Detection and Classification of Painted Road Objects for Intersection Assistance Applications

Radu Danescu, Sergiu Nedevschi, *Member, IEEE*

Abstract—**For a Driving Assistance System dedicated to intersection safety, knowledge about the structure and position of the intersection is essential, and detecting the painted road signs can greatly improve this knowledge. This paper describes a method for detection, measurement and classification of painted road objects that are typically found in European intersections. The features of the painted objects are first extracted using dark light dark transition detection on horizontal line regions, and then are refined using gray level segmentation based on Gaussian mixtures. The 3D bounding box of the objects is reconstructed using perspective geometry. The objects are classified based on a restricted set of features, using a decision tree and size constraints.**

I. Introduction

The intersection is the most complex part of any traffic scenario, the place where most of the classic driving assistance systems encounter significant problems. The motion of the obstacles is not restricted to mostly one direction, and the lanes are no longer compliant with simple models. For these reasons, in the intersection scenario the perception should be focused on extracting the most information out of disparate, independent cues. Instead of detecting lanes as a whole, we'll detect individual painted road objects, and classify them to identify their nature. The information extracted by this process will help a driving assistance system to accurately perceive the position, orientation and nature of the intersection itself with respect to the vehicle.

For many years, researchers dedicated their work to the aim of finding better extraction techniques for road markings, usually for the purpose of lane detection. In [1] a comparative analysis of the most popular marking extraction techniques is presented, and the best results are found to be provided by the algorithms that compare the brightness of the feature pixel with the brightness of its neighbors at specific distances, which account for the size of the marking to be found and for the perspective effect.

The identification of painted road objects mainly for lane detection does not deter some of the researchers in trying to identify the markings individually, and extracting some of their properties. The method presented in [2] uses histogram-based image segmentation and then applies decision trees on basic geometrical features to decide whether a painted object is a lane marking or not, and also to decide whether a set of objects are part of the same lane border. In [3] the painted road objects are extracted as polygons, and their basic properties such as length, parallelism and direction are used to decide whether they are possible lane delimiters or pedestrian crossings. In [4] we find a technique for extracting the rectangular painted objects by the use of an analytical model of the expected pixel intensity based on rectangle parameters, and the search for the parameters that minimize the error between the expectation and the image data. The aim of this technique was to identify lane delimiters in intersection scenarios.

A system that is able to identify and classify a wider range of painted road objects, not only lane marking delimiters, is presented in [5]. The segmentation is based on intensity levels and connected components analysis, and the classification uses a radial basis function classifier.

Instead of using the painted road markings for lane detection, some authors use the already detected lane as a guide in the search for specific objects such as arrows. The work presented in [6] uses the hypothesis that the arrows are in the middle of the lane, and therefore restrict their possible locations. The detection and classification of arrows is done by pattern matching on inverse perspective mapped images.

The purpose of this paper is to present a fast, robust and reliable algorithm for detection and classification of the most common painted road objects in the intersection.

II. Objectives

The painted road objects can be of multiple types: arrows, delimiters, lane markings, pedestrian crossings, traffic signs. Developing a system to detect and classify them all is not an easy task, and the computational power required may be more than we can spare. For this reason, we have selected a restricted set of objects, which are of highest importance and occur with great frequency in the European intersection scenario. Our aim is to develop a system that will reliably detect, measure and classify the types of objects shown in figure 1 with as little computational overhead as possible.

This paper will describe the process of detection and classification of painted road objects, applied step by step on the intersection scene presented in figure 2. The main steps of the algorithm are object segmentation, 3D reconstruction,

Manuscript received March 24, 2010. This work was supported by the European Commission through the INTERSAFE-2 FP7 Project, and by the POSDRU program, financing contract POSDRU/89/1.5/S/62557.

Radu Danescu and Sergiu Nedevschi are with the Technical University of Cluj-Napoca, Computer Science Department (e-mail: radu.danescu@cs.utcluj.ro). Department address: Computer Science Department, Str. Memorandumului nr. 28, Cluj-Napoca, Romania. Phone: +40 264 401457.

and object classification.

