A Stereovision-Based Lane Detector for Marked and Non-Marked Urban Roads

Radu Danescu, Sergiu Nedevschi, Thanh-Binh To
Technical University of Cluj Napoca, Volkswagen AG Wolfsburg
Radu.Danescu@cs.utcluj.ro, Sergiu.Nedevschi@cs.utcluj.ro, thanh-binh.to@volkswagen.de

Abstract

This paper presents a lane detection system that combines stereovision-specific techniques with grayscale image processing for maximizing the robustness and applicability against the difficult conditions found on the urban roads, marked or non marked. The lane marking features are extracting using a fast and robust dark-light-dark transition detector that's aware of the perspective effect. The clothoid lane model is matched to the extracted features using line segment fitting for two distance intervals, under special constraints that ensure correctness. Freeform lane border detection, independent of the geometry constraints, driven by lane marking features only, is used to solve the situations not suited for clothoid representation. Intensive validation techniques are used for tracking initialization, and for monitoring of the noise level on the road, in order to avoid false positives. The results of each detection method are fused together in a Kalman filter based framework.

1. Introduction

Driving assistance systems are already beginning to emerge as commercial products, as they have reached the acceptable reliability for highway operation. The urban scenario, however, requires additional research, as it presents a whole new range of problems. One of the earliest sensorial systems for urban driving assistance is described briefly in [2]. Since then, the advancements in computer hardware and image processing algorithms have enabled the scientific community to make significant progress towards robust and accurate vision sensors for the urban driving environment.

The urban lane detection work of our team benefited from the experience gathered in the framework of the past research projects. This work was aimed at designing a driving assistance sensor based on stereo grayscale cameras, for the highway and extra urban environment, able to detect the road geometry and the obstacle position, size and speed, using edge-based, general geometry, software stereo reconstruction. The lane detection algorithm is described in [3] and the whole system architecture is presented in [4].

Our research in urban stereo began with the aim of designing a robust, real-time sensor for the road and relevant object detection in the urban environment. The prerequisites have changed: the stereo processing is now done by a specialized hardware board, which means some more time is available for the high-level algorithms such as lane detection. It also means that we have more 3D points (the information is now dense, instead of edge only); on the other hand, there are more such high-level processing modules, targeted for various relevant aspects of the urban street: oriented object tracking, pedestrian recognition, pedestrian crossing detection, barrier detection, and so on.

The lane detection subsystem is little influenced by the new stereo system, as the lane can be estimated only by edges. On the other hand, we had to meet a new set of requirements: detection in crowded traffic, with limited visibility of the lane delimiters; detection of non-standard lane geometries; detection of higher curvature roads; discrimination of the lane delimiters in situations when the roads were full of false edge information, and handling the situations where only one lane delimiter is visible.

In order to meet the requirements, a complete system overhaul was needed. The lane detection system was completely redesigned, from the feature extraction algorithm to model matching and state update. The following chapters describe all these phases, and the way they are combined into a robust, real-time lane detector for the urban environment.

2. Lane detection system architecture

The urban lane detection system is organized as an integrator of multiple sensors. Instead of having multiple physical sensors, we have multiple detection stages, which all deliver results that will be used to update the lane model state parameters. The cycle begins with the prediction, and continues with all the detection algorithms, until the final update. When one algorithm updates the lane state, the resulted estimation becomes the prediction for the next stage. In this way, we can insert any number of algorithms into the processing chain, or we can temporary disable some of them, for testing or speedup purposes.

Figure 1 shows the organization of the lane detection system, the main processing modules and the relationships between them. In what follows, we give a brief description for each module, and then, in the next chapters, the most important ones will be described in detail.

