Where Will the Oncoming Vehicle be the Next Second?

Alexander Barth and Uwe Franke

Daimler AG, Group Research & Advanced Engineering
Sindelfingen, Germany
{alexander.barth, uwe.franke}@daimler.com

***Abstract*— A new image based approach for fast and robust tracking of vehicles from a moving platform is presented. Position, orientation, and the full motion state including velocity, acceleration, and yaw rate of a detected vehicle are estimated from a tracked 3D point cloud. This point cloud is computed by analyzing image sequences in both space and time, i.e. by fusion of stereo vision and tracked optical flow vectors. Starting from an automated initial vehicle hypothesis, the tracking is performed by means of Extended Kalman Filter. The filter combines the knowledge of where a point in the rigid point cloud has moved within a given time interval, with the dynamic model of a vehicle. The proposed system is applied to predict the driving path of other traffic participants and runs currently at 25Hz (VGA images) on our demonstrator vehicle UTA.**

I. INTRODUCTION

In many safety and advanced driving assistance applications the fast and reliable knowledge of where other moving objects are and how they are moving relative to the ego vehicle is essential. Safety applications require maximum robustness and accuracy, in particular automated emergency braking.

If one wants to design a system that *understands* how objects interact, for example at an urban intersection, one needs to be able, not only to detect other traffic participants, but also to predict their driving path. In curves or turning maneuvers an estimate of the yaw rate of other traffic participants is essential as can be seen in Figure 1. Using a common linear motion model, a collision of the oncoming vehicle with the ego vehicle would be predicted. However, if the yaw rate of the oncoming vehicle is estimated, the situation can be correctly interpreted as uncritical.

A lot of work has been done in the field of image based vehicle detection and tracking in the previous two decades. Systems proposed in the literature include mono and stereo camera settings. Many of these methods track image regions detected as a *vehicle* based on image statistics (color or graylevel histograms, edges, contours, templates,...) in the image plane. However, to predict the driving path of a vehicle it is not sufficient to *rediscover* an image region labeled as a vehicle in the next frame. 3D motion models combined with proper filter mechanisms have to be integrated with the tracking process.

Dellaert and Thorpe [1] use such a 3D motion model to predict position and shape of a bounding box model in the image with an Extended Kalman filter. The yaw rate of tracked vehicles are assumed to be approximately zero.

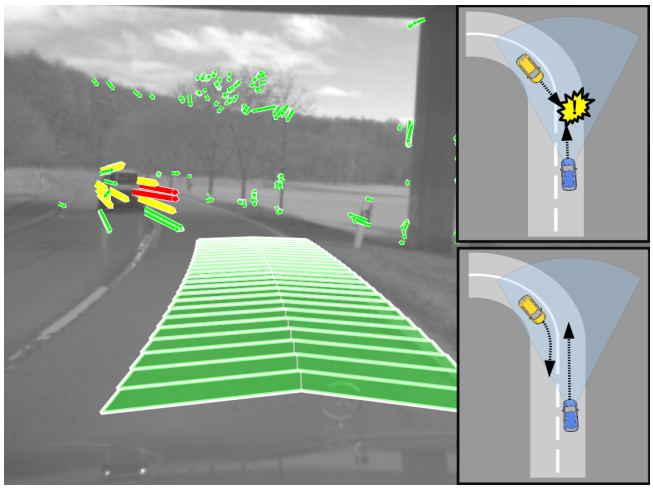


Fig. 1. Linear prediction of the driving path of an oncoming vehicle leads to false alarms in curves or turning maneuvers. With an estimate of the other vehicle's yaw rate, it is possible to predict where this vehicle will really be in the next second.

Dang et al. [2] apply an Extended Kalman Filter to track a rigid point cloud from a moving platform and extract the 3D rotation and translation parameters of this point cloud. Beside rigidity of the point cloud, no constraints regarding shape or motion are defined in this approach. In [3], a method combining a Lidar sensor with a mono camera for tracking of a cuboid object model is presented.

Intersection monitoring is often realized with a stationary camera observing the scene from elevated position [4], [5]. In [4], Koller et al. discriminate moving objects from the background on the basis of image flow, and combine a 3D vehicle motion model with a contour-based fitting of a 3D shape model to track road traffic.

