Application of Line Clustering Algorithms for Improving Road Feature Detection

Fabian Poggenhans, André-Marcel Hellmund
Mobile Perception Systems
FZI Research Center for Information Technology
76131 Karlsruhe, Germany
{poggenhans, hellmund}@fzi.de

Christoph Stiller
Institute of Measurement and Control Systems
Karlsruhe Institute of Technology
76131 Karlsruhe, Germany
stiller@kit.edu

***Abstract*— Although many algorithms have been proposed for the camera-based detection of road features (such as road markings, curbstones and road borders), truly contextual or relational information between the detections is rarely used. This is all the more surprising, since a lot of potential remains unused, regarding outlier rejection or compensating detection failures, multiple detections, misclassification or fragmentation. The aim of this paper is to present an approach that is suitable for such a task in both online and offline applications as a post-processing step after the actual detection and classification step. This is achieved by adapting a perception-based line-clustering algorithm that groups the pre-classified road features based on their relations and assigns them a final class. The grouped features are then fused to form continuous lines instead of individual dashes or fragmented lines. The evaluation on a 10 km drive in both rural and urban environment, as well as an online test on a short highway driving sequence shows that this approach is very well capable to increase the performance of road feature detection at a low computational cost.**

I. INTRODUCTION

The detection and classification of road features is a substantial part of modern driver assistance systems for quite some time. Since it also provides a significant contribution to behaviour generation and localisation for autonomous vehicles [1], high robustness and reliability is of great importance and needs to be further increased on existing road feature detection algorithms. We think that this can be archived by considering context-based information about the relative position of the road features.

Road feature in this context means elements that are meant to guide the driver along his lane. These include road markings (e.g. lane markings, stop lines or symbols such as arrows), curbstones, guardrails or road borders.

As the detection of road features has been an issue since the early days of driver assistance systems, a large number of classification algorithms can be found in literature regarding road markings, such as [2], [3], [4] or our own proposal [5]. A similar picture emerges with curbstones [6], guardrails [7] and road borders [8]. Most of the algorithms have been designed and evaluated for use on highways or rural roads and it is highly doubtable that they will perform well in scenarios as shown in Fig. 1 without using contextual information.

The number of publications in which contextual information is employed is very limited. As one of the few examples,

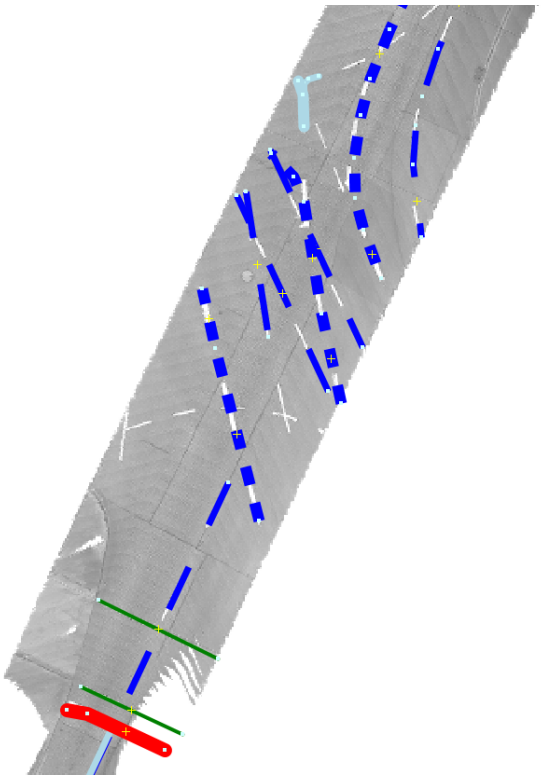


Fig. 1: A challenging road scenario even for modern marking detection algorithms and results from our proposed algorithm.

Mathibela et al. [9] use conditional random fields (CRF) to make use of the relative position of the markings in classification. In their approach, classification results improved in particular for complex scenarios like intersections by up to 20%. The main goal of this approach was to improve classification of the individual markings. It is, however, not designed to generate continuous lines from individual segments, which is needed for most motion planning algorithms such as lane keeping.

The work of Hur et al. [10] is an example where contextual information was used for continuous multi-lane detection, based on CRF as well. Despite a comparatively simple detection, without classification, this resulted in remarkable

recognition rates, which shows the potential of this approach.

