Road Junction Detection from 3D Point Clouds

Danilo Habermann[1], Carlos E.O. Vido[2], Fernando S. Osório[1], Fabio Ramos[2]

Abstract— **Detecting changing traffic conditions is of primal importance for the safety of autonomous cars navigating in urban environments. Among the traffic situations that require more attention and careful planning, road junctions are the most significant. This work presents an empirical study of the application of well known machine learning techniques to create a robust method for road junction detection. Features are extracted from 3D pointclouds corresponding to single frames of data collected by a laser rangefinder. Three well known classifiers—support vector machines, adaptive boosting and artificial neural networks—are used to classify them into "junctions" or "roads". The best performing classifier is used in the next stage, where structured classifiers—hidden Markov models and conditional random fields—are used to incorporate contextual information, in an attempt to improve the performance of the method. We tested and compared these approaches on datasets from two different 3D laser scanners, and in two different countries, Germany and Brazil.**

I. Introduction

For the last few years, research in unmanned ground vehicles (UGVs) has been growing steadily, and the field underwent great advances after the Defense Advanced Research Projects Agency (DARPA) Grand Challenges. While the first two (2004, 2005) took place in the unstructured terrain of the Mojave desert, the third challenge (2007) was in urban, more structured environment [1]. The introduction of UGVs into urban environments can bring several benefits. Optimised driving can help reduce fuel consumption, environmental impact [2] and traffic accidents, as well as improve mobility for handicapped individuals.

Despite the recent advances, safe autonomous urban navigation remains a challenging task. Safety relies not only on obstacle detection, but also on precise information about road network structure. For instance, in order to determine the safest speed and trajectory, the UGV must know the road's layout, such as whether there is a bend ahead and how sharp it is [3].

Road junctions are another important feature that an autonomous vehicle should be able to identify. They are crucial for global trajectory planning and optimisation, since they are usually the only places where a vehicle can change trajectory. In addition to that, junctions are also important for local planning: they typically concentrate vehicles and pedestrians moving in multiple directions, which increases the risk of accidents. In fact, Australian traffic accident data show that 44% of all accidents happen in junctions [4].

Navigating in unstructured and structured terrains are separate tasks associated to different challenges, and the same can be said of navigation near an intersection. Just as off- and on-road navigation rely on different sets of algorithms, junction detection systems can be used to determine when to activate specific algorithms designed to handle large quantities of pedestrians, predict the future position of nearby vehicles etc. in order to avoid accidents.

Road network structure can be acquired using a combination of Global Positioning System (GPS) and detailed metric maps. However, GPS localisation is not always precise—especially in areas surrounded by tall buildings—and maps are not always available. Without relying on this type of information, sensorial data must be used. Previous work in junction detection focuses on cameras and laser rangefinders. Algorithms reliant on visual data alone may also face difficulties in low luminosity settings. Combining visual data to rangefinder data helps overcome this difficulty, but at the cost of higher computational complexity [3].

This work aims at developing a robust method for road junction detection based only on data from a 3D point cloud generated from a rangefinder, our main contribution amounting to the experimental analysis of well known algorithms. The method proposed is part of the Carina 2 project [5], which, among other goals, aims at enabling GPS-less robotic navigation. Within this context, junctions are also used as landmarks that allow to determine the approximate location of the agent within a topological road map. The biggest hurdle of this approach is that rangefinders are expensive equipment, but this tends to change as they become increasingly popular in autonomous cars.

This paper is organised as follows: section II reviews some related work in this field. Section III describes the proposed method. Section IV conveys the results of the experiments done using two real datasets, one collected in São Carlos (Brazil) and one from the KITTI online repository[1]. Lastly, section V exposes our conclusions and future work perspectives.

II. Background

The work of Jochem, T. M. [6] was one of the first studies able to identify lanes and Y-junctions, using a video camera and an Artificial Neural Network (ANN). In [7], data from camera is fused with rangefinder data. Then techniques from image processing and computational geometry are used

[1]D. Habermann and F.S. Osrio are with Instituto de Ciências Matemáticas e de Computação, University of São Paulo, Brazil. danilo.habermann@gmail.com; fosorio@icmc.usp.br

[2]C.E.O. Vido and F.T. Ramos are with the School of Information Technologies, University of Sydney, NSW 2008, Australia. kadu@it.usyd.edu.au; fabio.ramos@sydney.edu.au

[1]http://www.cvlibs.net/datasets/kitti/

to extract a skeleton navigable region, thus providing the intersections. In [3], a camera and lidar fusion method is proposed for road intersection detection. A virtual predict trajectory space for autonomous vehicles is constructed, and lane information extracted beforehand is projected in this space. The possible intersection branches are detected from the fusion of lane information.