In this contribution a new in-car image based vehicle detection and tracking approach is introduced. 6D vision vectors [6], i.e. Kalman filtered 3D point tracks, are used to get initial object hypotheses. The underlying idea is that a group of points within a local neighborhood moving with equal velocity in the same direction is assumed to belong to the same object.

A coordinated turn motion model [7], which restricts lateral movements to a circular path based on velocity and yaw rate, predicts the object point cloud position ahead. Moving the point cloud in the world induces changes in the

image plane and can be observed in terms of deterministic optical flow and disparity changes.

A Kalman filter is used to solve for the inverse problem, i.e. relating these observations in the image to a movement of the point cloud in the world.

The main novel aspect of our approach is the estimation of the *yaw rate* of an observed vehicle. Real-time cycle times of $40ms$ are achieved on off-the-shelf PC hardware.

The remainder will be organized as follows. In Section II the detection and tracking approach is introduced including object model, system model, and measurement model as well as the automated object hypothesis generation used for initialization. We assume the reader is familiar with the basic concepts of the Extended Kalman Filter. For details we refer to the literature [7], [8]. Both simulated and real world experiments are presented in Section III and the results are discussed in Section IV.

II. APPROACH

For each vehicle a local *object coordinate system* is ideally defined with the origin at the center rear axis on the ground plane (street). This point corresponds to the *rotation point* of the system dynamics. x, y, and z represent the lateral axis, the height axis, and the longitudinal axis respectively (see Figure 2(a)). We will denote the coordinate system of the observing vehicle as *ego* system. All coordinate systems are left-handed.

A. Object Model

An object is represented by a set of 3D points P_i^o in *object coordinates* and a set of parameters describing the motion state and the object to ego coordinate transformation.

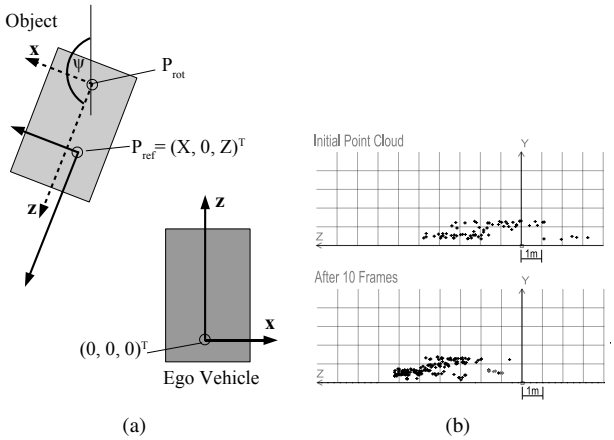
In the proposed method we focus on the estimation of the motion parameters and, thus, define the following Kalman filter state representation:

$$\vec{x} = (X, Z, \Delta X_{rot}, \Delta Z_{rot}, \psi, v, \dot{\psi}, a)^T \tag{1}$$

where X and Z represent the *reference point* P_{ref}^e in ego coordinates on the ground plane with $P_{ref}^e = (X, 0, Z)^T$ (in m). The reference point does not necessarily have to correspond to the center rear axis, which is usually not known at initialization. An additional offset $P_{rot}^o = (\Delta X_{rot}, 0, \Delta Z_{rot})^T$ is introduced representing the *rotation point* in object coordinates (in m). This allows for defining the object coordinate system at an arbitrary point, e.g. the center of gravity of the initial point cloud.

The object orientation is described by the rotation angle ψ (in rad) around the height axis between the ego and object coordinate system. The object's yaw rate is given by $\dot{\psi}$ (in rad/s). The absolute object velocity v (in m/s) is defined along the object's longitudinal axis with acceleration a (in m/s^2).

We separate the problem of motion estimation from the problem of shape reconstruction, i.e. the 3D points representing the object's structure are not included in the filter state to keep a computationally efficient representation. Let us assume for now the correct position of each point is known from beginning and it does not change (rigid body). In practice, the former does not hold due to the uncertainty of the sensor (see Figure 2(b)). Thus, a point update strategy is introduced in Section II-E.



