Autonomous Driving: Road-Estimation

Patricio Córdova, Yuqing Du
University of Toronto
27 King's College Circle, Toronto, ON M5S, Canada
patricio@cs.toronto.edu, yuqing.du@mail.utoronto.ca

Abstract

Road estimation is one of the main problems when designing a navigation system for autonomous driving. The current work tries to approach the problem by using computer vision and machine learning techniques. We tackle the problem by segmenting the road images, extracting relevant features that define a road, and modeling the relations between these features using machine learning models. The segmentation is performed by extracting superpixels from the original image using the Simple Linear Iterative Clustering (SLIC) algorithm. The main features we extracted from the superpixels are color, texture, location, histogram of gradient, size, and some more advanced features like distance between superpixels, color edges, etc. Then, we compare different machine learning models like Supporting Vector Machines (SVM), K-Nearest Neighbors (k-NN), AdaBoost, Random Forrest, and other models based on the generated features. Also, we use some more advanced techniques like Conditional Random Fields (CRF) and Structured SVM which model the relations between superpixels and aim to generate smoother classification boundaries. Finally, we evaluate the results. Among the non-structured models we achieved more than 94.5% accuracy by using the Random Forests model, and for the structured models, we reached more than 95% classification accuracy by using CRF Graph model.

1 Introduction

One of the main problems when trying to implement a navigation system for autonomous driving, based on computer vision techniques, is to detect objects like roads, pedestrians, other automobiles, traffic signs, etc. The current work tries to approach the first problem: road estimation, by analyzing sample road images. This is a binary classification problem in which we want to determine if a pixel in an image belongs to a road or not.

Detecting a road from an unstructured source, like an image, is a challenging problem mainly because of the variability of factors that influence how a road looks like. These can be environmental factors like changes in illumination due to weather conditions, time of the day, etc. Also, there are other factors like shadows generated by trees or buildings, obstacles on the road like other vehicles or pedestrians, road color variability, road shape shape variability, etc.

Machine learning is the top choice to solve this kind of problem. With the correct number of samples, and the correct distribution of samples (to capture the variability factors aforementioned), we can generate robust models to extract the patterns that define a road.

2 Approaches and Previous Work

With respect to feature extraction from superpixels, Tighe et al. [5] suggest to extract the superpixel shape, location of the superpixel in the image, texture histogram, RGB color mean and superpixel

thumbnail (appearance). Although Alvarez et al. [8] challenge the use of such factors arguing that the manual extraction of features like texture, color, etc. incurs high computational time in the inference process; for our project we decided to proceed with the approach suggested by Tighe et. al: we extracted all the features just mentioned, except appearance.

In order to solve the problem of road detection, different approaches have been proposed. The main types of approaches fall into the following main categories: algorithms that try to perform the road detection using mathematical models on top of specific features, like color [2, 3], and algorithms that perform road detection using feature extraction and machine learning models on top of them [1, 4]. Regarding the mathematical approach, it's not easy to build a model. Although simple models are robust, they are less accurate and more difficult to generalize to different scenarios: different illumination conditions, different road shapes, etc.

Regarding the machine learning approach, it's more robust and generalizable when the amount of training data is large and varied enough. State-of-the-art research suggest the use of models like Conditional Random Fields (CRF) and Structured Supporting Vector Machines (SSVM) in order to get the most optimal results.

Passani et. al [12] suggest to employ CRFs on top of a grid-shaped model (see Fig. 1) to define the labels of the superpixels and their relationships. More specifically, the nodes of this undirected graph correspond to a lattice of pixels in a 4-neighborhood. At each node i they introduce an output random variable y_i, taking a value from a set of labels L corresponding to the classes of interest, and the observable random variable x_i representing the value of local features.

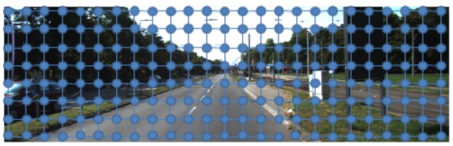

Figure 1: Graph of a CRF aligned with the images lattice of pixels [12]

More formally, the probability distribution can be factored in a product of unary (ψ_i) and pairwise (ψ_{ij}) potentials, also conditioned on $\mathbf{w}$:

$$p(\mathbf{y}|\mathbf{x};\mathbf{w}) = \frac{1}{Z(\mathbf{x};\mathbf{w})}\prod_i \psi_i(y_i,\mathbf{x};\mathbf{w}_i)\prod_{i,j}\psi_{i,j}(y_i,y_j,\mathbf{x};\mathbf{w}_{ij}) \quad (1)$$

The first product is over all individual pixels i, while the second one is over the edges in the graph which model neighboring pixels interactions. The optimal vector of parameters $\mathbf{w}$ is obtained by a supervised learning process that minimizes a loss function [9]. Once these parameters are computed from the training data, the model is applied for inferring pixelwise labels on test images.

