Radar/Lidar Sensor Fusion for Car-Following on Highways

Daniel Göhring, Miao Wang, Michael Schnürmacher, Tinosch Ganjineh
Institut für Informatik
Freie Universität Berlin
Germany

Abstract— **We present a real-time algorithm which enables an autonomous car to comfortably follow other cars at various speeds while keeping a safe distance. We focus on highway scenarios.**

A velocity and distance regulation approach is presented that depends on the position as well as the velocity of the followed car. Radar sensors provide reliable information on straight lanes, but fail in curves due to their restricted field of view. On the other hand, Lidar sensors are able to cover the regions of interest in almost all situations, but do not provide precise speed information. We combine the advantages of both sensors with a sensor fusion approach in order to provide permanent and precise spatial and dynamical data.

Our results in highway experiments with real traffic will be described in detail.

I. Introduction

The interest in autonomous cars has grown in recent years, as they provided new insights for general robotic systems in areas like safety, machine learning and environmental perception [2].

One key aspect for driving autonomous cars is the detection of obstacles and other cars. In our paper we focus on a highway scenario. Here, we describe an algorithm which enables our car to follow other cars at various speeds, while keeping a safe distance and providing braking in front of obstacles. We describe how data from Lidar and radar can be used and combined for precise obstacle and car detection at different velocities.

Sensor fusion of Lidar and radar combining the advantages of both sensor types has been used earlier, e.g., by Yamauchi [14] to make their system robust against adverse weather conditions. Blanc et al. [1] present an application that focuses on the reliable association of detected obstacles to lanes and the reduction of false alarms (phantom obstacles). A variety of applications of sensor fusion methods in cars are given by Kaempchen et al. in [7].

The remainder of this paper is structured as follows: Section II presents our data fusion approach. Section III and Section IV illustrate how the obstacles are incorporated in the path-planning process and velocity calculation for the autonomous car. Experimental results during a test run on public highways are given in Section V. Finally, Section VI draws conclusions and discusses future work.

II. Sensor Fusion

In car following applications it is mandatory to have continuous, precise, and accurate velocity and position information about vehicles driving ahead. Here, this is accomplished by combining data from Lidar and radar sensors.

Fig. 1. Autonomous Car "MadeInGermany"

Present-day laser scanners have a large range (up to 200 m) and a wide field of view, and can thus track objects even at big distances (mandatory at high speeds) as well as in curves. Their main drawback is that they completely lack dynamic information about the detected objects. Radar sensors, on the other hand, have a narrow field of view and reduced angular resolution, but use the Doppler effect to directly provide velocity information. The fusion of the data from both sensors can thus benefit from their complementary nature [4][11].

Section II-A describes the configuration of the sensors on "MadeInGermany" (*MIG*, see Figure 1) and the parameters of the sensors that are most important for the application. The fusion architecture is presented in Section II-B. Special focus lies on the treatment of the Out-Of-Sequence Measurement problem, that arises when combining data of unsynchronized sensors. Section II-C describes how we use one common movement model for state estimation. Experimental results are shown in Section V as part of our overall test scenario.

A. Sensor Configuration

The sensor configuration of our car is shown in Figure 2. Six ibeo Lux scanners, each having an horizontal field of view of 110°, are mounted on the car such that they provide almost 360° of view around the car (for the sensor fusion presented here only the front scanners are relevant). A TRW radar sensor is attached at the front of the car. It has a horizontal field of view of 12°. Both sensors have a similar range of up to 200 m (depending on the material and inclination angle of the

978-1-4577-0330-0/11/$26.00 ©2011 IEEE

reflecting surface) so that the fusion of both sensors can take place in the complete field of view of the radar sensor.

