Robotica (2003) volume 21, pp. 289–294. © 2003 Cambridge University Press
DOI: 10.1017/S0263574702004794 Printed in the United Kingdom

Cost-oriented virtual reality and real-time control system architecture

Dirk Wollherr and Martin Buss

Control Systems Group, Faculty of Electrical Engineering and Computer Science, Technical University Berlin, Berlin (Germany)
e-mail: http://www.rs.tu-berlin.de

(Received in Final Form: November 9, 2002)

SUMMARY
In this paper the use of inexpensive standard hardware and software is proposed in place of high-cost commercial solutions to set up real-time control experiments with a human in the control loop, including graphical and haptic virtual reality (VR). For this purpose a generic simulation environment for the implementation of control experiments using VR with haptic feedback and their evaluation, has been developed. As an example, setups of a car simulator with a human in the control loop and an inverted pendulum as an experiment for student laboratories are presented.

KEYWORDS: Real-time simulation; Haptic control design; Haptic interface.

1. INTRODUCTION

Important issues of interest in current research are tele-presence and teleembodiment. The goal is to immerse the user in the illusion to be present at a distant place (real or virtual) and not just to interact with a computer. In this context one usually has to generate a virtual world where the user can act. In the past very expensive, dedicated hardware has been necessary to implement VR and real-time dynamic simulators. With the increasing power of personal computers and powerful graphic accelerator hardware for the mass-market it is nowadays possible to use standard hardware to set up VR in combination with real-time embedded control systems.

In this paper a framework based on standard hardware components to rapidly implement VR simulations and hardware-in-the-loop simulators is presented. The idea is to generate simulation code directly using the MATLAB Real-time Workshop in connection with a self-programmed Linux Realtime Target. This allows to rapidly prototype a simulation environment, to generate and compile the simulation code. Such a system offers new possibilities for setting up experiments at moderate cost either for scientific use or as an experimentation environment for students. The framework presented here has been used to implement two example applications: a low cost car simulator with haptic and graphic feedback to the driver who is part of the control loop, and a SCARA robot balancing an inverted pendulum as a student laboratory experiment.

The developed car simulator is used to test a rollover avoidance controller (RAC)[1] for vehicles with a high centre of mass, assisting the driver electronically in dangerous situations. The car simulator allows to test the RAC with a real driver's input, rather than with computer generated, predetermined trajectories. Furthermore, the effect of new forms of haptic feedback on the driver can be evaluated: Future vehicles are likely to be equipped with a drive-by-wire system, i.e. there is no mechanical connection between the tires and the steering input device (steering wheel). Therefore it is no longer necessary for the driver to feel the forces exerted on the front tires. This opens up new ways of communicating additional information on the vehicle status to the driver; here a force proportional to the roll angle is used. Haptic feedback is provided in this implementation by a force feedback paddle, first developed at the Technische Universität München.[2]

In a second example requiring short development cycles the usability of the proposed framework is demonstrated by setting up a laboratory experiment for graduate students. Students are able to design a control algorithm and rapidly generate real-time executable code to compare simulation results with data obtained from hardware in the loop experiments. The system combines real-time vision sensing with an embedded controller, again implemented using standard hardware.

The problem of developing a general framework for simulations and experiments is currently being followed at several research institutions worldwide: e.g. at the Swiss Federal Institute of Technology Lausanne (EPFL) a group has successfully set up real-time control experiments for distance learning[3] using a real-time kernel for MacOS developed at the EPFL. The interface to the user and the access to data acquisition hardware is provided by LabView. Another promising approach is being followed at the FernUniversität Hagen,[4,5] where the use of commercial control design software has been abandoned. Instead a set of applets has been written used as a user interface.

The approach presented here aims at rapid development and implementation of new concepts in computer-aided control engineering. The goal is not to provide a tool for developing end-user applications, but to allow quick and inexpensive implementation of case studies that probe the feasibility of new concepts and help to preview and discuss problems arising in the practical realization. This approach combines the flexibility and versatility of open software with its full accessibility of existing source code together with advanced control design tools such as MATLAB.

https://doi.org/10.1017/S0263574702004794 Published online by Cambridge University Press

The organization of this paper is as follows: In Section 2 the basic system architecture of the here proposed controller environment is presented. This includes both, the hardware requirements and the software architecture. As example applications, a car simulation for rollover studies, and the setup of a laboratory experiment are presented in Section 3. Section 4 concludes the paper.