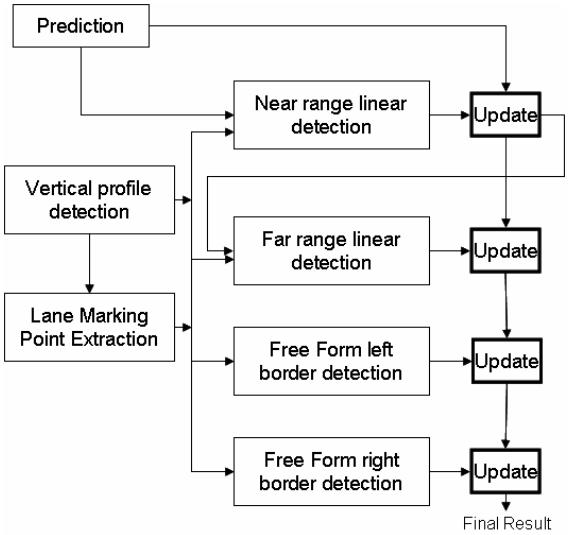

Figure 1. Lane detection system architecture

Prediction – This step applies the equations of state transition to the previous frame's estimation of the lane parameters, using the motion parameters of the ego vehicle (speed and curvature) and timestamp read from the CAN bus. The predicted state vector, along with its prediction covariance matrix, will be used in the detection phase to delimit search intervals, and in the update phase to filter the detection results.

Vertical Profile Detection – The detection of the pitch angle and of the vertical curvature is done using the same stereovision-based algorithm that we have used for the highway scenario [3], adjusted for the distance range of the urban traffic. This step will mark each edge point that has 3D information associated to it as either "road point" or "above road point". The road points are of interest in lane detection, the others are used for the obstacle detection routines.

Lane Marking Point Extraction – Together with edge detection, stereo reconstruction and road/above road labeling done by the vertical profile detection, the lane marking point extraction (classification) algorithm is part of the feature extraction methodology for urban lane estimation. This algorithm detects lane markings as pairs of 3D road points of similar in value but opposing in sign gradients, placed at the proper distance. This step is independent of the prediction, as it has to have universal, model-free application.

Near Range Linear Detection – The core of the model-based lane estimation process is the linear model matching. This algorithm fits two line segments (for the left and right lane border) to the perspective-projected road points, under several constraints that will ensure that these two segments are very likely to be the 2D projection of a section of the lane. First the linear matching is attempted to a range segment close to the ego vehicle, to ensure a minimum detection in restricted visibility conditions.

Far Range Linear Detection – If the near range linear model matching succeeds, the same algorithm is run for the next road section, in order to refine the estimation of curved roads. The far range linear detection is not attempted if in this range we have obstacles on the lane.

Free Form Left Border / Right Border Detection – These routines are independent of the model-based prediction and of the linear model matching algorithms, but they rely heavily on the lane marking extraction results. Each lane border is estimated independently as a chain of 3D points. The results of these routines are used for updating the lane model parameters, but they can be used also as standalone output.

Update – Each of the detection algorithms will update the lane model parameter vector and its associated covariance matrix, by means of Kalman filters. The result of one update will become the

prediction for the next detection algorithm, and the result of the final update will become the final output of the system.

3. Extraction of lane features

The highway scenario allows, in same cases, to perform decent lane detection using the most basic type of features, the edges. This is because the constraints of the lane model ensure the rejection of the noise features, while keeping the good ones. Another advantage of the highway scenario is that the obstacle-free look-ahead distance is usually high, and therefore the chances of getting the right features in the lane model matching process are also high.

These premises are not valid for the urban scenario. The urban scenario is, above all, complex. We may encounter highway-like portions of road, but we may also encounter situations where the free look-ahead distance is very small, where the obstacles are all over the place, where roads have complex textures that produces edges, and so on. Moreover, the lane may not always suit the model, no matter how complex this model is.

In our previous work [3], we have selected as features the edges that had a corresponding 3D height value compatible to the road surface. Thus, the feature detector was in fact the vertical profile detection algorithm, which allowed us to select the road edges. For the urban application, we need to classify these edges further, to extract the lane delimiting markings. An edge belonging to a lane marking will have priority in model matching, while the non-marking edges are useful when markings are not available.

The lane marking is detected using the classical Dark-Light-Dark (DLD) transition principle, which, in gradient terms, is seen as a pair of gradients of similar magnitude but opposable sign. A simple search for gradient pairs has, however, several drawbacks: the level of detail of the road surface decreases with the distance, and the distance between opposing gradients is also variable, both problems being caused by the perspective effect. We would like to filter out the noise in the near range of the road without destroying the edges in the farther range, and we would also like to have a range of acceptable distances between gradient pairs.

The first issue is solved using a variable width filter for computing the horizontal gradient, and the second by using a variable width interval for searching the gradient pairs. The width of the filter and the width of the search interval are computed from the 3D width of the lane markings (a possible range of widths) and the camera parameters, which enable us to compute beforehand the effect of the perspective.