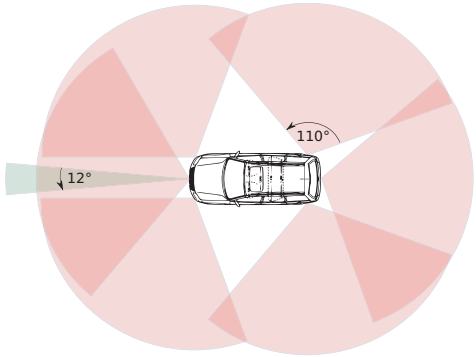


Fig. 2. Sensor Configuration: Six single Lux (red) and TRW radar (green).

B. Fusion Architecture

We use a centralized fusion architecture that is divided into sensor layer and fusion layer (see [3][13]). This is illustrated in Figure 3.

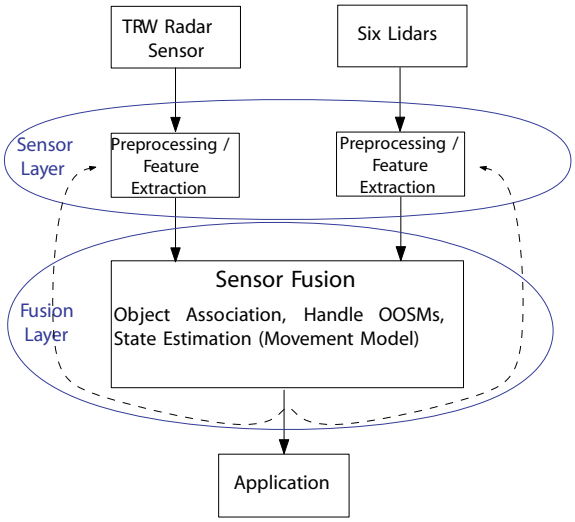


Fig. 3. The Fusion Architecture is separated into Sensor Layer and Fusion Layer. After pre-processing, the unfiltered obstacles are passed on to the fusion module, where they are incorporated into a common object model. Strong hypothesis can be fed back to the Sensor Layer.

The sensor layer is implemented separately for each sensor. First, it brings the scan data of each sensor into a common time and coordinate system. In order to make the system flexible for the integration of further sensors, the fusion layer should abstract as far as possible from the concrete data of the sensors. For that purpose, feature extraction and obstacle hypothesis generation are executed here as well.

The fusion layer maintains the obstacle tracks and handles the association and estimation. There is no hardware synchronization between the sensors, so that the fusion layer must cope with Out-Of-Sequence Measurements (OOSM) [6][12].

We use buffering and measurement reprocessing to handle this problem. It is easy-to-implement and finds an optimal solution to the OSMM problem in the sense that no data is lost and the latest information is immediately processed and passed on (not held back). As [12] correctly points out, the approach is memory (buffering) and time (reprocessing) consuming. We found that, in our case, this aspect could be neglected because the fusion region and thus the number of potential fusion objects is small. A short summary of algorithms for the OSMM problem is given in [12]. [6] discusses different synchronization strategies to minimize the worst case latency.

Our approach is illustrated in Figure 4. Each currently filtered obstacle O^f has a corresponding track T that maintains a history of the filtering process (state estimation is explained in Section II-C). Each entry in the history is a pair of obstacles $P_{t_i} = (O^f_{t_i}, O^m_{t_i})$, where $-d < i \leq 0$ and d is the depth of the history. $O^m_{t_i}$ is an obstacle that was observed by some sensor at time t_i (this can be any sensor used in the fusion) and has been used to update $O^f_{t_{i-1}}(t_i)$ (filtered obstacle $O^f_{t_{i-1}}$ predicted to time t_i), which then became $O^f_{t_i}$.

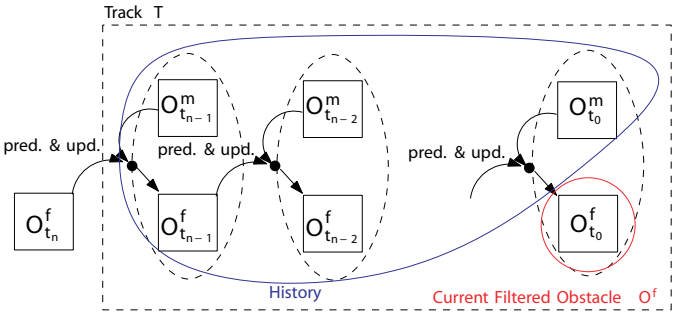


Fig. 4. Track: Maintains a history for each obstacle hypothesis.