(a) (b)

Fig. 2. (a) Object and ego coordinate system from bird's eye view. Since the position of the object's center rear axis is not known at initialization, the object origin or *reference point* has to be defined at an arbitrary position. The offset to the rotation point P_{rot} is estimated by the filter. (b) Example side view of an object point cloud corresponding to an oncoming vehicle at initialization ($46m$ distance) and after 10 frames ($31m$ distance). The points are referred to the object coordinate system.

B. System Model

The system model describes the dynamics of a tracked object and its points based on a coordinated turn, in plane motion model [7]. The movement of an object on a planar ground can be reduced to a translation and a rotation around the integrated yaw angle within a given time interval Δt and with respect to the rotation point. Both an approximately constant acceleration and yaw rate are assumed. Following the coordinated turn model, the translation vector T^o in object coordinates is computed as

$$T^o = \begin{pmatrix} \frac{v+a\Delta t}{\dot{\psi}}(1 - \cos(\dot{\psi}\Delta t)) \\ 0 \\ \frac{v+a\Delta t}{\dot{\psi}} \sin(\dot{\psi}\Delta t) \end{pmatrix} \tag{2}$$

In general, the transformation of any point P_i^o is given by a rotation around P_{rot}^o plus the translation T^o:

$$P_i^{o'} = R(\dot{\psi}\Delta t)\,(P_i^o - P_{rot}^o) + P_{rot}^o + T^o \tag{3}$$

with R a 3×3 rotation matrix around the height axis by angle $\dot{\psi}\Delta t$.

Applying (3) to the reference point $P_{ref}^o = (0, 0, 0)^T$ in object coordinates yields the predicted position of the vehicle in object coordinates. Since the reference point in the state vector is defined in ego coordinates, this position has to be transformed from object to ego coordinates by rotating around $-\psi$ and translating the result by $(X, 0, Z)^T$. It is possible to compensate for the motion of the ego vehicle using image based approaches [9], [6] or inertial sensors. Thus, for simplification we will assume a static observer in

the remainder of this paper although the system runs on a moving platform.

Finally, the following differential equations define the system model:

$$\begin{aligned}
\dot{X} &= R\left(\underline{\psi}\right)\left(R\left(\underline{\dot{\psi}}\Delta t\right)\left(-\underline{P^o_{rot}}\right)\right. \\
&\quad \left. + \underline{P^o_{rot}} + T^o\right) && (4)\\
\dot{Z} &= R\left(\underline{\psi}\right)\left(R\left(\underline{\dot{\psi}}\Delta t\right)\left(-\underline{P^o_{rot}}\right)\right. \\
&\quad \left. + \underline{P^o_{rot}} + T^o\right) && (5)\\
\dot{\Delta}X_{rot} &= 0 && (6)\\
\dot{\Delta}Z_{rot} &= 0 && (7)\\
\dot{\psi} &= \underline{\dot{\psi}} && (8)\\
\ddot{\psi} &= 0 && (9)\\
\dot{v} &= \underline{a} && (10)\\
\dot{a} &= 0 && (11)
\end{aligned}$$

For clarification, state variables are underlined in the equation above.

C. Measurement Model

Observations of the 3D object points in the stereo image pairs define the measurement model. The initial image position corresponding to the projection of a 3D world point onto the image plane is tracked by a feature tracker such as the well-known KLT tracker [10] between two time steps. In addition, a disparity measurement at the tracked image position is obtained from the stereo system each time step. Thus, a measurement $\vec{z_i}(t) = (u_i, v_i, d_i)^T$ of point P_i^o at discrete time t consists of a subpixel accurate image position (u_i, v_i) and the disparity measurement d_i at this position in the current frame, combining the information of space and time. All M measurements build the total measurement vector $\vec{z} = (z_0^T, ..., z_{M-1}^T)^T$.

