Detecting and positioning overtaking vehicles using 1D optical flow

Daniel Hultqvist[1], Jacob Roll[1], Fredrik Svensson[1], Johan Dahlin[2], and Thomas B. Schön[3]

***Abstract*— We are concerned with the problem of detecting an overtaking vehicle using a single camera mounted behind the ego-vehicle windscreen. The proposed solution makes use of 1D optical flow evaluated along lines parallel to the motion of the overtaking vehicles. The 1D optical flow is computed by tracking features along these lines. Based on these features, the position of the overtaking vehicle can also be estimated. The proposed solution has been implemented and tested in real time with promising results. The video data was recorded during test drives in normal traffic conditions in Sweden and Germany.**

I. Introduction

The development of active safety systems for vehicles is progressing rapidly as sensors such as cameras become more standard. In this paper, we address the problem of detecting and positioning overtaking vehicles. The ability to detect and estimate the position of vehicles relative to the ego-vehicle is crucial for analyzing the scene. The information can be used to take action if a dangerous situation arises, e.g. if an overtaking car cuts in too closely. For functionality of this kind, early detections of these events are essential for quick actions. Hence, the detection rate must be high, since each missed detection may pose a large safety risk.

[1] proposes a system which uses fixed detection zones and differential images to detect incoming cars. If a detection is made, the detection area is used as a template which is tracked in the next frame. Another approach for detection is to estimate 2D optical flow. [2] uses a fixed detection area in which 2D feature points are found. The feature points are tracked between consecutive frames using the Lucas-Kanade algorithm [3]. If more than a certain fraction of the motion vectors are moving in a particular direction, implying that the car is moving into the image, it is considered to be a detection. The 2D optical flow solution is combined with a radar system (sensor fusion) in order to get more reliable results. [4] implements a system using planar parallax and improves the robustness by using an inertial sensor as additional input in order to compensate for shakiness during a normal drive.

In this paper, we introduce a new, computationally efficient method for detecting overtaking vehicles. Although previous works show promising results, we seek a computationally cheaper solution with a lower complexity. While the detection of overtaking vehicles is of great importance, this problem is a small part of a system with limited resources, so a fast and reliable approach is needed. Our contribution consists of methods for detecting 1D feature points and a way of interpreting the tracked feature result for detecting overtaking vehicles. Our idea is further illustrated in Figure 1. It is also demonstrated that additional information can be extracted from the system, allowing us to estimate the position of the overtaking vehicle relative to the ego-vehicle.

The detection performance is evaluated using real video data collected during test drives. The evaluations show high detection rate and low false positive rate. Due to its low complexity, it is fast which allows it to be used in real time.

Fig. 1: Overview of the idea. 1D feature points are detected and tracked along so called detection lines (blue lines to the left). The slope of these lines are such that they intersect in the vanishing point, meaning that an overtaking vehicle is moving along these lines.

II. Problem formulation

We are interested in the possibility of detecting overtaking vehicles using a single camera setup. The information could be used to initiate or improve other algorithms already implemented such as vehicle tracking. If possible, the solution should also be able to estimate the relative position x_c, y_c and relative speed $\dot{x}_c, \dot{y}_c$ of the overtaking vehicle compared to the ego-vehicle (see Figure 2).

To simplify the problem, we make the assumption that the overtaking vehicle is moving in parallel to the ego-vehicle. This is valid throughout most of the overtaking events, especially in the beginning when the vehicle is first seen in the image. In a central-perspective view, this means that the overtaking vehicle will be moving towards the vanishing point (VP) in the image. This can be exploited using a line that intersects a starting point and the VP, as the axis along which the optical flow can be estimated.