When new obstacle data is added to the fusion module, the obstacles are associated to the maintained tracks. In order to compare a new obstacle O^m with a track T, we search for the correct time slice in the track history (binary search) and thus its potential corresponding filtered obstacle O^f. Once found, we predict O^f to the time of O^m and check if both can be associated [10]. A consensus between different potential associated tracks is found using a nearest neighbor approach. The total history is then reprocessed from O^f onwards by a chain of predict and update calls.

After each such iteration, the history can be cleaned from unnecessary data. It holds that data from the same sensor arrive in-sequence. When updating track T with O^m, every obstacle $O^m_{t_i}$ in T that stems from that same sensor becomes obsolete (except O^m). Obstacles that are older than some predefined threshold are marked as *obsolete* as well (the threshold depends on the respective sensor and their expected delays). We can then delete obstacle pairs from left to right in Figure 4 until we find a pair for which the measurement obstacle is not obsolete.

C. State Estimation

We use a constant acceleration model to describe the state of each object. It consists of the object's position p_x and p_y, its velocity v_x and v_y, and its acceleration a_x and a_y.

In [3] a fusion algorithm is presented that uses two models that are chosen depending on the information quality provided by the respective sensor. In addition to the point model it implements a bicycle model that is used when information about the object's orientation is available (e.g. by the laser scanner when the objects are close enough). This was not needed for the car following application described here, but is considered essential for future applications, especially when cars are turning tightly in dense traffic and thus orientation information is indispensable.

Our object model is implemented by a Kalman Filter [8][5]. The quality of the object features (position and velocity) computed from the scan data is very different for both sensors. For example, the velocity information from the radar is more accurate than that of the Lidar scanner, which must be computed from measurements at different times. This was accounted for by setting the values in the measurement covariance matrices appropriately when updating the state vector.

The Lux scanners provide a cloud of scan points, which must be processed to extract the essential features. Ideally, all points that belong to the same object in the real world are brought into a single cluster. For this purpose, we apply a simple distance-based cluster heuristic. As [3] suggests, this is improved by feeding back strong hypotheses of the fusion module back to the clustering phase.

We use the cluster's center of gravity as position feature for the state update. The stability of this feature depends on a large quantity of factors, like e.g. bias and distance of the followed car relative to the sensors. As the results in Section V show, we obtained a standard deviation of about 20 cm for this feature.

The velocity information cannot be computed from a single scan, but has to be estimated between scans at different times. We decided to compare consecutive measurements and thus obtained a standard deviation of about 0.8 km/h. When the time difference between both measurements becomes too small we raise the values in the covariance matrix, because in this case their combination can lead to false estimates.

The radar does not allow a mode that provides a scan's raw data. It sends a list of already filtered objects. Each object consists of its position defined by its distance to the sensor and its angle relative to the sensor's orientation and of its velocity relative to the sensor.

The object position information from the radar sensor is directly used as position feature, and similar to the Lidar feature we measured a precision of about 20 cm.

In contrast to the Lux, the radar directly provides information about the object's velocity by using the Doppler effect. This velocity is more precise than the velocity computed from the Lidar data (see experiments in Section V, $\sigma \approx 0.3$ km/h). Unfortunately, only the radial velocity component relative to the sensor is measured, which can lead to the following type of problem. In Figure 5, the radar detects an object that drives on the same lane ahead of us and that is shifted laterally with respect to the autonomous car. However, their velocity directions are equal. The radar only measures $v_{\parallel}$, so the car seems to leave the lane, although it actually does not (the same happens when a car approaches on the lane left to us). As a first approach we only use the component of $v_{\parallel}$ that is parallel to the forward direction of the own car, $v'_{\parallel}$. This is a tolerable approximation because the radar aperture angle is very small. Similar to the velocity computation of the Lux objects one could compare consecutive radar objects in order to get information about $v_{\perp}$. This is planned for the future.