The measurement equations result from a perspective camera model (neglecting lens distortions for simplicity):

$$\begin{aligned}
u_i &= h_{i,1}(\vec{x}) = f_u \frac{X_i^c}{Z_i^c} + u_0 && (12)\\
v_i &= h_{i,2}(\vec{x}) = -f_v \frac{Y_i^c}{Z_i^c} + v_0 && (13)\\
d_i &= h_{i,3}(\vec{x}) = f_u \frac{b}{Z_i^c} && (14)
\end{aligned}$$

where f_u and f_v are the horizontal and vertical focal length of the camera, (u_0, v_0) the principal point, and b the base line of the stereo system. $P_i^c = (X_i^c, Y_i^c, Z_i^c)$ is the point P_i^o in camera coordinates. The total transformation is composed of an object to ego transformation (parametrized by the state variables ψ, X, and Z), and the transformation between ego and camera system (including the extrinsic camera parameters).

Since both the system and the measurement model are non-linear, an Extended Kalman Filter approach is used for tracking. The measurement matrix H is given by

$$H = \begin{bmatrix}
\frac{\partial h_{0,1}}{\partial \vec{x}} \\
\frac{\partial h_{0,2}}{\partial \vec{x}} \\
\frac{\partial h_{0,3}}{\partial \vec{x}} \\
\vdots \\
\frac{\partial h_{0,M-1}}{\partial \vec{x}} \\
\frac{\partial h_{0,M-1}}{\partial \vec{x}} \\
\frac{\partial h_{0,M-1}}{\partial \vec{x}}
\end{bmatrix} \qquad (15)$$

The power of this approach comes from the large number of contributing points. The single point is not that important. This increases the robustness of the system significantly. A varying number of M points due to lost or new point tracks can be easily handled, since each point simply adds another measurement to the filter.

D. Initialization

Up to know we have assumed that the 3D points belonging to an object are known. In practice we need an initial segmentation to generate an object hypothesis and start the tracking. A very powerful image-based method for detecting and tracking the (linear) motion of 3D points with Kalman filters has been introduced in [11], [6]. Such filtered 3D point tracks can be represented as *6D vectors* [1], combining the 3D position with a 3D velocity vector. Clustering of compatible 6D vectors within a local neighborhood yields candidate point clouds. In our system a fast histogram based clustering is used, where *compatibility* of the velocity components is defined in terms of the Mahalanobis distance.

A candidate point cloud with sufficient support (e.g. number of points above a threshold) generates a new object hypothesis. The center of gravity $C^e = (C_x^e, C_y^e, C_z^e)^T$ in ego coordinates and the mean velocity vector $\bar{V} = (\bar{v}_x, \bar{v}_y, \bar{v}_z)^T$ of the corresponding point cloud define the initial filter state $\vec{x}_0$:

$$\vec{x}_0 = \left(C_x^e, C_z^e, 0, 0, \psi_0, \|\bar{V}\|, 0, 0\right)^T \qquad (16)$$

with $\psi_0 = \cos^{-1}\left(\bar{v}_z \|\bar{V}\|^{-1}\right)$. This initial state also defines the local object coordinate system to which all points in the point cloud are referred to.

To compensate for lost point tracks, at each cycle, new 6D vectors up to a user-defined maximum number are initialized at image positions that are good to track [12] and currently not covered by another 6D vector. It is possible to restrict the initialization method to objects exceeding a certain velocity threshold, direction, or dimensional constraint. Alternatively, the initialization could be replaced or extended by a radar or lidar sensor.

[1]The term *6D vector* will be used synonymously for a Kalman filtered 3D point track.

E. Point Update

In Section II-A we have assumed the position of a point in the rigid object model to be correctly known, neglecting the uncertainty of the stereo system. This uncertainty increases quadratically with distance. For an oncoming vehicle this means the probability of an erroneous distance measurement is larger at initialization, i.e. when the object is detected, and decreases as the vehicle comes closer. Accordingly, the structure of the initial point cloud does not have to describe the object correctly and, thus, has to be refined over time.

The optimal solution for this problem would be the integration of all point positions into the filter state leading to one large state vector. The Kalman filter then estimates both the motion parameters and the object structure. However, with an expected number of 50-200 points for one object, the larger filter state significantly increases the overall computation time. In a first approximation, the object's structure can be also refined outside the filter by point-wise averaging. After each cycle, the position of a point P_i^o in object coordinates is updated using a weighted average:

$$P_i^o(a_i + 1) = \frac{a_i P_i^o(a_i) + \tilde{P}_i^o}{a_i + 1} \tag{17}$$

where a_i is the age of the point i in the point cloud and $\tilde{P}_i^o$ the currently measured position. A new point is inserted into the point cloud with $a_i = 0$ and $P_i^o(1) = \tilde{P}_i^o$.