The use of CRF for an approach that combines forming of continuous lines with classification for generic situations, however, enormously raises the level of complexity due to the high number of possible combinations that have to be evaluated for inference in the underlying graph and the high number of training data required for learning the parameters in the graph.

This is a field, where Perception-based clustering algorithms have already been established as an alternative. In Ye et al. [11] such an algorithm is used to cluster connected sections in handwritten notes. Similar methods can be found for technical drawings [12], where they are used to obtain a higher semantic understanding of the elements contained. Another example of the early use of grouping algorithms is the contribution of Jonk et al. [13], in which simple line segments of an edge detection algorithm were combined with a clustering algorithm to form contiguous lines.

A theoretical overview for grouping algorithms provides the work of Engbers et al. [14] that served as a base for the algorithm presented here. It considers grouping "as the task to find groups of similar elements from a set S, where S is a finite set of basic elements $\{s_1, s_2, \ldots, s_n\}$ derived from an image." According to this, the difference between grouping and clustering is that in grouping algorithms one element can be part of several groups. Grouping is therefore a generalisation of clustering. Furthermore, Engbers et al. provide fundamental design considerations regarding grouping algorithms based on current insights into perception, as a continuation of the common Gestalt laws. We will discuss these in Section II with regard to the applicability on road features.

Later in this section we will discuss further details of the proposed algorithm. The suitability of our algorithm is then analysed in Section III with real data on a representative track and evaluated on a short section of a German highway with respect to its online fitness. In Section IV we will summarise the results and provide an outlook on our future work.

II. ALGORITHM OVERVIEW

A. Design considerations

As a starting point of our method, it is assumed that recognized road features are represented as linestrings with a global position, where every element is assigned a probability distribution over every possible type of marking. This could be the result of a classifier, as in the approach presented by us [5], but there are also simpler heuristics possible.

The basic idea of the proposed approach is to assign an explicit class to the pre-classified elements by grouping them with similar neighbouring elements, while taking into account their classification likelihoods and their relative location. As result, we want to obtain one continuous line that connects all lines within a group while approximating them as closely as possible.

It is assumed that the input data is subject to errors in the detection or classification step. Typically, these errors include the following:

- Inaccurate classification
- Multiple detections (e.g. through detections over multiple frames)
- Conflicting detections (e.g. different classification results for the same feature over multiple frames)
- Fragmentation of features in smaller segments
- Missing detections
- Detection of individual dashes instead of the whole dashed line

The features located in the immediate vicinity of roads have been created specifically for human drivers in order to be easily detectable and interpretable to avoid accidents as consequences of misinterpretation of the road ahead. Therefore, algorithms, such as that of Engbers at al., are particularly suitable because they are based on the principles of human perception.

In this context it is important to mention that grouping algorithms generally do not guarantee an optimal solution. The main reason is that the term cannot be clearly defined because multiple equivalent interpretations of the same scene are possible, depending on the context, scale and perspective. It specifically applies to hierarchical grouping algorithms that the grouping of two elements may be in conflict with another element and that conflict cannot be resolved in the further course of the algorithm. To obtain an optimal result here would mean pursuing both possible interpretations whenever a conflict occurs to see which path of possible decisions yields the best result.

However, this would lead to an explosion of complexity and thus computation time. For this reason we limit ourselves to pursue only the - at that time - most promising solution in the occurrence of conflicts.

B. Algorithm

We designed our approach as an extension of the algorithm outlined by Engbers et al. [14]. As mentioned in Section II-A, we assume we are given a set S of n detections as input, where for every detection $s_i \in S$ a probability $p_c(s_i)$ for a class $c \in C$ is assigned (where C is the set of all possible classes), so that $\sum_{c \in C} p_c(s_i) = 1$ and $0 \leq p_c \leq 1$. We also define a distance-probability function for every class $d_c(s_1, s_2)$ that assigns a geometrical probability for the two elements to be grouped (see Section II-D), with $0 \leq d_c \leq 1$.

If it is assumed that S contains outliers, an outlier class is useful in C. This class is ignored in the grouping stage.

