
# Adaptive Perception, State Estimation, and Navigation Methods for Mobile Robots

Habilitationsschrift

zur Erlangung der Lehrbefähigung für das Fach Informatik

vorgelegt dem
Fachbereich Mathematik und Informatik
der Freien Universität Berlin

von

Herrn Dr. Daniel Göhring

eingereicht am 13. Juli 2020

# 1 Summary

In this cumulative habilitation, my most relevant publications with focus on robotic perception, self-localization, tracking, navigation, and human-machine interfaces have been selected. These publications have been created during my post-doctoral research and during my time as a Juniorprofessor.

The author wants to clarify that the term “Mobile Robots” in the title of this work refers to a wide research area. Any robot which has the ability to change its own position can be considered to be mobile. Legged robots, aerial robots, and water robots, e.g., are mobile robots but have to solve different problems than wheeled robots do, e.g., planning a stable walk, maneuvering in the air, or localizing themselves and objects under water. Even though the modular arrangement of perception, world modeling, planning, and control is shared by many robots, the modules themselves can differ considerably for each domain. Thus, the presented robots in this work and the related publications provide a representative - even though not complete - overview about the the problems which need to be solved for wheeled mobile robots.

# 2 Robotics Hardware

I have been working in two different domains and mainly with two different hardware platforms. During my postdoctoral stay at ICSI and the EECS department of University of California at Berkeley, I have been working with a PR2 household robot in the Robotics Learning Lab of Professor Pieter Abbeel on vision and machine learning tasks, see Fig. 1.

Before and after my two-year stay in the United States of America, I was working at the AutoNOMOS-Labs at Freie Universität Berlin under the supervision of Professor Raúl Rojas, where I have been focusing on control, planning and object tracking aspects of the autonomous vehicles ”MadeInGermany” and ”e-Instein”, which have become the first autonomous vehicles in the city of Berlin, see Fig. 2.


Figure 1: The PR2 robot from Willow Garage.


Figure 2: The autonomous vehicles e-Instein (left) and MadeInGermany (right) on the former Berlin Tempelhof airport testing ground.

# 3 Arrangements of the Publications in the Robotic Context

For a better understanding how the different publications relate to each other and to particular robotics modules and solutions, the following paragraph will give a short summary.

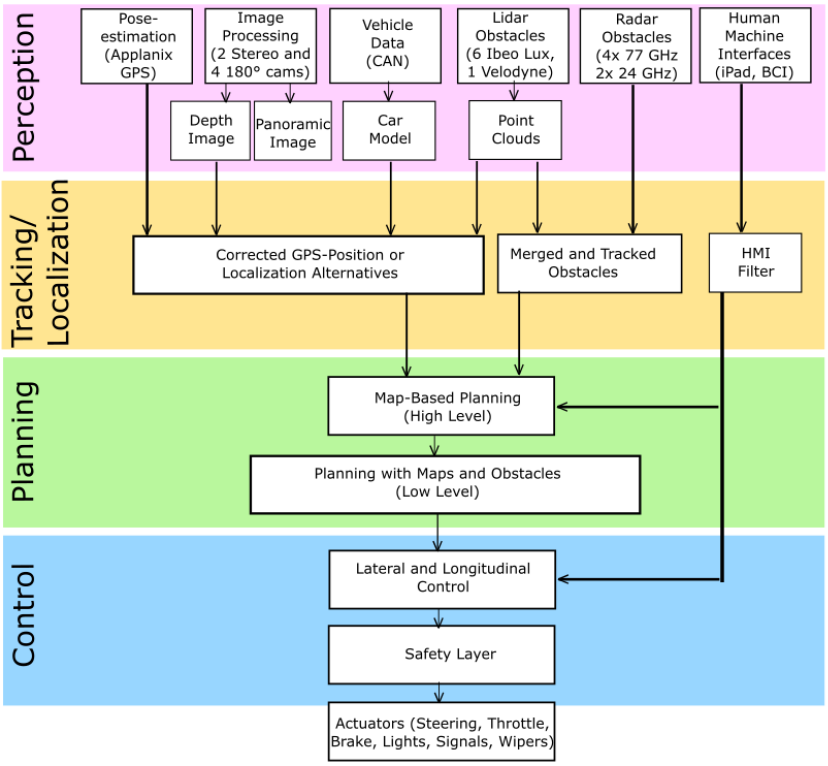



Figure 3: Module Configuration in the autonomous vehicle MadeInGermany.

## 3.1 Categorization of Publications

The author defines four categories of robotic modules within the robotic data processing chain: perception, world modeling, planning, and control. Even though other categorizations are possible and other researchers might want to add other problem classes or group existing ones together, this categorization is widely accepted.

The following paragraphs will explain, which type of modules belong to each of the four categories.

**Perception:** Perception modules, in the context of this work, handle sensory data. They compute object classes and object locations, e.g., within an image, or they cluster 3D points, which are provided by a LiDAR sensor to an object. Most applications of Deep Neural Networks to images can be considered to be a perception task. Also the generation of a depth image from two camera images and the detection of objects within the depth image is part of the perception layer. The perception layer ends (in this work) before sensory data is integrated over time.

**Tracking, Localization, World Modeling:** Tracking and localization are part of the world modeling layer. Here, recognized objects from the perception layer, from one or multiple sensors, are integrated over time in order to create a world model. As a result of this integration, the world model tells a robot what the positions and velocities of surrounding objects are. Also the prediction of the current scene into the future can be part of the world modeling layer. An important aspect of the world model is the representation of position uncertainty, velocity uncertainty, and other uncertainties as well as their inter-dependencies. As a form of representation, covariance matrices and entropy measures are chosen very often.

**Planning, Navigation:** In the planning layer, a robot needs to plan its actions under the given world model in order to achieve one or multiple goals. For a household robot, e.g., an important task is to grasp a specific object. Especially for autonomous vehicles but also for mobile robots in general, it is important to plan a path through a static or dynamic environment. In many cases, a robot has a map. But sometimes a map has to be created first or is not available at all. The planning layer can contain different sublayers: One part is usually responsible for an abstract plan, e.g., for an autonomous vehicle to create a plan from one city to another, but without knowledge about the traffic situation. Another layer is necessary to create a more detailed plan, e.g., for an autonomous vehicle to plan in what lane to drive and with what velocity - all that with a limited horizon of at most some hundred meters, which corresponds to the sensor range. The result is usually a 2D or 3D spline, defined over a time parameter or over a distance parameter. Desired spline properties are often that the spline function and its first and second derivatives are continuous.

**Control:** The control layer usually consists of modules which execute a given plan as precisely as possible. In an autonomous vehicle, the controller will process the amount of the throttle, brake, and steer commands. All the

| Paper | Perception | World Modeling | Planning | Control |
|---|---|---|---|---|
| [1] Lidar / Radar | | X | | |
| [2] iDriver | | X | | X |
| [3] BCI | | X | X | X |
| [4] Controller | | | X | X |
| [5] Grounding | X | | X | |
| [6] Object Det. | X | | | |
| [7] Grasp Aff. | X | | X | |
| [8] Acoustic | X | | | |
| [9] DNN / Lidar | X | | | |
| [10] Awareness | X | X | | |
| [11] Pole SL | X | X | | |
| [12] Path Graphs | | X | | |
| [13] LBP / HOG | X | | | |
| [14] TEB | | | X | |
| [15] A-Star | | | X | |
| [16] Mapping | | X | | |
| [17] Vector | | | X | |
| [18] TEB | | | X | |
| [19] Industry | X | | X | |
| [20] Collision | | X | | |
| [21] Head Pose | X | | | |

Table 1: This table shows, which papers cover which of the four research categories (perception, world modeling, planning, control).

calculations need to be processed periodically with a frequency of usually 100 - 1000 Hz. Another aspect of a controller is to make sure that a given plan will be executable while keeping a robot within safe dynamic state. E.g., a vehicle shall not apply a bigger steer angle than its tire friction, center of gravity, or other stability constraints allow.

## 3.2 Clusters of Own Research

To get a better understanding how the research papers of this work relate, they will be grouped together and their relation to each other will be explained briefly.

**Object Detection for Indoor Robots with Grippers:** The PR2 robot of publications [5,6,7] and the BMW robot [19] operate indoors and within a

defined environment. The presented publications which use these two robots focus on perception layer tasks. Objects need to be recognized in order to be grasped later. The PR2 robot needs to recognize different object classes within 2D images, their poses (position and orientation) using histograms of oriented gradients as features and support-vector machines as classifiers (abbreviated as HOG/SVMs). On the other hand, the BMW robot needs to detect boxes which contain car assembly parts and their poses using convolutional neural networks (CNNs) on 2D images.

**Human-Machine Interfaces for Autonomous Vehicles:** In [2] and [3], an iPad and a BCI were tested for their applicability to control an autonomous vehicle, or how the BCI can be used to influence the planner of an autonomous vehicle. In [10], an assistance system was tested, which checks the gaze direction of a driver with the help of an eye-tracker and warns the driver, if he or she missed to look at a traffic light or to check another vehicle.

**Sensor Fusion and Object Tracking** In [1] was analyzed how radar and lidar sensory data of an autonomous vehicle can be combined. In [8], two Kinect cameras were combined in order to localize other traffic vehicles by sound, and the approach was evaluated using lidar data. In [9], lidar data was used to support an image based object classification and detection method using CNNs for traffic scenarios. In [13], the data of two cameras were applied to create a stereo image which was used to detect other vehicles. For better precision, LBP and SVM based approaches were combined with depth information of the stereo image.

**Map Creation by Observation** In this research area, other traffic participants were observed by an autonomous vehicle to generate a map containing street lanes. In [12], this could be achieved using LiDAR sensory data to create path graphs, whereas in [16] the representation of the map was achieved through an orientation-based 2D grid.

**Path Planning Approaches** Different methods to create drive plans for autonomous vehicles were tested: In [14] and [18] a TEB-planner was tested which is able to create plans in space over time without a given map but by observing other traffic participants alone. [15] introduces an A* planner which uses a map and which also creates a plan in space over time. In [17] a vector field, provided by a map, is used to execute an action for an autonomous model car. Here, no plan is generated but the action is calculated immediately from the vector field. A simple planner (street follower),

together with the controller of the autonomous vehicle, is explained in [4].

**Further World Modeling and Situation Prediction Approaches** In [11], a stereo image is used to detect trees and other pole-like structures. The detected poles are matched against a map and the localization process, implemented as a Monte-Carlo particle filter in combination with a Kalman filter, is supported by the vehicle's odometry data. In [20], a collision risk is calculated for autonomous vehicles. In [21], a method to estimate body poses in a traffic scenario is presented using a monocular camera.

# 4 Selected Publications and Own Contributions

The following publications have been selected with respect to the research focus of this habilitation. Under each publication, my contributions are briefly described.

[1] **Daniel Göhring**, Miao Wang, Michael Schnürmacher, Tinosch Ganjineh, "Radar/Lidar Sensor Fusion for Car-Following on Highways.", In: *5th International Conference on Automation, Robotics and Applications, ICARA 2011, Wellington, New Zealand, December 6-8, 2011*, pp. 407–412, IEEE, 2011. DOI: 10.1109/ICARA.2011.6144918
URL: https://doi.org/10.1109/ICARA.2011.6144918

The idea of this work is to improve the accuracy of the velocity estimation of vehicle objects in front of an autonomous car with focus on highway scenarios. This problem is non-trivial, because of the relatively high longitudinal distances between vehicles on a highway, compared to distances in city scenarios. Higher distances result in fewer measurement points from the lidar which again results in more inaccurate distance and velocity measurements from the lidar. The radar sensor has a very limited field of view and as a result cannot perceive cars ahead in tight curves. In this work, a method for sensor fusion of lidar and radar data is presented, combining the high distance precision of lidar with velocity estimations from radar data. Another important aspect of this work is how the different frequencies of the lidar and the radar sensor can be handled by a Kalman filter, which also needs to deal with delayed or missing sensor information, see Fig. 4.

I contributed the conceptual idea, a substantial amount to the design of the Kalman filter, and to the evaluation. I was also the main test driver for

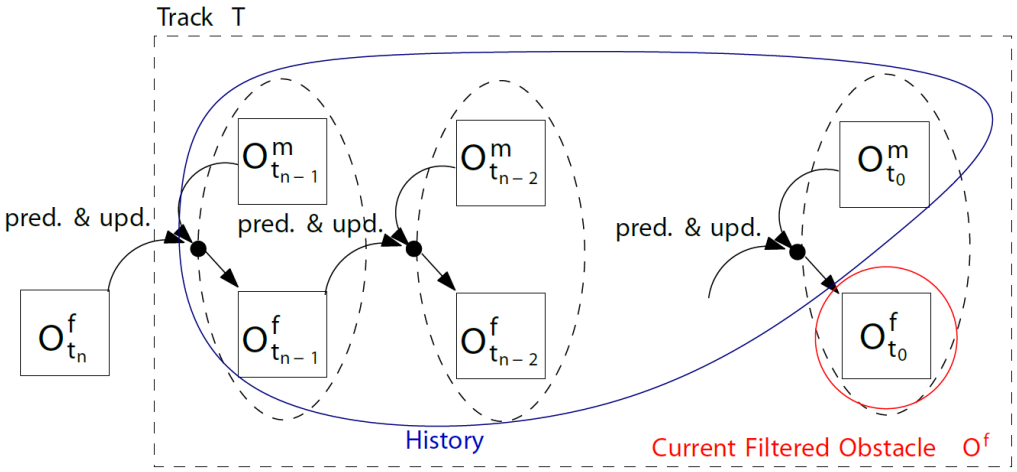



Figure 4: The Kalman filter concept for data fusion, with the ability to integrate delayed sensor data.

this work and wrote major parts of the paper.