3 Feature selection

Feature selection is a way of preprocessing the images that enables us to model relevant road characteristics, and is vital for the quality of our results. We tested different combinations of features and decided on a particular combination, further explained, which gives the best performance. The five different features we considered are:

3.1 Basic features

Intuitively, color is the most informative feature in this case. Since we are performing superpixel-based estimation, we took the average RGB values of all pixels within each superpixel. Generally, the color of a road can distinguish well between the road and its surroundings, but the appearance of shadows, and other objects like sidewalks, can somehow make the prediction rather confusing.

Apart from color, as we go through the images, we see that the road in most images is located at the centre bottom side of the image. This means that position is another representative feature.

Similarly, we selected the average X and Y coordinates of pixels in each superpixel as the location feature.

Furthermore, since different superpixels have different sizes (i.e. superpixels on the edge of the road tend to have smaller sizes than those in the centre), we simply count the number of pixels inside each superpixel and use it as size feature. This is more like a discriminant factor that tries to add smaller weights to pixels that have different sizes, than the ones belonging to a road.

3.2 Histogram of Orientation Gradient (HoG)

HoG is a popular feature used in object detection. The basic idea is that the local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. It is achieved by dividing images into small connected regions where HoG is calculated for pixels within each region. Moreover, this feature has the advantage that it emphasizes locality. Neighbour pixels tend to belong to the same class if their HoG difference is small. For extracting this feature, we used the Python library scikit-image [13] directly.

3.3 Texture

After observing the images, we find that the texture of a road can be quite different from its surroundings, especially compared to houses and trees. Roads tend to be flat and smooth, whereas trees and houses have more complex and compound textures. We use scikit-image GLCM texture feature which computes correlation and dissimilarity among pixels inside each superpixel. Like HoG, texture also gives us a smooth measure of the relation between labels.

3.4 Features Combination

Table 1: Feature Selection

	CLSHT	CL	CLH	CLT	CLHT	CLS
k-NN	93.05%	93.15%	93.18%	92.90%	93.18%	93.38%
Adaboost	92.86%	92.44%	92.82%	92.45%	92.82%	92.45%
RandomForest	93.67%	93.41%	93.53%	93.56%	93.44%	93.40%

We experimented with different combinations of features and then compared them. The results are shown in Table 2. Here "C" refers to color, "L" for location, "H" for HoG, "T" for texture and "S" for size. We tested the features using k-NN, AdaBoost and Random Forests models. From the results we can see that the combination CLSHT (Color, Location, Size, HoG and Texture) leads to the best performance, thus we decided to use it in our experiments.

4 Superpixels

Considering that the number of pixels in each image could be more than 5000,000, running the models at pixel level can be very costly. To alleviate the problem, we use the Simple Linear Iteration Clustering (SLIC) technique to compute the superpixels. This allows us to perform the classification by extracting features at superpixel level. There are two important parameters when parametrizing SLIC: the number of superpixels and the compactness. Compactness means the smoothness of the superpixel shapes, bigger compactness produces similar superpixel shapes. We tried out different superpixel extraction parameters. We show the results of our experiments in Tables 2 and 3.

From the results we conclude that by extracting 500 superpixels, at 30 compactness factor, we can get the best results. Therefore, these are the parameters used for our experiments.

Table 2: Superpixel extraction comparisons. Upper left: original image, upper right: 500 superpixels (sp) extracted with 30 compactness factor (c), lower left: 400sp-30c, lower right: 200sp-20c.