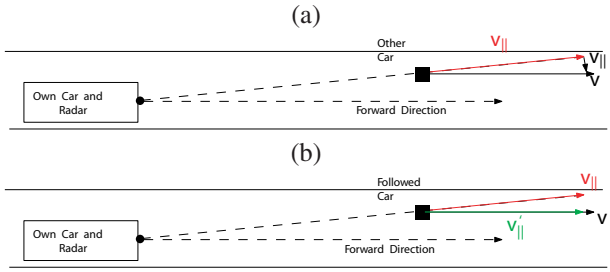


Fig. 5. Radar-Velocity-Measurement: (a) v is the actual velocity of the followed car, $v_{\parallel}$ the one measured by the radar sensor (b) and $v'_{\parallel}$ is the one we assume.

III. Obstacle Checking on Planned Trajectories

A behavioral module is in charge of generating a set of trajectories along the road network given a specific mission destination. From the perspective of the car's position, all outgoing roads are planned by a trajectory parallel to the smoothed center of the road and a predefined number of trajectories undergoing lateral shifts. Each trajectory is subsequently evaluated by various scores s_i, e.g. the path distance, eventual lane changes, and collisions with nearby obstacles. The planner selects the trajectory minimizing a weighted total evaluation score e with

$$e = \min_{\forall traj} \sum_{i=0}^{n} \alpha_i \, s_i. \tag{1}$$

and passes it on to a low-level controller where brake/acceleration and steering commands are computed. Figure 6 presents a view of the path-planning result in a simulator, which shows controller command data and interfering obstacles. In some cases, all generated trajectories may be blocked by static or dynamic obstacles for which the controller will need to generate speed commands to either come to a full stop or to maintain safe platooning distance (described in the next section).

To distinguish between static and dynamic obstacles an obstacle is assigned to a movement state m as being either *static* or *dynamic*, together with a holding time t_m of that movement state. An obstacle is allowed to change its movement state after a hysteresis with a dynamic to static threshold $thresh_{ds}$ of $7000ms$ and static to dynamic threshold $thresh_{sd}$ of $40ms$.

In particular, to evaluate one specific trajectory for given obstacles by sensor fusion data, we check each obstacle contour point for collision with the trajectory augmented with car dimensions. A majority of obstacles and their corresponding

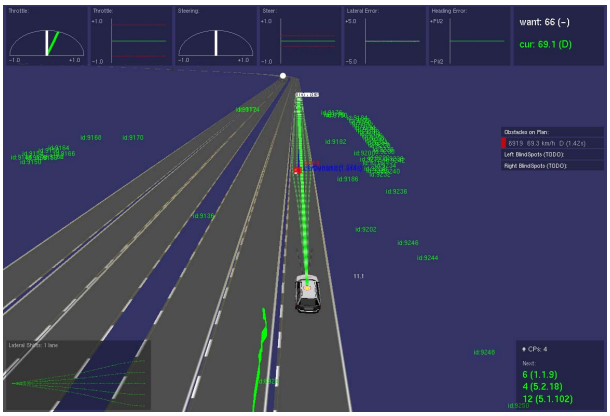


Fig. 6. Path-planning and low-level controller data (top bar) in a simulator. The interfering obstacle is marked as red.

contour points are usually located off-road including buildings, trees, and pedestrians. To distinguish them from more relevant obstacles on roads like moving cars, traffic islands or construction zones, each obstacle is categorized as being either *OFF-ROAD*, *NEAR-ROAD* or *ON-ROAD*. This is done by the sensor layer as each obstacle is checked for its distance to the nearest road segment. Thus, data points received by the behavioral executive are already filtered as relevant and irrelevant and only relevant points near or on-road need to be checked for collision checks with trajectories along the roads.