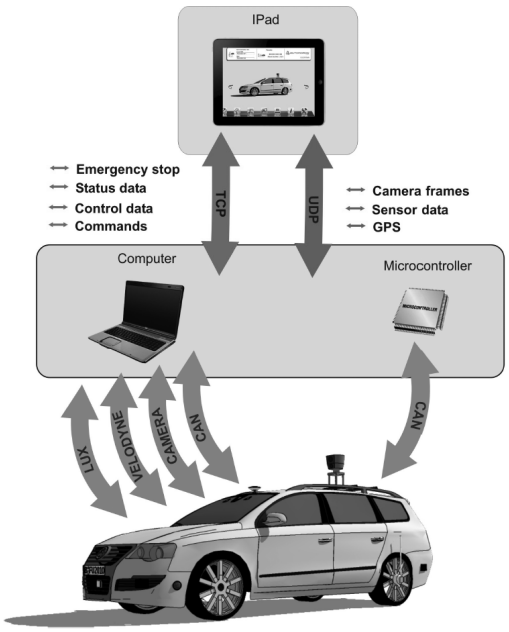



Figure 5: Data flow chart between the iPad and the vehicle's main computer.

[2] Arturo Reuschenbach, Miao Wang, Tinosch Ganjineh, **Daniel Göhring**, "iDriver - A Human Machine Interface for Autonomous Cars.", In: *Eighth International Conference on Information Technology: New Generations, ITNG 2011, Las Vegas, Nevada, USA, 11-13 April 2011* , pp. 435–440, IEEE Computer Society, 2011. DOI: 10.1109/ITNG.2011.83
URL: https://doi.org/10.1109/ITNG.2011.83

A remote control and data debugging app for the newly introduced iPad is presented. The remote control allows the user of the iPad to remotely control the automated vehicle, i.e., to control throttle, brake, and steering. Furthermore, lidar data and other information from the vehicle can be displayed on the iPad.

I contributed to the interface between iPad and main computer of the vehicle, see Fig. 5. I also implemented the CAN-communication in the vehicle and took care of the implementation of an emergency stop. I took precautions in terms of limiting the maximum steer angles as well as the maximum throttle and brake commands, in order to prevent the vehicle from entering (a subset of) unsafe driving states. I also helped evaluate the approach.

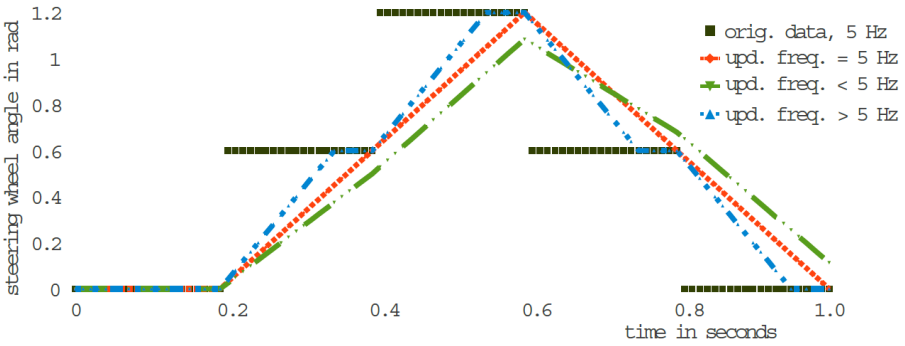



Figure 6: Steer Angle Smoother, the black dotted curve shows the raw angle values at 5 Hz as red from the BCI computer; the red curves shows the linearly interpolated steering angle with a control frequency of 100 Hz.

[3] **Daniel Göhring**, David Latotzky, Miao Wang, Raúl Rojas, "Semi-Autonomous Car Control Using Brain Computer Interfaces.", In: *Intelligent Autonomous Systems 12 - Volume 2 Proceedings of the 12th International Conference IAS-12, held June 26-29, 2012, Jeju Island, Korea,* Advances in Intelligent Systems and Computing, vol. 194, pp. 393–408, Springer, 2012. DOI: 10.1007/978-3-642-33932-5_37
URL: https://doi.org/10.1007/978-3-642-33932-5_37

This paper complements our research on human-machine interfaces for autonomous cars. After in [2] an iPad was used to control an automated vehicle, in this work a brain computer interface (BCI)-based control method for an autonomous vehicle is presented. A test person can generate up to four commands using a motor imagery based BCI from the Emotiv company. I implemented a controller which super-samples and smooths the input data for this work and simplifies the control of the car with the BCI, as shown in Fig. 6. A considerable amount of research included questions about how to avoid control oscillations by the human in order to drive straight after a curve, how fast the steering needs to be, if the steering wheel shall return by itself, how the throttle and brake commands shall be mapped to velocities or accelerations - just to name a few challenges. I also designed and conducted most parts of the experiments as the safety driver in collaboration with the test person and wrote major parts of the paper.

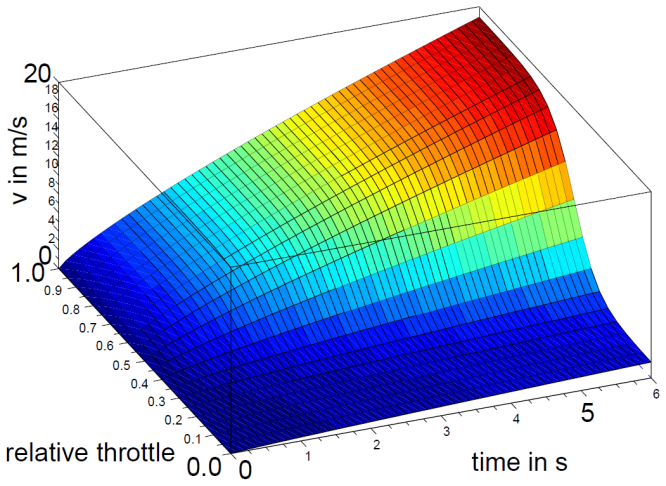



Figure 7: Approximated velocity function over time and over relative throttle values.

[4] **Daniel Göhring**, "Controller architecture for the autonomous cars: MadeInGermany and e-Instein.", *Technical Report, Freie Universität Berlin, Fachbereich Mathematik und Informatik*, 2012.
URL: http://dx.doi.org/10.17169/refubium-21789

In this work, the controller architecture and parts of the high level planner for the two autonomous vehicles at FU Berlin are described. I designed, implemented, and tested all aspects of the controller and most of the planning parts described in this technical report. I recorded and evaluated the experimental data, generated the dynamic vehicle models, and wrote the whole report. From the recorded vehicle data I created a prediction model, which tells the controller module, how much throttle or brake needs to be applied in order to achieve a certain acceleration under the condition of a given velocity. The model creation part and the inversion of the function were non-trivial, because of the non-linear character of the underlying function, see Fig. 7.

[5] Sergio Guadarrama, Lorenzo Riano, Dave Golland, **Daniel Göhring**, Yangqing Jia, Dan Klein, Pieter Abbeel, Trevor Darrell, "Grounding Spatial Relations for Human-Robot Interaction.", In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013,* pp. 1640–1647, IEEE, 2013. DOI: 10.1109/IROS.2013.6696569
URL: https://doi.org/10.1109/IROS.2013.6696569

This research work was created during my work at ICSI and UC Berkeley as a researcher in the DARPA funded BOLT project. The main question was how humans can communicate commands to a robot via human language. In this work, written sentences with focus on spatial prepositions were fed to a PR2 robot. The robot had to detect household objects, and to pick and place those objects accordingly to the content of the sentence received. A language parser was used. The meaning of spatial prepositions, e.g., in front of, behind, to the left of, inside, etc. were trained using annotated data in example scenarios. One of my parts in this systems paper was the implemen-

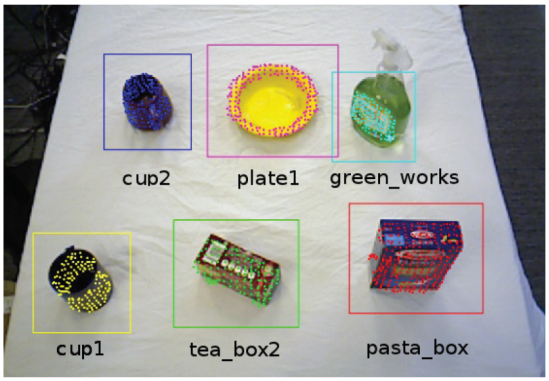



Figure 8: View from the mounted Kinect camera on the PR2 robot. Segmented objects are enframed, corresponding point cloud points are depicted, and object labels are shown.

tation of the data collection module under ROS with which I collected and automatically annotated the image data for training. Further, for the object detection with a trained image classifier, I designed and implemented the part of the object detection which needed to find the region of interest in the image using RGB-D data from a Kinect-like sensor, see Fig. 8. Those image patches were processed and sent to a linear SVM classifier, which was not trained by me. I evaluated the vision module and wrote the corresponding vision parts in the paper.

[6] **Daniel Goehring**, Judy Hoffman, Erik Rodner, Kate Saenko, Trevor Darrell, "Interactive Adaptation of Real-Time Object Detectors.", In: *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014* , pp. 1282–1289, IEEE, 2014.
DOI: 10.1109/ICRA.2014.6907018
URL: https://doi.org/10.1109/ICRA.2014.6907018

This work uses a simple webcam and an online-trained HOG/SVM classifier to detect household objects. The method can be used by robotics practitioners to quickly (under 30 seconds per object) build a large-scale real-time perception system. In particular, it is shown how to create new detectors on the fly using large-scale internet image databases (Image-net), thus allowing a user to choose among thousands of available categories to build a detection system suitable for the particular robotic application. Furthermore, it is shown how to adapt these models to the current environment with just a few in-situ images. Experiments on existing 2D benchmarks evaluate the speed, accuracy, and flexibility of the system.


Figure 9: Top: Demo setup with PR2 and objects to detect in front of it; Bottom left: Training user interface, object to learn in region of interest (red square); Bottom right: Detected objects in bounding boxes.

For this paper, I designed and implemented the vision part, e.g., the image extraction software which allows the user to capture in-situation images, see Fig. 9. Furthermore I implemented the in-situation image training part, and designed the modular system, i.e., the interaction of the different ROS-nodes under ROS. I wrote the corresponding sections of the paper. I also created the demo video for the paper. The video and the paper were submitted to ICRA and published. I presented the paper at ICRA 2014 in Hong Kong, China.

[7] Hyun-Oh Song, Mario Fritz, **Daniel Goehring**, Trevor Darrell, "Learning to Detect Visual Grasp Affordance.", In: *IEEE Transaction on Automation Science and Engineering,* vol. 13, no. 2, pp. 798–809, 2016.
DOI: 10.1109/TASE.2015.2396014
URL: https://doi.org/10.1109/TASE.2015.2396014

In this journal paper, a PR2 robot had to detect grasp points of household objects, e.g., pan handles using 2D image and 3D point cloud data as well as a trained image based HOG/SVM classifier. I wrote the image/point cloud fusion part, i.e., the selection of image regions (not the detection of grasp points) by taking advantage of 3D point clouds from an RGB-D camera sensor. I contributed to the system design under ROS and conducted the experiments on the PR2 together with another co-author.

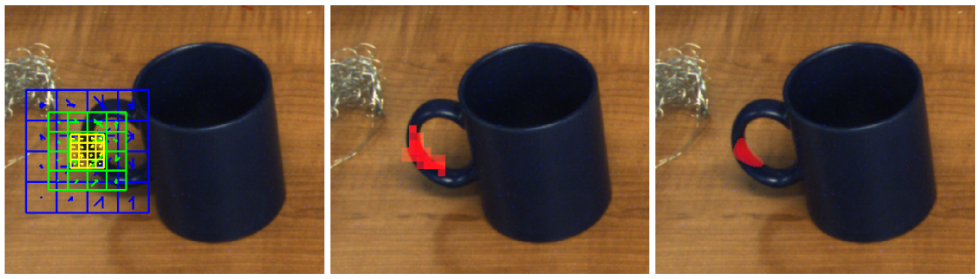


Figure 10: Local grasp region detection for a cup object. Left and center: SIFT descriptors on a key point were used to find suitable gripping points for the PR2 robot. Right: The point region is smoothed.

[8] Hamma Tadjine, **Daniel Goehring**, "Acoustic/Lidar Sensor Fusion for Car Tracking in City Traffic Scenarios." In: *3rd International Symposium on Future Active Safety Technology Towards zero traffic accidents (FAST-zero) '15, Gothenburg, Sweden, September 9-11, 2015,* pp. 67–72, 2015.
URL: https://research.chalmers.se/en/publication/222422

In the presented publication, a vehicle detects directions of moving objects based on the incoming acoustic data. Therefore, the microphones of two Kinect cameras (two microphones from each Kinect) were used in a way to make sure, that no ambiguities remain, no matter which direction the sound was emitted from. I wrote the nodes that process the acoustic data and generate an angular distribution, based on the time difference at which the acoustic signal reaches two microphones of each Kinect sensor. I also wrote the module which combines data of two Kinect sensors to disambiguate the calculated directions, conducted experiments, and wrote the paper.