2. SIMULATOR SYSTEM ARCHITECTURE

A VR simulation generally consists of two parts: the numerical simulation of the dynamical system model and the interface to the human user. From the different aspects of implementing the user interface, only visual and haptic feedback are considered in this paper, though stimulation of other human modalities such as the auditive sense might increase the impression of reality.

The basic idea about the proposed concept for designing simulation environments is to split up the software into different tasks like "haptic interaction", "visual feedback", or "simulation". These tasks are distributed over as many different computers as necessary to provide sufficient computational power. The use of several industry standard PCs is economically justifiable as they are generally cheaper than dedicated VR workstations or real-time simulators. These computers are linked over a LAN and exchange necessary data, like system states and coordinates, through TCP/UDP sockets or CORBA.

The following gives an idea of the hardware and software configuration which works for the example applications described in Section 3, concluding with a brief discussion of the performance achievable using the proposed architecture.

2.1. Hardware

The key components of the simulation prototyping environment are standard personal computers linked through a local area network (LAN). The choice of hardware for each PC depends on its task: real-time computation of dynamical system often requires extensive computational power, i.e. a powerful CPU, while the presentation of 3D virtual environments at adequate frame rates demands a video adapter with 3D acceleration.

These components being commercially available at moderate cost, low cost devices for haptic feedback usually neither have open programming interfaces nor provide sufficient power and accuracy. Hence the device adopted in reference [2] is used (see Figure 1). The operator can determine a control input by moving a lever attached to the axis of a motor into the desired position which is measured by a pulse encoder. If the motor exerts a moment on the lever, it gets bent. Strain gauges attached at the lower part of the lever are used to measure this flexion and estimate the moment resulting from the interaction between the human and the paddle.

All these data are processed by a Sensoray 626 multifunction I/O board providing ADC, DAC and quadrature decoders (see Figure 2). One DAC channel is used to control the motor through a PWM-amplifier, while an ADC measures the output of the strain gauges. The quadrature

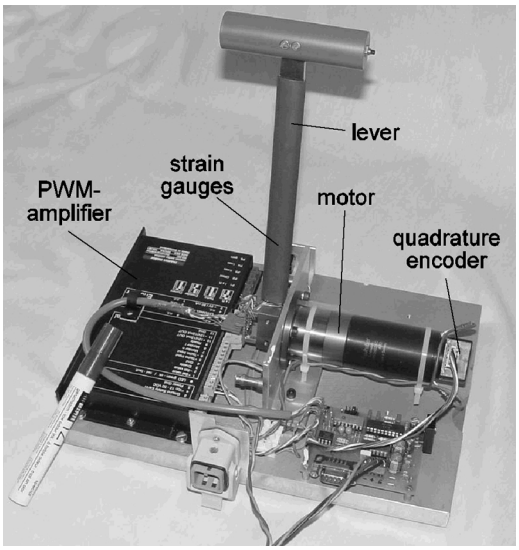


Fig. 1. Force feedback paddle.

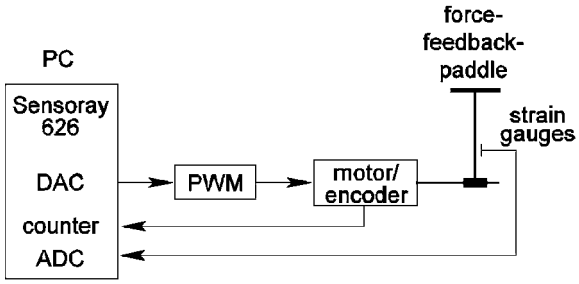


Fig. 2. Scheme for the control of the ff-paddle.

decoder determines the actual position and velocity of the lever. With this hardware architecture force, position, or impedance control can be realized.

The example of the force feedback paddle demonstrates how new devices needed for experiments can be built using a (small) number of standard hardware components, like multifunction I/O-boards, PWM-amplifiers and motors with quadrature encoders. Devices assembled from parts of this "building kit" have the virtue for the researcher to be known in detail and to be adaptable exactly to the individual needs.

Another strategic impact comes to the choice of software running on the selected hardware.

2.2. Software

As an operating system for the computers Realtime (RT) Linux was chosen. Besides its economic advantage – RT Linux is distributed free of charge – this system is technically competitive. While Linux itself already is capable of handling soft real-time requirements, its real-time extension RT Linux (http://www.rtlinux.org) meets hard timing constraints, i.e. it is possible to start processes at a fixed scheduled time. A RT Linux periodic task runs within 35 microseconds of its scheduled time on x86 hardware, the delay between the detection of an interrupt and the launching of the corresponding program is less than 15 microseconds.[6]

Another reason for choosing RT Linux is the availability of an enormous amount of software in its source code. The

https://doi.org/10.1017/S0263574702004794 Published online by Cambridge University Press

researcher can build upon this existing, well tested, efficient code, and adapt it to the individual needs. The time needed for development is shortened significantly.