[1]Autoliv Electronics AB, Linköping, Sweden {jacob.roll, fredrik.svensson}@autoliv.com

[2]Div. of Automatic Control, Linköping University, Sweden johan.dahlin@isy.liu.se

[3]Dept. of Information Technology, Uppsala University, Sweden thomas.schon@it.uu.se

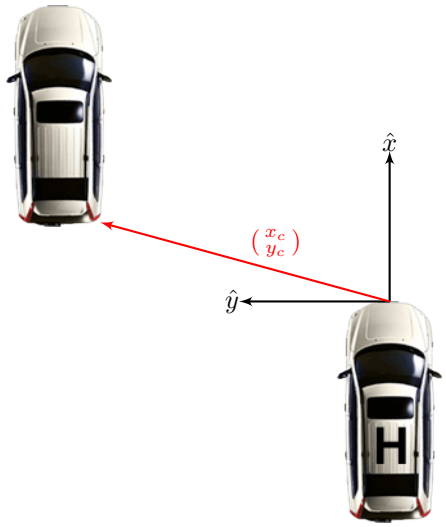

Fig. 2: An overview of the coordinate frame used. x_c and y_c are the coordinates of the right back corner of the overtaking vehicle, in the ego-vehicle coordinate system; $\dot{x}_c$ and $\dot{y}_c$ are their time derivatives.

As the overtaking vehicle is moving further into the image, it will be moving along this line. With this knowledge, a set consisting of several detection lines is created in a region near the edge of the image, which is illustrated in Figure 1. In general, two regions are needed on separate sides of the frame due to multiple lanes or left and right hand traffic. In this paper, we use only the left region as proof of concept.

We split our solution into two parts, one for detection and one for position estimation. The detection part is applied to every frame, while the position estimation is activated once an overtaking vehicle has been detected. The two parts are briefly outlined below.

Detection

1) Create regions of interest where incoming vehicles are searched for. The regions are placed at fixed positions along the edges of the frames. See Figure 1.
2) In each frame, detection lines are created within the detection regions. The intensity values along each line are bilinearly interpolated. In order to reduce noise and effects due to shakiness, the intensity value of each pixel on the line is averaged with its neighbours orthogonal to the line.
3) One-dimensional features are found along each line. The features are tracked into the next frame using a Newton-Raphson algorithm.
4) If the ratio of tracked features that are moving in a certain direction is above a specified threshold τ_d, a detection is considered to be made.

Position estimation

1) If a detection is made, 2D features are located within the detection zone.
2) The feature points are tracked in the next frame using the Lucas-Kanade algorithm [3]. Feature points can be fused if they are close to each other.
3) Using the information about the position and displacements of the 1D feature points, the position of the car can be estimated.

III. Detection lines

We start by deciding a region of interest (ROI) where overtaking vehicles can be detected. The image coordinates for the ROI, i.e. the top and bottom left and the right coordinates, can be determined from world coordinates, e.g., by specifying the closest lateral overtaking distance and that the ROI should range from the ground and up to a specified maximum height. As overtaking vehicles should be detected as soon as possible, the region is placed to the far left of the image. We then estimate the VP using existing algorithms, e.g. [5], whereafter the VP is used to define a set of lines, originating from the left edge of the frame and pointing towards the VP (see Figure 1), which will be parallel assuming a flat road. These lines are referred to as detection lines. For each detection line, the intensity values along the lines are bilinearly interpolated. The line creation phase is summarized in Algorithm 1. Once the detection lines have been created, we detect features on each of the lines.

Algorithm 1 Create the detection lines

Input: Vanishing point v_p, camera height above ground z_{cam}, closest lateral distance y_{min}, maximum height z_{max}, ROI left edge p_x^l, ROI right edge p_x^r, number of lines n_ℓ, line step length ℓ_s.
Output: Detection lines with interpolated values.

- Calculate ROI bottom left and top left corners
  $p^{bl} = [p_x^l, v_{py} + \frac{z_{cam}}{y_{min}}(v_{px} - p_x^l)]$
  $p^{tl} = [p_x^l, v_{py} + \frac{z_{cam} - z_{max}}{y_{min}}(v_{px} - p_x^l)]$
- Calculate line spacing margin $m = (p_y^{bl} - p_y^{tl})/(n_\ell - 1)$
- **for** k in $0,\ldots,n_\ell - 1$ **do**
  - Compute starting point, $p_s = p^{bl} + [0, mk]$.
  - Evaluate the end point, $p_e = [p_x^r, p_{sy} + \frac{p_x^r - p_x^l}{v_{px} - p_x^l}(v_{py} - p_{sy})]$.
  - Interpolate intensity values on the line between $[p_s, p_e]$ with step length ℓ_s.

  end for

IV. One-dimensional features

In the two-dimensional case, there are several common methods to find feature points, e.g. *SIFT* [6], *FAST* [7] and the Shi-Tomasi detector [8]. In the one-dimensional case, however, there are not so many standard methods available.

