PREADJUSTMENT OF A VISION-BASED LANE TRACKER

USING VIRTUAL TEST DRIVE WITHIN A HARDWARE IN THE LOOP SIMULATOR

*** Kilian von Neumann-Cosel (neumannc@in.tum.de)**
**** Mirko Nentwig (mirko.nentwig@audi.de)**
*** Daniel Lehmann (lehmannd@in.tum.de)**
***** Johannes Speth (speth@ini.tum.de)**
*** Prof. A. Knoll (knoll@in.tum.de)**

* TUM - Technische Universität München
Informatik VI: Robotik & Embedded Systems
Boltzmannstr. 3, D-85748 Garching

**Audi AG, Elektronik Entwicklung
Hardware-in-the-Loop Funktionserprobung
D-85045 Ingolstadt

*** TUM - Technische Universität München
Lehrstuhl für Realzeit-Computersysteme
Arcisstr. 21, D-80299 München

Abstract

In this paper, it is shown that synthetic images can be used to preadjust a lane tracking algorithm which is developed by Audi. This was achieved by setting up a heterogeneous hardware-in-the-loop testbed. It includes the highly configurable and extendable simulation “Virtual Test Drive”. The main components are a traffic simulation, visualization and a sensor model which supplies ground truth data about the street lanes. Additionally, the visualization is used to generate synthetic camera sensor data. The testbed also contains a realistic driving dynamics simulation and a real image processing ECU (which is represented as a standard PC in the early development stages). One of the modules on the image processing ECU is a lane tracking algorithm. The algorithm is designed to calculate the transition curves while driving. This information can be used as input for driving assistance functions, e.g. lane departure warning. By running the lane tracker on a synthetic image it is possible to compare the results of the lane tracker with the ground truth data provided by the simulation. In this particular case, the information has been used to adjust the expected system error variance of one state variable.

Résumé

Introduction

As cameras and other sensors become increasingly common in modern production cars, image processing allows new active approaches to improve comfort and safety. Popular examples are pedestrian or vehicle detection systems to avoid collisions but also lane detection to help the driver stay on track.

Those algorithms are conventionally tested with real sensor data and their results are validated with ground truth. However, ground truth is difficult to measure. For example video data for a car tracker would need to be labelled with the exact positions and velocities of both the ego and the target cars and the size and orientation of the other cars. Creating those labels is very time-consuming. For these reasons generating synthetic images to simulate vision-based algorithms is of raising importance [5].

Testbed Architecture

The Virtual Test Drive (VTD) simulation framework allows users to fulfil a wide range of different simulation tasks throughout the entire development process of driver assistance and active safety systems. Depending on the exact use case of the tasks, different types of simulators can be used [1].

In this case, a PC-based VTD-simulator is connected to a hardware-in-the-loop (HIL) simulator and an image processing electronic control unit (ECU). Additional electronic automotive systems, such as ECUs, sensors, and actuators, can be connected to the HIL-simulator. With this extended system it is possible to test complete functions from top to bottom, e.g. a lane departure warning lamp can be activated as soon as the ego car is crossing a lane inside the virtual environment.