A crossroad detection method in rural areas using 3D LIDAR data was proposed in [8]. The captured laser data is stored in a binary occupancy grid and a distance transform operator is applied. The presence of maximum value in the center of this grid indicates a crossing. Once the crossing detection is computed, the branches are extracted by iteratively exploring the grid cells with values next to the maximum.

In [9], to detect the crossing as far as possible, an admissible space is extracted by searching the free region ahead the sensor. A simulated 2D rangefinder is used to trace beams in the elevation map, inside the admissible space. The data generated is analysed and its peaks are extracted through a peak-finding algorithm. The number and position of peaks are statistically processed to detect the crossroads. Similarly, [10] and [11] also use a virtual rangefinder within a static grid map. Data generated by this sensor is analysed by a Support Vector Machine to identify the junctions. The main drawback of the methods using simulated rangefinders is that they rely on static maps, ignoring vehicles and pedestrians that would exist in a real world scenario.

III. Intersection Detection Method

During operation, 3D rangefinders create pointclouds at regular intervals, which can be called frames. Each frame contains a set of environment readings from a single location. The objective of the algorithm is to classify a single frame as a road junction or simple road. This is done in three steps: feature extraction, initial classification using a conventional classifier and refined classification using structural modelling techniques to incorporate contextual information from the neighbouring frames. The process is illustrated by the diagram in Figure 1.

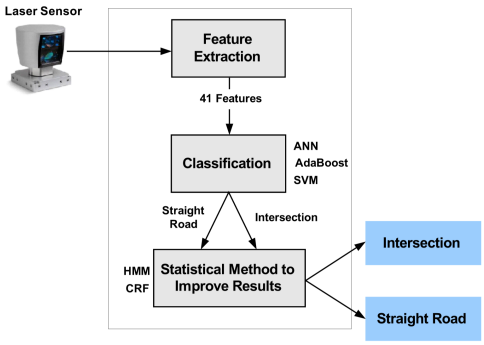


Fig. 1: Schematic representation of the algorithm

A. Feature Extraction

A laser rangefinder collects environmental data at a frequency of 10Hz, in the form of a three-dimensional pointcloud. Each point represents the position where a beam emitted by the sensor hit an obstacle. A few examples can be seen in Figure 2. Note that, while some frames have very clear vertical cues—buildings, trees, guard rails etc—that help the detection task, others have little beside the curbs (Figure 2c).

Features are extracted from the pointcloud using the method described by [12]. This method was chosen because it generates features that do not change when the vehicle is rotated. This is an important property, as the vehicle often might change movement direction at junctions. The 41 features extracted capture geometrical and statistical properties, summarised in Table I.

TABLE I: Feature list

Property	Associated Features
Volume	2
Average Range	2
Standard Deviation of Range	2
Sphericity	3
Centroid	3
Maximum Range	2
Distance	3
Regularity	1
Curvature	2
Range Kurtosis	2
Relative Range	4
Range Difference	6
Range Histogram	9

It is important to stress that the input data used is raw—unlike the methods described on [9], [10], [11], **we do not remove cars, pedestrians or other dynamic obstacles**.

B. Classification

The features are passed as input to an initial classifier, and the goal is to yield output similar to what is illustrated in Figure 3. The experiments performed compared three conventional classifiers.

1) Artificial neural networks (ANNs) [13] are a nonlinear statistical machine learning technique inspired by biological neural networks, widely used for pattern recognition. They are essentially directed graphs. This work uses *feedforward* ANNs, which are also acyclical. The nodes (or *neurons*) are organised in layers, and each receives as input the outputs of all neurons in the layer immediately before it. Figure 4 shows an example of a simple ANN.

The objective of the learning algorithms for ANNs is to adjust the weights on all the edges. Within each neuron, the weighted inputs from the previous layer are then combined and go through an *activation function*, which traditionally constrains its output into $[0;1] \in \mathbf{R}$. Sigmoid functions are widely used for this purpose, although other functions are possible.