A. Feature detection

Here, we evaluate three alternative feature point types, listed in Table I. An example of outputs from each of the three approaches can be seen in Figure 3, where the signal consists of interpolated intensity values from one of the detection lines.

The slope approach may be the most intuitively appealing, since it often corresponds to an edge in the image. The standard deviation (STD) approach usually provides the highest number of points, but is harder to grasp intuitively. Here, the slopes approach is selected since it provides sufficiently many feature points per line and gives interpretable results.

Name	Description
Extrema	Find all minima and maxima of the interpolated signal values (i.e. first derivative close to zero) where the magnitude of the second derivatives exceed a certain threshold.
STD	Find all minima and maxima of the interpolated signal values with a standard deviation above a specified threshold within an interval around the extreme point.
Slopes	Find all minima and maxima of the gradient of the interpolated signal values.

TABLE I: Methods of finding one-dimensional features.

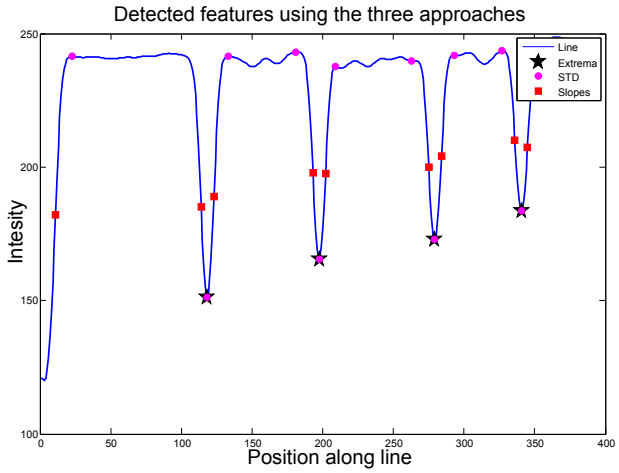

Fig. 3: Detected features for the interpolated pixel values along a detection line for three different approaches.

Algorithm 2 Feature detection

Input: Interpolated values from line ℓ, slope threshold τ_s, suppression distance d_s.
Output: A set of feature points f.

- Estimate first and second derivative of the line ℓ using finite difference.
- Extract points with a second derivative close to zero to a set F.
- **for each** point p in F **do**
  - If first derivative in point p is close to zero, remove p from the set F.
  - If the distance to the previous point is below the threshold d_s, remove p from the set F.

  end for

The detected feature points are clustered by a simple distance interval merging. This is done by starting with the leftmost feature point and removing all features within an interval to the right of the feature point. When the feature points are removed, we move on to the next remaining feature point, etc. The slope feature detection method is summarized in Algorithm 2. The final set of selected feature points is then passed on to the tracker.

B. Feature tracking

A neighbourhood of 15 pixels around each feature point is cut out and used as a template. We then search for the best match of the template along the corresponding detection line in the next frame. We choose the $\mathcal{L}^2$-norm as the distance metric and define the distance $\epsilon(h)$ according to

$$\epsilon(h) = \sum_{x \in T} |F(x+h) - G(x)|^2, \tag{1}$$

where T denotes the relevant interval around feature point x, F denotes the previous frame, G the current frame, and h denotes the translation offset between the two frames. The distance (1) is iteratively minimized to find h by,

$$h_0 = 0,$$
$$h_{k+1} = h_k + \frac{\sum_{x \in T} F'(x+h_k)[G(x) - F(x+h_k)]}{\sum_{x \in T} F'(x+h_k)^2}. \tag{2}$$

A feature point is considered successfully matched if the distance $\epsilon(h)$ is below a specified threshold τ_m. If the difference threshold has not been met within a specified maximum number of iterations, or if the step length is below a limit and the difference threshold is not met, the tracking is considered to have failed. The tracking algorithm is summarized in Algorithm 3. The threshold τ_m can be chosen based on test data.

Algorithm 3 Feature movement estimation

Input: Current frame G, previous frame F, the vanishing point v_p, number of lines n_l, line margins m and start position p. Error threshold τ_m and maximum number of iterations $n_{\max}$.
Output: A set of feature points f with corresponding offsets.