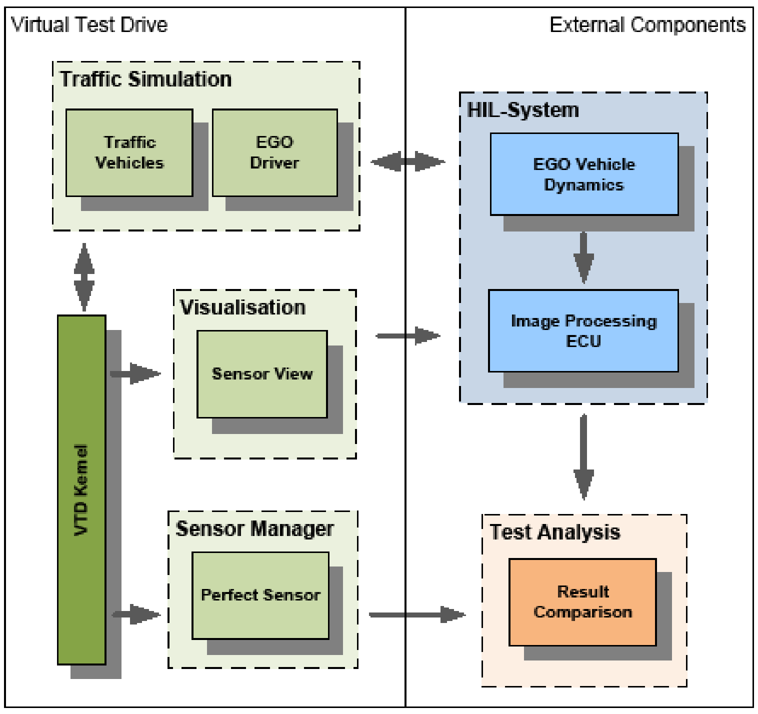


Figure 1: Testbed Architecture

Each simulator type uses the same core components to handle the data flow within the simulation. The extended simulation components, for example vehicle dynamics, driver model, visualization and traffic simulation, are used to simulate road users and the environment. Target components are related to the development of an aimed assistance system, e.g. choosing of appropriate sensors, developing algorithms and evaluating the use of certain actuators. The fully modular framework allows varying extended and target components in each simulator type. It is running in real-time on a regular PC.

In this use case, the visualization is used to generate virtual camera images. Before generating these images however, it is necessary to set up the intrinsic and extrinsic camera parameters, e.g. its installation position and projection matrix. These parameters are also used by the image processing ECU to calculate the corresponding results. The virtual images are transferred to the HIL with a dedicated connection.

Another component in the Virtual Test Drive architecture is the sensor manager. It consists of a plug-in architecture which allows the user to include arbitrary sensor models in the simulation framework. In this configuration, a perfect sensor plug-in is used, which returns position data of traffic vehicles and road lane information. This is used as ground truth data to compare to the results of the tested lane tracking algorithm.

The driver component of the Virtual Test Drive environment is “driving” the external vehicle dynamics. The external driving dynamics needs an input vector by the driver model, including pedal states, steering wheel angle and the four wheels’ positions vector in six degrees of freedom.

The HIL-system calculates the vehicle dynamics of the ego car within its own simulator. In order to accomplish these tasks, an intensive communication between VTD and the HIL-simulator is necessary. This is achieved by using a broadcast oriented network protocol which allows the computer cluster to work on a virtual shared memory architecture. Each node of the network has the ability to share data with all other network nodes at same time. The communication layer is implemented using the Automotive Data and Time triggered Framework [2].

The HIL-system is simulating all missing electronic components, such as sensors, actuators and ECU. It provides a simulation of necessary bus data and stimulates discrete inputs. Furthermore, the simulated sensor data has to match the position of detected obstacles in the video image.

The lane tracking algorithm is part of an image processing ECU that is connected to the simulation framework. Through an interface of the hardware controller, virtual camera images from Virtual Test Drive can be received and processed. The recognized lanes can be displayed inside additional tools or be written to files [2].

Also, the ground truth data can be stored in text files. Then, these files can be loaded into an independent analysis application where several plots and diagrams can be generated to compare the results.

Lane Tracking Algorithm

The lane tracker is a computer vision based system which predicts the curvature of the road lanes in front of the car.

The implementation of the lane tracker is based on the “4D-approach” described by [4] and has been developed by the Audi department for Advanced Driver Assistance Systems. The steps of the general tracking procedure are only described briefly here; for more detailed information on the underlying models the reader is deferred to [3].