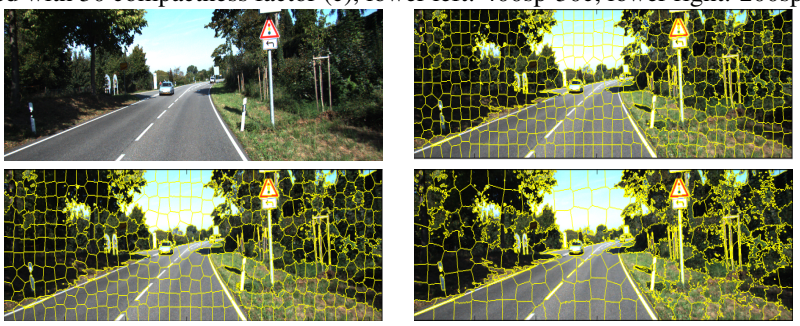

Table 3: Superpixel parameters

	500-30	400-30	200-20
k-NN	93.05%	92.53%	92.20%
AdaBoost	92.86%	92.63%	91.31%
Random Forests	93.78%	93.74%	92.53%

5 Models

5.1 Naive Bayes

Naive Bayes is a simple and fast generative classifier based on Bayes theorem. It holds strong assumptions that both superpixels and different features are independent, but in this case when neighbour superpixels are closely related, it oversimplifies the classification resulting in a relatively poor estimation. Figure 2 is a sample output estimation. It basically shows the worst performance compared to other models we tried. Also, since the superpixels are treated independently, the prediction is not smooth. We see superpixels located in the middle of the road being predicted as non-road probably because of the white lane line.

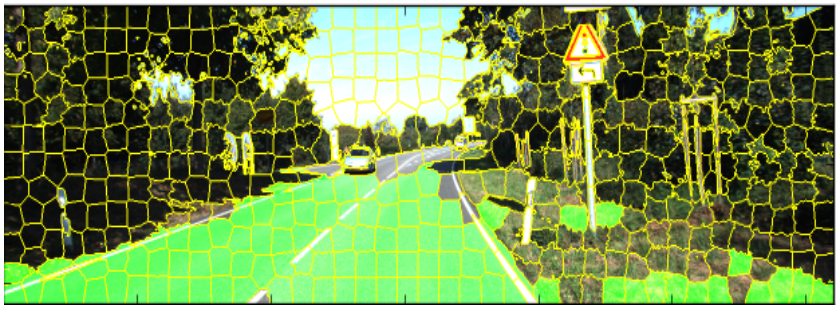

Figure 2: Naive Bayes Road Estimation.

5.2 K-Nearest Neighbors

K-Nearest Neighbors (k-NN) is a non-parametric method used for classification. It memorizes the training data during training and performs the classification by finding the k closest training samples to the new sample during the testing phase. The output class membership is decided on the major class membership of the k nearest samples.

To parametrize k-NN, setting the k and $distance$ parameters is an important step. During experiments we find that using manhattan_distance instead of euclidean_distance gives better predictions.

We then tried different k values on validation data using manhattan_distance. The results are showed in Figure 3. Estimation accuracy reaches its peak value when $k = 15$. After using these settings, the sample output in Figure 4 shows relatively good accuracy except for some superpixels near the edges. Unlike Naive Bayes, no superpixels in the middle of the road were misclassified since the decision of each superpixel is made purely based on the k neighbour superpixels.