2) Adaptive boosting (AdaBoost) [14] belongs to a family of methods collectively called *boosting*, meta-algorithms that combine the outputs of other learning algorithms ("weak

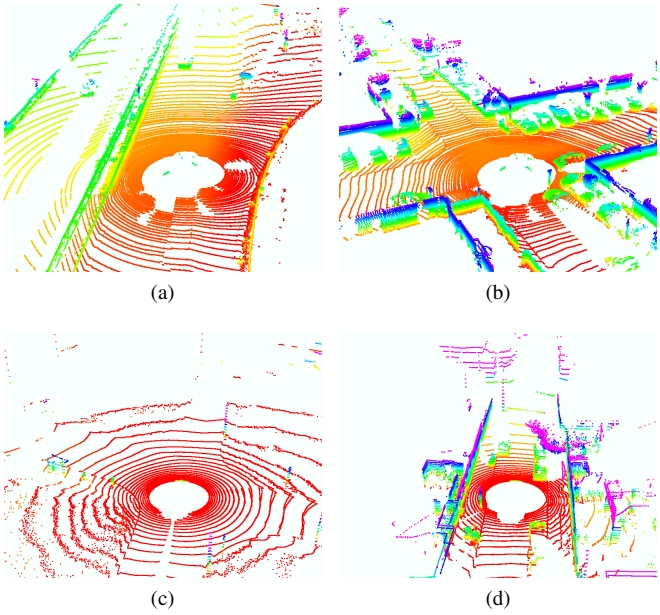


Fig. 2: Velodyne data frames. Colour relates to height. (a) KITTI Y-junction with relatively few vertical cues; (b) KITTI cross-junction rich in vertical cues; (c) São Carlos cross-junction with almost no vertical cues; (d) São Carlos straight road rich in vertical cues.

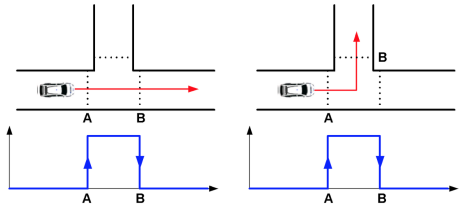


Fig. 3: Desired response for the classifier. Note that the trajectory of the car does not influence the classification.

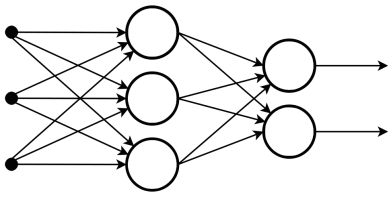


Fig. 4: Schematic representation of an ANN with the input layer on the left, on hidden layer with three neurons and the output layer with two neurons on the right.

classifiers") to make predictions. Boosted classifiers usually take the form

$$F_T(\mathbf{x}) = \sum_{t=1}^{T} f_t(\mathbf{x}), \tag{1}$$

where f_t are the weak classifiers, each producing an answer that places input $\mathbf{x}$ into either the positive or the negative class. The magnitude of f_t is the confidence of each classifier in the prediction, so the sum should favour the class to which most classifiers are confident that the input belongs.

AdaBoost is a method to train boosted classifiers. Using a training dataset $\mathcal{D}$, each iteration t adds an extra layer to a $t-1$-layer classifier built in previous iterations, F_{t-1}. The new layers are constructed by assigning a weight α_t to a candidate weak classifier's hypothesis h, such that the error E_t is minimised:

$$E_t = \sum_{\mathbf{x}\in\mathcal{D}} E[F_{t-1}(\mathbf{x}) + \alpha_t h(\mathbf{x})]. \tag{2}$$

This process discards classifiers that do not improve the model's predictive power, which helps avoid overfitting. It may also improve execution time in relation to other boosting methods with the same number of layers, since it can avoid computing some.

3) Support vector machines (SVMs) [15] are a supervised learning model that can be used for classification. The inputs are represented as points in space, and the algorithm tries to fit a hyperplane through the input space, making them a non-probabilistic binary linear classifier. The hyperplane that leaves the largest margin between both classes is the best fit. Figure 5 provides a visual example to illustrate this concept.

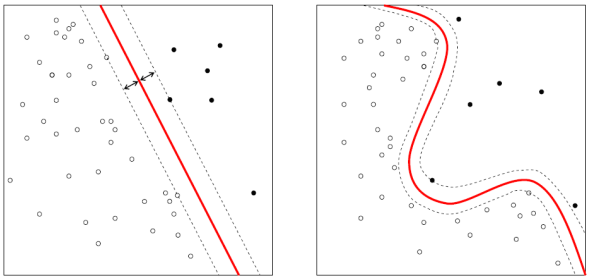


Fig. 5: SVM example in two dimensions. (a) Linear case. The hyperplane is indicated in red, the margin between classes indicated with black arrows. The support vectors are the inputs that sit on the dashed line. (b) Non-linear case. A kernel function must be used to create a feature space where the classes are separable by a hyperplane. Once that's done, the task becomes the same as in the linear case.