The centre of a lane is modelled as a transition curve whose curvature at distance l along its pathway is described by the equation

$$c(l) = c_0 + c_1 \cdot l$$

where $c_0 = 1/r_0$ is the curvature of the circle with radius r_0 in the starting point of the transition curve. Parameter c_1 describes the change in curvature along the pathway. To calculate the expected positions of lane markers, bordering the own lane, the width B of the lane must be taken into account. In order to compensate for the relative position of the ego vehicle towards the position of the transition curve's origin some other parameters are estimated:

- pitch angle θ of the vehicle's x-axis towards the ground plane,
- yaw angle ψ of the vehicle's x-axis towards the tangent to the transition curve at its origin,
- lateral offset Y_{off} of the vehicle towards the origin of the transition curve.

In each iteration of the tracking loop the estimated state vector $\hat{\vec{x}}_{k-1}$ is predicted to the current time step k using a transition matrix. Using the predicted state vector $\hat{\vec{x}}_k^*$ the lane centre transition curve model is calculated and support points along the pathway are created. For each support point the expected points on the left and the right lane border are produced by moving a distance of $\pm B/2$ along the perpendicular to the tangent to the transition curve model at the particular support point. In a next step these points are projected to the imaging chip using the known camera pose within the vehicle and a pin hole camera model. This result in pixel positions where edge points of lane markers are expected and can be looked for with an adaptive convolution mask, as described in Figure 2.

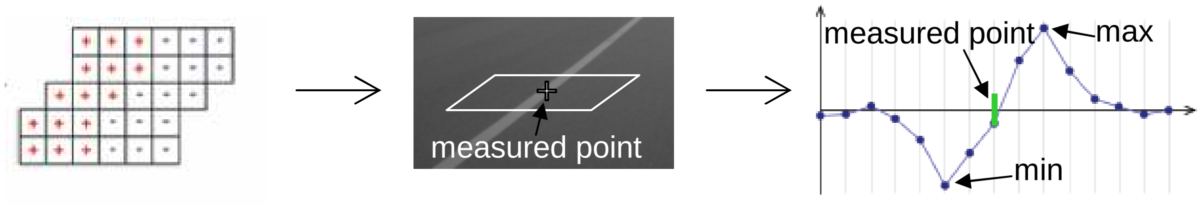


Figure 2: Example of adaptive convolution mask

In the update step of the Extended Kalman Filter the measurement residuals $(z_k - H_k \hat{\vec{x}}_k^*)$ are determined, weighted by the Kalman Gain K and added to the last predicted state vector:

$$\hat{\vec{x}}_k = \hat{\vec{x}}_k^* + K(z_k - H_k \hat{\vec{x}}_k^*)$$

Thus, the prediction for the next time step can be performed. For simplicity reasons the equations for the computation of the error covariance matrix are omitted here.

An example of the lane tracker working on real images can be seen in Figure 3. Figure 4 shows the algorithm working on rendered synthetic image.