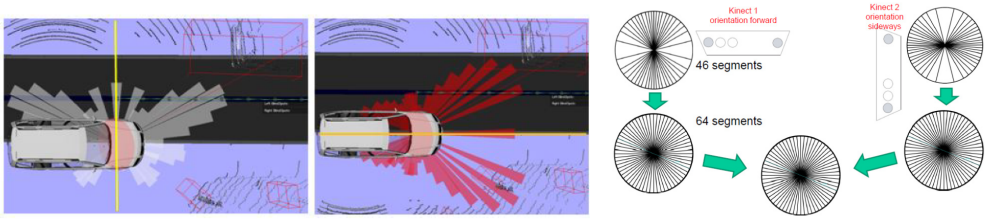



Figure 11: Left and center: Symmetries of angular distribution for front facing Kinect and sideways facing Kinect, respectively. Right: Data fusion scheme.

[9] Stefan Lange, Fritz Ulbrich, **Daniel Goehring**, "Online Vehicle Detection using Deep Neural Networks and Lidar based Preselected Image Patches." In: *2016 IEEE Intelligent Vehicles Symposium, IV 2016, Gothenburg, Sweden, June 19-22, 2016,* pp. 954–959, IEEE, 2016.
DOI: 10.1109/IVS.2016.7535503
URL: https://doi.org/10.1109/IVS.2016.7535503

This work describes one of my first approaches to combine 3D lidar data with CNN-based image classifiers for an autonomous vehicle. To improve the detection accuracy and to reduce the required processing power, lidar data was used to select regions in a 2D image, see Fig. 12. Only the regions, not the complete images were classified by a CNN. The CNN was trained and executed within the Caffe framework, without the need of a GPU. I contributed the paper idea, helped conduct experiments on a content level and as a safety driver. I also wrote considerable amounts of the paper. Much of the work had to be put into the integration of the Caffe software chain into our modular robotics framework (OROCOS).

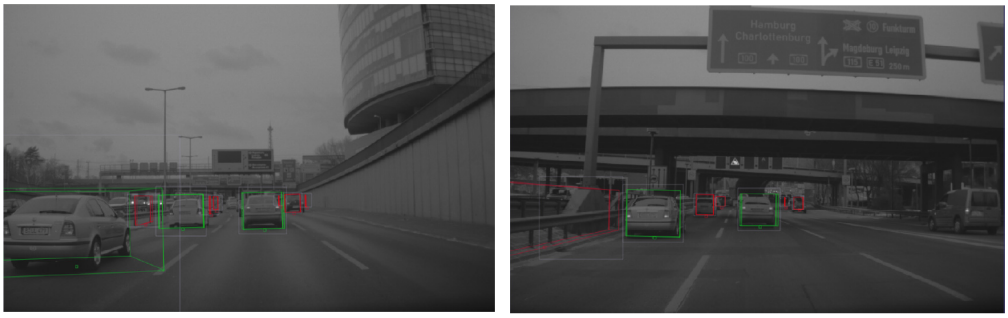


Figure 12: Examples with a good classification precision on the Berlin Autobahn A100. Green boxes in the b/w images are correctly classified vehicles.

Figure 13: Left: The driver is wearing eye-tracking glasses from SMI, a camera in the glasses is perceiving the fiducial markers and localizes the head relative to the car. Right: The depth and image information of the stereo camera is projected as a colored point cloud into the 3D space with respect to the car.

[10] Tobias Langner, Daniel Seifert, Bennet Fischer, **Daniel Goehring**, Tinosch Ganjineh, Raul Rojas, "Traffic Awareness Driver Assistance based on Stereovision, Eye-tracking, and Head-Up Display." In: *2016 IEEE International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016,* pp. 3167–3173, IEEE, 2016.
DOI: 10.1109/ICRA.2016.7487485
URL: https://doi.org/10.1109/ICRA.2016.7487485

This paper combines an eye-tracking device, camera data of an autonomous vehicle, and an onboard display to warn distracted drivers whenever they have missed features of their environment, e.g., looking at a traffic light. I contributed ideas to the design of the system, conducted experiments with the eye-tracking device, and helped test the overall system in the autonomous vehicle and in real traffic, see Fig. 13. In addition, I helped to structure the paper and wrote considerable parts of it.

[11] Robert Spangenberg, **Daniel Goehring**, Raul Rojas, "Pole-Based Localization for Autonomous Vehicles in Urban Scenarios." In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, Daejeon, South Korea, October 9-14, 2016,* pp. 2161–2166, IEEE, 2016.
DOI: 10.1109/IROS.2016.7759339
URL: https://doi.org/10.1109/IROS.2016.7759339

This paper presents a self-localization method for a self-driving vehicle us-

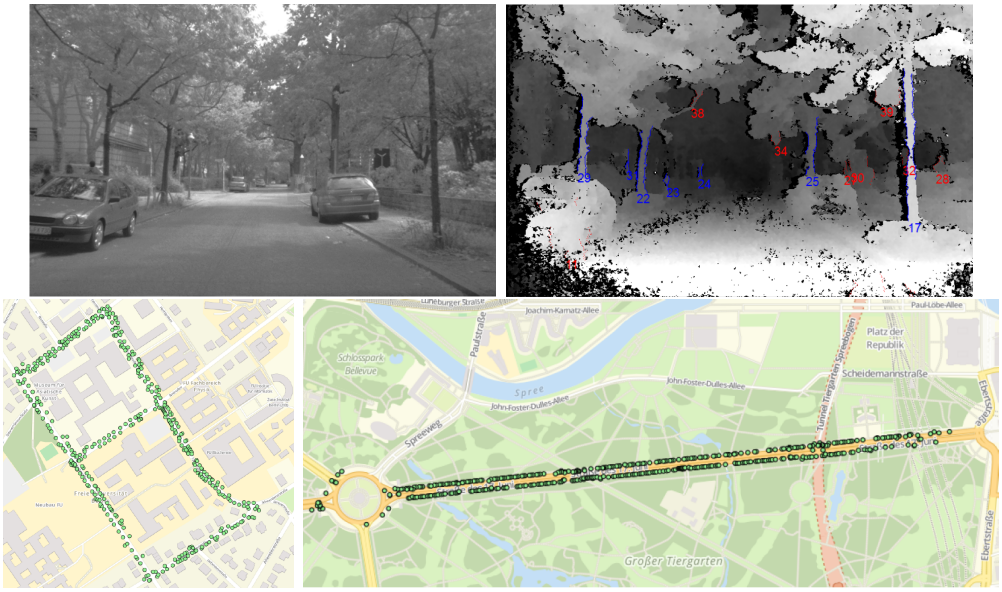



Figure 14: Upper left: Rectified image from the FU campus; Upper right: Depth image with recognized trees; Lower left: Tree map from FU campus, Takustraße; Lower right: Tree map of Straße des 17. Juni.

ing two stereo cameras, odometry data, and a map of pole-like structures. I contributed ideas to the design of the odometry data as well as to the data assignment of detected poles to poles in a map, see Fig. 14. Instead of a greedy matching approach for the sensor data to map data, a more global solution was implemented. I supported the experiments on many occasions as a safety driver, discussed options for the underlying Kalman filter, and wrote parts of the paper. In addition, I created the demo video for the paper which was submitted to the IROS conference.

[12] Fritz Ulbrich, Simon Rotter, **Daniel Goehring**, Raul Rojas, "Extracting Path Graphs from Vehicle Trajectories." In: *2016 IEEE Intelligent Vehicles Symposium, IV 2016, Gothenburg, Sweden, June 19-22, 2016,* pp. 1260–1264, IEEE, 2016. DOI: 10.1109/IVS.2016.7535552
URL: https://doi.org/10.1109/IVS.2016.7535552

In this paper, a lane map is created while observing trajectories of other vehicles. Other traffic participants are tracked via lidar data. The idea of this paper was born after observing people driving on snowy roads, i.e., with

Figure 15: Generated lane trajectories after observing other vehicles.

imperceptible lane markings, as well as in Mexico City on roads without lane markings. It appeared to the authors of this publication that people cluster themselves into instantaneous lanes, depending on the vehicles in front and to the side of them. I contributed to the data recording of real traffic data and with ideas to the mapping process, i.e., how different interrupted traces of observed vehicles can be combined to continuous paths, see Fig. 15.

[13] Daniel Neumann, Tobias Langner, Fritz Ulbrich, Dorothee Spitta, **Daniel Goehring**, "Online Vehicle Detection using Haar-like, LBP and HOG Feature based Image Classifiers with Stereo Vision Preselection." In: *IEEE Intelligent Vehicles Symposium, IV 2017, Los Angeles, CA, USA, June 11-14, 2017,* pp. 773–778, IEEE, 2017. DOI: 10.1109/IVS.2017.7995810
URL: https://doi.org/10.1109/IVS.2017.7995810

Stereo image data are used to detect object regions, which are then classified using different classification methods. I contributed the paper idea and helped design the system architecture. I also helped record traffic data, gave input on how to evaluate the data, and on how to calculate and to visualize the results.

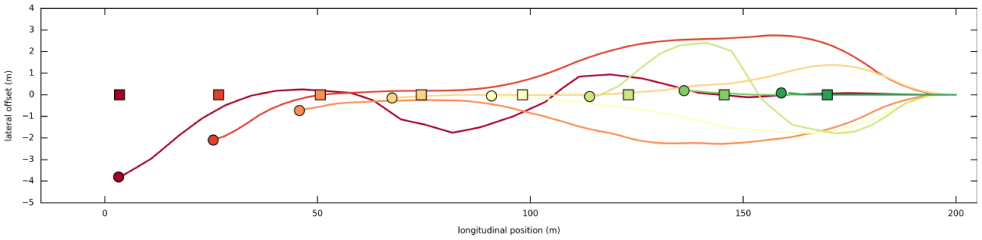



Figure 16: Different planned paths in space over time using the presented timed elastic bands variant.

[14] Fritz Ulbrich, **Daniel Goehring**, Tobias Langner, Zahra Boroujeni, Raul Rojas, "Stable Timed Elastic Bands with Loose Ends." In: *IEEE Intelligent Vehicles Symposium, IV 2017, Los Angeles, CA, USA, June 11-14, 2017,* pp. 186–192, IEEE, 2017. DOI: 10.1109/IVS.2017.7995718
URL: https://doi.org/10.1109/IVS.2017.7995718

This paper presents a planning method for autonomous vehicles using timed elastic bands (TEB). A challenge that autonomous vehicles need to cope with is planning in space with moving obstacles. Generating a plan for a static scenario is not enough since the environment changes over time. Therefore, a path planning algorithm needs to generate a trajectory with respect to a time parameter. Constraints, which represent how fast other vehicles can accelerate or turn have to be included into the TEB-solving algorithm. This work focuses on how the limited perception aspect, i.e., where the free space configuration in a few hundred meters is unclear, can be considered by a TEB-planner, see Fig. 16. I contributed ideas to the different constraints in the TEB-framework, especially with my knowledge about the dynamic and kinematic constraints of our test vehicle MiG, helped refine the approach during real-world tests, discussed the experimental design, and executed self-driving experiments as a safety driver.

[15] Zahra Boroujeni, **Daniel Goehring**, Fritz Ulbrich, Daniel Neumann, Raul Rojas, "Flexible Unit A-star Trajectory Planning for Autonomous Vehicles on Structured Road Maps." In: *2017 IEEE International Conference on Vehicular Electronics and Safety, ICVES 2017, Vienna, Austria, June 27-28, 2017,* pp. 7–12, IEEE, 2017. DOI: 10.1109/ICVES.2017.7991893
URL: https://doi.org/10.1109/ICVES.2017.7991893

An A*-planner is presented in order to generate plans in dynamic traffic

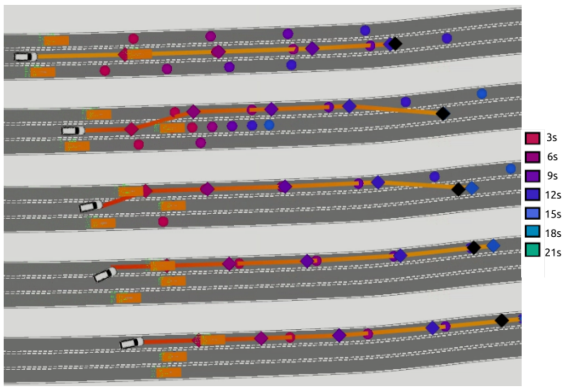



Figure 17: Generated plans, different colors depict different points in time and the corresponding position in space over time.

environments to be able to execute lane changing and overtaking maneuvers. Compared to TEB-planners, the resulting trajectories of the introduced A*-planners are more predictable and stable over time and can be adjusted to the street boundaries more easily. I sketched the main idea of the approach, i.e., how to plan over time, see Fig. 17., how to branch the search tree into combinations of acceleration and lane changing maneuvers, and what metric to use for the cost and heuristic function of the A*-algorithm. In addition, I provided most of the input on how to create the experiments in a simulation and how to visualize the experimental results, which consisted of tracked objects over time and space and the planned trajectories. The kinematic and dynamic model of the simulated vehicles were provided by me, but not the simulation framework. Finally, I contributed to the writing process on the content and language side.

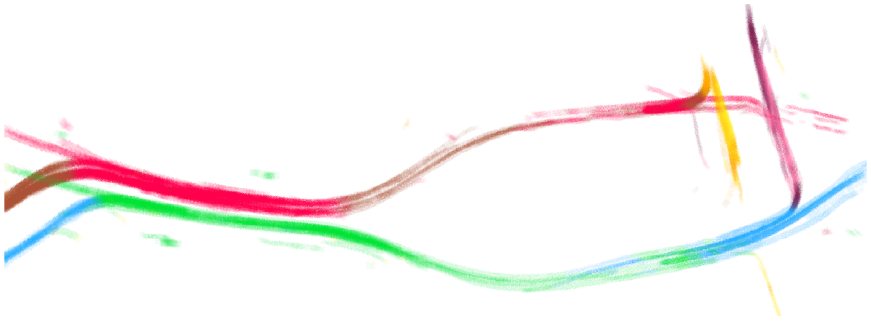


Figure 18: From observed vehicles a 2D orientation histogram (which corresponds to a drive direction histogram) is created. The experiments were executed in a simulation.