All relevant obstacles are stored in a spatial kd-tree [9] to test containment in the trajectories domain. If interfering obstacles are found along the trajectory they are evaluated by a sigmoidal function over distance to the obstacle:

$$e_{collision} = \lambda * \frac{1}{1 + \exp(\,\mu_1\, d_{obst} - \mu_2\, d_s)} \tag{2}$$

where d_{obst} denotes the longitudinal distance between car and obstacle and d_s a safety distance to be maintained. The symbols λ, μ_1 and μ_2 denote tuning parameters.

Equation 2 is computed when the preceding car is either marked as static or its speed is slower than $1m/s$, or slower than half of the maximum allowed speed on that road segment. Otherwise the preceding car is driving fast enough within permitted speed limits so that it can safely be followed by regulating a safety distance to it.

Additionally, each time Equation 2 is calculated a possible passing condition is met and co-numerated. The evaluation is additionally increased once the passing condition is met more than a certain times in a row. In case of a static obstacle it is also integrated over the movement state holding time t_m. Thus, the evaluation of the given trajectory is worsened by time and possible trajectories including lane change to neighboring lanes are more preferred if they exist.

In a subsequent step, after the best trajectory is selected, the category of all obstacles with at least one interfering contour point colliding with that trajectory are upgraded from *ON-ROAD* to *ON-TRAJECTORY*. Moreover, the nearest obstacle with respect to the car is especially marked as *FIRST-ON-TRAJECTORY*. This way, succeeding modules interested in only interfering obstacles (such as the low-level controller) can limit their search to a smaller set of obstacles. Furthermore the obstacle labeled as *FIRST-ON-TRAJECTORY* is propagated back to the obstacle tracker to verify the performance of the car following behavior.

IV. Velocity Planning

The velocity of a car is constrained by a number of different aspects. The car has to brake, whenever an obstacle has been recognized to be on the planned trajectory, or moving towards it. Second, the velocity is constrained by the curvature of the street in front of the car. Furthermore the car has to reduce speed in front of traffic lights, stop signs and intersections. Also, if a car is followed, the velocity of the followed car must be taken into account. Therefore, parameters define the comfortable brake and centrifugal acceleration.

A. Braking Trajectory for Obstacles

If an obstacle has been detected on the planned trajectory, we want the car to decrease speed with a constant negative acceleration. A very simple approach is using the distance to the obstacle, assuming the obstacle is static. A drawback of not using the speed of an object is that the braking distance is too long or too short, because the obstacle might have moved while approaching it. Further, usually one does not want to brake directly in front of the obstacle but to keep a safety distance. The following paragraphs will discuss the different cases and resulting equations.

1) Static Objects: In the 1-D case, given a current car position s_0, the position of a static object s_E, a safety distance to the object s_{safe} and a given wanted negative acceleration $a_{brake} = -2m/s^2$ we can calculate the desired current velocity v_0 using Equation 3.

$$v_0 = \sqrt{2a_{brake}(s_0 - s_E + s_{safe})} \tag{3}$$

2) Dynamic Objects: When the velocity of the object in front is known and the object is moving away from us, the car does not have to stop but to reduce its speed to the obstacle's velocity, keeping a constant safety distance s_{safe}. Since we assume a highway scenario we only consider the longitudinal component v_E of the velocity of the vehicle in front of us. For positive v_E we have:

$$v_0 = \sqrt{2a_{brake}(s_0 - s_E + s_{safe}) + v_E^2} \tag{4}$$

For negative v_E (the followed car moves towards us) our car has to make a full stop. Humans follow other cars not with a fixed safety distance, moreover the distance depends on the speed of the car they follow. As a good approximation we assume the optimal following distance to be a combination of a fixed part s_{safe} and a dynamic part $2v_E$. As a modified formula we get:

$$v_0 = \sqrt{2a_{brake}(s_0 - s_E + s_{safe} + 2v_E) + v_E^2} \tag{5}$$

The velocity function over position for a dynamic object and given braking acceleration, see equation 4, is depicted in Fig. 7. Given the distance to an obstacle and the obstacle's velocity one can calculate the desired velocity of the car. A given safety distance would shift this function to the left.