Figure 2(b) gives an example of how a point cloud of an oncoming vehicle is adapted over time. The initial point cloud (at $46m$ distance) spreads over approximately $8m$ in longitudinal direction and hardly describes the shape of a car. After 10 frames, this point cloud has converged. Hood, windshield, and roof are visible in the side view.

Points with lost feature tracks or missing disparity measurements are removed from the object's point list. On the other hand, the same mechanism used for generating new object hypothesis as introduced in Section II-D can be adapted to find new points that are likely to belong to an object already being tracked.

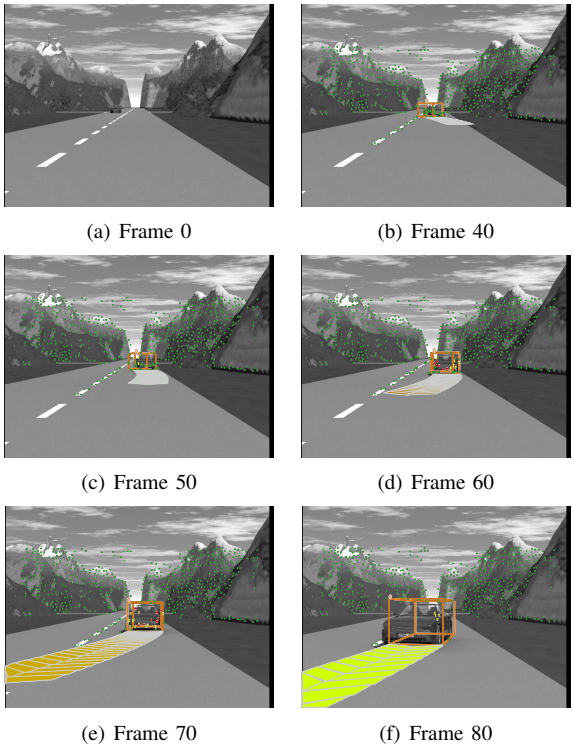
In addition, points not matching the estimated motion model are detected by the filter. An extended 3σ test is applied to remove such outliers from the point list based on the residual between measurement and prediction.

III. EXPERIMENTAL RESULTS

The proposed system has been tested both on simulated data with known ground truth motion parameters of the vehicles in the scene and on real world data.

A. Simulated Results

In Figure 3 selected frames of a rendered scene are shown. The scene contains a vehicle driving towards the stationary ego vehicle starting at $60m$ distance on the opposite lane with a constant velocity of $15m/s$. After 30 frames the observed vehicle changes the lane to simulate a potentially critical situation and turns back to the opposite lane just in front of the ego vehicle. The constant time between two frames is $0.04s$. The 3D bounding box in the images indicates position, dimension, and orientation. The predicted driving path for the next second is visualized as carpet on the ground plane. Its curvature is related to the yaw rate, while the length of the path is based on the current velocity and acceleration.



(a) Frame 0 (b) Frame 40

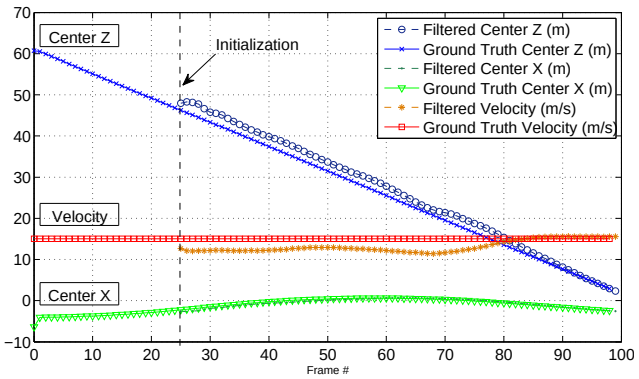
(c) Frame 50 (d) Frame 60

(e) Frame 70 (f) Frame 80

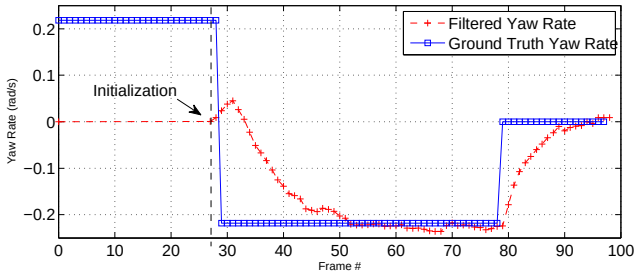
Fig. 3. Selected frames of virtual sequence at different time steps. The object's bounding box and the predicted motion path within the next second estimated by the Kalman filter is visualized. The marked points correspond to 6D vector positions used for initialization.