[16] Nicolai Steinke, Fritz Ulbrich, **Daniel Goehring**, Raul Rojas, "Traffic Mapping for Autonomous Cars." In: *2018 IEEE Intelligent Vehicles Symposium, IV 2018, Changshu, Suzhou, China, June 26-30, 2018,* pp. 689–694, IEEE, 2018. DOI: 10.1109/IVS.2018.8500601
URL: https://doi.org/10.1109/IVS.2018.8500601

In this work, a grid map is created by observing other vehicles in a simulation, see Fig. 18. This work extends the map creation idea from [12] but uses a 2D grid to represent the drive direction distribution of vehicles w.r.t. a plane. I contributed to the evaluation process, i.e., which experiments to execute and how to to visualize the results.

[17] Zahra Boroujeni, Mostafa Mohammadi, Daniel Neumann, **Daniel Goehring**, Raul Rojas, "Autonomous Car Navigation using Vector Fields." In: *2018 IEEE Intelligent Vehicles Symposium, IV 2018, Changshu, Suzhou, China, June 26-30, 2018,* pp. 794–799, IEEE, 2018.
DOI: 10.1109/IVS.2018.8500446
URL: https://doi.org/10.1109/IVS.2018.8500446

This work describes how vector fields, as shown in Fig. 19, can be used to guide a simulated vehicle or a model car on a track. The motivation for this work came from the question in how far planning and control tasks can be stored in a map, instead of being calculated in each time frame. Therefore, the map has been discretized into a look-up table, where for each position and velocity a force vector is computed, pulling the vehicle into a certain di-

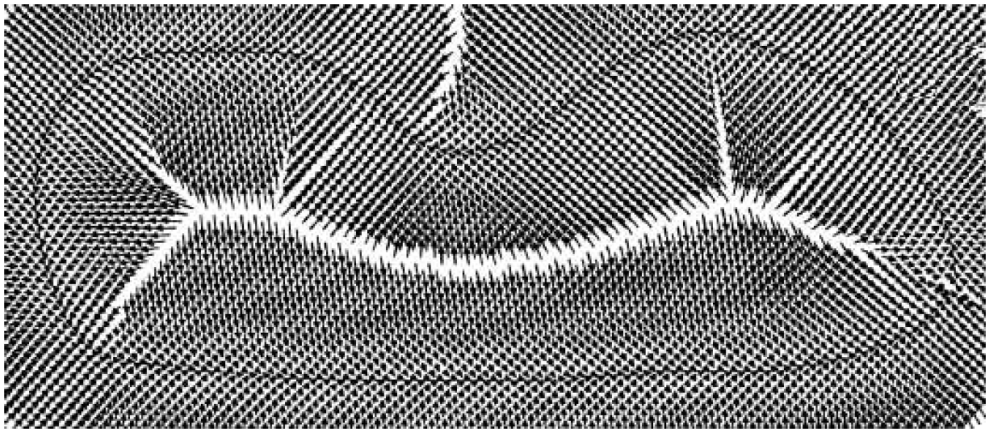


Figure 19: A vector field obtained for a constant velocity of 2 m/s.

rection. I contributed ideas to the experimental evaluation and gave support during the writing process.

[18] Fritz Ulbrich, Tobias Langner, Stephan Sundermann, **Daniel Goehring**, Raul Rojas, "Following Cars With Elastic Bands." In: *2018 IEEE Intelligent Vehicles Symposium, IV 2018, Changshu, Suzhou, China, June 26-30, 2018,* pp. 1529–1536, IEEE, 2018. DOI: 10.1109/IVS.2018.8500481
URL: https://doi.org/10.1109/IVS.2018.8500481

This work continues the research on how TEB-planners can be used to guide a real autonomous vehicle in dynamic traffic scenarios. I supported the design of the different contraints and metrics, e.g., the cost to switch to a virtual lane (virtual, because this approach does not require maps), a suitable following velocity and distance, and how to suppress oscillations. I contributed to conducting the experiments in real world traffic, even though most experiments presented in the paper were performed in a simulator, which I helped design.

Figure 20: Different boxes in a facility which need to be recognized and gripped by a robot.

[19] Christian Poss, Olimjon Ibraginov, Anoshan Indreswaran, Nils Gutsche, Thomas Irrenhauser, Marco Prueglmeier, **Daniel Goehring**, "Application of Open-Source Deep Neural Networks for Object Detection in Industrial Environments." In: *17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018, Orlando, FL, USA, December 17-20, 2018,* pp. 231–236, IEEE, 2018. DOI: 10.1109/ICMLA.2018.00041
URL: https://doi.org/10.1109/ICMLA.2018.00041

A robot within a facility of BMW needs to recognize boxes containing car parts or tools in a cluttered environment using Deep Neural Networks. I supervised the lead author during his external PhD-thesis at BMW AG and discussed his ideas on a number of occasions. This included the analysis of different method candidates, which data to use, and how to evaluate and to present the data gained from experimental results.

There are many similarities between the research field of this work to the research area of [5] and [7] using a PR2 robot. In both research domains, a robot needs to detect objects using 2D and 3D data in order to grasp them. One difference for the industrial robot is that it does not need to distinguish different box categories as long as it is able to grasp each box. On the algorithmic side, the facility robot uses CNNs, as, e.g., from the YOLO-framework, whereas the PR2 is mainly using HOG/SVMs. In order to grasp objects, the PR2 robot is using two mechanical grippers. The industrial robot facilitates a suction device. For both robot types, a suitable placement of the gripping or suction device is essential in order to succeed in the gripping and lifting task.

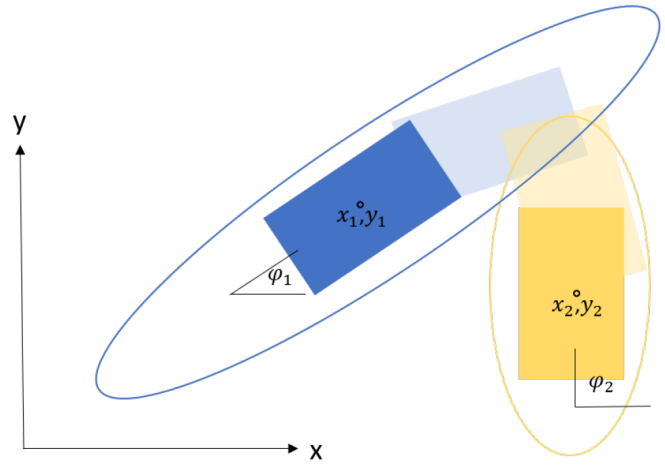



Figure 21: Overlap uncertainty of two oriented rectangles at a point in time.

[20] Andreas Phillipp, **Daniel Goehring**, "Analytic Collision Risk Calculation for Autonomous Vehicle Navigation." In: *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019,* pp. 1744–1750, IEEE, 2019. DOI: 10.1109/ICRA.2019.8793264
URL: https://doi.org/10.1109/ICRA.2019.8793264

This work presents an approach to evaluate the collision risks of a vehicle in traffic scenarios. Since collision checking and avoidance is an import part of the perception and planning system for autonomous driving, this work presents an analytic approach to calculate the probability of a future collision, e.g., as shown in Fig. 21, and extends another already known solution. I have been supervising the lead author during his PhD-thesis and how to make his approach applicable under real-time constraints, how to design the state space of different vehicles, which experiments to conduct, and how to prepare the results for the paper.

Figure 22: Samples of unlabeled data, the first row shows unsorted, the second row clustered samples.

[21] Michaela Steinhoff, **Daniel Göhring** "Pedestrian Head and Body Pose Estimation with CNN in the Context of Automated Driving." In: *Proceedings of the 6th International Conference on Vehicle Technology and Intelligent Transport Systems, VEHITS 2020, Prague, Czech Republic, May 2-4, 2020,* pp. 353–360, SCITEPRESS, 2020. DOI: 10.5220/0009410903530360 URL: https://doi.org/10.5220/0009410903530360

In this work, which can be considered as an experimental approach towards a driver assistant system for an automated vehicle, a camera based head position estimation is described. The motivation of this work is to get a better prediction accuracy of what a pedestrian who stands next to the street is planning to do, which is part of the research area of intention recognition. In this work, training data from pedestrians, as shown in Fig. 22, has been annotated and used to train a CNN. The lead author is an external PhD-student of mine, working at IAV GmbH. I gave support on the system design, provided feedback with respect to the applicability of the approach as well as on the visualization of experimental data.

# 5 List of Co-Authors

The publications of the last section have been created in collaboration with the following researchers (in order of appearance in the publication list). After each name, the corresponding collaborative publications have been listed:

1. Miao Wang [1][2][3]
2. Michael Schnürmacher [1]
3. Tinosch Ganjineh [1][2][10]
4. Arturo Reuschenbach [2]
5. David Latotzki [3]
6. Raul Rojas [3][10][11][12][14][15][16][17][18]
7. Sergion Guadarrama [5]
8. Lorenzo Riano [5]
9. Dave Golland [5]
10. Yangqing Jia [5]
11. Dan Klein [5]
12. Pieter Abbeel [5]
13. Trevor Darrell [5] [6]
14. Judy Hoffman [6]
15. Erik Rodner [6]
16. Kate Saenko [6]
17. Mario Fritz [7]
18. Hamma Tadjine [8]
19. Stefan Lange [9]
20. Fritz Ulbrich [9][12][13][14][15][16][18]
21. Tobias Langner [10][13][14][18]

22. Daniel Seifert [10]
23. Bennet Fischer [10]
24. Robert Spangenberg [11]
25. Simon Rotter [12]
26. Zahra Boroujeni [14][15][17]
27. Daniel Neumann [13][15][17]
28. Dorothee Spitta [13]
29. Nicolai Steinke [16]
30. Mostafa Mohammadi [17]
31. Stephan Sundermann [18]
32. Christian Poss [19]
33. Olimjon Ibraginov [19]
34. Anoshan Indreswaran [19]
35. Nils Gutsche [19]
36. Thomas Irrenhauser [19]
37. Marco Prueglmeier [19]
38. Andreas Philipp [20]
39. Michaela Steinhoff [21]

# Radar/Lidar Sensor Fusion for Car-Following on Highways

Daniel Göhring, Miao Wang, Michael Schnürmacher, Tinosch Ganjineh



DOI: 10.1109/ICARA.2011.6144918
URL: https://doi.org/10.1109/ICARA.2011.6144918

# iDriver - A Human Machine Interface for Autonomous Cars

Arturo Reuschenbach, Miao Wang, Tinosch Ganjineh, Daniel Göhring



DOI: 10.1109/ITNG.2011.83
URL: https://doi.org/10.1109/ITNG.2011.83

# Semi-Autonomous Car Control Using Brain Computer Interfaces

Daniel Göhring, David Latotzky, Miao Wang, Raúl Rojas



DOI: 10.1007/978-3-642-33932-5_37
URL: https://doi.org/10.1007/978-3-642-33932-5_37

# FREIE UNIVERSITÄT BERLIN

Controller Architecture for the Autonomous

Cars: MadeInGermany and e-Instein

Daniel Göhring

B-12-06
November 2012


FACHBEREICH MATHEMATIK UND INFORMATIK
SERIE B · INFORMATIK

**Abstract.** In this paper a realtime controller architecture for our autonomous cars, a highly equipped conventional car and an electric vehicle is presented. The key aspects for controlling a car are stability, accuracy and smoothness, which constrain design critereas of all kinds on controller components. This report presents solutions for a variety of controller components, and aspects of a controller implementation of an autonomous car. The algorithms described proved their applicability in dense urban Berlin traffic as well as on the Berlin Autobahn.

## 1 Car Introduction

In our project AutoNOMOS we use three different platforms. The Spirit Of Berlin, a Dodge Caravan marks the first autonomous car which was programmed at the Artificial Intelligence Group of Prof. Raúl Rojas at the Freie Universität Berlin since 2006. This car successfully participated in the Darpa Urban Challenge and qualified for the semi finals. Since 2010 we are working on a Volkswagen Passat, called "MadeInGermany" (MiG). MiG is a very modern car with modern sensors and a drive by wire architecture where we have access to throttle and brake actorics via CAN bus. Since 2011 we are also working on an electric vehicle, a Mitsubishi iMiev, called "e-Instein".


**Fig. 1.** Autonomous Cars (f.l.t.r.): Spirit of Berlin, e-Instein, MadeInGermany

## 2 Controller Specification

At the beginning of the development of a controller, one has to think about the environment the controller should work in and about the constraints the

controller has to satisfy. For our autonomous cars, the following specification aspects were defined:

- Safety: The controller must be fail safe on a high level with regards to software or hardware failures. Further, the controller must never perform maneuvers, which are not safe with respect to the physical limitations of the car or the environment. It must never operate beyond the mechanical limitations of the actoric. To give an example, a controller which operates on a flat street can to make other assumptions about grip in comparison to a controller which has to operate in off-road scenarios. Regarding actoric, a quickly responding and precisely adjustable actoric allows more aggressive maneuvers than the one reacting with huge time delays. As a rule of thumb translational and centrifugal forces are limited to 40 percent of physically possible values.
- Accuracy: Reaching desired control values quickly is most important for a safe driving of the car. Imprecise or slow controllers usually lead to oscillations within the upper controller level or behavior layer. Oscillations are one reason for an uncomfortable driving experience. For MadeInGermany, the goal was to have a lateral error to the planned trajectory of less than 10 cm at 100 km/h and a velocity error of less than 0.5 km/h.
- Comfort: Besides limiting the amount of lateral and longitudinal forces to a level which feels comfortable to a modest driver, another important aspect is to limit acceleration changes, i.e., the function of the acceleration vector over time must be continuous. Changes within the acceleration vector must remain small for a comfortable feeling of drive.