The value of the horizontal gradient of a point of coordinates (x, y) is given by equation (1):

$$G_N(x,y) = \frac{\sum_{i=x+1}^{x+D} I(i,y) - \sum_{i=x-1}^{x-D} I(i,y)}{2D} \qquad (1)$$
$$D = KernelSize(y)$$

Applying the above formula directly is computationally expensive, as D's value may even go beyond 20, for the lowest image lines. However, we can observer that the formula for the gradient of a point differs very little from the formula for the gradient of its previous neighbor. In order to take advantage of that, we have to defer the division by 2D to the end of the image line. Let's denote the un-normalized gradient by G_U:

$$G_U(x,y) = \sum_{i=x+1}^{x+D} I(i,y) - \sum_{i=x-1}^{x-D} I(i,y) \qquad (2)$$

G_U can be computed using a recurrent equation:

$$G_U(x,y) = G_U(x-1,y) + I(x+D,y) - I(x,y) + I(x-D-1,y) - I(x-1,y) \qquad (3)$$

Normalization takes place at the end of each line:

$$G_N(x,y) = \frac{G_U(x,y)}{2D} \qquad (4)$$

Figure 2. Original grayscale image (left) and the result of applying the adaptive horizontal gradient filter (right)

The next step is to eliminate the intermediate values, leaving only the points of minimum and of maximum

gradient. A point of maximum and a point of minimum form together a DLD pair if the distance between them falls inside an acceptable range and the absolute values of the gradients are similar.

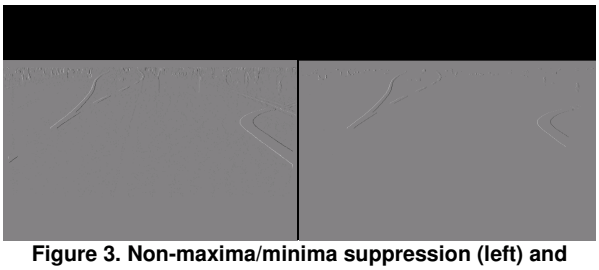

Figure 3. Non-maxima/minima suppression (left) and pairing (right)

The final step is to eliminate the DLD pair edges that do not belong to the road surface, using the 3D information provided by stereovision.

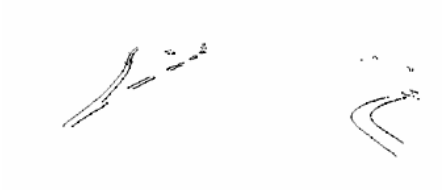

Figure 4. Extracted lane marking edges

4. Linear model matching

In a sufficiently short distance range, the lane geometry can be approximated by a linear (rectangular) segment, described by angle, offset and width. The boundaries of this region are two straight, parallel lines, which maintain their linear property when projected in the image space. The resulted image lines can be detected using standard edge-based line extraction techniques, and they will have to obey several constraints so that they can be regarded as the projection of a 3D lane. These constraints are the following: the lines have to be parallel in 3D (which, in 2D, translates by the fact that they have to intersect at the vanishing line), they have to form a lane of a certain width (not too wide, not too narrow), and this lane must not exclude the ego vehicle (we have to be inside it). These constraints are not only used as an accept/reject system, but they are also a measure of the quality of the possible lane, which allows the selection of the best of all possible candidates.

After the vertical profile is detected, we transform the state prediction $(\overline{\mathbf{X}}(k), \overline{\mathbf{P}}(k))$ into measurement prediction, $(\overline{\mathbf{X_D}}(k), \overline{\mathbf{C_D}}(k))$, using the state-to-image space representation mapping that we have employed in the past for the highway lane detection, method described in [3]. The vector $\mathbf{X_D}$ has the form $\mathbf{X_D}$=(x_{1L}, x_{2L},…x_{nL}, x_{1R}, x_{2R},…x_{nR}), containing the horizontal image coordinates (x coordinates) of the left and right lane borders, for known y coordinates y_i. The y coordinates are in a 1:1 correspondence to a series of Z coordinates, projected using the equations of the vertical road profile.

For the linear lane detection, we require only two consecutive x coordinates of the lane model image space prediction for each side (a single segment).