During the planning process of the software architecture possible package delays due to congested LANs must be taken care of. Therefore it must be assured that control loops cannot become unstable with long delays or loss of single packets. A careful choice of the form of the data can speed up the communication process: for data-transmission absolute values like world coordinates are recommended, instead of incremental changes with respect to the previous data. This way the loss of single packets during transmission often is acceptable, hence faster UDP sockets can be used. In contrast to the TCP protocol, the UDP protocol does not check for correct delivery of packages which reduces the communication overhead of the protocol and hence speeds up data transmission.

Coding a problem by hand being very time consuming and susceptible to errors a new target for MATLAB's RT Workshop has been developed. This target allows a generation and compilation of standalone real-time code for RT Linux from a SIMULINK model. The control loop to be implemented is composed of ordinary SIMULINK blocksets; hence the migration from designing a controller in offline mode to evaluating it in an experiment is subject to substituting the system model by hardware in the loop which can also be accessed through SIMULINK blocksets. Another benefit of this procedure is the clear structure compared to a large number of manually linked files of source code. Consequently maintenance and modification of the model become easier and less time consuming. MATLAB was chosen as a platform as it is widely used. Of course any other modern simulation tool, capable of producing a standalone code, could have been chosen instead.

The visual output of the virtual environment is realized on a separate computer also receiving all necessary coordinates and system states from the simulation task through TCP- or UDP-socket connections. The graphics task does not run as a real-time application as the soft real-time capabilities of Linux are sufficient to ensure time consistency between the simulation and the feedback to the driver.

The proposed distributed architecture raises the question of achievable performance and limitations using the framework.

2.3. *Performance of the proposed architecture*

An important performance measure for tasks distributed over several applications is the delay for communicating data between them.

Due to the special software architecture of RT Linux,[7] the realtime process is only capable to exchange data through dedicated RT fifos; these RT fifos are serviced by a special, non-real-time application, further on referred to as the "data handler" (see Figure 3). This data handler is able to forward the received data to other external applications by any means of network communication protocol; here UNIX sockets are used.

The latency for data exchange through RT fifos is small and very constant. Measurements showed an expectation

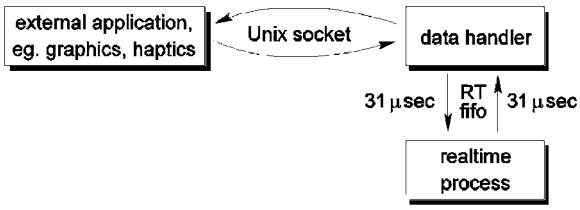


Fig. 3. Fundamental communication structure.

value of 62 μsec for the round trip data handler "RT process" with a variance of 1.2 μsec.[2]

The communication delay to other applications highly depends on a number of factors such as network congestion, choice of communication protocol, packet length, etc. In the here presented case UDP/IP packets containing 8 bytes of user data were used. The average round trip delay between two computers – i.e. a packet is sent one PC, mirrored by a second PC and received again by the first PC – was 87 μsec with a variance of 249 μsec.[2] The same measurement for communication of two separate tasks on the same computer showed an average round trip delay of 31 μsec with a variance of 34 μsec.[2] These communication delays should be taken into account when distributing tasks over different computers as delays in a control loop might cause instability.

In the following, two example applications based on the proposed framework are presented demonstrating the implementation of practical problems.

3. APPLICATIONS

3.1. *Rollover avoidance*

The framework described above has been used to implement a car driving simulator used for the evaluation of a rollover avoidance controller (RAC). Following the idea in reference [8] a driver assistance system for vehicles with high centre of mass was suggested in references [1, 9] which limits the range of the steering angle accessible by the driver. The RAC determines how close the vehicle is to tilting from the measured state $\boldsymbol{x}$ of the vehicle and the steering angle δ^w adjusted on the front wheels by evaluating the rollover coefficient

$$R(\boldsymbol{x}, \delta^w) = \frac{F_{z,R} - F_{z,L}}{F_{z,R} + F_{z,L}}$$

While the wheels on both sides of the vehicle have contact to the ground, $|R| < 1$, $|R| > 1$ means the car is tilting. If the driver chooses a steering angle which makes the vehicle tilt, i.e. $|R|$ becomes close to 1, the RAC determines all steering angles which prevent rollover. From all admissible angles the one closest to the angle commanded by the driver is set at the front wheels. For further information on the controller please refer to references [1, 9].