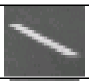
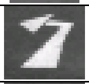
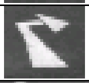
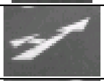
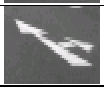
	Lane marking or pedestrian crossing element
	Forward arrow
	Left arrow
	Right arrow
	Forward + left arrow
	Forward + right arrow

Fig. 1. Painted road objects to be classified.

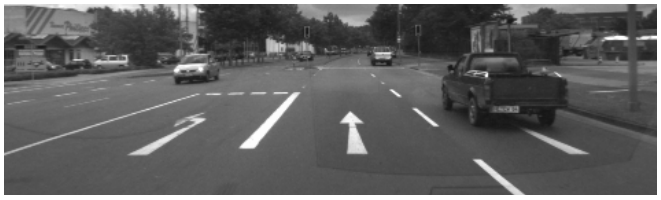


Fig. 2. Painted road objects at an intersection.

III. Image Segmentation

A. *Initial segmentation*

In order to identify the painted road objects, we have to search for dark-light-dark transition patterns. Instead of pairs of gradients of opposing signs and similar magnitude, which is the traditional way of lane marking extraction, we use the already existent road edges (image edges that have a 3D correspondent located on the road surface, figure 3). The 3D road points are selected after the pitch of the vehicle is detected using the method described in [7].

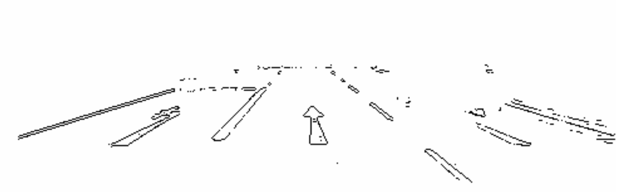


Fig. 3. Road edges.

The edges have the role of dividing each horizontal image line into regions, as shown in figure 4. A region is the part of an image line that covers the horizontal distance between two edge points.

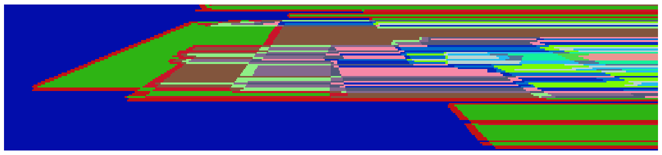


Fig. 4. Image lines are divided into regions by the road edges.

For each region, the average intensity of the corresponding pixels in the original grayscale image is computed. The average intensity of each region is compared to the average intensity of the neighboring regions. A valid region is one that has the average intensity higher than the intensity of its neighbors, and the intensities of the neighbors are similar. Another validation condition for a line region is imposed on its width: the width must be lower than the perspective projection of the widest acceptable object, for the specific image line. The widest acceptable object is the pedestrian crossing element.

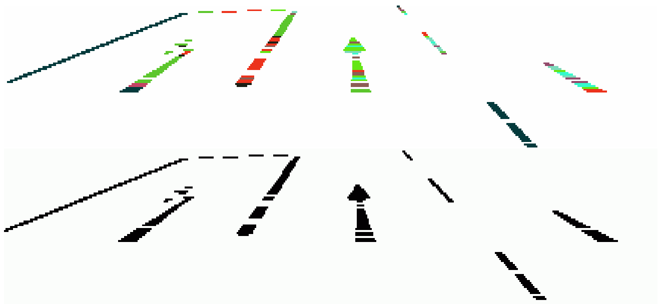


Fig. 5. Valid line regions (top) lead to initial segmentation (bottom).

The valid line regions that remain after validation select the initial pixels of the painted road objects (fig. 5).

B. *Refining the segmentation*

The results of the initial segmentation, based solely on the validity of the line regions, show that the road objects may appear incomplete. In what follows, we'll refine the segmentation based on the intensity of the markings and of the background. First, an intensity histogram of the already segmented markings is computed and normalized so that it will approximate a probability density function. This probability density function is multi-modal, and for this reasons we'll try to approximate it by a mixture of Gaussians. Instead of using complex algorithms such as Expectation-Maximization, we have developed a Greedy style approach, which works reasonably (sub-optimal) well, in linear time, and does not require prior knowledge about the distribution (such as the number of Gaussians).