Many applications, however, have classes that are not linearly separable in the input space. To perform non-linear classification, a kernel function can be used to perform what's known as the *kernel trick*, creating a feature space where the classes are linearly separable.

C. Structured Classification

The sensor produces 10 frames every second, so it is expected that a number of consecutive frames belong to the same class—it is unlikely that in a fraction of a second the vehicle can go in and out of a junction. However, since the classifier is only aware of a single frame at a time, it does not have access to this type of contextual information about the studied phenomenon. As a result, the classifier might make noisy predictions (see Figure 6).

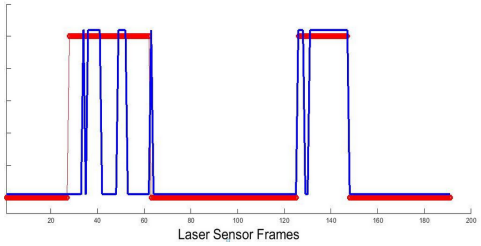


Fig. 6: Prediction made by AdaBoost on a small subsample of the KITTI dataset. There are only two junctions (higher values in red), but due to noise, the classifier (in blue) register six peaks, and the UGV would consider it passed six junctions and miscalculate its position in a topological map. Red: ground truth, blue: prediction.

We want the system to be reliable enough to be used to determine the approximate position of the UGV in a topological map. The behaviour shown in Figure 6 hinders this application. To overcome this, we use structured models that incorporate our prior knowledge about the system. Two different approaches were tested to address this.

1) Hidden Markov Models (HMMs) [16] are a statistical method that model a sequence of observations (called *emissions*) by assuming they are the product of a sequence of hidden states, which cannot be directly observed. They can be interpreted as a special case of a directed probabilistic graphical model. Figure 7 illustrates this using a simple model.

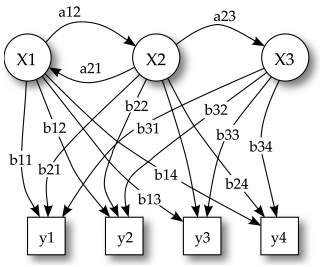


Fig. 7: Schematic representation of an HMM. X are the states, y the emissions, a the transition probabilities and b the emission probabilities.

If a sequence of states and the emissions generated by them are known (*e.g.*, a labeled training set for a classifier), they can be used to calculate the *emission matrix* (the probabilities of each of the states generating each of the observations) and the *transition matrix* (the probability of each state leading to each other state). If, instead, the emission and transition matrices are known, they can be used to calculate the sequence of states that generated the observations.

HMMs make two independence assumptions: that each state depends solely on the immediate previous state and that each emission depends solely on the current state.

2) Conditional Random Fields (CRFs) [16] are a type of undirected probabilistic graphical model trained discriminatively. As opposed to Markov random fields, they model the posterior probability of hidden variables given observations directly, therefore being able to capture complex relationships between observations and hidden states. CRFs have been extensively used in speech recognition and computer vision. Linear-chain CRFs are the main concern of this work.

Linear CRFs can be seen as a generalisation of HMMs in which the transition probabilities—instead of being fixed values—are a function of the observations. The dependence of each state on the observations is given by a fixed set of *feature functions*. The model is trained by learning these functions and the conditional distributions between states from the dataset. This can be then used to determine the most likely states that would have generated a set of observations.

IV. Experiments and Results

To measure the performance of the proposed method, two separate real world datasets were used. One of them was collected by members of the Carina 2 project in São Carlos, Brazil, using a vehicle equipped with a Velodyne 32 laser rangefinder, which takes 70 thousand readings per frame. The other is from the KITTI repository [17], [18], [19], and it was collected using a Velodyne 64 rangefinder, which produces denser pointclouds with twice as many points per frame. Both sensors operate at a frame rate of 10Hz. Table II has some additional information about the datasets used. SC1 and SC2 are the training and test subsets sampled from the São Carlos dataset, K1 and K2 are the training and test subsets sampled from the KITTI dataset. They come from the odometry benchmark set, K1 includes folders 00 to 07 and K2, folder 08. All four datasets are disjoint.