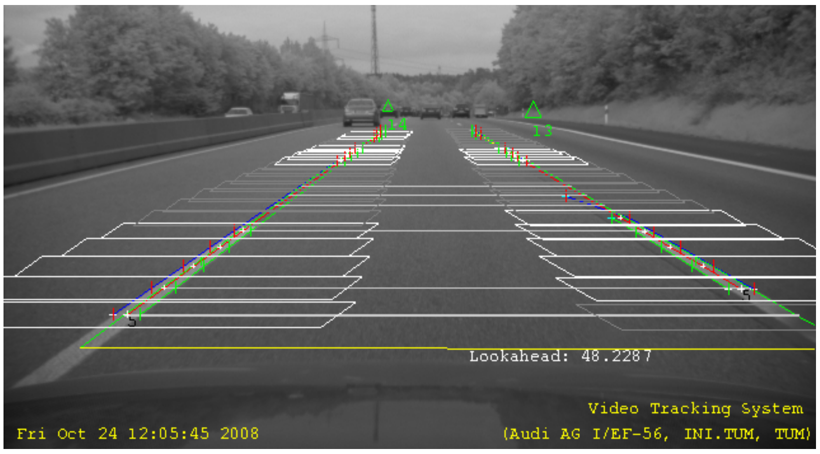


Figure 3: Lane tracking on a real image

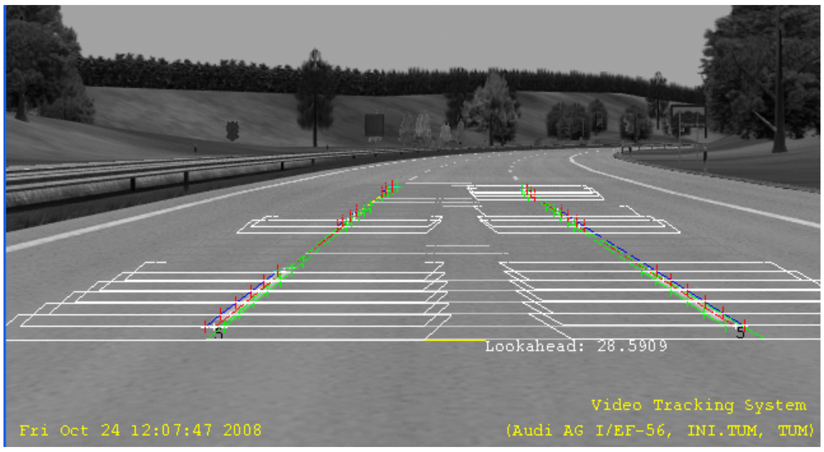


Figure 4: Lane tracking on a synthetic image

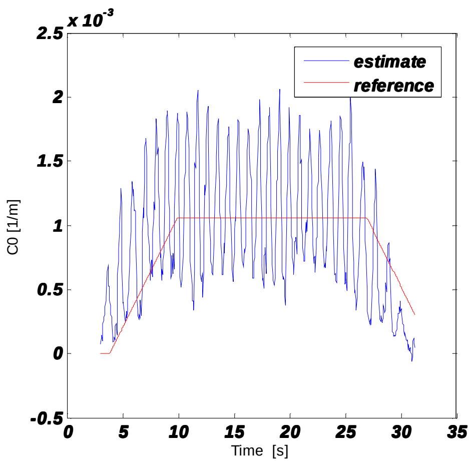
Preadjustment of the Lane Tracking Algorithm

The following paragraph describes one possibility to preadjust the lane tracking algorithm for the estimation of the lane curvature c_0 using the simulation environment.

The scenario chosen is a car ride on a flat, empty virtual highway in a right curve (Figure 4). The sensor inputs required by the lane tracker consists of the camera image, in this case the rendered view of the scene and dynamic information (speed and yaw rate) Latter is usually provided by the ESP. In this case, it is replaced by EGO vehicle dynamics data.

During the adjustment of the algorithm, the expected values are compared against measured values. The expected values represent ground truth values of the simulation, whereas the measured values are result values of the running algorithm.

One drawback is that certain parts of the simulation are not perfect, for example lane markings are modeled as sequences of straight lines. This leads to a minimum expected system error which would be achieved once the models of the simulation on one hand and the tracking system on the other hand are synchronized and properly calibrated. The partly unknown effects of the differences between the simulated world and the internal model of the tracker can be compensated by the tracker through adjusting the system error covariance matrix Q of the Kalman filter.



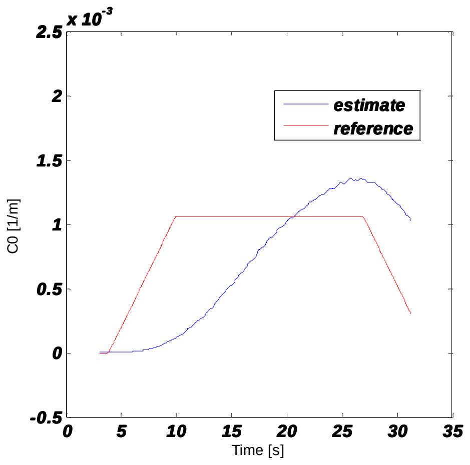


Figure 5: Test 1, too high uncertainty

Figure 6: Test 3, too low uncertainty

The system error assumption directly affects the impact of the new measurement on the last predicted values. A higher uncertainty results in a higher influence of the measurement. Therefore, assuming high uncertainty leads to noisy estimates, since the noise of the measurement directly affects the estimated state variable. Thus, it is interesting to know the minimum value of the assumed system error for which the estimator can still follow reality (here: virtual reality) within the allowed maximum latency time. This leads to the optimal assumption for the requirements of a specific driver assistance system.