First, we'll define the function $S_H(\mu,i)$, which can compute the square of the standard deviation for a peak μ of a normalized histogram *H*, by analyzing the ratio between the histogram value in the peak and an arbitrary value at position *i* near the peak (equation 1).

$$S_H(\mu,i) = \frac{(i-\mu)^2}{2\ln\frac{H(\mu)}{H(i)}} \qquad (1)$$

Then, the computation of the standard deviation for a certain peak can be done by averaging the values of

$S_H(\mu, i)$ in a window of radius d around the peak.

$$\sigma_H^2(\mu, d) = \frac{1}{2d+1} \sum_{i=\mu-d}^{\mu+d} S_H(\mu, i) \qquad (2)$$

The window d should be smaller than the expected standard deviation, so that other peaks have less chance of influencing the estimation.

The extraction of Gaussians from a histogram H is done by the algorithm described in figure 6. The constants of the algorithms are d, the averaging window size, and t, the stopping threshold, which describes what proportion of the total distribution should be covered (ideally t should be 1, but in practice a value of 0.8 ensures that we don't select a lot of irrelevant peaks).

Algorithm *Gaussian fit*
Inputs: H – normalized histogram
Outputs: L_μ - list of means, L_σ - list of standard deviations, L_w - list of weights

```
$L_\mu = \Phi$
$L_\sigma = \Phi$
$L_w = \Phi$
$c = 0$
while c<t
    $\mu = \arg\max_i (H(i))$
    $\sigma = \sqrt{\sigma_H^2(d, \mu)}$
    $w = 0$
    for k= $\mu - 2\sigma$ to $\mu + 2\sigma$
        $w = w + H(k)$
        $H(k) = 0$
    end for
    $c = c + w$
    $L_\mu = L_\mu \cup \{\mu\}$
    $L_\sigma = L_\sigma \cup \{\sigma\}$
    $L_w = L_w \cup \{w\}$
end while
```

Fig. 6. The algorithm for Gaussian mixture approximation

The algorithm works in a Greedy manner, finding maxima in the histogram, declaring it the top of a Gaussian peak, and then removing the whole Gaussian bell from the histogram, until no more histogram data remains.

The same algorithm is applied to the foreground (markings) and to the background (asphalt). The results for our test case are shown in figure 7.

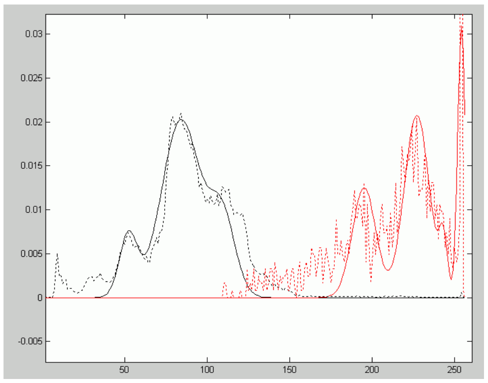


Fig. 7. Histograms and Gaussian mixture approximations for background (left, black) and markings (right, red).

The next step is to label each marking point with the identifier of its associated (nearest) Gaussian peak. Then, each labeled marking point is extended into the neighboring pixels by a process of morpho-dilation (figure 8). In this way, the gaps between marking fragments are closed.

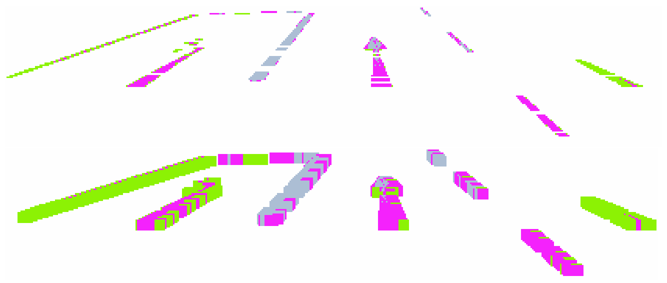


Fig. 8. Marking points labeled by their associated Gaussian peak (top) and the result of dilation (bottom).

Now, each dilated point i is subjected to the probability test (equation 3), which tests whether it belongs to a marking or to the background.

$$p(marking \mid i) = \frac{p(i \mid marking)}{p(i \mid marking) + p(i \mid background)} \qquad (3)$$

The probability of the pixel i given the marking distribution is given by the associated Gaussian bell for the label of i, l_i (resulted from dilation) and the intensity of the grayscale image at point i, I_i.

$$p(i \mid marking) = w(l_i) Gauss(\mu(l_i), \sigma(l_i), I_i) \qquad (4)$$

The probability of the pixel i given the background distribution is given by the background mixture directly, without restriction to any peak.