The resulting state estimates for position (X and Z) and velocity compared to ground truth data are shown in Figure 4(a). The vehicle is detected at approximately $50m$ distance at frame 25 and initialized with a velocity of $12m/s$. The plot shows an almost ideal reconstruction of the lateral position, while the longitudinal distance is a bit larger in the filtered results due to the under estimated velocity up to frame 82. The *root mean square error* (RMSE) of the lateral position is $0.2728m$ over the total sequence and reduces to $0.1287m$ if only the results after frame 80 are considered. Accordingly, the RMSE of the longitudinal distance is $2.0044m$ over the total sequence and $0.8565m$ for the results from frame 80 respectively. The under estimated initial velocity leads to a total RMSE in velocity of $2.2538m/s$ and $0.4934m/s$ from frame 80.

Figure 4(b) visualizes how the filter reacts to the sudden (unnatural) changes of the yaw rate. As can be seen, these changes can not be adapted directly by the filter, leading to larger deviations from the ground truth and a total RMSE of $0.0980rad/s$. However, this smooth adaptation corresponds to the expected, characteristic filter behavior. After convergence, the ground truth value is well approximated. The results also show that the filter converges faster as the



(a) Position + Velocity



(b) Yaw Rate

Fig. 4. Filtering results of the virtual sequence compared to ground truth data.

vehicle comes closer due to the lower uncertainty of the measurements.

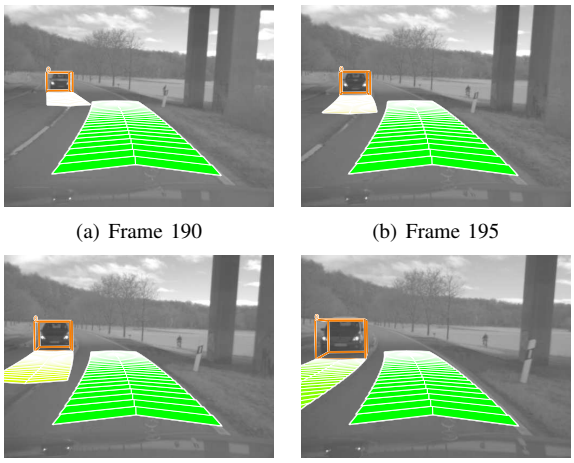
B. Real World Results

The proposed method has been integrated into our Mercedes-Benz demonstrator car *UTA*.

Figure 5 shows selected frames of a sequence with an oncoming vehicle on a curved country road. It illustrates how the Kalman filter sequentially corrects the inaccurate initialization within the first $500ms$ of tracking. The overlaid carpets represent the predicted driving corridor of the ego vehicle based on inertial sensors and the estimated driving path of the oncoming vehicle respectively. At frame 188, the oncoming vehicle is detected at about $38m$ distance using the purely image-based method as introduced in Section II-D.

Tracking the object over a few frames indicates the object is not driving straight ahead (critical), but on a curved path (uncritical). With the assumed motion model, the movement of all observed points on the object can be best explained as a vehicle with a negative yaw rate. The estimated yaw rate of the oncoming vehicle is plotted in Figure 6. It can be seen that the absolute value of the estimated yaw rate of the oncoming vehicle is slightly larger compared to the ego vehicle, since the oncoming vehicle is driving on the inner lane of the curve on a smaller radius.

It is important to state the system does not include any lane recognition. The estimated yaw rate depends purely on the point motion and the underlying coordinated turn motion model.