Therefore, we can calculate a grouping measure for a given pair of detections s_1, s_2 and a class c as follows:

$$p_{group}(s_1, s_2, c) := p_c(s_1) \cdot p_c(s_2) \cdot d_c(s_1, s_2)$$

The structure of the algorithm is then as shown in Algorithm 1.

The use of a threshold to stop the grouping is advisable because of the greedy nature of the algorithm. Otherwise all elements of the same class will be grouped unless there is a conflict between them. In practice, chosing a good value was very simple as the gap in p_{group} between elements worth grouping and not worth grouping is usally very wide.

```
// initialization
for every pair s_1, s_2 in S, and every c in C do
    calculate p_group(s_1, s_2, c) and store it in a list L of
     descending order;
end
initialize the list of groups G with
 G = {{s_1}, {s_2}, ..., {s_n}};
// grouping
while L is not empty do
    remove the first value from L and find the
     (s_1, s_2, c)-tuple assigned to it;
    if the value falls below a threshold t then
        break;
    end
    if (s_1, s_2) can be grouped without conflicts, and are
     not part of the same group in G then
        merge the two groups in G that contain s_1 and
         s_2;
        set p_c(s_1) = p_c(s_2) = 1 (and to zero for all
         other classes);
        update L with new p_group for all elements
         connected to s_1, s_2;
    end
end
```

Algorithm 1: Enhanced grouping algorithm

We will now discuss important algorithm details in the following sections.

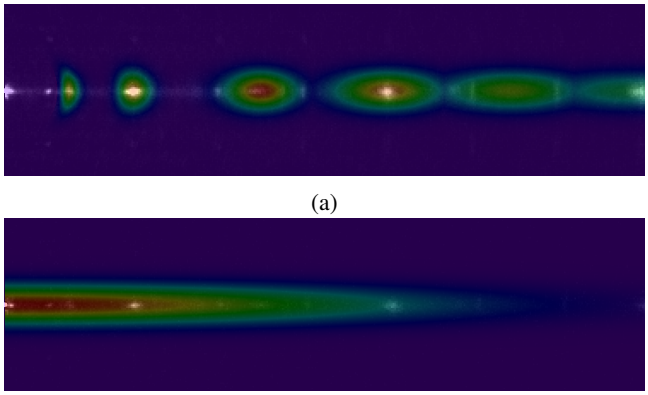
C. Conflict detection

An essential element in the algorithm is conflict detection. These conflicts cannot be easily captured by the distance measure, because they are affected by other elements in the vicinity. Conflicts can occur at any time during the grouping process, therefore, they must be continuously checked. We identified and used the following conflict conditions when trying to group two elements s_1 and s_2:

- s_1 or s_2 is to be grouped with a different group than an element s_3 on the (almost) same position,
- s_2 is already in a group, and its direct neighbours have a much lower grouping probability with s_1 (and vice versa).

The rationale behind the first condition is that if two detections are on the same spot but have been classified differently, this is most likely a misclassification. In this case, we prefer the detection that was grouped first, because it apparently fits better into the context.

The reason for the second condition is that lines on the road may split (e.g. on highway exit points or crossings). In this case, p_{group} between two elements after a split will be much smaller than between the element before the split respectively. Without this condition, both lines after the split would become part of the same group, which is highly undesirable.



(a)

(b)

Fig. 2: Distance function for the dashed line class (a) and straight line class (b). The origin is on the left border in the middle. White points stand for positions of real lines in our dataset.

A further conflict check is required to avoid that two elements selected for grouping are in groups with different classes associated to them. However, this is already guaranteed by the update step of the algorithm.

D. Distance function

The distance function expresses on which position an element is to be expected from the perspective of another element, given a certain class. The design of these functions (one for each class) is crucial for the grouping results, as it influences how much one class will be preferred against all others. It becomes even more important if the grouping step needs to compensate unreliable classification results from the classifier.

For our case, we defined three types of neighbourhood functions based on the type of road features:

- *Point features (e.g. Arrow detections):* We assume point features are singular detections without any repeating pattern. In this case, we only want to cluster detections around this point.
- *Straight lines:* Due to errors in the detection process, lines can be fragmented into smaller segments. Here, we expect other lines directly in front of the line, but not at the sides.
- *Dashed lines:* These appear in fixed patterns on the road in ratios of 1:2, 1:1 or 2:1 between dash and gap lengths. Because we assume that dashes can be missing in between, the neighbourhood function needs local maxima for higher ratios as well.