- Create detection lines set L in previous and current frame as described in Algorithm 1.
- **for each** line l in L **do**
  - Check if the line standard deviation is large, if not continue with next.
  - Detect features along the detection line in previous frame using Algorithm 2.
  - **for each** feature point p **do**
    + Find the feature points displacement in the current frame using (2).

  end for

end for

Figure 4 presents an example of successfully tracked features. We see that there is a large number of successfully tracked features. On average, 61.2% of all features are successfully tracked, i.e. with $\epsilon(h)$ converging below τ_m within maximum number of iterations. While this might seem low, a large number of features could be extracted in each frame due to the low complexity of our solution. In our tests, 50 detection lines have been used, setting a maximum of six features on each line, rendering up to 300 features in each frame. This would give us more than 150 successfully tracked features.

C. Issue of repeating patterns

A common problem is repetitive patterns in the signal. An example of this is the poles separating highway lanes. This can also be seen in Figure 3, where each dip represents a pole. During the matching phase, a valid solution can be

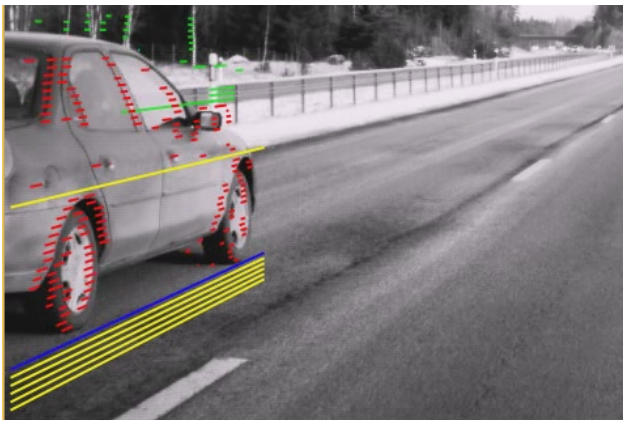

Fig. 4: Tracked features on an overtaking vehicle. Red lines represent features moving towards the vanishing points, green lines moving away from the vanishing point and yellow line were unsuccessfully tracked.

found in each of the dips. Due to the short distances between poles, the matching will be done incorrectly if the ego-vehicle is moving too fast. An example of the pole problem can be seen in Figure 5.

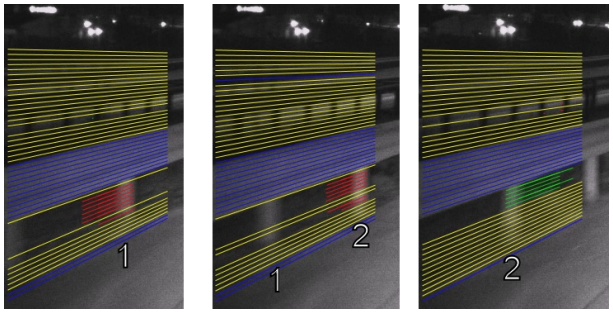

Fig. 5: Sequence illustrating the repeating pattern problem. The poles are marked by an id number to follow them between the frames. As can be seen, pole nr 1 is falsely matched with pole nr 2 in the middle frame. This is because pole nr 2 has moved such that the distance to the previous position of pole nr 1 is shorter than the distance pole nr 1 has moved between the frames.

Our approach to solve the issue is to investigate if the tracking result is unique. This is done by using multiple starting points for each feature. We first detect features in both the previous and the current frame. The feature points from the previous frame are then one by one tracked in the current frame, using the feature points from the current frame as starting points. The tracked feature point is discarded if more than one match is found. This works well as most features have a straightforward correspondence.

Another possible countermeasure is to use a track-retrack approach. First, the feature points from the previous frame are matched in the current frame. When the feature points have been successfully tracked, the tracking is applied in reverse, using the matches in the current frame as start. If the result of the retrack is approximately the original starting point, the tracking is considered successful.

D. Detecting a vehicle

The movement of each feature along each line is now known. During normal driving, i.e. not during an overtaking situation, all features will in theory be moving away from the vanishing point. If a vehicle is overtaking, all feature points on the line that resides on the vehicle will instead be moving towards the vanishing point. When the vehicle has moved sufficiently far into the image, most of the feature points will be on the vehicle.