By restricting the marking probability to the peak of its dilation label, we are enforcing the condition that the pixel that initially was not found as a marking point must have similar intensity to its initially extracted neighbors. This is a consistency condition that ensures the reduction of false positives.

For our test case, the values of the marking probability function, scaled in the interval 0-255 and negated, are shown in figure 9.

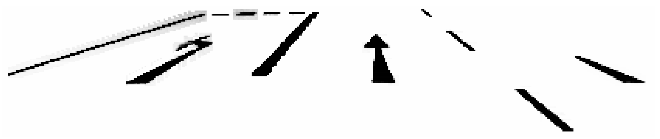


Fig. 9. Object probability image

Now, the probability image can be thresholded. Due to the fact that the difference between objects and background is quite clearly visible, a fixed threshold of 0.5 is enough. After thresholding, a labeling algorithm is applied to identify individual objects.

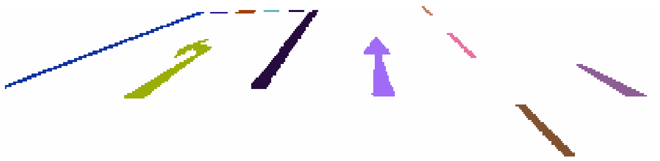


Fig. 10. Individual objects identified by labeling

IV. Extraction of 3D Cuboids

The output of the road objects detection system will be a list of 3D cuboids (of height zero), with position, size and orientation, which can be used by an intersection environment perception application.

One possible method of extraction of 3D information from 2D image objects is to use an Inverse Perspective Mapping (IPM) transformation, as we know that the objects must be located in the road surface. However, IPM tends to amplify the flaws in segmentation, and damage the shape of the objects. We can avoid using IPM if we reason about the perspective effect in the image space.

The first thing that we need to find is the orientation of the object. Because the objects may not be symmetrical, as in the case of left or right arrows, we cannot use the axis of elongation. Instead, for the left and right sides of the object, we fit lines using the RANSAC technique [8]. By counting the number of points that are located on the resulting lines, we can select which side is closer to the linear model. The line of this side is the dominant line (figure 11), and this line will give us the orientation of the whole object.

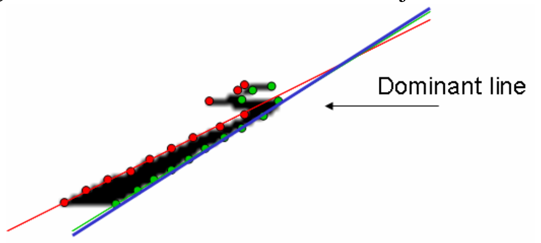


Fig. 11. Line fitting to the object's sides, using RANSAC, and finding the dominant line.

We'll assume that the 3D correspondent of the image space dominant line is parallel to the orientation of the 3D object. This means that the projections of the 3D object's sides in the image space will intersect the dominant line at the horizon, and will pass through the extreme points of the object. This condition leads to the algorithm of finding the object's bounding box:

1. Intersect the dominant line with the horizon line (which can be found using the pitch angle, computed from lane detection, or by finding a vanishing point of the road), as shown in figure 12.
2. From the intersection point, cast rays that pass through the leftmost and rightmost points of the object. Intersect these lines with the top and bottom horizontal limits of the object, as shown in figure 13.

At this point, we have bounded our 2D object with two lines which are parallel to its orientation in 3D, and pass through its extreme points.

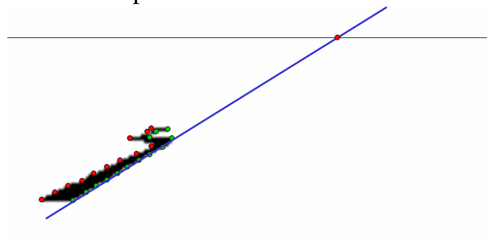


Fig. 12. Intersection between the dominant line and the horizon line.