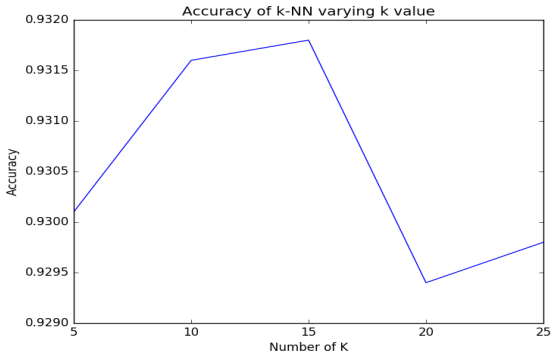

Figure 3: k-NN accuracy with different *K* values

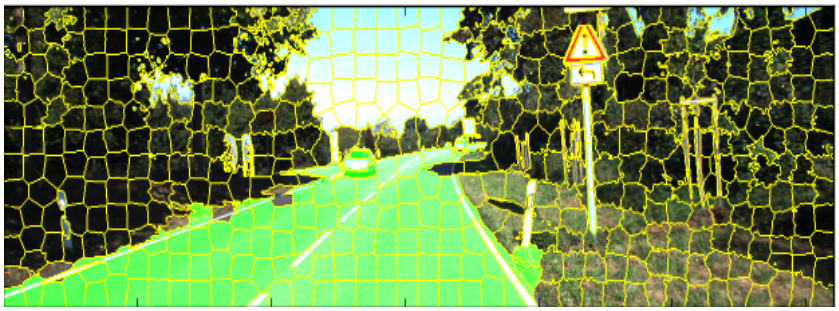

Figure 4: k-NN Road Estimation

5.3 AdaBoost

AdaBoost is an ensemble method which combines weak classifiers to form a stronger one. AdaBoost focuses on hard examples by giving bigger weights to misclassified examples during previous training cycles. The output of the model is a weighted sum of outputs of each weak classifier.

We use the scikit-learn package [13] and the AdaBoost (ADA) routine to perform the classification. We adjust the number of classifiers, Figure 5 shows the validation accuracy after adjusting the number of classifiers. We see that as the number of estimators increase, the performance gets better. However, when it exceeds 100, the performance goes down mainly because of overfitting. Therefore, 100 estimators are used for AdaBoost. See Figure 6 for a sample output.

5.4 Random Forests

Random Forests is another ensemble method which operates by constructing a multitude of decision trees at training time and outputting the class that is the mean prediction of the individual trees. Compared to AdaBoost, it avoids overfitting the training set by largely randomizing input data. Here we also tried different number of estimators and we show in Figure 7 how 100 estimators give the best performance. See Figure 8 for the sample output.