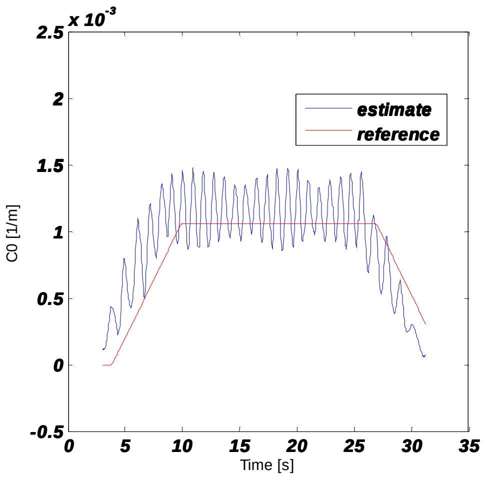


Figure 7: Test 2, medium uncertainty

The charts above show both the ground-truths of the curvature as well as the estimated results of the lane detector as a function of time, where positive values represent a right curve.

In Figure 5 the system error was set to an extremely high value, so there is a great noise on the estimated curvature. Figure 6 shows the results for a too low error assumption. While this low value completely eliminates the jitter effects, the estimator could not follow the ground truth anymore. For Figure 7 the value was set to a medium value, so there is less jitter, but the estimator can still follow the dynamic of the reference value. It can be seen, that there is an optimal value for the error assumption, for which the results should lie between the results shown in Figure 6 and Figure 7.

Conclusions

The Virtual Test Drive simulation environment combined with a hardware- in- the- loop simulator offers the application engineer a unique chance to preadjust and test real electronic control units in a lab. It is now possible to do functional trials in a closed loop set-up. This approach can help to decrease the amount of real road trials and thereby to save costs and to reduce risks for man and machine, respectively.

The example use case described in this paper demonstrates the potential of this simulation testbed. The results proved that it is possible to optimize the estimation quality by adapting the covariance matrix. The major benefit of this testbed is the opportunity to automate parameter studies using ground truth data and a weighting function to measure the quality of the results. Nonetheless, it is always necessary validate the adaptations in real road trials.

Bibliography

[1] K. von Neumann-Cosel, M. Dupuis, C. Weiss, *Virtual Test Drive – Provision of a Consistent Tool-Set for [D,H,S,V]-in-the-Loop,* In Proceedings on Driving Simulation Conference Europe, 2009, Monaco, provided that this paper will be accepted by the DSC scientific committee

[2] R. Schabenberger, *ADTF: Framework for Driver Assistance and Safety Systems,* In Proceedings on International Congress of Electronics in Motor Vehicles 13, 2007, Baden-Baden, Germany

[3] B. Mysliwetz: *Parallelrechnerbasierte Bildfolgen-Interpretation zur autonomen Fahrzeugführung,* Dissertation, Universität der Bundeswehr München, Fakultät für Luft- und Raumfahrttechnik, Institut für Systemdynamik und Flugmechanik, 1990, Neubiberg

[4] E.D. Dickmanns: *4-D Szenenanalyse mit integralen raum/zeitlichen Modellen*, In E. Paulus (Hrsg.): Mustererkennung 1987, Informatik Fachberichte 149, Seite 257-271, Springer-Verlag, 1987

[5] Redmill, K. A., J. I. Martin und U. Ozgluner: *Virtual Environment Simulation for Image Processing,* Sensor Evaluation. In: Intelligent Transportation Systems 2000 Proceedings, S. 64 _ 70. IEEE, 2000.