3.1.1. Realization. In this setup the tasks of the simulator have been split into three parts (Figure 4): (i) simulation of the dynamical system model; (ii) control of the haptic interface; (iii) 3D graphical visualization. Each of these

https://doi.org/10.1017/S0263574702004794 Published online by Cambridge University Press

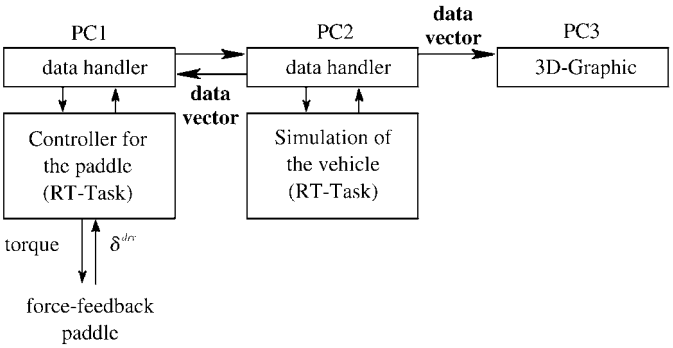


Fig. 4. Communication structure of the simulation software.

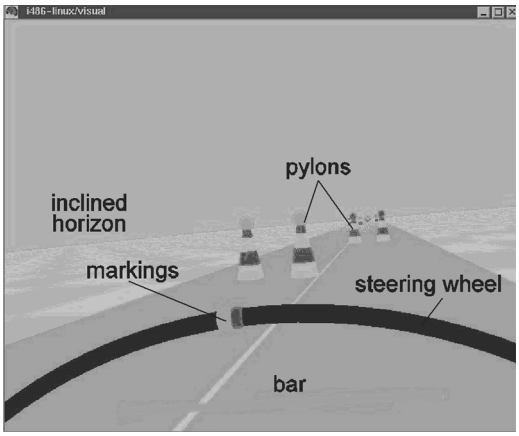


Fig. 5. 3D simulation environment: street with pylons and virtual steering wheel.

tasks runs on a separate computer. The PCs are all equipped with AMD Athlon 900 processors and 256 MB RAM. They are linked through a 100 Mbit switched LAN.

The interface to the user is split into two channels: the visual feedback and the haptic feedback through the steering device. The virtual environment has been programmed in *C*, using the Maverik library, a toolkit for VR application programmers using OpenGL/Mesa for low level rendering. Figure 5 shows an abstract driving situation with the driver looking through the vehicle windscreen and the virtual steering wheel in front of him/her. Pylons on alternating sides of the road represent obstacles which the driver is supposed to avoid. The inclination of the horizon corresponds to the tilting angle of the vehicle. There are two markings on the steering wheel. The light grey marking in 5 denotes the steering angle δ^{drv} chosen by the driver; the dark grey one represents the invariance angle δ^{inv}. A bar is located on the lower edge of the screen, whose length proportionally depends on the actual rollover coefficient R.

Through the force feedback paddle the driver obtains additional information about the rollover coefficient. Experiments showed that this information helps the driver to better understand the complex dynamics of the vehicle. In these experiments the driver feedback was a moment

$$M_L = (e^{-5(R+1)} - e^{-5(-R+1)})\, M_L^{\max},$$

with $M_L^{\max}$ being the maximum admissible moment.

3.1.2. Experimental Results. The vehicle model used in the experiments is the so-called nonlinear one-track model

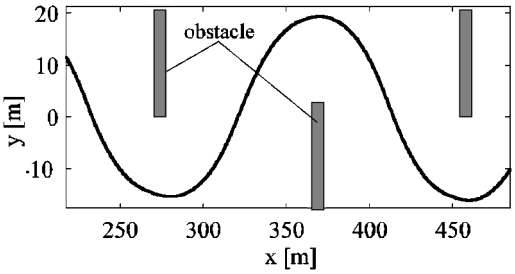


Fig. 6. Driven trajectory.