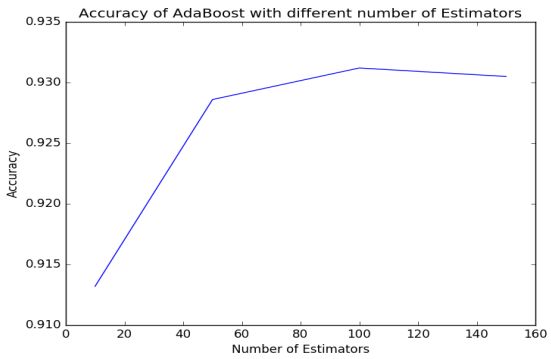

Figure 5: AdaBoost with different number of estimators

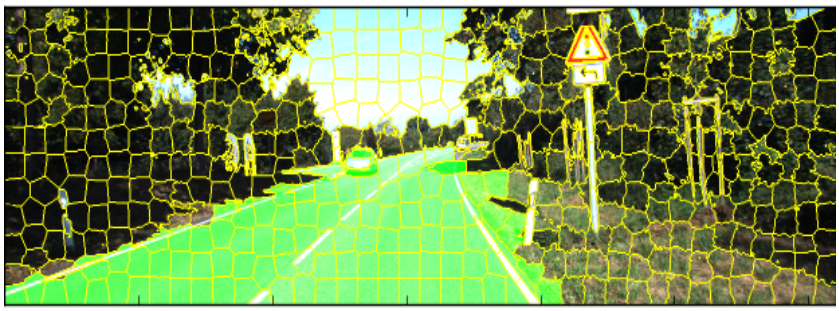

Figure 6: AdaBoost Road Estimation

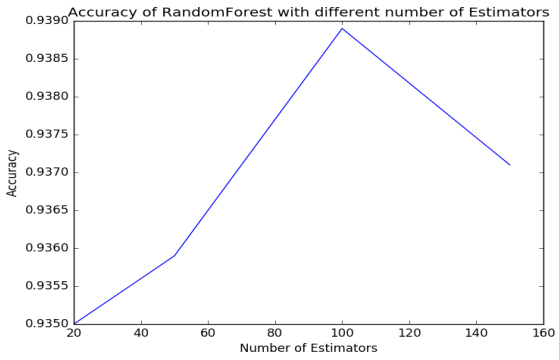

Figure 7: Random Forests with different number of estimators

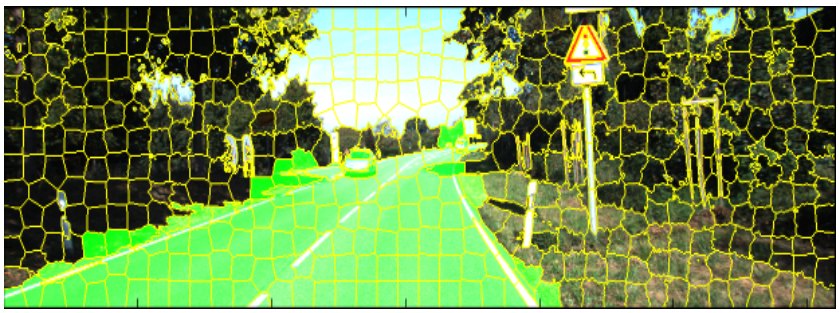

Figure 8: Random Forests Road Estimation

5.5 Support Vector Machines

Support Vector Machines (SVM) is a discriminative classifier which maximize the margin between data near the classification boundaries. Because it only cares about samples near the margin, it usually avoids the problem of overfitting. Given the case when data is not linear separable, there are kernel functions to help alleviate the computation when mapping the data into some higher dimensional, linear separable, space. In this case, we use Radial Basis Function (RBF) kernel provided in the scikit-learn library. RBF is a popular SVM kernel and a real-valued function whose value depends only on the distance from the center.

Unfortunately, the major downside of SVM is that it can be painfully inefficient to train compared to other models and the prediction, although good, is certainly not the best (see the results section). Figure 9 gives a sample estimation.

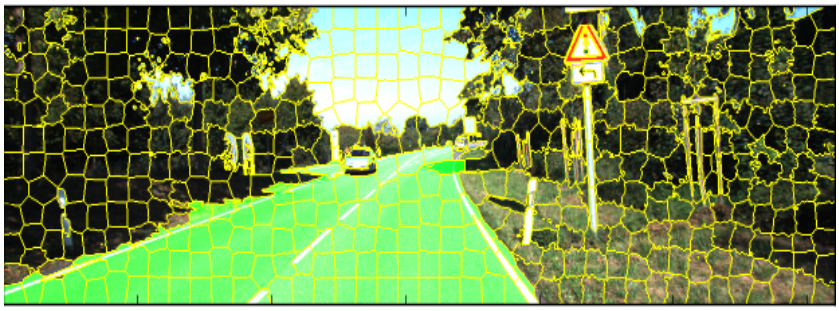

Figure 9: SVM Road Estimation

5.6 Quadratic Discriminant Analysis

Quadratic Discriminant Analysis (QDA) is a closely related model to linear discriminant analysis. This classifier has the advantage that it has closed-form solutions which can be easily computed. It also can result in relatively good performance, and it has no hyper-parameters to tune so it's easier to train. See Figure 10 for a sample output.

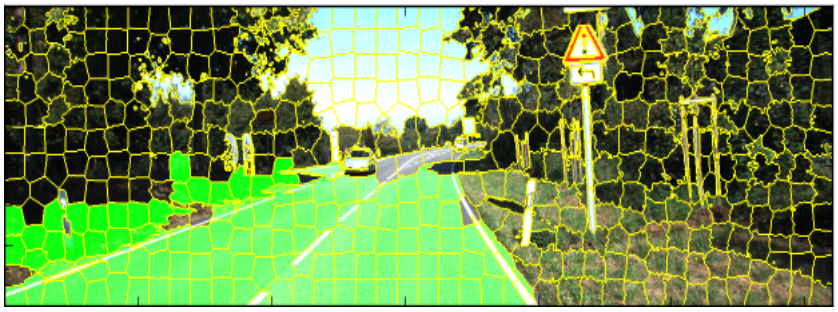

Figure 10: QDA Road Estimation

5.7 Structured Learning

Structured prediction is a generalization of the standard paradigms of supervised learning, classification and regression. In structured prediction the goal is not to predict a single label or a number, but a possibly much more complicated object like a sequence or a graph, in which the prediction of an element depends on "nearby" predictions.

5.7.1 Conditional Random Fields

Conditional random fields (CRFs) is a probabilistic framework for labeling and segmenting structured data, such as sequences, trees and lattices. The underlying idea consists in defining a conditional probability distribution over label sequences given a particular observation sequence, rather than a joint distribution over both label and observation sequences.

More formally, for observations set $\boldsymbol{X}$ and random variables set $\boldsymbol{Y}$, $(\boldsymbol{X}, \boldsymbol{Y})$ is a conditional random field when the random variables $\boldsymbol{Y_v}$ conditioned on $\boldsymbol{X}$ obey the Markov property with respect to the graph G:

$$p(\boldsymbol{Y_v}|\boldsymbol{X}, \boldsymbol{Y_v}, w \neq v) = p(\boldsymbol{Y_v}|\boldsymbol{X}, \boldsymbol{Y_v}, w \sim v)$$

where $\boldsymbol{Y} = (\boldsymbol{Y_v})_{v \in V}$ and $\boldsymbol{w} \sim v$ means that w and v are neighbors in G [7].