We choose to detect an overtaking vehicle by checking if a majority of the tracked feature points are moving towards the vanishing point. Normally, a large number of features can be found in the background due to well-textured common objects such as trees, signs, buildings etc. In order to reduce the effect of these, we start evaluating the ratio with a group consisting of the first twenty lines counting from the bottom. If a detection is not made, an additional line is added and the ratio is recalculated. This is done until a detection is made or if all lines have been added to the group. The detection algorithm is summarized in Algorithm 4.

Algorithm 4 Vehicle detection

Input: Feature point set F. Total number of features n_{tot}, start size n_s of detection set and detection threshold τ_d.
Output: Detection decision.

- Create initial set $\mathcal{O}$ with n_s detection lines starting from bottom.
- **while** no detection and not all lines included in set $\mathcal{O}$ **do**
  - Add new line to set $\mathcal{O}$
  - Extract tracked feature translation offsets from F corresponding to lines in set $\mathcal{O}$. (tracked in Algorithm (3))
  - Calculate ratio r of features moving towards VP.
  - **If** ratio r is above τ_d, set detection found and break.

  end while
- Return detection result

V. Position estimation

Having detected an overtaking vehicle, the next step would be to estimate its position (in particular, the position of its closest corner(x_c, y_c)) and longitudinal velocity. To do this, we can realize that the detected feature points give a reasonable estimate of the image position of the vehicle. Using a flat earth assumption, which will in most cases be a good approximation at short distances, we can transform this image position to a world position. This can be used as a measurement in an Extended Kalman Filter (EKF), using a simple vehicle model with the state variables $(x_c\ y_c\ \dot{x}_c\ \dot{y}_c)$ (as defined in Figure 2).

VI. Experimental results

The results reported here are obtained using a MATLAB implementation, which was implemented as a proof of concept. In order to prove real-time performance, the algorithms

have been ported to C++. The latter implementation is unoptimized, but still achieves a processing time of maximum 5 ms for detection and an additional 2 ms for the position estimation.

We used a dataset consisting of approximately 35 minutes of driving mainly on highways. The first half of the dataset is recorded on German highways during the summer. The environment is mostly country side during a sunny day. The second half is recorded in Sweden with mixed seasons with ten minutes driving at night in city environments. The results reported here are obtained using a MATLAB implementation.

A. Vehicle detection

The ability to detect an overtaking vehicle is evaluated by inspecting the ratio of features moving forward during the actual overtake. During normal driving, with no overtaking vehicle, the ratio of features moving forward is on average 5%. An example of an overtake situation can be seen in Figure 6 with corresponding ratio of features moving forward in Figure 7.

As the vehicle enters the image, at approximately frame 18, the ratio of features moving forward increases radically. As the ratio goes above 50%, we declare it as a detection. Note that this graph includes all tracked features. As mentioned previously, in order to reduce the effect of features moving in the background, we start by evaluating the ratio with a set consisting of the bottom 25% of the lines first and increase the set with one line until a detection is met or all lines are included. Because of this, we get a detection sooner, already at frame 19.

Table II presents the detection rate in a dataset consisting of 35 minutes of randomly selected video sequences with overtaking situations.

Lane	True overtakes	Detected	Missed	False det.
Adjacent	32	32 (100%)	0	5
Further away	18	11 (61.1%)	7	0

TABLE II: Detection performance in test dataset. The adjacent lane is the lane directly left of the ego-vehicles. The further away lines are two or more lanes left of the ego-vehicles.

The detection rate is high and all overtakes in the adjacent lane are detected. As a vehicle is moving along the detection lines, there are almost always plenty of features that are successfully tracked, which increases the trust in the system. Darker cars, in particular black, give fewer detected features due to less defined edges, but yet delivers enough to get a detection. The missed vehicles are all driving in the second adjacent lane, and are thus only partly driving within the set of detection lines. This is not a large problem since a proper overtake is considered to be in the adjacent lane.