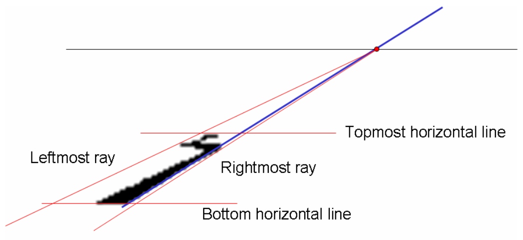


Fig. 13. Extraction of object boundaries in the image space.

Now, the reasoning must move to the 3D space. For that, we have to find the 3D correspondents of the four image space corners. In order to do that, we can use IPM, or dense stereo information about the road surface. In our implementation, the best results were found using a pitch-adjusted IPM (the pitch angle of the road surface was computed from dense stereo information, and this angle allows us to adjust the IPM parameters dynamically with the pitching of the road).

The resulted object is not a rectangle in 3D, but a parallelogram (figure 14). The final step is to compute the encompassing rectangle, and, from this, the parameters of the 3D object: size (width and length), position and orientation.

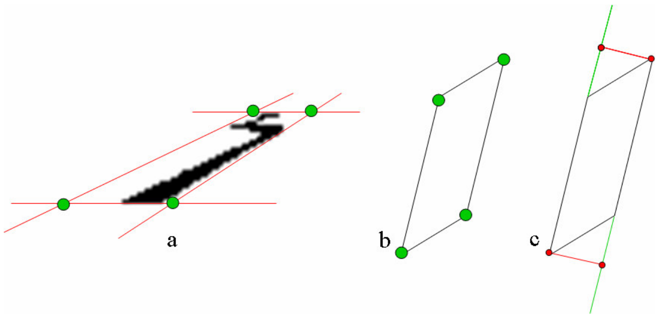


Fig. 14. Reconstruction of the 3D oriented rectangle: a) trapezoid in the image space, b) parallelogram in the 3D space, c) rectangle in the 3D space.

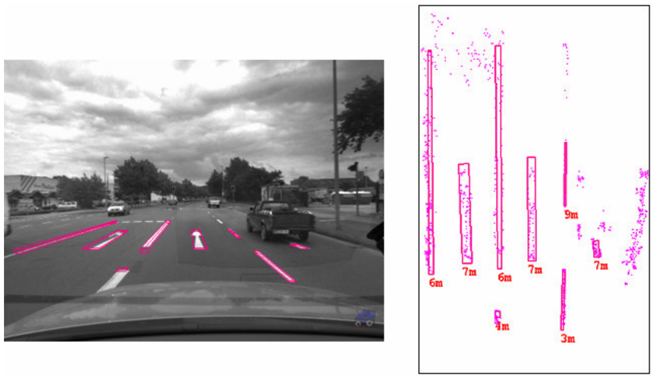


Fig. 15. Reconstruction results, in perspective view (left) and bird-eye view (right).

V. Object Classification

A. Features for classification

A classifier's work is greatly facilitated by the selection of a small number of meaningful features. We have found that the relation between an object's border points with the RANSAC extracted lines gives us clues about the object's class. Also, the line parameters are already computed, and therefore the extraction of our features will bring little computational overhead.

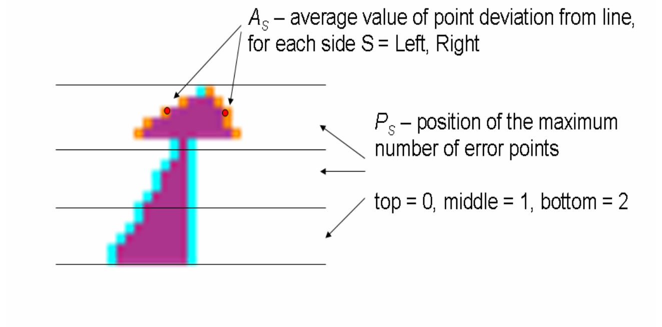

A_S – average value of point deviation from line, for each side S = Left, Right
P_S – position of the maximum number of error points
top = 0, middle = 1, bottom = 2

Fig. 16. Features used for classification.

For each side $S \in \{Left, Right\}$ we extract the following features (see figure 16):

R_S – ratio between the number of points that do not fit on the line, and the total number of points on a border (left or right).