(a) Frame 190 (b) Frame 195

(c) Frame 200 (d) Frame 205

Fig. 5. Selected frames of a real world scene with one oncoming vehicle captured in a left curve. The driving corridor of the ego vehicle based on inertial sensors is visualized in all images. It can be seen how the filter successively corrects the erroneous initial assumption of the driving path.

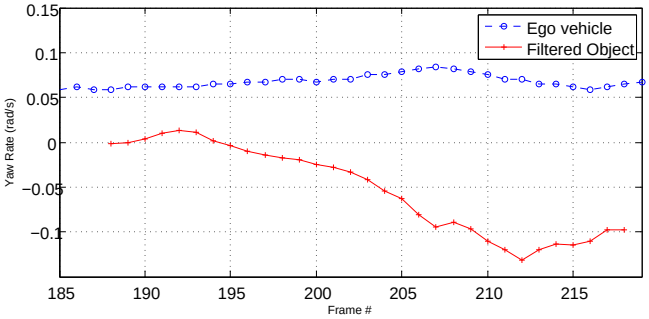


Fig. 6. Filtering results of the real world sequence. The filtered yaw rate indicates the oncoming vehicle is steering to the right, while the yaw rate of the ego vehicle is consistent with a slight left turn.

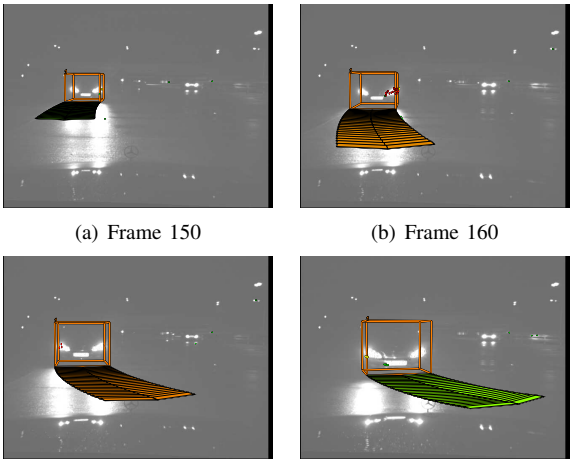
A second sequence (see Figure 7) demonstrates the robustness of the proposed method. An oncoming vehicle is detected and tracked in a rainy night. The vehicle moves toward the ego vehicle and suddenly turns to the left to drive around the obstacle. Only the lights and the number plate illuminated by the ego vehicle are visible. This information, however, is sufficient to predict the driving path of the vehicle accurately.

The tracking approach is not restricted to oncoming traffic. The system is designed to track objects in all moving directions as long as the object is visible in the image. An example of a tracked leading vehicle in a right curve is given in Figure 8.

IV. DISCUSSION

We proposed a method for image based real-time tracking of vehicles from a moving platform.

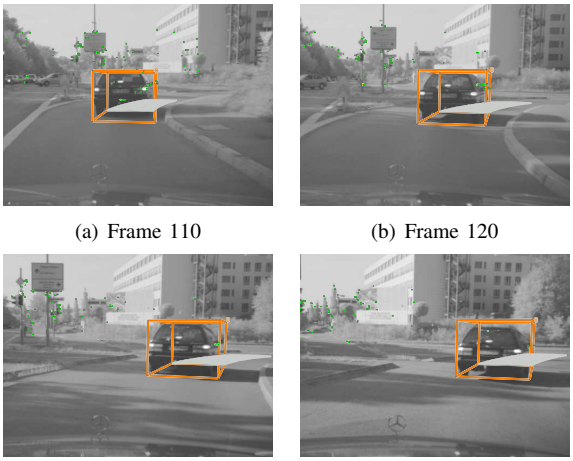
The simulated data has shown the system is able to estimate the motion parameters as well as position and orientation of an object very accurately. However, as with all filters, the time until the state parameters describe the real values well, depends on the quality of the initialization.



(a) Frame 150 (b) Frame 160

(c) Frame 170 (d) Frame 180

Fig. 7. Tracking results of a rainy night scene. Although only points on the lights can be tracked, the driving path is well estimated.



(a) Frame 110 (b) Frame 120

(c) Frame 130 (d) Frame 140

Fig. 8. Tracking results of a leading vehicle in a right curve.