The false detections in the adjacent lane primarily result from of two different cases. The first kind of issue arises during bumpy rides. If the view of the camera shakes significantly, the assumption of moving strictly forward on

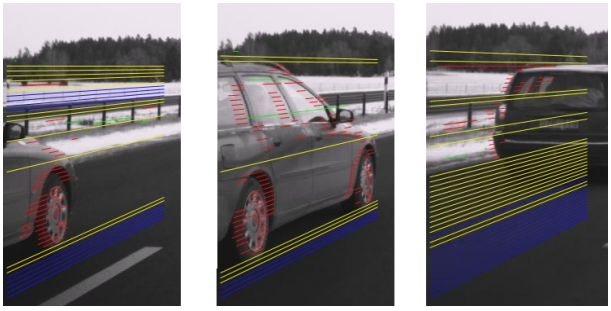

Fig. 6: Example of overtake sequence. As the car moves further into the detection lines, more and more features are detected and tracked. The detection is made already in the first frame because of many tracked features on the wheels. The last detection is made on the third image.

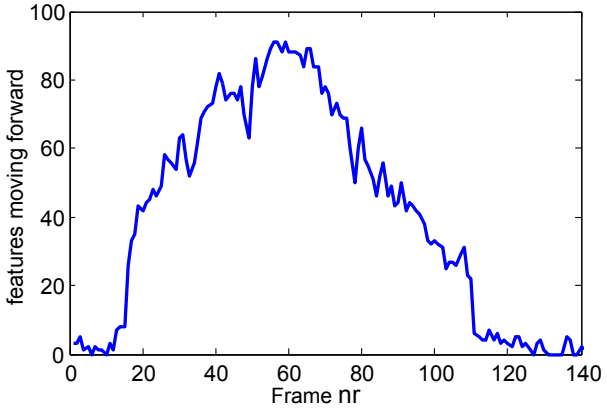

Fig. 7: The ratio of feature points estimated moving forward during an overtake. The ratio increases radically when the overtaking car enters the image. At frame 60, the overtaking car completely covers the detection lines from start to end.

a plane breaks. Due to this, the feature tracking will not be valid and can give a large amount of badly tracked features. It would be straightforward to combine our approach with a dynamic image stabilization functionality to reduce this problem.

The second kind of issue is due to the repetitive patterns arising from the poles between the highway lanes, which is discussed in Section IV-C. This is most noticeable when driving in the lane closest to the poles, where the repetitive pattern is harder to detect. The poles are often on the bottom lines, which will have large impact in our approach of determining detection, since we start with a detection group consisting of the last lines. Our counter-measures using multiple starting points works well when driving second lane from the poles, as there are zero false detections.

B. Position estimation

For the position estimation, there was no ground truth data available, and therefore no proper statistics evaluation has been made. However, as an indication of how the position estimation may work, let us consider a typical overtaking sequence. Figure 8 shows the unfiltered estimated relative

world coordinates for the overtaking vehicle, while Figure 9 shows the unfiltered estimated horizontal image coordinate of the back-right corner during the sequence (see also Figure 10 for a one-frame example). As can be seen, the estimated vehicle position seems reasonable and gives accurate image coordinates for the corner of the vehicle, especially considering that the exact location of the corner may be somewhat ambiguous for many cars.

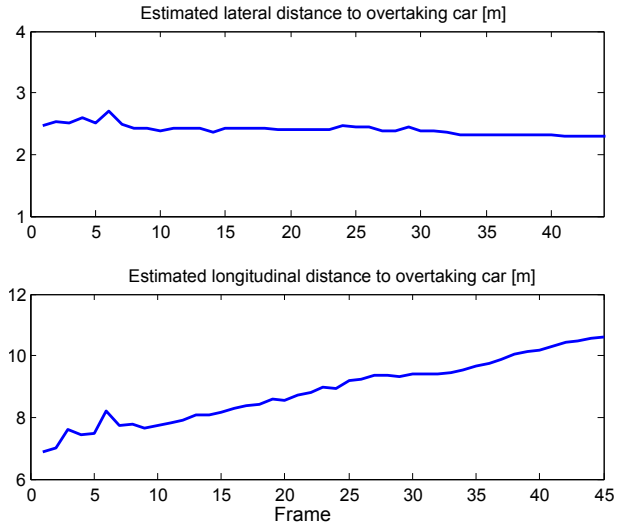

Fig. 8: The estimated world coordinates over the frames.