A_S – the average error (deviation from line) for the non-compliant points, for each side.

P_S – the position of the maximum number of error points for each side. The height of the object is divided into three regions, top, middle and bottom, and each region gets a position value, from 0 to 2.

Thus, the two sides of the object give us a set of six numerical features, R_{Left}, A_{Left}, P_{Left}, R_{Right}, A_{Right}, P_{Right}. Now, we have to find a classifier that will use these features to put the object into one of the six relevant classes.

B. The classifier

Initially, we have tried to generate a classifier automatically, by training a decision tree using Weka [9]. However, we have found that we can get a simpler and better classifier by simply encoding in a decision tree some rules about the shape of the object to be classified (figure 17). Such rules are: the lane marking components have a low amount of error points on both sides; the simple arrows have most of the error points in the top section; the composite arrows have most of the error points of one side in the middle section. The decision tree resulted from Weka was used only for threshold selection.

```
**if** ($R_{Left} < 0.3$ **or** $R_{Right} < 0.3$)
    **if** ($R_{Left} < 0.2$ **and** $R_{Right} < 0.2$)
        **return** (lane marking)
    **else**
        **if** ($P_{Left} = 0$ **and** $P_{Right} = 0$)
            **if** ($A_{Left} < -1.34$ **and** $A_{Right} > 1.34$)
                **return** (forward arrow)
            **if** ($A_{Left} < -1.34$ **and** $A_{Right} <= -3.34$)
                **return** (left arrow);
            **if** ($A_{Left} >= 3.34$ **and** $A_{Right} > 1.34$)
                **return** (right arrow)
        **else**
            **if** ($A_{Left} < -4$ **and** $A_{Right} > 1.34$ **and** $P_{Left} = 1$ **and** $P_{Right} = 0$)
                **return** (forward left arrow)
            **if** ($A_{Left} < -1.34$ **and** $A_{Right} > 4$ **and** $P_{Left} = 0$ **and** $P_{Right} = 1$)
                **return** (forward right arrow)
```

Fig. 17. The classification decision tree.

After the object's features are run through the decision tree, and each object receives a class, a final validation based on the 3D size is applied. This will ensure that the object's size is consistent to its class. The following image shows the final product of our system, a set of 3D road objects with their associated class.

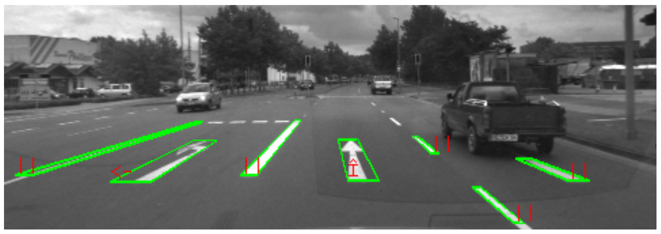


Fig. 18. Painted road objects, detected and classified.

VI. Results

The system was tested on real city images, in different visibility conditions. For our specific setup, which uses a 6.5 mm focal lens, the objects are detected at the maximum distance of 15 meters, as they are heavily affected by the loss of detail of the road surface due to the perspective effect. For comparison, reliable stereovision-based obstacle detection with the same camera setup can be performed at about 30 meters, as the features of the vertical surfaces are not so heavily affected by perspective.

For the classification performance, it is difficult to devise an objective metric. If we count the number of correctly classified instances and compare it to the total number of occurrences, the resulted number is not impressive, because when the object enters the field of view it cannot be reliably classified. The same happens when the object is exiting the image frame, and only parts of it are visible. In this case, the correct rate of classification is about 80%. If, on the other hand, we estimate the classification rate only when the object is in the most visible positions, the classification rate increases to more than 95%. The most important factor that can cause errors in classification is the quality of the road objects. If the paint is worn out, the segmentation algorithm will only get a partial object, which cannot be properly classified.

The computation overhead of the painted road objects extraction system is less than 10 ms, provided that the road edges are already extracted.

Note: a video file showing results can be downloaded from the address http://users.utcluj.ro/~rdanescu/painted.avi .