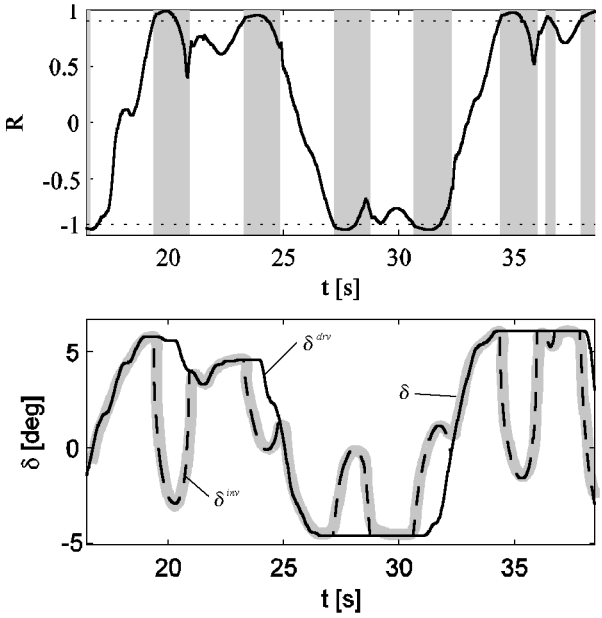


Fig. 7. Simulation results.

as described in reference [10] extended by an additional mass following.[11] Forces exerted on the tires are modeled by the HSRI tire model.[12] A detailed description of this model can be found in reference [1].

The system parameters have been chosen to describe a 14.3 ton lorry with its centre of gravity at a height 2.28 m above the ground and a track width of 0.93 m. From this data one can see that the vehicle is unrealistically unstable and tilts easily. The choice of parameters was done to demonstrate the capabilities of the controller.

Figure 6 shows a slalom maneuver steered by a human driver trying to avoid three obstacles. The corresponding steering angle δ^{drv} of the driver is shown in the lower plot of Figure 7 as a solid line. As soon as the rollover coefficient R in the upper graph of Figure 7 exceeds the threshold $|R| = 0.9$ (dotted lines), the RAC becomes active and commands the invariance angle δ^{inv}, marked as a dashed line in the lower plot. Due the characteristics of the steering dynamics of a 2nd order system $|R|$ still exceeds the threshold 0.9, but one can see that the threshold has been chosen to keep the truck from tilting. Note that a vehicle not equipped with a RAC would have tilted as soon as R reached the value ± 1.

3.2. Inverted Pendulum

The second example application of the presented framework is the control of an inverted pendulum. This setup is used as

https://doi.org/10.1017/S0263574702004794 Published online by Cambridge University Press

an experiment for students on an advanced control course laboratory offering the possibility to test and compare various control strategies. This problem – at the first sight quite different to the one described before – has similar requirements on the underlying software: due to the limited time the students have for the laboratory, an environment allowing short development cycles is needed. The proposed approach provides an efficient framework for such laboratory experiments.

In this experiment the SCARA robot in Figure 8 is to balance an inverted pendulum in the upright position. Pulse encoders at the two driven joints of the robot allow one to determine the position of the actuator of the robot. The orientation and the inclination angle of the pendulum are measured with a video camera tracking the upper end of the pendulum which is marked with an LED for easier visual tracking.

In this setup the software has been split into two tasks running on different PCs: the controller and the visual tracking system. The software XVision (http://www.cs.jhu.edu/CIPS/xvision/) is used for tracking. It was modified to directly send the tracking coordinates to the controller through a socket connection. The controlling computer is equipped with a Servo-To-Go motor control board featuring DAC, ADC and pulse encoders on a single board. It is used to access the motors at the joints of the SCARA robot.

Students can design controllers using Matlab and test them in offline simulations using a model of the SCARA robot to be controlled. When offline simulations of the controller work, real-time code is directly generated and the controller can be evaluated with hardware in the loop. This is very useful to demonstrate the effects of potentially not modeled system properties, such as friction and stiction.

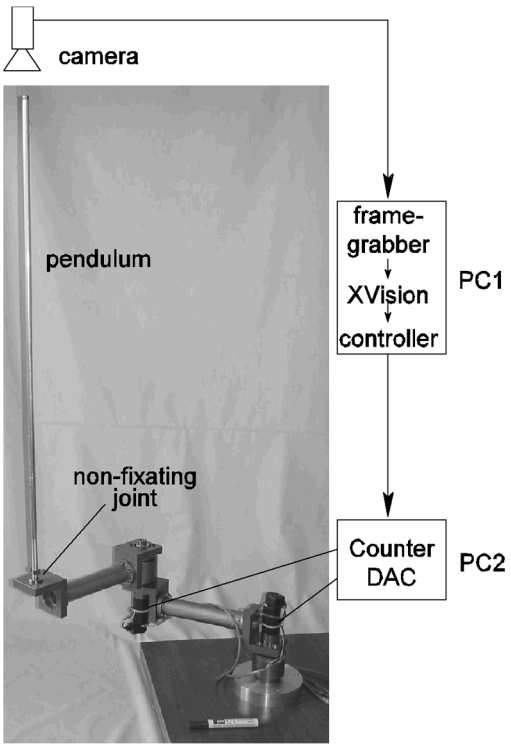


Fig. 8. SCARA robot with an inverted pendulum.