Some of these aspects contradict each other, e.g., safety and comfort. A safe controller might try to apply appropriate maneuvers as fast as possible but this can be uncomfortable, because humans prefer slow changes of accelerations. Further, comfortable maneuvers, e.g., braking late, can sometimes result in dangerous situations. The same holds for accuracy. A controller which tries to be too precise can result in an uncomfortable feeling of drive.

## 3 Controller and Planning Modules Overview

This section introduces the module chain and the corresponding module function description. Modules will be distinguished by their level of abstraction in terms of the platform on which they run. An overview of the given modules, their input parameters and their implementation name within the OROCOS-Framework are given in Fig. 2. The code within the AutoNOMOS project was designed to serve as a middle ware for a variety of autonomous platforms. Therefore, reusability of code was a very important aspect. Only the low controller modules are platform specific, high level behavior modules are independent of the platform they are executed on.

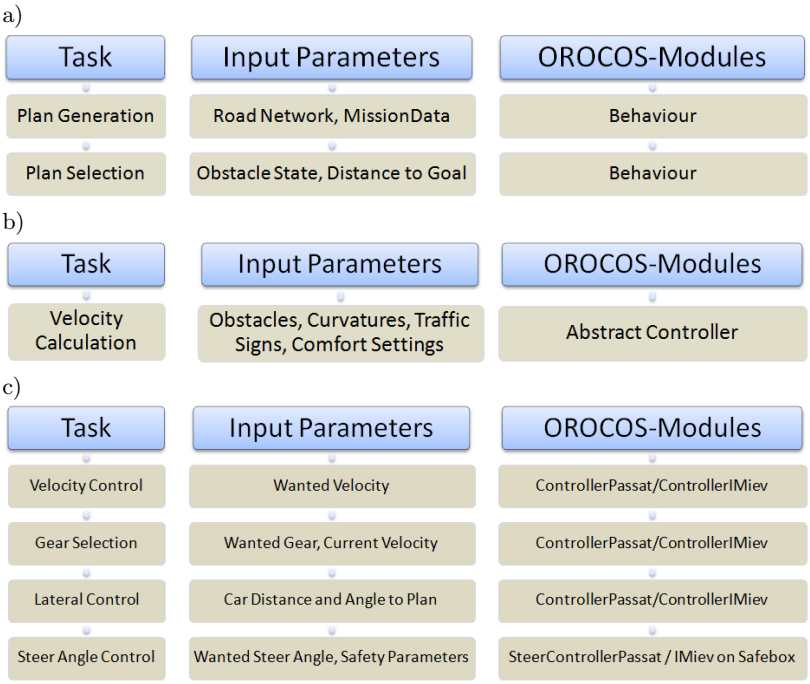



**Fig. 2.** Controller and behavior models. a) For these models, code and data are vehicle independent. b) Code is vehicle independent but not data independent. c) These modules are designed for a certain vehicle (VW Passat (MiG)).

## 4 Steer Control Chain (MiG)

The complete steer control chain includes the module for the behavior, which generates a planned trajectory. The behavior uses a road network definition file which includes the streets, and a mission file which defines the checkpoints to be visited while traveling to a certain destination. Just to mention, the behavior generates in each time frame a new, updated plan, mainly it does not remember old trajectories. Now, the generated planned trajectory is fed into a lateral controller. The task of the lateral controller is to compare the current position of the car with the alignment of the planned trajectory. Depending on other aspects as comfort and the current velocity, the lateral controller generates a wanted (desired) steering wheel angle. It was assumed there is a linear dependency between steering wheel angle and the angle of the front wheel, even though it is known that this assumption is just a rough approximation. However, for small angles this assumption provides a good estimate and bigger steering angles are executed at low velocities only, where the exact execution of the generated wanted steering angle is not as important as it is for high velocities. Future work remains to approximate the real function between the steering wheel angle and

front wheel angle, and also the function between velocity, steering wheel angle and wheel slip. Bringing back the focus to the control loop, the desired steering wheel angle is fed into the steer controller, who generates a desired momentum for the steer motor. The control loop is depicted in Fig. 3. In the next sections the steer angle controller and later the lateral controller are described "bottom up".

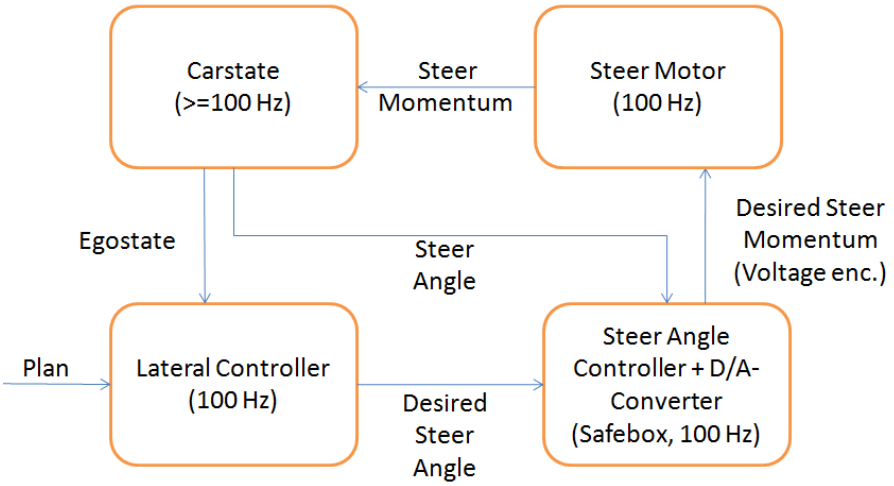



**Fig. 3.** Steer controller chain for MadeInGermany. The lateral controller generates a desired steer angle, feeds it to the steering wheel controller, which generates a desired momentum, encoded within a voltage.

### 4.1 Steer Angle Controller (MiG)

The steer angle controller gets a desired steering wheel angle as an input and generates a torque value. To avoid oscillations on the lateral controller, which is executed right before the steer angle controller within the control loop, the steer angle controller shall generate torque values to minimize the difference between desired steering wheel angle and current steering wheel angle as quick as possible. The steer motor, manufactured from Maccon, has no built in angle sensor. Thus, for feedback, we use an external angle sensor, which is built in stock in the MiG-Passat and sends its data via CAN bus with an update frequency of 100 Hz to the controller gateways. 100 Hz is also our control frequency. The sensory data are accumulated in the car state. This architecture is not optimal with regards to system feedback. Further, every system has a certain reaction time. To handle these imminent delays, a predictive controller was the solution of choice.

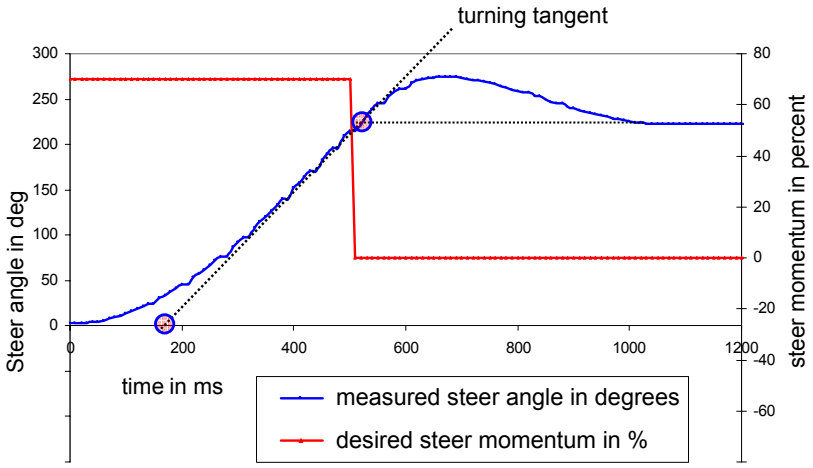



**Fig. 4.** Impulse response test for the steer motor.

**Predictive control.** To get a rough estimate about the delay within the control loop, we sent a torque impulse to the steer motor and measured the resulting angle change over time. The response curve was plotted in Fig. 4. The measured response time was about 150 ms. One can see the steering wheel accelerating, the maximum steer angle velocity was measured at $600^o/s$.

To overcome the system delay, a predictive PID-controller was used. The controller does not use the current steering wheel angle $\theta_{sw}$ as feedback, but a prediction value of what the steering wheel angle will be $\theta'_{sw}$, assuming a constant change of the steering wheel angle within the next $t = 0.12s$ - which is the estimated system delay - while moving constantly with the currently measured steering wheel angular velocity $\omega_{sw}$. We get:

$$\theta'_{sw} = \theta_{sw} + t \cdot \omega_{sw} \tag{1}$$

Now, where we calculated the values for the control loop, we feed them into the PID-rule. We use a classic PID-controller with a limited maximum integral for the integral term. This is important to avoid big overshoots.

**Velocity dependent momentum adaptation.** To accommodate fact that the necessary steer momentum varies with the car's velocity, a scale factor was introduced, which is multiplied with the controller output, c.f., Fig. 5. When the car is standing, one has to use a higher force to turn the front wheels, because the tire has a high friction with the concrete. When moving slow, the steering wheel can be turned quite easily. The higher the velocities are, the harder it gets to turn the steering wheel in. There are several causes for this: At first, the centrifugal

force gets bigger, pushing the front wheels back to their neutral position. Second, the rotational impulse gets bigger, to change the impulse vector, a higher torque is necessary. Third, car manufacturers limit the support of the steering servo motor to get the car more stable at higher velocities - our steer motor relies on the support of the steering servo motor. Experiments showed that the controller, in its precise mode, i.e., with maximum scale factors reaches a desired angle within one second with a precision of less than 0.5 degrees, after 1.5 seconds the accuracy is better than 0.15 degrees.

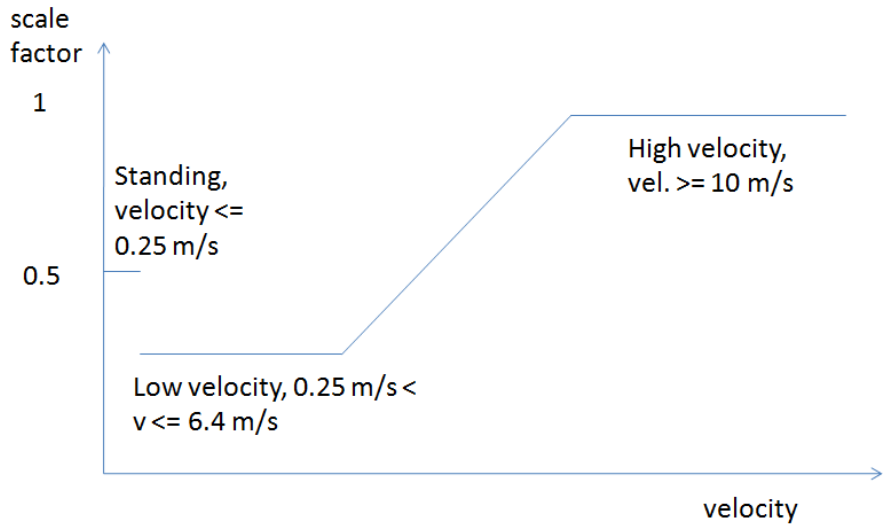



**Fig. 5.** Steer momentum scale factor function, modeled as a piecewise linear function. Medium momentum is necessary for a standing car, small momentum for low velocities and higher momentum for higher velocities - especially while steering away from the zero angle.

**Steer Angle Limiter.** To avoid the execution of wanted steering wheel angles, which might pose a threat to the car's safety and stability, a steer angle limiter was implemented, which calculates the maximum allowed steering angle for a given velocity under the assumption of a maximum allowed centrifugal force, see steer angle limiter code example. The steering wheel controller, together with the steer angle limiter runs on the safe box hardware, that's why it cannot be negatively affected by the other modules, running on the main computer.

```
/******* Steer Angle Limiter *******/
float getMaxSteeringWheelAngle(tolerance) {
mSinMaxFWAngle = fabs(WHEELBASE * MAXLATERALACCELERATION /
(mAvgFrontWheelSpeeds * mAvgFrontWheelSpeeds));
if ((mAvgFrontWheelSpeeds != 0.0) && (mSinMaxFWAngle < 1)) {
    maxWheelAngleAtGivenSpeed = asin(mSinMaxFWAngle) * 180 / Pi;
    return fabs((maxWheelAngleAtGivenSpeed /
    MAXWHEELANGLE * MAXSTEERINGWHEELANGLE) + tolerance);
}
return MAXSTEERINGWHEELANGLE; //in degrees
}
```

## 5 Steer Controller Chain (e-Instein)

The main difference of the e-Instein steer controller chain to the MiG controller chain is that the output voltage of the safe box encodes a steer angle for the Paravan module. The lateral controller generates a desired steer angle, feeds it to the steering wheel controller, which generates a calibrated angle value for the safe box. The safe box generates a voltage, encoding the steering wheel angle non-linear and feeds it to the Paravan system, see Fig. 7. The different modules are described in detail in the following sections.