Figure 2 shows the distance functions for dashed and straight lines. For point features (not depicted) the distribution becomes a simple Gaussian distribution based euclidean distance. Note how the distribution for dashed lines has repeated maxima that are sharper close to the origin and become more diffuse in the distance to capture inaccuracies in calculating the line length. Positions of real detections in our dataset are shown as light dots in the image. Especially

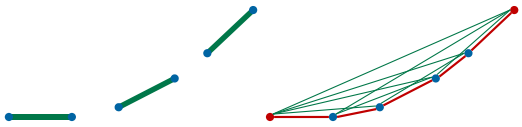


Fig. 3: Finding the best connection through the dashes shown in the left image. Rejected connections in the graph are marked in green, final path in red.

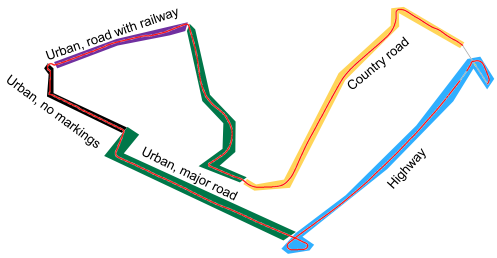


Fig. 4: Test drive used for evaluation

the distribution for dashed lines in Figure 2a corresponds very well to real locations.

Because lines on the road are generally not straight, we use an approach similar to the approach proposed by Jonk et al. [13] that is based on the delta angle $\Delta\phi$ between the two elements: Instead of evaluating the distance function in the direction of one of the two lines, we evaluate the distance function along a line at $\frac{1}{2}\Delta\phi$, because we can expect the next element along this line if the lines form a circle. To avoid that lines are clustered at a high delta angle, we multiply the result with a von Mises distribution $p(\Delta\phi)$.

E. Post-processing

After the grouping is completed, the final classes and the groups of the elements are known, but we need to calculate the final lines from the groups. To first determine the correct sequence of the line elements, we build a neighbourhood graph containing all points of the linestrings as vertices and the connection between them as edges. Additionally, we connect the endpoints of every linestring with all other endpoints in the vicinity. Finally, we calculate the minimum spanning tree (MST) from this graph. The resulting line is then the longest remaining path in the MST.

This method ensures that the final line is not a direct connection between the two most distant endpoints, but instead connects most of the line points without major jumps and without self-intersections.

The process is shown in Figure 3. The resulting path (in red) might still contain sharp edges, e.g. if we have two detections on the same space. Because of that we simplify the resulting line using the Douglas-Peucker algorithm [15] and finally smooth and supersample the line using non-uniform rational B-splines (NURBS).

III. EVALUATION

A. Set-up and method

The described algorithm is evaluatad on two different scenarios. The first scenario is composed of an 11 km test drive in the east of Karlsruhe, Germany. The test drive was recorded three times and a ground truth was manually labelled containing all visible markings along the track. The drive contains a variety of different scenarios (see Figure 4).

Images were recorded using our research vehicle "AnnieWay" with a stereo set-up to the rear of the vehicle and at a resolution of 1200 px×597 px at a frame rate of 10 Hz.

The second evaluation is done online along a German highway to evaluate the results of the algorithm with a more standard online line detector. The hardware setup consisted of a grey-scale monocular camera (2.8 Mpx) recording images with a frequency of 15 Hz.

The algorithm itself was implemented in C++ (single-threaded). The distance functions were parametrized based on the German legal instructions for road markings [16].

B. Offline evaluation on test drive

The marking detection algorithm used for offline evaluation requires top-view images. These are obtained by warping the input images to top-view perspective using stereo reconstruction and merging them with each other based on their relative position. Positions are calculated using a visual SLAM approach [17]. Markings are detected using an algorithm proposed by us [5]. For comparison with the ground truth, we sample points along manually labelled reference lines and compute the distance to the next detection of this class. If this is larger than 20 cm, we consider that as a false negative, otherwise as a true positive. The false positives are calculated by sampling along a detected line and calculating the distance to the next ground truth label.

For point features, like arrows, the centroid distance between detection and ground truth is used instead.