TABLE II: Dataset description

	Dataset			
	SC1	SC2	K1	K2
Junctions	57	12	61	19
Frames	4150	892	5046	1834
Junction frames	2794	645	3323	1280
Road frames	1356	247	1723	554
Velodyne	32	32	64	64

Tests were done in two stages. First, to compare the performance of the three base classifiers used, we train and test the algorithm in each of the available datasets separately. Once the best classifier is determined, we combine it to the structural modelling techniques and test it in a disjoint dataset to measure if the training generalises well.

The performance metrics used for the tests were accuracy, precision and recall. If we take the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FP), we have:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} , \tag{3}$$

$$Precision = \frac{TP}{TP + FP} , \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \ . \tag{5}$$

A. Training

The first step is to determine which base classifier has the best performance. For this, all must be trained and tested using a single dataset at each time.

For this work, we have used ANNs with two hidden layers. To determine the number of neurons within each, several configurations were tested, ranging between five and twenty neurons per layer. The input layer has 41 neurons (one for each feature), and the output layer has two (one for each class, junction or road). All ANNs were trained using scaled conjugated gradient and 10-fold cross validation.

We trained SVMs using two different kernels, polynomial and linear. Both were able to learn the training set well, but even with a small number of training iterations, the models learned were not able to make good predictions on the test sets. We interpret this as indication that, in this scenario, the method is prone to overfitting.

Different configurations of AdaBoost were trained, changing the number of weak classifiers. For the São Carlos datasets, the performance improvement was negligible above 50 weak classifiers. For the KITTI dataset, 100 weak classifiers were used.

For the structured classification step, the HMM and CRF algorithms take as input the labels generated by the first round of classifiers. As such, their training set are the predictions over the SC1 and K1 datasets. CRF was trained with sets of 20 sequential frames.

B. Classification Performance

The best results obtained by each classifier are summarised in Table III. In our tests, AdaBoost had the best overall performance in both datasets, and proved itself more robust and less prone to overfitting than the other classifiers used.

TABLE III: Trained classifier results

Dataset	Classifier	Accuracy	Precision	Recall
SC2	ANN	0.7088	**0.870**	0.8100
	AdaBoost	**0.8700**	0.6478	**0.8466**
	SVM	0.7220	0.7854	0.4987
K2	ANN	0.8049	0.6606	0.6867
	AdaBoost	**0.9027**	**0.8736**	**0.8190**
	SVM	0.3626	0.8267	0.3009

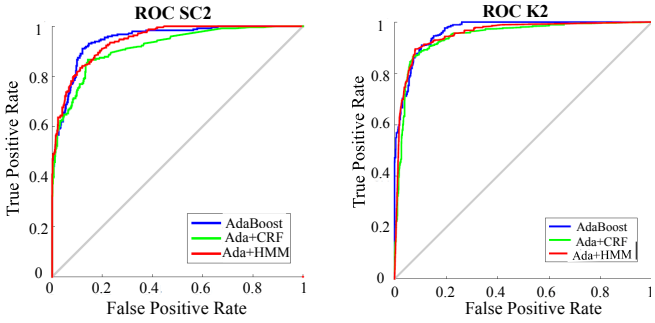
C. Structured Classification Performance

In this section, we analyse whether we can improve the predictions made by AdaBoost using HMMs or CRFs to incorporate contextual knowledge.

Table IV has the results for the structured models. All three methods show comparable performance, the fluctuations in accuracy, precision and recall are small, and these metrics seem to suggest a small performance loss for CRF. To further investigate this effect, we analysed the receiver operating characteristic (ROC) curves for these three methods. ROC curves show the performance of a binary classifier as the discrimination threshold is varied, in terms of its true positive rate (TPR) and false positive rate (FPR), with a larger area under the curve indicating a better performance. The results of this analysis are in Figure 8 and Table V.

