges applied on heat map
(g) largest connected component (h) heat map dilated (i) result

Figure 4: `seq0002/I00875.jpg`: Prediction and morphing with `LibLinear`

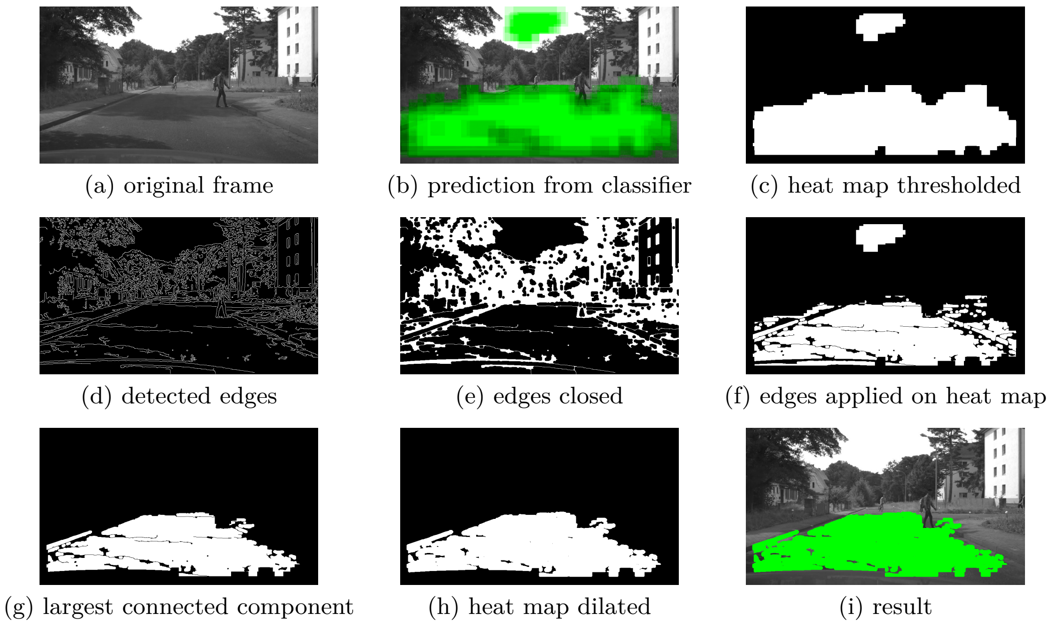
the two examples Figure 4 and Figure 5 you see the seven steps used on the prediction. First the heat map is thresholded with a fixed value to get a segmentation(c). Then we compute the edges for the original frame with the canny edge detector(d). Here we ensure fine-grained edges as we assume the roadway has few strong gradients but the surroundings have these. When we then morphologically close the edges(e), the fine-grained structures present in the surroundings of the street become ideally one face or at least of big connected component. As the heat map and the edges are present as binary images we can multiply the heat map with the complement of the edges to subtract the edges from the heat map(f). In the fifth step the largest connected component is selected and so possible lone misclassifications are discarded(g). Probably the now remaining component has fine incisions. For the last morphological step the heat map gets dilated so these small cuts at least get filled(h). The finished morphed classification is reapplied on the original frame to get a frame for the result video(i).

3 Analysis

Using the described algorithm, four models were computed. Two using the linear classifier and two using the random forest. As ten videos have training data the dataset was split into training and testing sets. One linear and forest model was trained with four videos (named A) and one different four videos (named B). Because of memory restrictions in the working machine the sets could not be larger. With these models the training videos were processed and the predictions saved.

3.1 Labeling performance

To get some performance measurement all four models were used on the complementary training data. Therefore the original labeled boxes were put into the models and the predicted labels were compared with the original label. Both methods only predict classes and no floating point score so one data-wide precision and recall value was calculated. The precision is the fraction of the as road predicted instances that are actually road. The recall is the fraction of all road instances that were predicted. Additionally we look at the F-measure. As seen in Table 1 all ratings are above 0.98 and therefore show that all four models reached a high accuracy in predicting the complementing testing data. The random forest trained on videos 5 to 8 performed slightly better then the other three. Nevertheless is this performance not inevitably transferred into the visually seen performance to predict the actual roadway shape. Without a more, near pixel accurate labeling of the roadways the actual desired classification performance can not be measured.



(a) original frame (b) prediction from classifier (c) heat map thresholded
(d) detected edges (e) edges closed (f) edges applied on heat map
(g) largest connected component (h) heat map dilated (i) result

Figure 5: `seq0007/I00415.jpg`: Prediction and morphing with the random forest

	linear A	forest A	linear B	forest B
Precision	0.9865	0.9899	0.9854	0.9893
Recall	0.9938	0.9899	0.9942	0.9923
F-Measure	0.9901	0.9899	0.9898	0.9908

Table 1: Performance measurements for the models

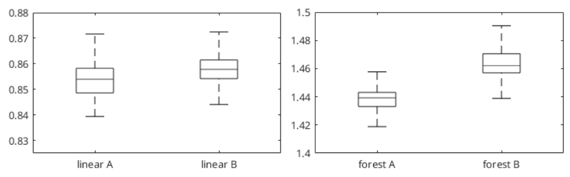


Figure 6: Ranges of processing times in seconds

3.2 Time performance

The application of such algorithms in the real world strongly depends on good time performance. A car without real time information processing and decision system is mostly pointless. Also because of my own technical limitations there was a strong focus on processing speed. Stopping the time for reading the frame, making the prediction and applying the Morphology but without possible saving operations the ranges of time periods is visualized for the four models and `seq0000` in Figure 6. `LibLinear` performed clearly faster in both trainings set cases than the random forests. This depends mostly only on the actual prediction time the libraries take. All models process a frame in under 1.5sec. With linear classification the prediction takes way under a second which was aimed at. Taken the nearly same labeling performance into account the speed seems a huge drawback of using a random forest.
Most task were able to be calculated in parallel on the multi-core CPU but as I have seen the factor of around $\frac{1}{4}$ is not as helpful as an focus on a fast implementation.

4 Conclusion

In conclusion the practicum was really interesting to put work in. But even though the labeling performances are outstanding that solely comes from missing possibilities to test its accuracy. Also the most important drawback is the huge similarity of the training data. As written in the introduction this task asks for a method that can detect nearly any road under many different environmental conditions. There is no performance measuring done for a different environment but it is to assume that the results would be way worse. Using HOGs is a good way to get texture and color invariant information about what makes a road. But also they are not rotation-invariant what strongly limits their ability to predict in different environments. Moreover the reliance on the intensity histogram is difficult. A red brick street with a lot of shadows is a street that happens to exist often but has a entirely different intensity distribution. One possible way to improve this method would be to implement a additional on-line step which considers the last frames for the next heat map. In that you could gain speed and a sharper labeling performance.

At the end we want to at least see the best classifier against one image outside the test data:

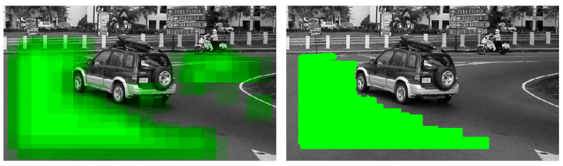


Figure 7: Random toy example from the internet

For computation it was always used:

Intel Core i5-3210M @ 2.50GHz

8022MB memory

MATLAB R2015b 64-bit

References

[1] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 9, pp. 1627–1645, 2010.

[2] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[3] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.