To measure the improvement compared to the raw detections where the classification result is actually a probability distribution over all classes, we use the most probable classification result as reference. Because the detector only detects single dashes instead of whole lanes, we created a second ground truth containing only dashes for better comparability.

The results of the grouping algorithm are shown in Table I for all the markings detected over one of the three drives and for grouping the detections over all three in one large step. The execution time on an Intel Mobile i7 CPU at 2.5 GHz was 3.34 s for the single drive and 35.05 s for all three.

The results show that our algorithm is able to greatly increase the recognition rate over all line types. The longest resulting line is tracked over its full length of 1.1 km (and similarly for other lines). The improvement with the 25 cm lines is lower because they are usually shorter so that the use of contextual information confers a smaller advantage.

In the results from our original detector, straight lines are often detected as fragments and erroneously classified as dashes. Our algorithm corrects this mistake because these lines appear in a random pattern instead of fixed ratios. This greatly increases the precision for dashed lines and the recall for straight lines.

Class	Exist	Recall			Precision		
		Detected	One drive	Three drives	Detected	One	Three
Dashed line 12 cm	8050 m	92%	93%	98%	45%	76%	73%
Dashed line 25 cm	2348 m	81%	80%	90%	77%	82%	67%
Straight line 12 cm	7028 m	57%	70%	82%	96%	93%	80%
Straight line 25 cm	2343 m	61%	73%	76%	77%	80%	82%
Stop line	168 m	77%	78%	77%	81%	82%	68%
Pedestrian crossing	688 m	64%	64%	83%	84%	83%	85%
Total	**20625 m**	**71%**	**80%**	**88%**	**61%**	**82%**	**76%**
Zebra crossing	3	100%	100%	100%	43%	75%	75%
Zig-zag	10	80%	70%	80%	67%	78%	80%
Arrows	131	73%	73%	80%	97%	97%	94%
Tempo Limits (30)	2	100%	100%	100%	100%	100%	100%
Total	**146**	**73%**	**72%**	**83%**	**91%**	**94%**	**92%**

TABLE I: Results of the grouping algorithm for one drive and for three drives compared to the raw detections from the markings detector. Arrows and pedestrian crossings are summarized from multiple subclasses for better legibility.

However, the point features listed on the lower half of Table I profit not so much from grouping in the single drive. This is expectable because every element is detected only once and contextual information is not of use here. This changes when detections are clustered over multiple drives, because now we can group multiple detections on the same position which is helpful to reject outliers.

While the recall rate increases for multiple drives in general, the precision was lower, mostly because of the higher risk of random outliers but also because the position of the three drives calculated by the SLAM approach diverges by over 1 m for the last 20% of the track, so that the positions of the markings no longer match.

Some images from the results of our algorithm are shown in Fig. 5 and 6. Dashed lines from the algorithm are shown as blue (thick or thin) dashes (dash length is not related to the real dash length), stop lines in red, pedestrian and zebra crossings in green and symbols with their respective symbol.

The real stop lines are not visible under the red lines but were always classified correctly except for one false positive in Image 5c, at the start of a pedestrian crossing.

The images show that our algorithm performs very well even in complex scenarios like crossings where many different line types meet and even intersect at a small distance. Circles or curves as in roundabouts are not a problem.

However, one problem we found difficult to handle was that our algorithm grouped some lines that end right before an intersection or pedestrian crossing and then continue some 30 m after it. In these cases two interpretations of the same scene are possible: Two separate lines or one continued line that crosses the intersection. Our ground truth data only contains the first interpretation yielding a disproportionately lower precision value.

Most false positives originate in false detections from the detector if those detections appear in a pattern that fits well to one of the distance functions. One issue here is tram rails on the road like in Fig. 6a, because once they are erroneously detected by the detector, the grouping algorithm is unable to distinguish them from regular road markings. A similar problem arises when curbstones are detected as road markings as well (see Fig. 6b).