VII. Conclusion and Future Work

This paper presents a complete solution for detection, measurement and classification of six of the most common types of painted road objects that are found at intersections. The key ideas of our algorithm are the two-step segmentation process, combining dark-light-dark transitions and Gaussian mixtures for a complete reconstruction of the object shape, the extraction of the 3D information from the camera perspective, and the fast classification process that uses a reduced set of relevant features.

There are multiple ways the system may be improved, and one of them is tracking. Tracking can be employed in multiple stages: at segmentation, the histograms of the present may be combined with the histograms of the past; at classification, the class consistency of the objects can be ensured from one frame to another; at 3D reconstruction, the position and size can be filtered in time. Another improvement is the analysis of the relations between individual objects. For instance, pedestrian crossing elements can be combined into a pedestrian crossing type object, or, if an arrow is detected as several smaller objects, the results can be combined and the original object reconstructed.

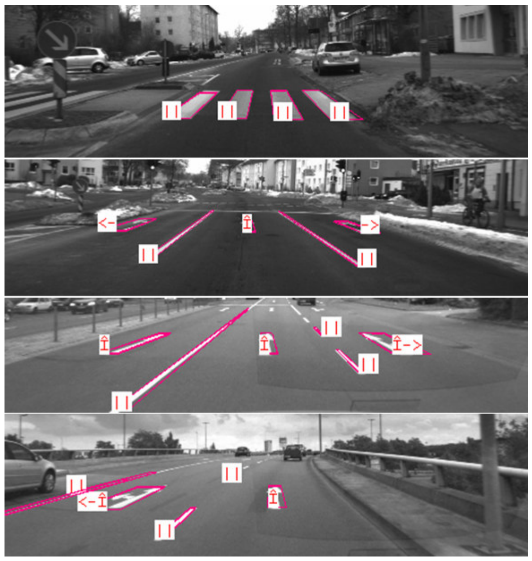


Fig. 19. Results – objects correctly detected and classified.

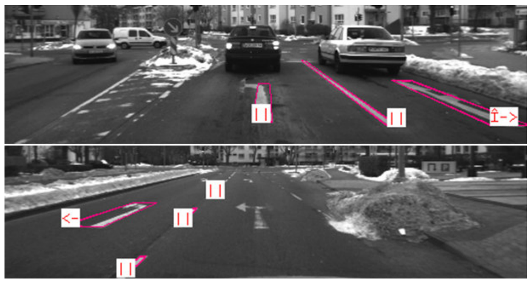


Fig. 20. Results – errors in classification. Top – left arrow misclassified as lane marking, due to partial visibility; Bottom –arrow completely missed due to low visibility.

References

[1] T. Veit, J. P. Tarel, P. Nicolle, P. Charbonier, “Evaluation of Road Marking Feature Extraction”, in proc of *IEEE Conference on Intelligent Transportation Systems 2008 (ITSC 08)*, pp. 174-181.

[2] J. P. Gonzalez, U. Ozguner, “Lane Detection Using Histogram-Based Segmentation and Decision Trees”, in proc of *IEEE Intelligent Transportation Systems Conference 2000* (ITSC 2000), pp. 346-351.

[3] F. Paetzold, U. Franke, “Road Recognition in Urban Environment”, in proc of *IEEE International Conference on Intelligent Vehicles*, 1998, pp. 87-91.

[4] C. Duchow, “A novel, signal model based approach to lane detection for use in intersection assistance”, in proc of IEEE Intelligent Transportation Systems Conference 2006 (ITSC 2006), pp. 1162-1167.

[5] U. Franke, D. Gavrila, S. Goerzig, “Vision based Driver Assistance in Urban Traffic”, in proc of *World Congress on Intelligent Transportation Systems (ITS)*, 2000.

[6] S. Vacek, C. Schimmel, R. Dillman, “Road-marking analysis for autonomous vehicle guidance”, in proc of *European Conference on Mobile Robots*, 2007.

[7] R. Danescu, S. Nedevschi, “Probabilistic Lane Tracking in Difficult Road Scenarios Using Stereovision”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, No. 2, June 2009, pp. 272-282.

[8] M. A. Fischler, R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". *Comm. Of the ACM* 24: pp. 381–395, 1981.

[9] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten (2009); The WEKA Data Mining Software: An Update; *SIGKDD Explorations*, Volume 11, Issue 1.