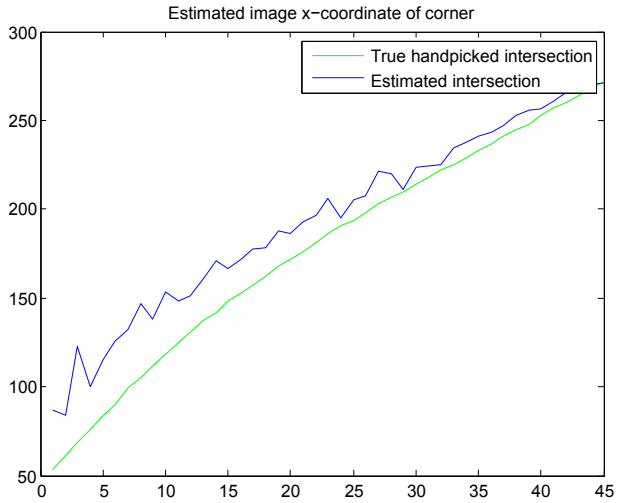

Fig. 9: The estimated horizontal image position of the back-right corner of the vehicle, compared to hand-labelled ground truth.

VII. Conclusions and Future Work

The developed system shows robust performance with a low false positive rate. The 1D feature detector produces many features suitable for tracking. The estimated movement of the feature points provides a simple way of detecting overtaking vehicles. The size of the estimated movements for all features can be used to model the corner position of the overtaking vehicle relative to the ego-vehicle.

A potential performance increase is to implement image stabilization to reduce effects due to shakiness. The position estimation could benefit from using a more advanced model, perhaps with a particle filter, estimating probabilities for each possible position. In the future, the performance of the position estimation should be further evaluated against ground truth data.

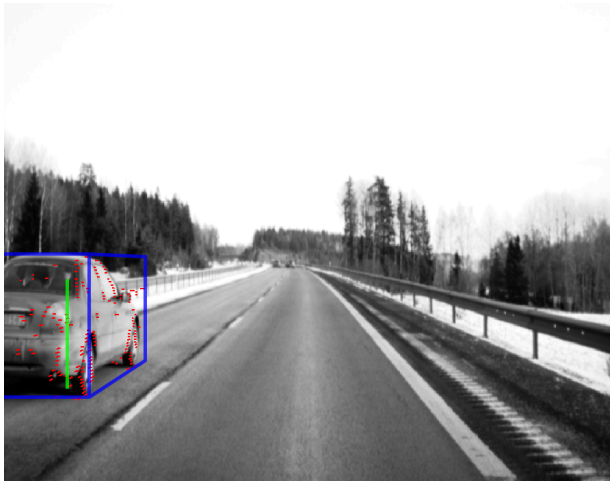

Fig. 10: A single frame example of the estimated horizontal image position of the back-right corner of the vehicle, represented by the closest right edge of the blue cuboid, compared to hand-labelled ground truth, represented by the green line.

References

[1] H. Morizane, H. Takenaga, Y. Kobayashi, and K. Nakamura, "Cut-in vehicle recognition system," in *Proceedings of International Conference on Intelligent Transportation Systems*, Tokyo, Japan, October 1999, pp. 976–980.

[2] F. Garcia, P. Cerri, A. Broggi, A. de la Escalera, and J. M. Armingol, "Data fusion for overtaking vehicle detection based on radar and optical flow," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, Alcala de Henares, Spain, June 2012, pp. 494–499.

[3] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th international joint conference on Artificial intelligence*, Vancouver, B.C., Canada, August 1981.

[4] D. Baehring, S. Simon, W. Niehsen, and C. Stiller, "Detection of close cut-in and overtaking vehicles for driver assistance based on planar parallax," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, Las Vegas, NV, USA, June 2005, pp. 290–295.

[5] N. Gupta, H. Faraji, D. He, and G. Rathi, "Robust online estimation of the vanishing point for vehicle mounted cameras." in *International Workshop on Machine Learning for Signal Processing*, Beijing, China, September 2011.

[6] D. G. Lowe, "Object recognition from local scale-invariant features." in *Proceedings of the International Conference on Computer Vision*, vol. 2, Kerkyra, Corfu, Greece, September 1999, p. 1150–1157.

[7] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking." in *Proceedings of the International Conference on Computer Vision*, vol. 2, Beijing, China, October 2005, pp. 1508–1511.

[8] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, June 1994, pp. 593–600.