C. On-line evaluation on highway

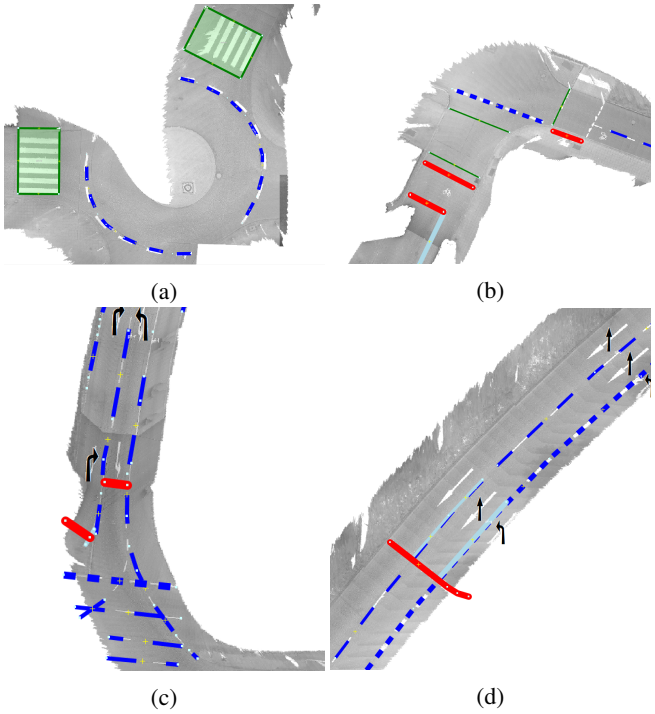
To showcase the application of our presented algorithm for online processing of detected lane markings, we evaluate its performance for a simplified scenario on a German highway. Clusters of lane markings are firstly extracted from the image sequence using a common lane marking extraction algorithm. The detected lines are transformed into top-view perspective and class probabilities are assigned for each marking cluster by applying simple fuzzy rules. Finally, the transformed clusters are accumulated according to the vehicle motion obtained by visual odometry [18].

For evaluation purpose, we selected a test track of approximately 2 km length on a German highway. The size of the accumulation window is 2 s, or 30 frames respectively. Firstly, we analysed the runtime for 1000 time steps in total on an Intel Mobile i7 CPU. As a result, on average, 141 marking clusters are passed to the grouping algorithm which took approximately 37 ms per time step to compute the grouping result.

In total, 86.9% of all labelled lane markings are correctly classified as thin/thick, dashed/straight lines, while the percentage of incorrectly classified and missing lane markings is 7.6% and 5.5% respectively. Considering runtime and classification performance, the grouping algorithm is highly suitable for online algorithms, e.g. for lane-level accurate localisation of an autonomously driving vehicle on highways since two or more lane markings groups are correctly classified in approximately 94% of the time.

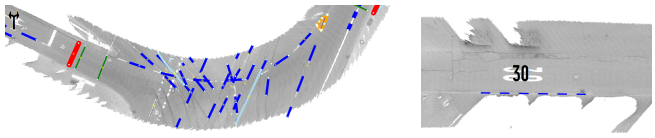
IV. CONCLUSION AND FUTURE WORK

In this paper, we presented a method for improving the detection of road features based on their relative position. The evaluation shows that the proposed algorithm is well suited for increasing the performance of a road feature detection algorithm at a low computational cost. We found the proposed algorithm to be very robust even in complex environments – like intersections – where many conflicting interpretations of the detected markings are possible. We were also able to show that or algorithm is able to run not only off-line for mapping purposes, but also on-line to increase the reliability of an on-line lane marking detector while at the same time providing fully connected dashed



(a) (b) (c) (d)

Fig. 5: Examples for good results of the grouping algorithm.



(a) (b)

Fig. 6: Problematic regions: (a) Rails, (b) Random detections on curbstones

lines instead of individual dashes and rejecting outliers and multiple detections of the same marking.

Our approach achieved an overall increase in the detection rate from 71% to 80% for line markings. The precision increased even more by over 35% overall. This shows that the approach is very well suited to improve the reliability of road feature detection.

The approach is also able to handle detections of several recorded tracks of the same route. In our evaluation we were able to reach a further improvement from 71% to 88% for line markings and similar for point-typed markings. However, the algorithm performance decreases from a certain number of drives on the same route, because the increasing probability of random detections in the proximity of an outlier can lead to random groupings that worsen the result. This effect is partly responsible for the drop in precision in the multi-drive results.