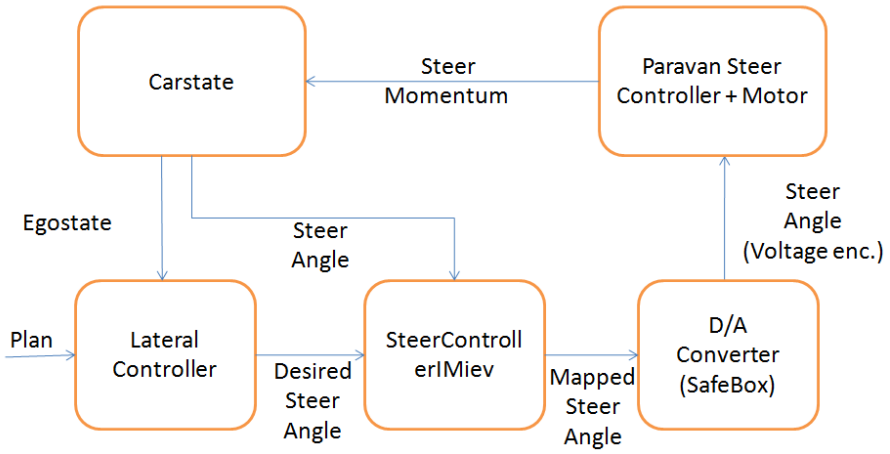



**Fig. 6.** Steer controller chain for e-Instein.

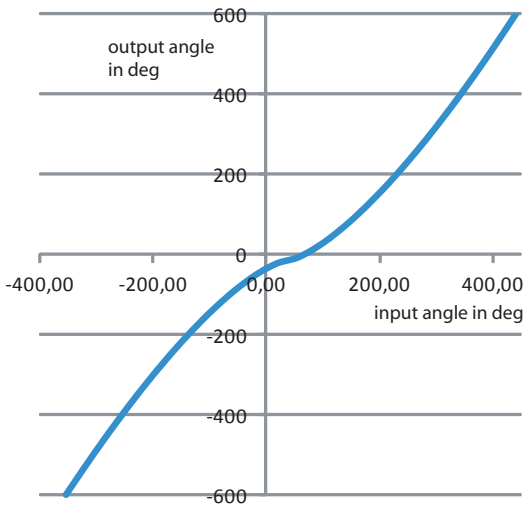



**Fig. 7.** An input angle is mapped to an output angle. The voltage at the paravan is not linearly dependend on the resulting steer angle. The function, of the form: $a \cdot (x-b)^c + d$ does not pass the origin of the coordinate system.

### 5.1 Steer Angle Controller (e-Instein)

Originally the steer controller for the e-Instein was assumed to be unnecessary, because the lateral controller generates a desired steering wheel angle and the Paravan module takes a voltage, encoding a steering wheel angle as an input. Unfortunately the mapping between both is not linear and we were missing a specification. Therefore, the steering wheel angle implements a linearization function, c.f., Fig. 7, to map the desired angle to a linearized angle, which feeds the Paravan system, c.f., Equation 2. Parameters $a, b, c, d$ had to be chosen wisely to approximate the experimentally derived function.

$$linearizedAngle = \text{sgn}(desiredAngle - b) \cdot a \cdot (|desiredAngle - b|)^c + d \quad (2)$$

This function, unfortunately, gives only a rough estimate for the control voltage to reach a certain wanted angle. The Paravan system implements a dead zone, thus, it depends from which direction (left or right) the steering angle turns into a desired angle. This prediction function *pred* is sufficient to reach the steering wheel angle with an accuracy of +/- 5 degrees. To reduce the remaining difference between desired and current steering wheel angle below 5 degrees, a PID controller, somewhat similar to the one in MiG is used. The PID-controller can only compensate differences smaller than 10 degrees to avoid oscillations. The steering motor in the e-Instein is very slow (approx. 360 degrees per second), a non limited integral in the PID controller could result in harsh overshoots. Combining the prediction function and the PID-controller, we get the summarized control Equation 3, resulting in a linearized Angle, which can be linearly

encoded to a voltage, which again feeds the Paravan box.

$$linearizedAngle = \mathbf{pred}(desiredAngle) + \mathbf{pid}(desiredAngle, currentAngle) \tag{3}$$

The combined control form allow to reach a desired steering wheel angle with an accuracy below 1 degree, within one second - assumed the desired steering wheel angle is not too far away from the current one.

### 5.2 Lateral Controller (MiG and e-Instein)

The task of the lateral controller is to generate a desired steer angle to stabilize the car on the planned trajectory. Thereby the car must not swing left and right of the trajectory. To fulfill the control task a velocity adaptive PD controller was implemented.

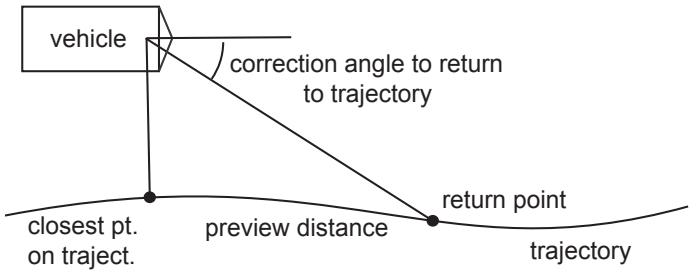



**Fig. 8.** Lateral controller sketch, showing closest point on trajectory, return point and correction angle.

**Velocity adaptive PD-Controller (MiG and e-Instein)** The velocity adaptive PD controller uses two weight sets, one weight set for low velocities and one set for high velocities. For medium velocities in between, the two weight sets are linearly interpolated. The resulting function is somewhat similar to the output scale function of the steering wheel controller of MadeInGermany, see Fig. 5. The control task is visualized in Fig. 8. Control input values are the current car orientation and the angle between the car's forward looking direction and the direction from the car's front axis' middle point to the return point of the trajectory, c.f., Fig. 8. The return point on the plan is calculated velocity dependent. The faster the car goes, the more distant the point is. This is useful to stabilize the car for higher velocities and to be able to perform sharp turns while going slow. Still, this approach tends to short cut curves by a small amount. The distance of the return point to the closest point on the trajectory (returnDist) is calculated as shown in Equation 4.

$$returnDist = \max(staticDist, velocity \cdot velocityScale) \tag{4}$$

The static distance value makes sure that the distance to the return point never gets smaller than a certain value, also when the car is standing. It was set to 2 meters in praxis, the scale factor was set to 1 s, which means that the return point is usually reached within one second, if the plan would be kept. In praxis, the car converges to the plan, only, because the desired angle is calculated each time again, converging to zero but never reaching zero.

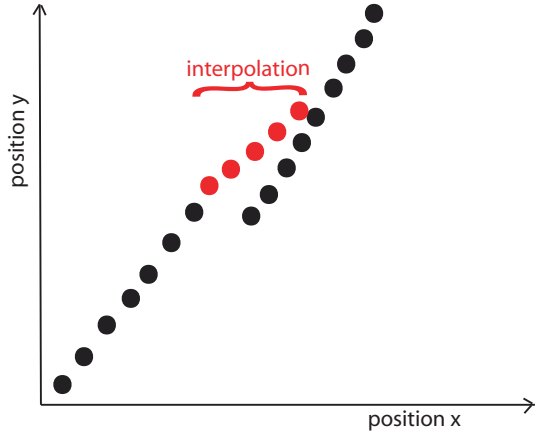



**Fig. 9.** Ego state over time from bottom left to top right. Suddenly an ego state jump occurred, was recognized by the controller. The jump in interpolated within a certain time, a useful value was $t = 2s$.

**EgoState jump filtering (MiG).** A second feature of the controller is to memorize the last egostate. Using a prediction, it can detect differences between the expected egostate in the next time step and the current egostate. If the projected difference to the lateral direction of the car is bigger than a certain amount, e.g., 0.1 meters, it is considered to be a critical ego state jump. The two dimensional egostate difference is substracted from the current egostate and interpolated to zero within two seconds. If within these two seconds another jump is detected, an accumulated difference is calculated and interpolation starts from new.

The ego state jumps are supposed to have different causes. Loss of UMTS connection result in loss of correction data and in ego state jumps. Multi path connections to localization satellites could also be a reason.

**Car angle to IMU angle online calibration.** A further feature is the online calibration of the supposed car orientation angle to the Inertial Measurement Unit (IMU) angle of the Applanix GPS. The Applanix calibrates its IMU at

every restart. Thus, the difference angle between IMU and car forward direction changes every time about 0.05 degrees, in worst case scenarios, whenever the calibration routine was ignored up to 0.5 degrees. At velocities of 100 km/h, 0.5 degrees wrongly calibrated IMU can result in a trajectory offset of 0.5 - 1 meter. Therefore a second calibration routine was implemented in the controller. When going almost straight and going faster than 60 km/h, the controller integrates the error to the trajectory over a certain time span, a useful value was $t = 4s$. If the integral was significantly below or beyond zero, an small positive or negative offset value is added to the car orientation angle. Thus, after ten iterations, i.e., within a minute, the IMU offset could be successfully neutralized.

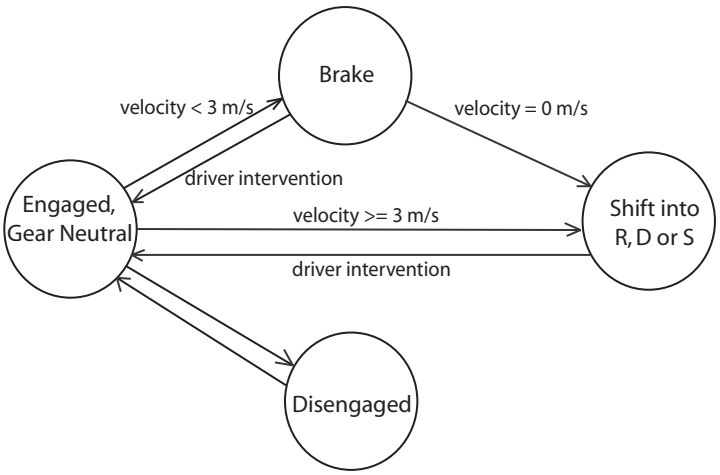



**Fig. 10.** Finite state machine of the gear shifting routine, enabling the driver to give the control back when going fast enough, or braking and shifting whenever going slow or standing. The disengage state can be reached from all states (left out in the figure).

## 6 Gear Selection (MiG and e-Instein)

The gear selection routine has to access the gear over CAN bus, and, if necessary send brake commands. The gear selection routine must fulfill at least two tasks: shift into drive whenever the car is engaged, i.e., made ready for autonomous drive. Initially in N, the brake must be pressed with 10 bar, then the gear can be set to drive, reverse or sport. Another task is to give back the control to the car, whenever a driver intervention occurred, which means, that the driver intervened by pressing throttle or brake (MiG only). After releasing brake or throttle, the computer should regain control over the car without stopping. Luckily, the gear allows shifts from neutral to drive, whenever the car is faster than 10 km/h or

approx. 3 meters per second. The finite state machine, depicted in Fig. 10 gives an idea about the functionality of the gear setting routine.

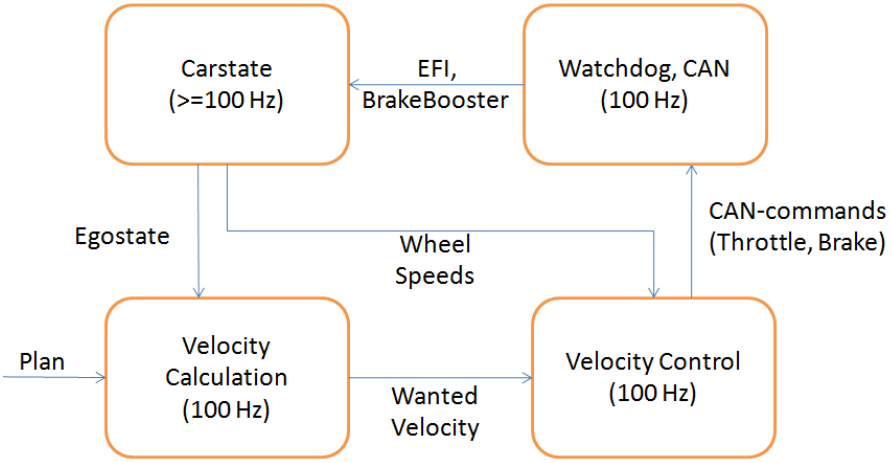



**Fig. 11.** Velocity control chain: Given a plan, a desired velocity is calculated by the velocity calculation module and fed into the velocity controller. Feedback is given by the ego state and the wheel speeds.

## 7 Velocity Control Chain (MiG)

The velocity control chain starts again with a planned trajectory from the behavior module. The velocity planner calculates a wanted/desired velocity on this trajectory. This wanted velocity is handed to the velocity controller, which generates throttle and brake commands, which are fed through the cargate and watchdog to the CAN bus of the MiG-Passat. The different modules are described in detail within the next sections.

### 7.1 Velocity Control (MiG and e-Instein)

Velocity control is executed by a PI-controller, running with a 100 Hz. Input values are a wanted velocity from the velocity calculator and a current velocity, coming via CAN bus and stored in the car state.

**Brake and throttle controller. (MiG)** The PI-controller compares the wanted velocity to the current velocity. Also here the integral is limited to a certain value. The output of the controller is mapped to throttle or brake. Negative outputs are mapped to brake commands and positive outputs are mapped to throttle. The

most important aspect here lies in the fact, that throttle and brake commands have different effects, i.e., 50 percent of brake results in much higher acceleration amounts than 50 percent of throttle. That is why positive outputs of the controller were weighted with a throttleConstant, which was set to 4.