For our project, we use the Structured Supporting Vector Machine (SSVM) model to fit the CRF Graph. SSVM is a Support Vector Machine algorithm for predicting multivariate or structured outputs. Unlike regular SVMs, however, which considers only univariate predictions like in classification and regression, SSVM can predict complex objects too.

Our approach is to generate two CRF models: the first one considers only the superpixel edges. The second one is based on the extraction of feature edges like color and distance, and mixing them with the regular edges. We use the Pystruct pairwise CRF on a general graph class (GraphCRF) for the first model, and the pairwise CRF class with features/strength associated to each edge (EdgeFeatureGraphCRF) for the second model. We use the non-structured models that produced the best classification results: Random Forrest, AdaBoost and k-NN, as the unary inputs for the CRF model. The results are presented in the results section. Figure 11 shows a sample output.

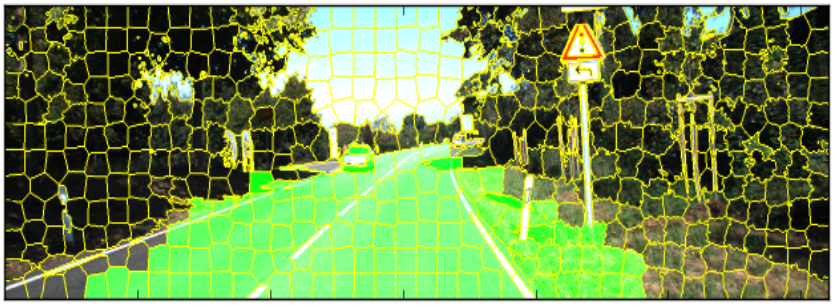

Figure 11: CRF Road Estimation

6 Experiments

6.1 Dataset

We use the base kit of KITTI [10] with: left color images, calibration and training labels data. The kit consists of 289 road images and 290 labeled images. There are three different categories of road scenes: uu - urban unmarked (98 images), um - urban marked (95 images), umm - urban multiple marked lanes (96 images). The marked images are images with multiple lanes divided and marked. For labels, we use only the road ground truth.

We split the training data randomly into training (60%), validation (10%) and testing (30%) for the purpose of our experiments.

6.2 Evaluation Metrics

Since we performed a superpixel based classification, we tried to look also at the pixel level accuracy. However, after some experiments we saw that there were no big differences between superpixel accuracy and pixel accuracy, thus we decided to focus only on superpixel evaluations.

Following the standard evaluation method of the KITTI dataset, we used **Precision**, **Recall** and **F1-Meaure** as the model performance evaluation metrics. The computation of these three metrics are given as follows:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 - Measure = \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

Apart from the model performance evaluations, we also considered the running time of the models, which is a simple indication of their complexity.

7 Results

For our project we use the Python scikit-image [13], scikit-learn [6], and pystruct [11] libraries.

After running the models on test set, with the best parameters detailed in the former section, Tables 4 and 5 show the results. Among the non-structured classification algorithms, Random Forests and SVM give the best performances in all three metrics. And regarding training time, SVM is the slowest model among all models tried, as it could have been predicted since it involves heavier computation. Naive Bayes was the fastest model, which could also be foreseen due to its simplified assumptions. Generally speaking, considering both training time and classification performance, Random Forests and AdaBoost are the best classifiers within the non-structured classifiers category. We can conclude that ensemble methods are very efficient for this problem.

By comparing non-structured methods with structured methods, we can tell that structured methods have even better performance than the best non-structured models. This can be seen by understanding the fundamental goal of structured methods. If we look the pictures as graphs of superpixels, and we consider that the road is a portion of the graph, by estimating the relationship between the nodes that belong strictly to the road portion, we can generate a more connected and thus smoother prediction. Table 4 show our CRF classification results.

Table 4: CRF Graph model results based on Random Forrests, AdaBoost and k-NN models.