A better way to further improve the performance would be to detect much more different features than only road markings, like curbstones or – in our case – even rails. Because the algorithm accepts only one valid detection in a certain range, this can serve to reject outlier, e.g. false marking detections on curbstones, in a more intelligent way.

A further possible improvement could be achieved by considering the relations between different classes, too, by providing distance functions not only for two detections of the same class, but for any combination of classes. That way, contextual information of the kind ”Dashed lines often appear in parallel with staight lines” could be modelled. We did not investigate this in our paper, because such an approach extremely increases complexity and thus computation time. Still, the results of Mathibela et. al [9] indicate that such an approach can further increase the detection results when only little other contextual information is available.

References

[1] M. Schreiber, C. Knöppel, and U. Franke, “Laneloc: Lane marking based localization using highly accurate maps,” *IEEE Intelligent Vehicles Symposium*, 2013.

[2] B. Qin, W. Liu, X. Shen, Z. Chong, T. Bandyopadhyay, M. Ang, E. Frazzoli, and D. Rus, “A general framework for road marking detection and analysis,” in *International Conference on Intelligent Transportation Systems (ITSC)*, pp. 619–625, IEEE, 2013.

[3] P. Charbonnier, F. Diebolt, Y. Guillard, and F. Peyret, “Road markings recognition using image processing,” *IEEE Conference on Intelligent Transportatoin Systems*, 1997.

[4] J. C. McCall and M. M. Trivedi, “An integrated robust approach to lane maring detection and lane tracking,” *IEEE Intelligent Vehicles Symposium*, 2004.

[5] F. Poggenhans, M. Schreiber, and C. Stiller, “A universal approach to detect and classify road surface markings,” in *International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1915–1921, IEEE, 2015.

[6] M. Enzweiler, P. Greiner, C. Knöppel, and U. Franke, “Towards multi-cue urban curb recognition,” in *Intelligent Vehicles Symposium (IV)*, pp. 902–907, IEEE, 2013.

[7] A. Wimmer, T. Jungel, M. Glück, and K. Dietmayer, “Automatic generation of a highly accurate map for driver assistance systems in road construction sites,” in *Intelligent Vehicles Symposium (IV)*, pp. 281–286, IEEE, 2010.

[8] T. Kühnl and J. Fritsch, “Visio-spatial road boundary detection for unmarked urban and rural roads,” in *Intelligent Vehicles Symposium Proceedings*, pp. 1251–1256, IEEE, 2014.

[9] B. Mathibela, P. Newman, and I. Posner, “Reading the road: road marking classification and interpretation,” *IEEE International Conference on Intelligent Transportation Systems*, 2015.

[10] J. Hur, S.-N. Kang, and S.-W. Seo, “Multi-lane detection in urban driving environments using conditional random fields,” in *Intelligent Vehicles Symposium (IV)*, pp. 1297–1302, IEEE, 2013.

[11] M. Ye, H. Sutanto, S. Raghupathy, C. Li, and M. Shilman, “Grouping text lines in freeform handwritten notes,” in *International Conference on Document Analysis and Recognition*, pp. 367–371, IEEE, 2005.

[12] N. Nayef and T. Breuel, “Statistical grouping for segmenting symbols parts from line drawings, with application to symbol spotting,” in *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 364–368, IEEE, 2011.

[13] A. Jonk and A. Smeulders, “An axiomatic approach to clustering line-segments,” in *International Conference on Document Analysis and Recognition*, vol. 1, pp. 386–389, IEEE, 1995.

[14] E. Engbers and A. Smeulders, “Design considerations for generic grouping in vision,” *Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 4, pp. 445–457, 2003.

[15] U. Ramer, “An iterative procedure for the polygonal approximation of plane curves,” *Computer Graphics and Image Processing*, 1972.

[16] *Richtlinie für die Markierung von Straßen (Guidelines for Road Markings)*. German Department of Transport, 1980.

[17] M. Sons, H. Lategahn, C. G. Keller, and C. Stiller, “Multi trajectory pose adjustment for life-long mapping,” *IEEE Intelligent Vehicles Symposium*, 2015.

[18] A. Geiger, J. Ziegler, and C. Stiller, “Stereoscan: Dense 3d reconstruction in real-time,” in *Intelligent Vehicles Symposium (IV)*, 2011.