TABLE IV: HMM and CRF results

Dataset	Classifier	Accuracy	Precision	Recall
SC2	AdaBoost	0.8700	0.6478	0.8466
	AdaBoost + HMM	**0.8789**	0.6680	**0.8639**
	AdaBoost + CRF	**0.8789**	**0.8145**	0.7287
K2	AdaBoost	0.9027	0.8736	0.8190
	AdaBoost + HMM	**0.9104**	**0.8889**	0.8285
	AdaBoost + CRF	0.9044	0.8393	**0.8484**



(a) (b)

Fig. 8: ROC curves. (a) São Carlos data; (b) KITTI data

TABLE V: ROC analysis

Dataset	Classifier	Area under the ROC curve	FPR for TPR=0.9
SC2	AdaBoost	0.9433	0.3147
	AdaBoost + HMM	**0.9446**	**0.1907**
	AdaBoost + CRF	0.9143	0.2124
K2	AdaBoost	**0.9662**	**0.1027**
	AdaBoost + HMM	0.9559	0.1074
	AdaBoost + CRF	0.9350	0.1319

Again, all three methods performed well. The area under the ROC curve, also a measure of accuracy, was higher than 0.9 for all three methods. CRF seems to have the worst performance of the methods. Figure 9 presents the final classification on both test datasets. Upon close inspection, it reveals an interesting pattern.

In real world deployment, a proper road junction detection algorithm must be able to accurately detect the number of junctions the platform on which it runs has crossed. However the width of each junction or its distance to the next one might be unknown at the instant when the measurements are taken, so an additional junction will be added to the count whenever the classifier changes state. This change of state can be seen as a peak on the class per frame graph. Therefore,