Graph	Base Model	Time	Precision	Recall	F1-Measure
Edges	RF	57.086s	95.07%	94.97%	95.01%
Edges	AdaBoost	55.230s	93.98%	93.56%	93.70%
Edges	k-NN	54.821s	94.69%	94.50%	94.57%
Edges, Color, Distance	RF	90.496s	94.17%	94.22%	94.09%
Edges, Color, Distance	AdaBoost	125.377s	90.88%	90.45%	89.97%
Edges, Color, Distance	k-NN	91.654s	93.15%	93.21%	93.02%

From observation of the data and the classification results, we believe that the misclassifications can be caused by shadows on the road, white lane lines, sidewalks, objects on the road, and other factors which are difficult to eliminate. As we can see from the output images, the misclassified superpixels are usually along the edges of the road, or in the case of Naive Bayes, the superpixels over the white lane lines in the middle of the road. Our results can be seen in Table 5.

8 Conclusion

In this project, we compared the performance of different classifiers in the task of road estimation, using both non-structured models and structured models. Our work was carried out through 4 phases: feature selection, superpixel setting, models training and evaluation of performance. The best performance of our non-structured models reached more than 94.5% using a Random Forrest model with 100 estimators, and the best performance for the structured models was achieved by CRF, improved on top of the Random Forrest classification results, and fitted using SSVM. It reached more than 95% accuracy.

Table 5: Experiment final results

	Time	Precision	Recall	F1-Measure
SVM	464.961s	93.94%	93.99%	93.96%
NB	0.034s	89.35%	88.91%	89.09%
k-NN	0.145s	93.29%	93.30%	93.30%
AdaBoost	13.424s	93.44%	93.43%	93.43%
RF	22.494s	94.61%	94.67%	94.63%
QDA	0.0937s	90.58%	88.09%	88.72%
CRF Edges	57.086s	95.07%	94.97%	95.01%

References

[1] Mike Foedisch and Aya Takeuchi. "Adaptive real-time road detection using neural networks". In: *Proceedings of the 7th International IEEE Conference on Intelligent Transportation Systems, Washington, DC*. Vol. 396. Citeseer. 2004.

[2] Yinghua He, Hong Wang, and Bo Zhang. "Color-based road detection in urban traffic scenes". In: *Intelligent Transportation Systems, IEEE Transactions on* 5.4 (2004), pp. 309–318.

[3] Miguel Angel Sotelo et al. "A color vision-based lane tracking system for autonomous driving on unmarked roads". In: *Autonomous Robots* 16.1 (2004), pp. 95–116.

[4] Sha Yun, Zhang Guo-Ying, and Yang Yong. "A road detection algorithm by boosting using feature combination". In: *Intelligent Vehicles Symposium, 2007 IEEE*. IEEE. 2007, pp. 364–368.

[5] Joseph Tighe and Svetlana Lazebnik. "Superparsing: scalable nonparametric image parsing with superpixels". In: *Computer Vision–ECCV 2010*. Springer, 2010, pp. 352–365.

[6] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[7] Charles Sutton and Andrew McCallum. "An introduction to conditional random fields". In: *Machine Learning* 4.4 (2011), pp. 267–373.

[8] Jose M Alvarez et al. "Semantic road segmentation via multi-scale ensembles of learned features". In: *Computer Vision–ECCV 2012. Workshops and Demonstrations*. Springer. 2012, pp. 586–595.

[9] Jens Domke. "Learning graphical model parameters with approximate marginal inference". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35.10 (2013), pp. 2454–2467.

[10] Jannik Fritsch, Tobias Kuehnl, and Andreas Geiger. "A New Performance Measure and Evaluation Benchmark for Road Detection Algorithms". In: *International Conference on Intelligent Transportation Systems (ITSC)*. 2013.

[11] Andreas C. Müller and Sven Behnke. "pystruct - Learning Structured Prediction in Python". In: *Journal of Machine Learning Research* 15 (2014), pp. 2055–2060. URL: http://jmlr.org/papers/v15/mueller14a.html.

[12] Mario Passani, J Javier Yebes, and Luis M Bergasa. "CRF-based semantic labeling in miniaturized road scenes". In: *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*. IEEE. 2014, pp. 1902–1903.

[13] Stéfan van der Walt et al. "scikit-image: image processing in Python". In: *PeerJ* 2 (June 2014), e453. ISSN: 2167-8359. DOI: 10.7717/peerj.453. URL: http://dx.doi.org/10.7717/peerj.453.