In the real world left curve sequence, due to the initial error in orientation and with the fact that the filter is initialized with $\dot{\psi} = 0$, it takes 10-15 cycles (or 0.4-$0.6s$) until the real driving path of the oncoming vehicle is estimated well. In this sequence, however, both vehicles have a velocity of about $50km/h$, i.e. a relative velocity of $100km/h$, thus, there is not much more time than a second between detection and passing.

State of the art radar systems or laser scanners (Lidar sensors) allow for good estimations of distance and relative velocities of objects in the longitudinal direction of the ego vehicle. The integration of such sensor information could accelerate convergence, e.g. one could use the radar velocity for initialization.

A multi filter approach could also improve the system. A number of Kalman filters parametrized with variances for different scenarios (e.g. linear movement, curve movement, or turn maneuver) could be run in parallel with an automated selection of the currently best matching model.

The proposed method is very robust since the dynamics are constrained by a 3D vehicle motion model. This makes it easier to reject point tracks that do not match the motion model, compared to approaches estimating an unconstrained rotation and translation. The power of this approach stems from the large number of contributing points, which reduces the effect of inevitable modelling inaccuracies, and increases the accuracy and reliability of the tracking process. The night scene has shown that it is possible to estimate the yaw rate of an oncoming vehicle, even if only the lights are visible and the image is corrupted by a lot of noise.

The system is not restricted to oncoming traffic and can be used for tracking a leading vehicle in the same way without any adaptations.

On current PC hardware (Intel Core 2 Quad), the proposed system requires a total processing time of less than $40ms$ for VGA images, i.e. runs with a stable frame rate of $25Hz$, without exploiting all capacities of parallel processing or computation on the graphics card. The tracking of up to 5 objects at one time requires $< 15\%$ of the total time only. The remaining processing time includes stereo and point track computation, ego-motion estimation, image-based object detection, and visualization.

REFERENCES

[1] F. Dellaert and C. Thorpe, "Robust car tracking using kalman filtering and bayesian templates," in *Conference on Intelligent Transportation Systems*, 1997.

[2] T. Dang, C. Hoffmann, and C. Stiller, "Fusing optical flow and stereo disparity for object tracking," *IEEE 5th International Conference on Intelligent Transportation Systems*, pp. 112–117, 2002.

[3] N. Kaempchen, "Feature-level fusion of laser scanner and video data for advanced driver assistance systems," Ph.D. dissertation, University Ulm, 2007.

[4] D. Koller, K. Daniilidis, and H. Nagel, "Model-based object tracking in monocular image sequences of road traffic scenes," *International Journal of Computer Vision*, vol. 10, no. 3, pp. 257–281, June 1993.

[5] H. Veeraraghavan and N. Papanikolopoulos, "Combining multiple tracking modalities for vehicle tracking at traffic intersections," in *IEEE Conf. on Robotics and Automation*, 2004.

[6] C. Rabe, U. Franke, and S. Gehrig, "Fast detection of moving objects in complex scenarios," *Intelligent Vehicles Symposium, 2007 IEEE*, pp. 398–403, 13-15 June 2007.

[7] Y. Bar-Shalom, X. Rong Li, and T. Kirubarajan, *Estimation with Applications To Tracking and Navigation*. John Wiley & Sons, Inc, 2001.

[8] D. Simon, *Optimal State Estimation*. John Wiley & Sons, Inc, 2006.

[9] H. Badino, "A robust approach for ego-motion estimation using a mobile stereo platform," in 1^{st} *Internation Workshop on Complex Motion (IWCM04)*, Guenzburg, Germany, October 12 - 14 2004. [Online]. Available: citeseer.ist.psu.edu/757895.html

[10] C. Tomasi and T. Kanade, "Detection and tracking of point features," Carnegie Mellon University, Tech. Rep. CMU-CS-91-132, April 1991. [Online]. Available: citeseer.ist.psu.edu/tomasi91detection.html

[11] U. Franke, C. Rabe, H. Badino, and S. Gehrig, "6D-vision: Fusion of stereo and motion for robust environment perception," in *27th DAGM Symposium*, 2005, p. 216.

[12] J. Shi and C. Tomasi, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994. [Online]. Available: citeseer.ist.psu.edu/shi94good.html