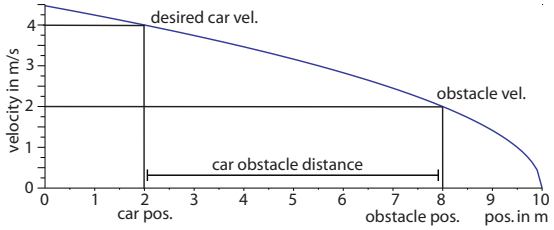


Fig. 7. Velocity function over position. The resulting car velocity is calculated from the current car position, the followed car's position and the followed car's velocity.

3) Stopping at Stop Signs: Stopping at stop signs or intersections is similarly handled as braking in front of static obstacles. However, they usually do not have to be detected by sensor readings, moreover their position is known a priori from digital road maps. The velocity function for braking towards a stop signs or intersections can be calculated with equation 3.

V. Experimental Results

To evaluate the results of the sensor fusion and velocity planning we set up an autonomous test scenario on a highway. Section V-A describes the test setup. The improvement of the precision of the single-sensor velocity and distance estimations achieved by sensor fusion was the main issue. Section V-B shows our results and consequences for the velocity planning of the autonomous car.

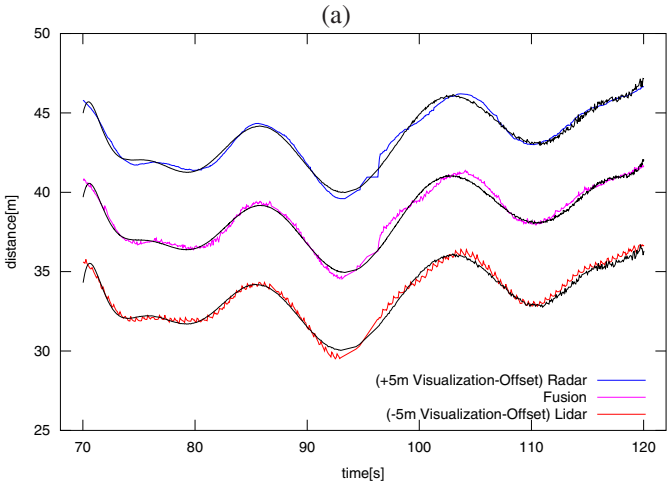
A. Test Setup

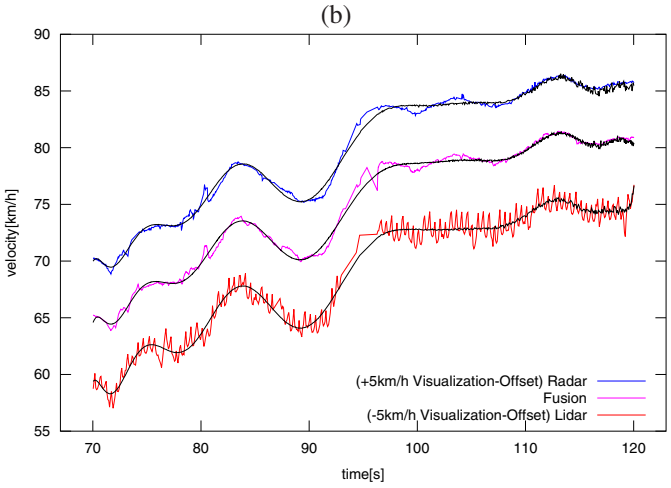
The Automobil-Verkehrs- und Übungs-Straße (AVUS), a highway passage of Autobahn A115 and former motor racing circuit in the south-western districts of Berlin, Germany, was chosen as testing course. Data was recorded from the entrance ramp Kleinmachnow to the exit ramp Berlin-Hüttenweg during an autonomous drive of "MadeInGermany" in real traffic. A manual controlled car stayed in front of "MadeInGermany" for it to follow at various speeds.