**Throttle and brake limitations. (MiG)** For a comfortable ride, the brake commands were limited to 32 percent of their maximum value, which can result in up to $6ms^{-2}$ deceleration. This was especially important for testing the obstacle tracker. To give an example, wrongly classified obstacles could have a devastating effect when going with a 100 km/h on the highway, if they resulted in a full brake, endangering especially the following cars. Also the throttle commands were limited to 50 percent of their maximum during normal velocities. When starting to move, this value was limited to 35 percent and linearly interpolated to 50 percent at a velocity of 5 meters per second.

For a racetrack scenario, much higher thresholds were applied and can be activated through a flag. In this case, the steering angle limiter from Section 4.1 must be deactivated on the safe box as well.

**Dynamic Handbrake. (MiG and e-Instein)** Stopping a car needs special treatment. A car with an automatic gear tends to start rolling, even without any throttle or brake commands. The integral part of the PI-controller must have an integral weight together with a maximum integral value, big enough to stop the car. This alone is no difference to any other velocity to control. But if the desired velocity is set to zero, the controller tends to convergence to the wanted velocity or it swings around the wanted velocity, resulting in a stop and go. Thus, without special treatment, it can take several seconds to reach a full stop with the car. Further, if pressing the brake not hard enough, the Passat car (MiG) will not fully open the clutch of its direct shift gear, resulting in a semi closed clutch, which again causes severe damage to the clutch over time. A nice solution is to add an increasing amount of brake pressure over time, whenever the wanted velocity is set to zero, combined with the fact that the car is already slower than 4 meters per second. If the car is actually standing, a further fixed amount of brake pressure is added to the brake, to make sure, the clutch is fully open. This "dynamic handbrake" is also useful when stopping at a steep mountain to make sure the car will stop. In experiments the controller proved to be able to keep all velocities up to 100 km/h with an accuracy of 0.5 km/h.

### 7.2 Velocity Calculation (MiG and e-Instein)

The velocity calculation module is independent of the car it is running on. Important parameters for the velocity calculation are maximum allowed centrifugal accelerations and brake accelerations. These values are referred to as "comfort settings". The velocity calculation is generated instantly and recalculated each time step again. There are different aspects which have an effect on the currently desired velocity: static obstacles or traffic signs/lights on the planned trajectory,

dynamic obstacles on the trajectory, curves in front with a certain curvature and at a certain distance, , and of course the officially allowed velocity. All aspects on the trajectory result in a maximum velocity at which they can be approached currently, e.g., a curve in a distance of 5 meters results in a lower currently desired velocity than the same curve in 50 meters distance. For reasons of safety, from all desired velocities the smallest one is returned to the velocity controller.

The following sections will give a glimpse about the different calculations, which all use the same core function.

**Velocity calculation for static obstacles.** A first assumption when braking towards a static obstacle was, that the braking acceleration $a$ shall be constant over the whole time. Let us assume, a static obstacle was detected in a certain distance $\Delta s$. To brake down with a constant acceleration $a$ (brake accelerations are negative) and to stop right in front of the obstacle, the current desired velocity $v_d$ is calculated as follows:

$$v_d = \sqrt{-2a\Delta s} \tag{5}$$

$$v_d = (-2a \cdot (\textbf{obstaclePosition} - \textbf{carPosition}))^{0.5} \tag{6}$$

$$v_d = (-2a \cdot (\textbf{obstaclePosition} - \textbf{carPosition}))^{b} \tag{7}$$

In Fig. 12 a desired velocity over distance to obstacle function is shown for a constant example brake acceleration. For a generalization, parameter $b$ was introduced and is, for the case of constant accelerations, set to 0.5. Acceleration parameter $a$ was set to -1.5.

In practice it showed that static accelerations do not fit the human perception of comfort. Humans tend to accept higher accelerations at high velocities, at low velocities the same accelerations feel uncomfortable. Therefore Equation 5 was changed to allow higher accelerations at higher velocities. This can be achieved through changing the values of parameter $a$ and $b$, parameter $b$ defines, how constant the acceleration is over time, $b = 0.5$ means constant acceleration. Through experiments was found, that the current function with $a = -0.65; b = 0.57$ leads to much better results regarding a higher acceleration at higher velocities and smaller accelerations while stopping:

$$v_d = (-2 \cdot (-0.65) \cdot (\underbrace{\textbf{obstaclePosition} - \textbf{carPosition}}_{\Delta s}))^{0.57} \tag{8}$$

The function is plotted in Fig. 12. Both functions result in the same desired velocity of approx. 100 km/h for a distance of 300 meters, or, with other words, the planned braking distance for a velocity of 100 km/h is the same, when approaching a stop sign. Not to increase the stopping distance for higher velocities was an important aspect within the design process, because the sensors in the car have a limited distance at which they can detect obstacles. In case of an unexpected obstacle, the planned velocity will instantaneously jump to a small value, resulting in a high braking pressure.

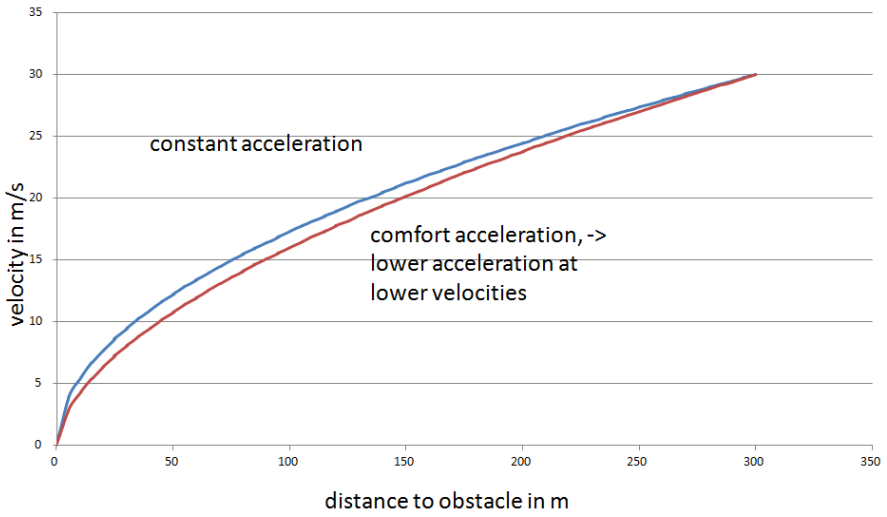



**Fig. 12.** Desired velocity function over distances to a static object. Blue curve depicts the function with a constant brake acceleration, red curve for a variable brake function.

**Stopping for traffic lights.** One exception of static obstacle are traffic lights. If turned red, they are treated as stop signs, when green, they can be just ignored. If a traffic light turns from green to yellow, it depends on the proximity of the car to the traffic light, if a braking maneuver is executed or not. If the car is closer than 42 percent of the necessary distance to brake down comfortably, which can be calculated by another form of Equation 8, where $\Delta s$ is isolated, the car just passes the traffic light. This 42 percent point is called the "point of no return".

**Velocity calculation for dynamic obstacles.** The introduced function can be extended to braking down to dynamic obstacles easily. Therefore, an accurate estimate of the obstacle's velocity is necessary. Positive velocities mark an object which is moving away from the car. Lateral velocities are not taken into account. The desired velocity Equation 8 can be extended as follows, when $v_o$ marks the velocity of the obstacle, $s_c$ and $s_o$ stand for the position of the car and of the obstacle, respectively:

$$v_d = (-2 \cdot a \cdot (\mathbf{s_o} - \mathbf{s_c} + (\mathbf{v_o}^{(1/b)})))^b \tag{9}$$

$$v_d = (-2 \cdot a \cdot (\underbrace{\mathbf{s_o} - \mathbf{s_c}}_{\Delta s} - \mathbf{v_o} + (\mathbf{v_o}^{(1/b)})))^b \tag{10}$$

The first equation assumes that the car will accelerate/decelerate to the obstacles velocity and approach the obstacle with zero distance. The velocity over distance to the obstacle function is depicted in Fig. 13. The second Equation keeps a safety distance depending on the obstacle's velocity. With the second equation, a very

comfortable car following behavior could be implemented, which worked both at high autobahn velocities as on inner city traffic.

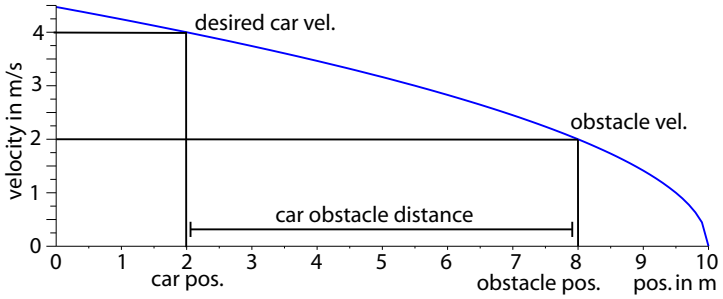



**Fig. 13.** The velocity over distance to an obstacle function, as shown in Eq. 9.

**Velocity calculation in curves.** Velocity calculation in front of curves is somewhat similar to braking in front of multiple moving obstacles. At first the planned trajectory is sampled in different distance steps, as Fig. 14 shows. For each point of the sample set and on the trajectory, the curvature $c$ is calculated. For each given curvature $c \neq 0$, which is the reciprocal of the curve radius $c = 1/r$, given a maximum allowed centrifugal force $F_z$, we get a maximum allowed velocity $v$:

$$F_z = v^2/r \tag{11}$$

$$F_z = v^2 \cdot c \tag{12}$$

$$v = \sqrt{F_z/c} \tag{13}$$

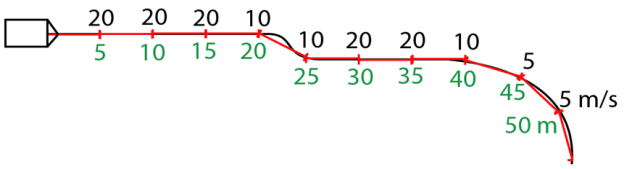



**Fig. 14.** Above the trajectory the maximum allowed velocity are shown; below, the distance of the car to that point is shown.

Now, with a maximum velocity $v_p$ and a distance $s_p$ for each of the sampled points on the trajectory, the first variant of Equation 9 can be applied, assuming

that $v_p$ is the velocity of the obstacle ($s_o$) and $s_p$ the distance of of the obstacle ($\Delta s$). After doing this for all points on the sampled trajectory, we take the smallest velocity as the wanted velocity for curvatures.

### 7.3 Combination of forces using Kamm's circle (MiG and e-Instein)

To avoid that the maximum lateral and longitudinal forces occur together, a second variant for the velocity calculation in front of curves can be applied. Here we take advantage of the shape of the "Kamm's Circle". In our case, both maximum forces are modeled as an ellipse, different shapes are also possible. For the most distant sample point on the trajectory the curvature and the maximum allowed velocity are calculated. Now, this velocity is propagated stepwise back to the car, but a new constraint here is, that in case of maximum lateral acceleration, no longitudinal acceleration must occur in the time step before. While going step by step back to the car, the propagated velocity it compared to the maximum allowed velocity in the next sample point and the minimum of the current and the propagated velocity is taken. The advantage of this method is, that both forces, lateral and longitudinal must not occur in arbitrary combinations, making the driving experience more comfortable.

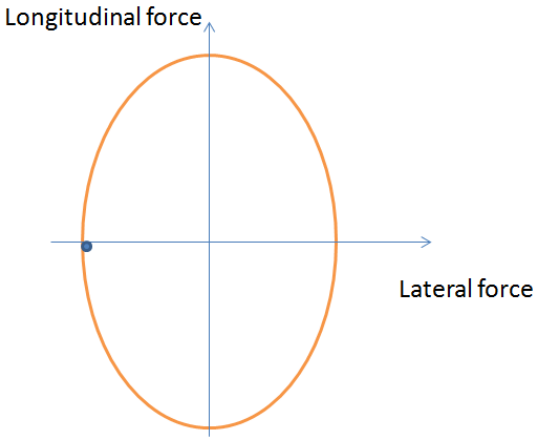



**Fig. 15.** Example form of a Kamm's Circle.

## 8 Derivation of a Car Model (MiG)

A precise car model is important for a predictive control and for a good controller simulation. In this section, a short overview of the steps taken to acquire an accurate model for MadeInGermany are presented.