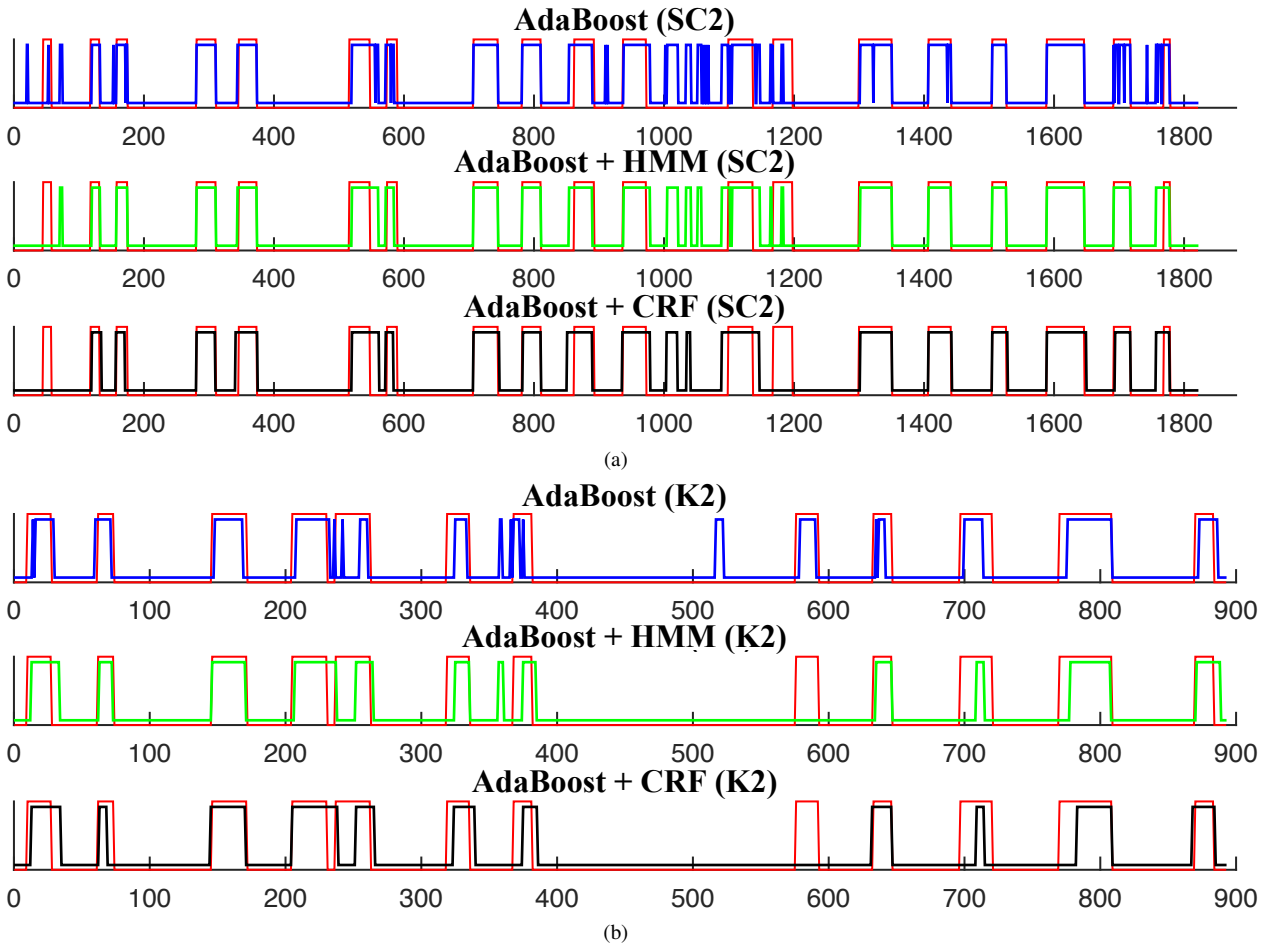


Fig. 9: Structured classification results. (a) São Carlos dataset; (b) KITTI dataset. Red indicates the ground truth.

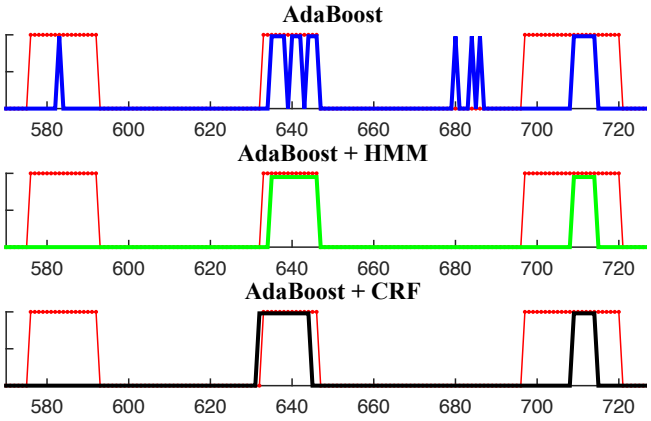


Fig. 10: Results for a subsample of the São Carlos dataset, illustrating possible types of detection error. After structural modelling, the first junction was misclassified. Pure AdaBoost, however, misclassified a road between the second and third junctions. Red indicates the ground truth.

the frame-by-frame analyses made on Tables III to V might not fully measure the success of the methods analysed.

To exemplify this, Figure 10 contains a sample of the SC2 dataset. It comes from a region with three junctions, as can be seen in the ground truth. For this region, AdaBoost registered 8 peaks, while HMM and CRF both had 2. This graph, however, evidences something more important: even if AdaBoost had the best performance on a frame-by-frame analysis, it still has very noisy readings that were eliminated by the smoothing effect of the structural models. On the other hand, they smoothed out a peak that should have been in the final result.

To indirectly measure how each method would perform on the task of localising an UGV inside a topological map, we propose four separate metrics that identify the phenomena observed in Figure 10:

True Detections per Junction (TDJ): a detection is true when a single peak appears within a junction. The result is divided by the number of peaks in the ground truth. In Figure 10, all methods have TDJ $= {}^{2}/_{3}$—AdaBoost has three peaks within the second junction, the other two failed to detect the first.

Noisy Detections per Junction (NDJ): when multiple peaks happen inside a single junction. The result is divided by the actual number of junctions. In Figure 10, this happened for

AdaBoost in the second junction, for which NDJ $= {}^1/_3$.

False Negatives per Junction (FNJ): a detection is false when no peaks appear within the length of the junction. The result is divided by the number of peaks in the ground truth. In Figure 10, this happened for both structured models in the first junction.

False Detections (FD): when a peak happens outside a junction. This is kept In Figure 10, this only happened for AdaBoost between the two last junctions.

Table VI contains analyses of these metrics for both test datasets. By these standards, we can clearly observe that the incorporation of a structured model improves on the results obtained. By smoothing out noisy and false detections, both methods obtained considerably better results. CRF's biggest advantage was in reducing noisy detections.

TABLE VI: Separate metrics

Dataset	Classifier	TPJ	NDJ	FNJ	FD
SC2	AdaBoost	0.4166	0.3333	0	12
	AdaBoost + HMM	0.9167	0.0833	0.0833	1
	AdaBoost + CRF	0.9167	0	0.0833	0
K2	AdaBoost	0.6842	0.3684	0	20
	AdaBoost + HMM	0.8421	0.0526	0.0526	4
	AdaBoost + CRF	0.8947	0	0.1053	4

V. Conclusion

In this work, we presented a road junction detection method for urban areas. The pointcloud collected using a rangefinder goes through three steps: feature extraction, classification and structural modelling. We have compared the performance of ANNs, SVMs and AdaBoost for the second step, and of HMMs and CRFs for the last.