The laboratory setup offers a variety of possible experiments: Designing a position or velocity controller for the SCARA robot is an introduction to classical problems in robotics; the inverted pendulum, being highly nonlinear, allows to compare nonlinear control strategies such as feedback linearization or sliding mode control. The proposed framework provides the flexibility needed to run several experiments on a single hardware setup without modification.

This example shows how a laboratory experiment can be set up at moderate cost using inexpensive hardware. Future plans include extending the setup by a third computer providing a VR representation of the system. This computer can serve *java* applets on a distant computer enabling Internet experiments. A further step is the implementation of augmented reality offering the possibility to disturb the experiment through the Internet.

4. CONCLUSIONS

In this paper a novel framework for rapid control prototyping and development of VR simulation environments is presented. It allows fast development of new simulation environments at moderate cost. Furthermore the framework is versatile and easily expandable as all programming interfaces, except Matlab, are open in the public domain.

The example application of vehicle rollover avoidance control can be seen as a first step towards Computer Aided Control Design (CACD) with haptic feedback. During the development of the RAC this kind of CACD proved to be very helpful to evaluate the RAC.

ACKNOWLEDGMENTS

Technical support by U. Weidauer and A. Bergmann is greatly appreciated. The inverted pendulum experiment has been set up by C. Rissler during his diploma thesis.

References

1. D. Wollherr, J. Mareczek, M. Buss and G. Schmidt, “Rollover Avoidance for Steerable Vehicles by Invariance Control”, *Proceedings of European Control Conference*, Porto, Portugal (2001) pp. 3522–3527.
2. H. Baier, “Model-Based Teleoperation of a Drilling Machine in Uncertain Communication Barriers.” *Internal Report* (Institute of Automatic Control Engineering, Technische Universität München, April, 1997).
3. C. Salzmann, D. Gillet, and P. Huguenin, “Introduction to Real-time Control using LabView with an Application to Distance Learning”, *The International Journal of Engineering Education* **16**(3), 255–272 (2000).
4. A. Jochheim and C. Röhrig, “The Virtual Lab for Teleoperated Control of Real Experiments”, *Proceedings of the Conference on Decision and Control*, Phoenix, Arizona, USA (1999) pp. 819–824.
5. C. Röhrig and A. Jochheim, “The Virtual Lab for Controlling Real Experiments via Internet”, *Proceedings of the IEEE International Symposium on Computer Aided Control System Design*, Kohala Cost, Hawaii USA (1999) pp. 279–284.
6. M. Barabanov, “A linux-based real-time operating system”, *Master’s Thesis* (New Mexico Institute of Mining and Technology, Scorro, New Mexico, 1997).
7. FSM Labs, Inc., http://fsmlabs.com, Getting Started with RTLinux (2001).

https://doi.org/10.1017/S0263574702004794 Published online by Cambridge University Press

8. J. Ackermann and D. Odenthal, "Damping of Vehicle Roll Dynamics by Gain Scheduled Active Steering", *Proceedings of European Control Conference*, Karlsruhe, Germany, (1999).
9. J. Mareczek, D. Wollherr, M. Buss and G. Schmidt, "Überschlagsvermeidung bei Kraftfahrzeugen durch Invarianzregelung", *at – Automatisierungstechnik* **50**(2), 70–78 (2001).
10. P. Riekert and T. Schunck, "Zur Fahrmechanik des gummibereiften Kraftfahrzeugs", *Ingenieur Archiv.* **11**, 210–224 (1940).
11. L. Segel, "Theoretical prediction and experimental substantiation of the response of the automobile to steering control", *Proc. IMechE* (1956–1957) pp. 310–330.
12. M. Mitschke, *Dynamik der Kraftfahrzeuge*, **Vol. C** (Berlin, Germany: Springer, 1990).

https://doi.org/10.1017/S0263574702004794 Published online by Cambridge University Press