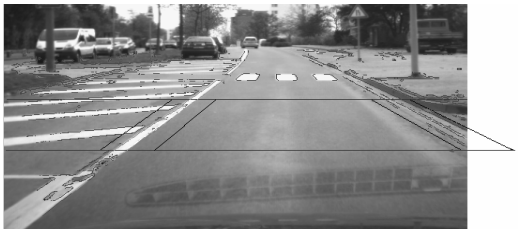

Figure 5. Search areas for linear detection, and the lane delimiting feature points

The primitives for the linear lane model-matching algorithm are the lines, and therefore we need a method that will extract the lines from the road points. The classical method for line extraction is the Hough transform. The standard Hough transform uses a parametrical representation of the line, in the form of a distance ρ from a point of origin, and an angle of orientation θ. For our convenience, we use a modified version of HT, where the line is defined by the x coordinates of the intersections with the top and the bottom limit of the search area: Line(i) = {xtop(i), xbottom(i)}.

For each *xbottom*, each of the edge points is taken into consideration and the *xtop* coordinate is computed. A histogram is incremented at position *xtop*, using a weight given by the edge point's class (if it is a lane marking, the weight is higher). For each *xbottom*, the value of *xtop* corresponding to the highest histogram value is kept, and the others are discarded.

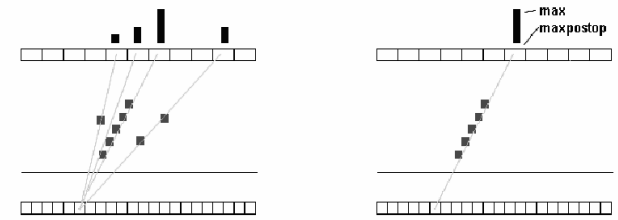

Figure 6. Extracting the lines using a modified version of Hough Transform

What we have so far is a number of *n* lines for the left side, and a number of *n* lines for the right side, which results in a *n* x *n* search space for the best left-right pair. If we had allowed the set of lines to be represented as a 2D Hough space, we would have now to search a n^4 space!

The pair search algorithm is the following:

```
MaxWeight = 0
For each line L_L of the left set
   For each line L_R of the right set
      If Not Valid (L_L) continue
      If Not Valid (L_R) continue
      If Not ValidPair (L_L, L_R) continue
      Weight = ComputeWeight (L_L, L_R)
      If (Weight > MaxWeight)
      MaxWeight = Weight
      BestPair = (L_L, L_R)
If MaxWeight > 0 return BestPair
```

Individual line validation: An individual line can be validated only based on the number of its points. If a line has too few points, it is discarded.

Line *pair validation:* a line pair is valid if it fulfills three conditions:

1. The lane formed by the two lines has an acceptable width
2. The lane formed by the two lines contains the ego vehicle
3. The two lines correspond to parallel 3D lines

The first two conditions are verified using the *xbottom* value of the two lines, as it is not hard to find a 1:1 mapping between the 3D lateral position and the bottom x value (under a reasonable error tolerance, as this is only a validation, not a precise measurement, which will be done later). The third condition, translated in 2D, means that the two lines must intersect at the horizon line – which, knowing the pitch angle, is not difficult to compute.

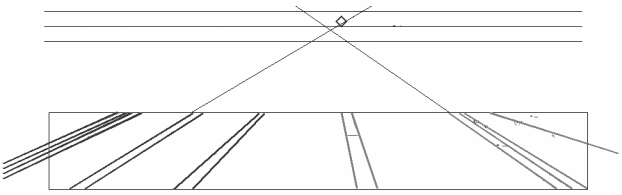

Figure 7. The left and the right line must intersect at the horizon line

Computation of the line pair weight: the best left-right line pair is selected based on a composite weight:

$$Weight = W_{Edge} \cdot W_{Width} \cdot W_{Lateral} \quad (5)$$

W_{Edge} is proportional with the number of edge points belonging to both lines. W_{Width} is maximal for the standard width of 3.5 m, and decreases towards zero as we approach the limits of the acceptable width. $W_{Lateral}$ is maximal if the lane is centered, and decreases to zero as the ego vehicle approaches the lane boundaries.

Figure 8. Linear lane detection result

If the pair detection has failed (no valid pair has been detected), partial detection is attempted. This method uses the prediction of one of the lane delimiters as a pair for the other, whose detection is attempted.