AdaBoost was considered the best classifier, as it managed to learn the training set without overfitting, generalising well to the test set. On a frame-by-frame analysis, subsequent use of CRF and HMM do not seem to improve from the results obtained by AdaBoost itself. However, Figure 9 shows that both methods remove a lot of the classification noise, generating an output that allows to more clearly detect the start and end of a road junction.

Further testing is needed to determine whether the method presented in this work can be reliably trained and tested with data coming from different parts of the world. The ability to perform well under these conditions is a feature we believe is important in the future of autonomous cars, as it would allow manufacturers to prescind from owning training grounds or datasets from every country where they market their vehicles.

This work can be extended in several ways. One aspect observed is that classification is much harder in regions with low urban density, because they typically have less vertical information (see Figure 2c). A lot of the previous work on this field relies on obstacles being above a set height for detection, which leads us to conclude this is an inherent difficulty of the field. In the future, we plan to make the algorithm more robust against this situation by incorporating methods for curb detection and discrimination between X, Y and T junctions, as described in [20].

References

[1] C. Ilas, "Perception in autonomous ground vehicles," in *International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 2013.

[2] T. Luettel, M. Himmelshach, and H.-J. Wuensche, "Autonomous ground vehicles - concepts and a path to the future," in *Proceedings of the IEEE, Centennial Issue*, 2012.

[3] Y. Nie, Q. Chen, T. Chen, Z. Sun, and B. Dai, "Camera and lidar fusion for road intersection detection," in *IEEE Symposium on Electrical and Electronics Engineering (EEESYM).*, 2012.

[4] Road safety in new south wales: Statistical statement for the year ended in 31 december 2012. [Online]. Available: http://roadsafety.transport.nsw.gov.au/downloads/crashstats2012.pdf

[5] L. C. Fernandes, J. R. Souza, G. Pessin, P. Y. Shinzato, D. Sales, C. Mendes, M. Prado, R. Klaser, A. C. Magalhães, A. Hata, D. Pigatto, K. C. Branco, V. G. Jr., F. S. Osorio, and D. F. Wolf, "Carina intelligent robotic car: Architectural design and applications," *Journal of Systems Architecture*, vol. 60, no. 4, pp. 372 – 392, 2014.

[6] T. Jochem, D. Pommerleau, and C. Thorpe, "Vision-based neural network and intersection detection and traversal," in *Intelligent Robots and Systems*, 1995.

[7] P. Mukhija, S. Tourani, and K. Krishna, "Outdoor intersection detection for autonomous exploration," in *IEEE Conference on Intelligent Transport Systems*, 2012.

[8] A. Mueller, M. Himmelsbach, T. Luettel, F. von Hundelshausen, and H.-J. Wuensche, "Gis-based topological robot localization through lidar crossroad detection," in *International IEEE Conference on Intelligent Transportation Systems*, 2011, pp. 2001–2008.

[9] C. Tongtong, D. Bin, L. Daxue, and L. Zhao, "Lidar-based long range road intersection detection," in *International Conference on Image and Graphics (ICIG)*, 2011, pp. 754–759.

[10] Q. Zhu, L. Chen, Q. Li, M. Li, A. Nüchter, and J. Wang, "3d lidar point cloud based intersection recognition for autonomous driving." in *Intelligent Vehicles Symposium*. IEEE, 2012, pp. 456–461.

[11] Q. Zhu, Q. Mao, L. Chen, M. Li, and Q. Li, "Veloregistration based intersection detection for autonomous driving in challenging urban scenarios," in *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2012, pp. 1191–1196.

[12] K. Granström, T. B. Schön, J. I. Nieto, and F. T. Ramos, "Learning to close loops from range data." *I. J. Robotic Res.*, vol. 30, no. 14, pp. 1728–1754, 2011.

[13] C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

[14] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[15] B. Schölkopf and A. J. Smola, *Learning with Kernels*. MIT Press, 2001.

[16] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[17] J. Fritsch, T. Kuehnl, and A. Geiger, "A new performance measure and evaluation benchmark for road detection algorithms," in *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[18] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[19] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[20] A. Hata, D. Habermann, D. Wolf, and F. Osório, "Crossroad detection using artificial neural networks," in *International Conference on Engineering Applications of Neural Networks (EANN)*, 2013.