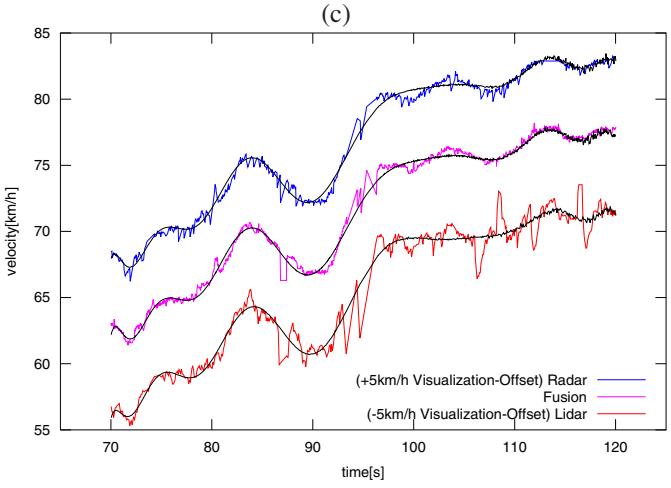
B. Evaluation

In Section IV we showed that the velocity planning depends to a great extent on the other vehicle's velocity and position relative to the autonomous car. Thus it is desirable to provide permanent and precise information about these qualities. Their loss can result in a dangerous, abrupt and false reduction or elevation of speed and their imprecision potentially leads to an uncomfortable journey due to small but permanent speed adaptations to the new situation. Both are reduced by our sensor fusion algorithm.

Figure 8(a) and 8(b) show the velocity and distance of the followed car over time. The objective is to compare the







(d)

	σ_{radar}	σ_{Lidar}	σ_{fusion}
Velocity [km/h]	0.301	0.795	0.340
Distance [m]	0.080	0.094	0.072
Desired Speed [km/h]	0.405	0.829	0.369

Fig. 8. Comparison between the precision of the (a) velocity, (b) distance and (c) desired speed computed by radar, Lidar, and their fusion. Table (d) shows the standard deviations of the parameters with respect to their polynomial fit curve.

precision of the fusion features with respect to that of the single sensor features. This was only possible where both sensor data was available (actually, there exist some small time intervals where no radar data was available due).

We have no ground truth of the features at hand. Instead, we approximate the actual values by a Least Squares polynomial fit. The order of the polynomial is chosen such that it best fits the expected function by visual judgment and such that the polynomials for the fusion, Lidar and radar results are similar. The polynomials are drawn as black lines in Figures 8(a), 8(b), and 8(c). The standard deviation of the actual values (with the artificially created ground truth taken as mean value) is taken as indicator for their precision (shown in Table 8(d)). Note that the artificial ground truth adds an error to the analysis, but we considered it adequate enough to be significant.

For enhanced visualization, radar values are given a constant positive offset and Lidar values a constant negative offset. In general, red indicates Lidar data, blue indicates radar data and magenta indicates data from fusion of both.

1) Velocity: As mentioned before, the velocity estimates computed from the Lidar data (≈ 0.8 km/h) are worse than the estimates of the radar (≈ 0.38 km/h). The fusion clearly exploits the good precision of the radar velocity (≈ 0.34 km/h). Remember, that the radar can only improve the component of the velocity that is parallel to the direction of the radar. If the Lidar velocity has a perpendicular nonzero velocity component it could negatively effect the precision.

We only consider the magnitude of the velocity vector, because the velocity planning does not directly depend on the velocity direction (see Section IV).

2) Distance: The precision of the distance estimation computed from sensor fusion is similar to Lidar and radar sensor data solely. One can see that the precision depends on the distance to the sensors. In Region 1 the precision is below 0.1m (≈ 0.08 m), whereas in Region 2 it increases but stays below 0.3m (≈ 0.26 m).