The results of the linear lane detection are used to update the 3D lane model state parameters directly through the Extended Kalman filter, without a prior transformation to the 3D space. The measurement vector is composed of the *xbottom* and *xtop* values for the left and for the right lines if both are detected, or for a single side otherwise.

The linear detection is applied twice: for the near and for a far range. The near range results will update the lane state; then a new search area will be generated for the far range, and the linear detection algorithm is applied again, followed by a new update. In this way we are able to estimate curves faster than with a single linear detection.

5. Freeform Lane Detection

Sometimes the clothoid model-based lane detection is not accurate enough to describe reality. This is true especially for the urban scenarios, due to the ad-hoc design of the roads. For these scenarios, we need to find a less restrictive solution, while also maintaining the robustness. The benefits of using the clothoid model are undisputed: small set of parameters, simple and realistic motion model for tracking, geometry

constraints that ensured robustness even on noisy roads, and the ability to function in the absence of lane markings. It is not easy to discard all these advantages, and still maintain a reasonable robustness, and therefore compromise has to be made. We have decided to rely on the lane markings alone for the freeform border detection, because, in the absence of a guiding model, we have to be pickier about the features we use.

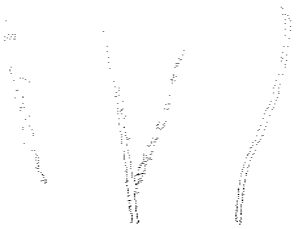

Figure 9. Bird-eye view of the 3D lane marking points for a lane not suitable for clothoid modeling

The top view of the lane marking 3D points is transformed into a binary image. However, fitting a curve to a binary image is not easy, due to the discrete nature of the data, which leads to an "all or nothing" measure of the quality of matching. A solution to overcome this problem is the Distance Transform - the distance transformed image has the property that each pixel has a value proportional to the distance between its coordinates and the coordinates of the nearest edge point.

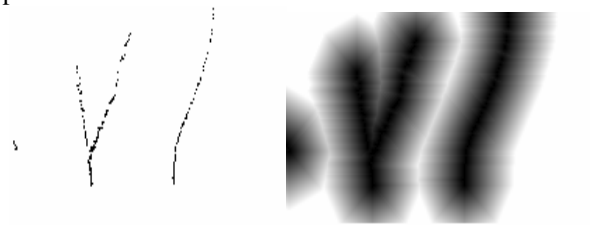

Figure 10. The binary image generated by the top view of the marking points is subjected to Distance Transform

The freeform lane is represented as a Catmull-Rom spline. The Catmull-Rom spline [1] has been used for lane detection before, as described in [5], where the detection was done in the perspective image space. This type of curve is a smooth interpolator of a set of control points **Pi**, having the supplementary property that the tangent to the curve in the control point $\mathbf{P_i}$ is parallel to the line formed by the control points P_{i-1} and P_{i+1}. One other convenient property is that this type of curve actually passes through the control points, making it very easy to handle, and very useful for drawing complex curves from a limited set of control points.

Our spline is based on a set of four control points. The vertical coordinate of these points is fixed, leaving the horizontal coordinates to be searched for – this results in a four-dimensional search space. Figure 11 depicts the search problem. The distance between the curve and the image data is computed using a subset of the curve points – the control points and six intermediate points.

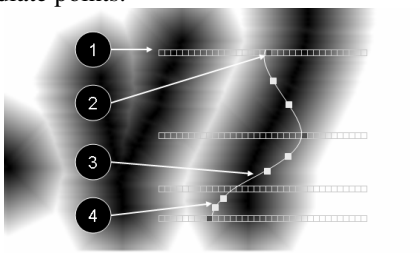

Figure 11. The search problem for freeform detection: 1 - The search space for one of the control points. The *y* coordinate is fixed; 2 - A control point instance (hypothesis); 3- The Catmull-Rom spline generated by the control points hypotheses in the current iteration; 4 - The intermediate points, which, together with the control points, are used for evaluating the distance between the curve and the image

The search for the best delimiter is performed using simulated annealing. The initial position o the curve is in the center of the image, and the random factor of SA is biased towards left or right, depending on which delimiter we want to detect. The final result is converted back into the 3D space, and is represented as a chain of 3D points for each lane delimiter.

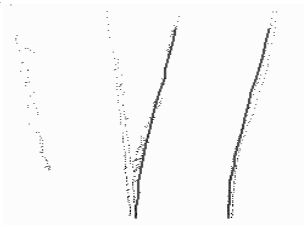

Figure 12. Top view of the 3D polylines representing the lane borders

Besides delivering them as polylines to a driving assistance application, we use the results of freeform detection for updating the clothoid-based state, through the Kalman filter. The measurement vector has the form $\mathbf{Y} = [X_1, X_2, ..., X_n]^T$, a series of X coordinates for given distance coordinates Z.

6. Decreasing the number of false positives

Lane tracking is especially vulnerable in the

initialization phase. The search space is very broad, and therefore a lot of false features can be regarded as road delimiting features. This is not the worse problem, however, as the first detected result is not validated for output. The problem is that the first detection result will define the search regions for the next detection. If these regions are erroneous, the next frames will also lead to false results, either positive or negative, but never correct. Such a case is presented in the following figure; the road surface is full of edges, and therefore a random positive result may appear.

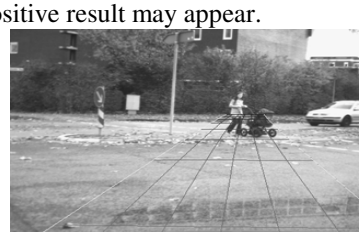

Figure 12. Initializing the tracking without care can lead to severe false positives

The idea for correction is the following: if there are a lot of road features, but they do not define a truly valid road, successive detections started from the zero-time condition (with full search region) will provide different results. If a lane is present, the results will be similar, and tracking can begin.

The tracking initialization system is implemented as a ring buffer of size N. The system requires N consecutive short range linear lane detections of both sides of the lane. If a failure occurs before this count is reached, everything is reset. If the count is reached, the differences between lane parameters of consecutive results are compared, and the maximum difference is evaluated against a threshold. If the values are below the chosen thresholds, the tracking is initialized, and we operate in tracking mode (the search regions are decreased in size, the results are filtered, etc).

The number N must be adequate to the scenario. If we are on a highway and travel at high speeds, the scene may change too fast, and we won't have the time to analyze a lot of frames before tracking initialization (we may never reach tracking state). We choose N by looking at the speed of our vehicle (read from CAN), in the following way:

Speed interval (km/h)	N
0-10	6
10-20	4
20-70	3
70-	2

This method of tracking initialization has proven successful in eliminating the vast majority of false positives, while having a very low impact on the cases of correct detection (a delay in lane output since the system starts or since the scene changes severely is the most frequent problem). However, this solution is useless in the situation when the lane tracking is already initialized in good (or acceptable) conditions, but somehow the scene changes into a very edge-rich, noisy image.

The following image shows a situation when the lane is correctly tracked, even in the presence of severe noise, as it was initialized earlier.

Figure 13. Lane correctly tracked, before the noisy zone

The situation soon changes for the worst. The noisy features in the image increase and the valuable features are all nearly extinct. Lacking clear data, the lane model is matched to the wrong features, causing false results, as shown in the next picture.

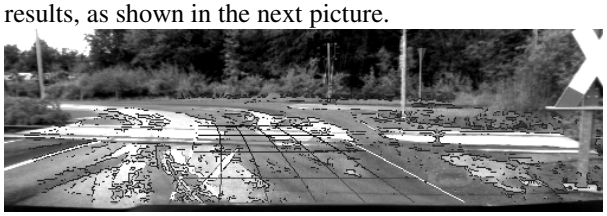

Figure 14. False results caused by multiple noise edges, after the tracking has been initialized

In order to solve this problem, we have to monitor the edge density of the road. If the edge density of the surface between the (predicted) lane borders is comparable or greater than the edge density around the borders, then we have a dangerous situation.

In order to monitor the edge density, we build a polar histogram, which counts the road edge points along rays that pass through a pole which is the predicted vanishing point (fig. 15).

Then, we look for histogram maxima in three regions of equal angular size: around the left delimiter, around the right delimiter, and between delimiters.