3) Desired Velocity: Here, we analyze the desired velocity of our car to follow the car in front. This is done again for Lidar, radar and fusioned input data separately. The results are shown in Figure 8(b). Tests demonstrated that the fusioned input data resulted in a stable wanted speed, comparable to the velocity given by the radar. This led to an efficient control effort with few braking and accelerating actions. The resulting velocity of the vehicle became more constant as well.

VI. Conclusion and Future Work

We presented a real-time efficient radar/Lidar obstacle fusion approach, combining the advantages of both - accurate and highly available position estimation with Lidar and precise velocity estimation with radar. The described algorithms have been tested on our autonomous car and led to a comfortable following behavior on the German Autobahn in real traffic conditions. In experiments we could show that by fusion of Lidar with radar data we increased the precision compared to the Lidar velocity and also achieved a good position and velocity estimation whenever radar data were unavailable, thus compensated the narrow field of view of the radar sensor.

Future work will focus on further integration of rear orientated blind spot radar to implement a safe and robust lane changing behavior. Further, a multi-object tracking and fusion system for road intersections, maybe with occlusions will be part of our future research.

Acknowledgments

The authors wish to thank the German federal ministry of education and research (BMBF).

References

[1] C. Blanc, L. Trassoudaine, Y. L. Guilloux, and R. Moreira. Track to track fusion method applied to road obstacle detection. In *IEEE International Conference on Information Fusion*, 2004.

[2] M. Buehler, K. Iagnemma, and S. Singh. Special issue: Special issue on the 2007 darpa urban challenge. *Journal of Field Robotics*, 25(8):423 – 566, August 2008.

[3] M. Darms, P. Rybski, and C. Urmson. An adaptive model switching approach for a multisensor tracking system used for autonomous driving in an urban environment. In *AUTOREG 2008, Steuerung und Regelung von Fahrzeugen und Motoren: 4. Fachtagung*, pages 521–530, Duesseldorf, February 2008. VDI-Verlag.

[4] J. Dickmann, F. Diewald, M. Maehlisch, J. Klappstein, S. Zuther, S. Pietzsch, and S. Hahn. Environmental perception for future integrated safety systems, 2009.

[5] J. B. Gao and C. J. Harris. Some remarks on kalman filters and for the multisensor fusion. In *Information Fusion*, volume 3, pages 191–202, 2002.

[6] N. Kaempchen and K. Dietmayer. Data synchronization strategies for multi-sensor fusion. In *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, 2003.

[7] N. Kaempchen, K. C. Fuerstenberg, A. G. Skibicki, and K. C. J. Dietmayer. Sensor fusion for multiple automotive active safety and comfort applications. In J. Valldorf and W. Gessner, editors, *Advanced Microsystems for Automotive Applications*, volume 2, pages 137–163. Springer, 2004.

[8] R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, 82:35–45, 1960.

[9] B. C. Ooi. Spatial kd-tree: A data structure for geographic database. In *BTW*, pages 247–258, 1987.

[10] B. L. Scala and A. Farina. Choosing track association method. In *Information Fusion*, volume 3, pages 119–133, 2002.

[11] M. Skutek. *Ein PreCrash-System auf Basis multisensorieller Umgebungserfassung*. PhD thesis, TU Chemnitz, 2006.

[12] X. H. Wang, W. and M. Wang. Out-of-sequence measurement algorithm based on gaussian particle filter. *Inform. Technol. J.*, 9:942–948, 2010.

[13] H. Winner, S. Hakuli, and G. Wolf. *Handbuch Fahrassistenzsysteme*. Vieweg+Teubner and GWV Fachverlag GmbH, 2009.

[14] B. Yamauchi. All-weather perception for man-portable robots using ultra-wideband radar. In *In IEEE International Conference on Robotics and Automation (ICRA)*, 2010.