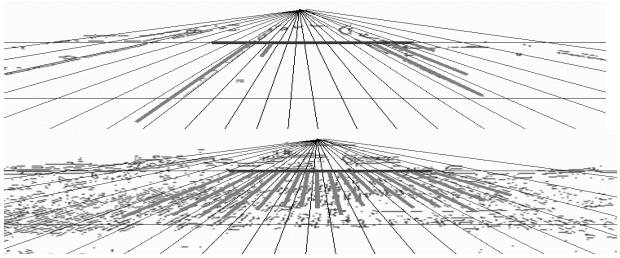

Figure 15. Polar histograms to check the edge density. Top – the shape of a histogram for a clean road. Bottom – the shape of a histogram for a noisy road

If the road is clean, the value of the middle maximum should be significantly lower than the value of the left or of the right maxima. Our condition is that the value of the middle maximum should be at least four times lower than the side maxima, if the road is to be looked at as clean.

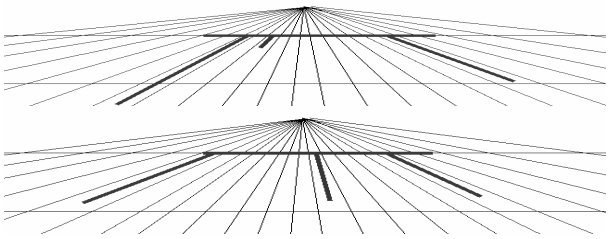

Fig. 16. Comparison between regional maxima: Top – clean road, the middle maximum is much lower than either side maximum. Bottom – the middle maximum is almost equal to the side maxima.

If, according to the test described above, the road is not clean, we have to impose additional constraints to lane detection. The constraints are applied selectively, to the border that fails the test (left, right or both), and consist of eliminating the non-marking edges from the model matching data set. In this way, if we have a very noisy road that has some visible lane markings, lane detection will be performed on those markings, thus leading to good results, or, if there are no markings, or they are not detected by us, the lane will not be detected at all. In both situations we avoid delivering a false result. Thus, we enforce a "lane marking only" policy for the noisy roads, while allowing a simple edge-based detection for cleaner roads that have partial or non-existent lane markings.

Figure 17 shows the results of applying the edge density test and remedy for the situation of figure 14, but this time the result is no longer affected by the high density of the noise.

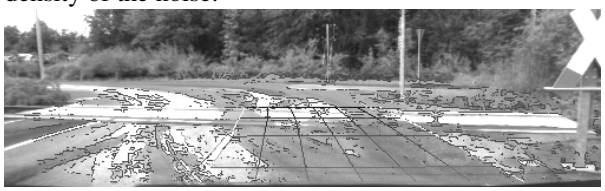

Figure 17. Using the region edge density test, the lane detection results remain correct.

7. Results

The urban lane detection system was tested in multiple real-life traffic scenarios, which covered the following difficulty factors: heavy vehicle traffic, incomplete lane delimitation – permanent or temporary absence of one delimiter, high curvatures, complex lane geometries or widths, bad visibility of lane delimiters, noise features on the road which can be mistaken for lane delimiters, bad general visibility conditions. Thanks to the integration of lane detection techniques the system was able to robustly solve the vast majority of situations, while maintaining a real-time behavior – lane detection processing time takes less than 30 ms on a standard P4.

Figure 18. Lane detection results

8. References

[1] E. Catmull, R. Rom, "A Class of Local Interpolating Splines", in Computer Aided Geometric Design, Academic Press, 1974, ISBN 0120790505

[2] U. Franke, D. Gavrila, S. Goerzig, F. Lindner, F. Paetzold, C. Woehler, "Autonomous Driving approaches Downtown", IEEE Intelligent Systems, 1999

[3] S. Nedevschi, R. Danescu, T. Marita, F. Oniga, C. Pocol, S. Sobol, T. Graf, R. Schmidt, "Driving Environment Perception Using Stereovision", Procedeeings of IEEE Intelligent Vehicles Symposium, (IV2005), June 2005, Las Vegas, USA, pp.331-336.

[4] S. Nedevschi, R.Schmidt, T. Graf, R. Danescu, D. Frentiu, T. Marita, F. Oniga, C. Pocol, "3D Lane Detection System Based on Stereovision", IEEE Intelligent Transportation Systems Conference (ITSC), 2004, Washington, USA

[5] Y. Wang et al., "Lane detection using spline model", Pattern Recognition Letters, 2000