### 8.1 Derivation of a Car Model

A car model is crucial for a strong simulation of the car's dynamics and to establish an accurate controller. As a model for position and orientation prediction over time an Ackermann drive was assumed for our car as long as centrifugal forces and accelerations were below $5ms^{-2}$. As a further abstraction a bicycle model for the car's dynamic was used, which consists of only one front and one left wheel in the middle of each axis. Given an shaft distance $L$, a velocity $v$ at the front middle axis point and a steering angle $\alpha$ of the front wheel, the change of the car's orientation angle $\omega$ can be easily calculated:

$$\omega = \frac{v(t)\sin(\alpha)}{L} \tag{14}$$

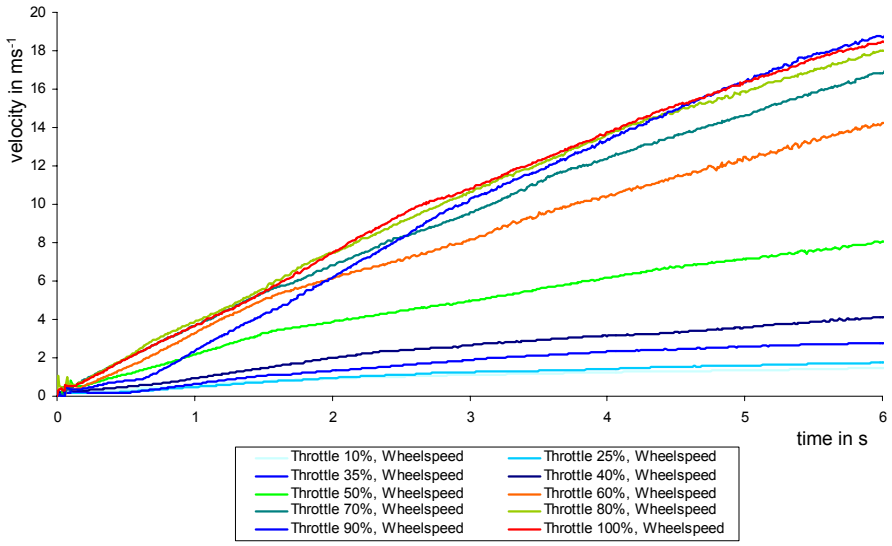



**Fig. 16.** Velocity of MiG over time for different throttle values.

### 8.2 Modeling Acceleration and Braking

For an accurate control of a wanted speed it is useful to have a model of the acceleration and braking behavior of the car, given the gear is set automatically from the gearbox. The desired function has as arguments a current velocity $v_{t-1}$, a time duration $t$, a current throttle or brake command $u_0$ and returns a new velocity $v_t$.

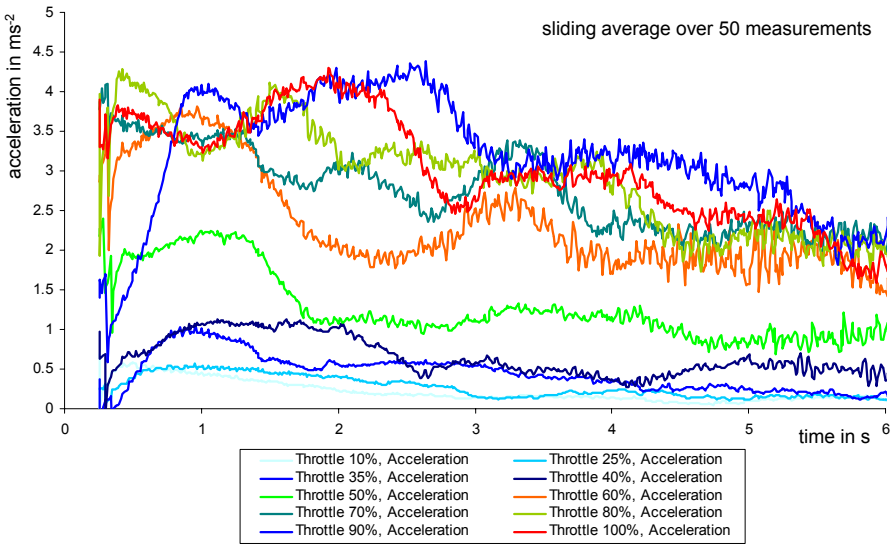



**Fig. 17.** Acceleration of MiG over time for different throttle values.

$$v_t = f(v_{t-1}, u, t) \tag{15}$$

For reasons of simplicity, the acceleration, including braking experiments were started with a standing car, then the car accelerated for various seconds, but not faster than 80 km/h. The braking experiments were performed at 30 and 50 km/h.

The velocity was provided by the four velocity sensors in the wheel.

**Throttle model** First, we were interested in the car's acceleration properties when using the throttle command. We recorded different velocity over time functions for different, but constant throttle commands. From the velocity over time functions we can derive the acceleration over time functions, as well.

Now a velocity over time function was be approximated. The velocity over time functions can be approximated with good accuracy by a polynomial function of the form:

$$v_t = at^b \tag{16}$$

The following parameters were derived through gradient descent, where $b$ is constant 0.8, and where $a$ depends on the chosen throttle strength $u$. Now a function for $a$, depending on the given throttle command $u$ can be derived. Therefore a sigmoidal function of the form showed to be useful:

$$a = \frac{4.1}{1 + e^{15(0.55-u)}} + 0.4 \tag{17}$$

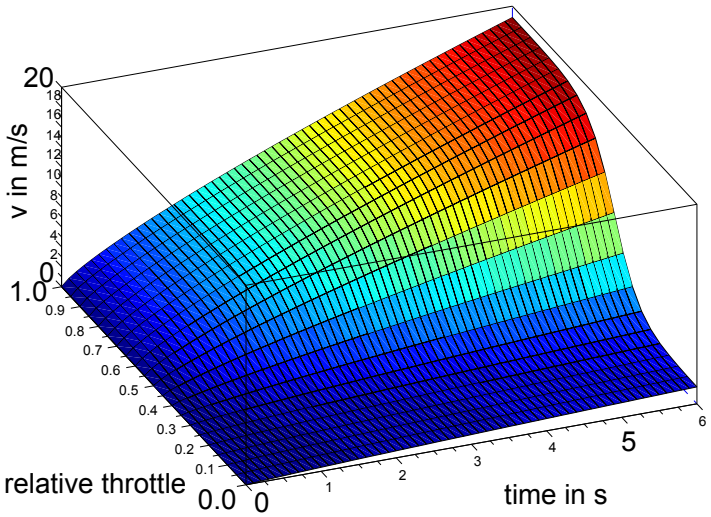



**Fig. 18.** Approximated velocity function over time and over relative throttle values.

The complete function for the car velocity $v$ over time $t$, given throttle command $u$ was assumed as:

$$v_t = f(u,t) = (\frac{4.1}{1+e^{15(0.55-u)}} + 0.4)t^{0.8} \tag{18}$$

A plot of the function is given in 18. Therefrom, the acceleration over time function can be derived easily:

$$a_t = f'(u,t) = 0.8(\frac{4.1}{1+e^{15(0.55-u)}} + 0.4)t^{-0.2} \tag{19}$$

**Brake model** Second, we wanted to analyze the car's braking behavior. Therefore we drove the car with 30 km/h and 50 km/h and executed constant brake commands, the resulting brake acceleration over time function is depicted in Fig. 19 (b). We measured the velocity over time until the car stopped.

As before we derived the acceleration over time functions for the different brake commands, the function for braking the car, e.g., from a speed of $v_0 = 8.3\frac{m}{s}$ was approximated by:

$$v_t = \max(0, f^{-1}(u,t) = v_0 - (13u^{0.8} + 0.2)t) \tag{20}$$

The two-dimensional function plot for a braking car is shown in Fig. 19 (c). The control delay of the throttle and of the brake was about 0.1 s each.

### 8.3 Car Model Conclusion

The acceleration and brake functions for a given current velocity and given throttle/brake command were successfully applied to our simulator. The lower

level controllers still had to be tuned in the real car because of the complex car dynamics, especially tire slip, mass distribution, system delays and slippage within all kinds of gears. However, this simple car model helped significantly to design the behavior and the higher level controllers within the simulator.

## 9 Summary

This technical report focussed on the main components of the autonomous cars "MadeInGermany" and "e-Instein". The specification described at the beginning was successfully met. MiG was able to safely drive through inner city traffic and drove also on the Berlin Autobahn with up to 100 km/h, where its mean lateral error to the trajectory was less than 10 cm. The velocity error was about 0.5 km/h. For e-Instein these values are still higher, but work is in progress. Future work will focus on energy efficient control routines to increase energy efficient control.

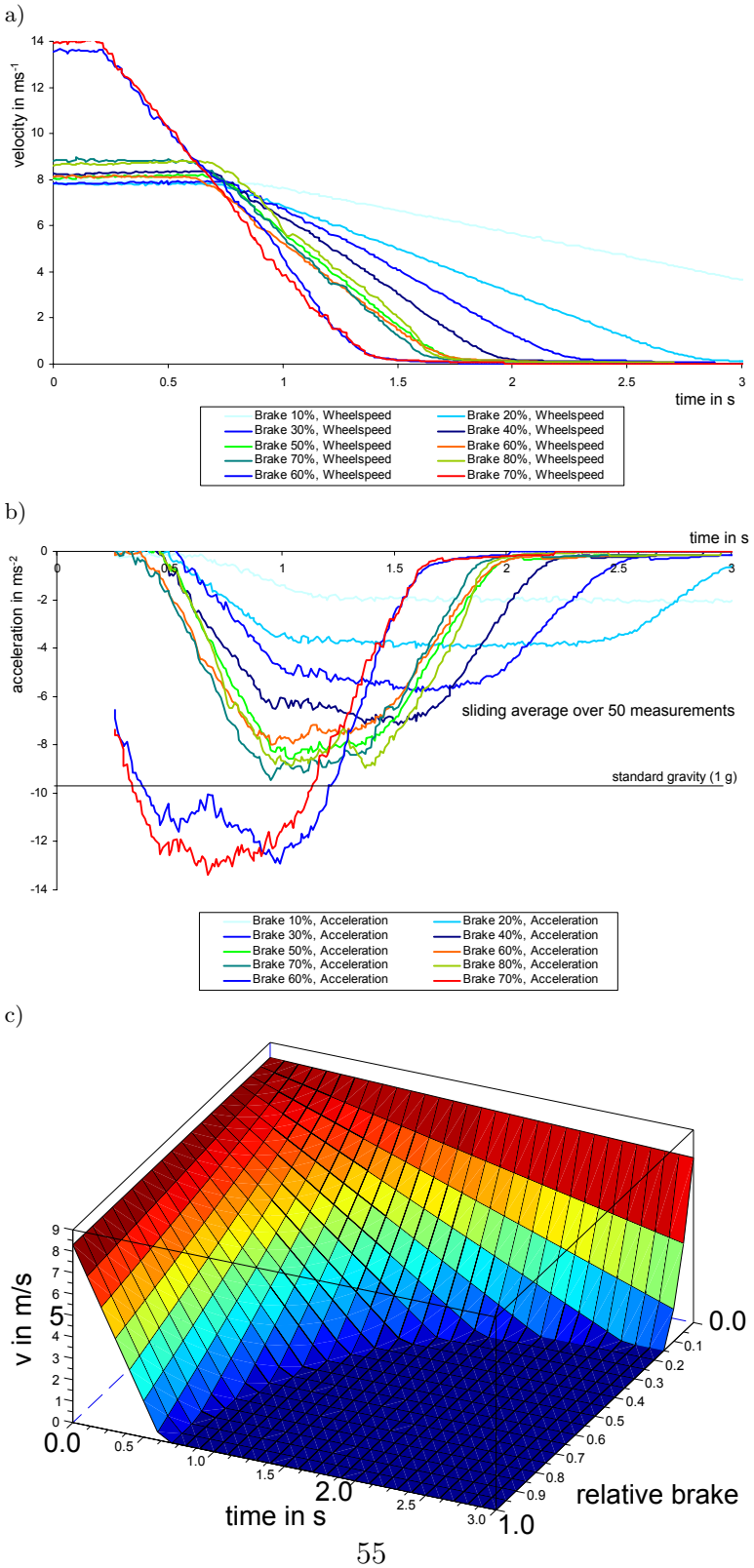





**Fig. 19.** Brake acceleration over time. (a) Ground truth for different relative brake pressure values (MiG). Velocity over time function. (b) Ground truth, acceleration over time function. (c) Approximated velocity over time function and over relative brake pressure values.

# Grounding Spatial Relations for Human-Robot Interaction

Sergio Guadarrama, Lorenzo Riano, Dave Golland, Daniel Göhring,
Yangqing Jia, Dan Klein, Pieter Abbeel, Trevor Darrell



DOI: 10.1109/IROS.2013.6696569
URL: https://doi.org/10.1109/IROS.2013.6696569

# Interactive Adaptation of Real-Time Object Detectors

Daniel Goehring, Judy Hoffman, Erik Rodner, Kate Saenko, Trevor Darrell



DOI: 10.1109/ICRA.2014.6907018
URL: https://doi.org/10.1109/ICRA.2014.6907018

# Learning to Detect Visual Grasp Affordance

Hyun-Oh Song, Mario Fritz, Daniel Goehring, Trevor Darrell



DOI: 10.1109/TASE.2015.2396014
URL: https://doi.org/10.1109/TASE.2015.2396014

# Acoustic/Lidar Sensor Fusion for Car Tracking in City Traffic Scenarios ⋆

**Hamma Tadjine** * **Daniel Goehring** **

* *IAV GmbH, Carnotstraße 1, Berlin, 10587, Germany (e-mail: Dr.Hadj.Hamma.Tadjine@iav.de).*
** *Freie Universität Berlin, Arnimallee 7, 14195 Berlin (e-mail: daniel.goehring@fu-berlin.de)*

**Abstract:** In this paper we describe a sound source localization approach which, in combination with data from lidar sensors, can be used for an improved object tracking in the setting of an autonomous car. After explaining the chosen sensor setup we will show how acoustic data from two Kinect cameras, i.e., multiple microphones, which were mounted on top of a car, can be combined to derive an object's direction and distance. Part of this work will focus on a method to handle non-synchronized sensory data between the multiple acoustic sensors. We will describe how the sound localization approach was evaluated using data from lidar sensors.

*Keywords:* Sound source localization, Autonomous driving, Sensor data fusion.

## 1. INTRODUCTION

The ability to quickly detect and classify objects, especially other vehicles within the surrounding is crucial for an autonomous car. However, cluttered environments, occlusions and real-time constraints under which autonomous vehicles have to operate let this task remain a key-challenge problem. In recent years, tremendous progress has been made in the field of self-localization, world modeling and object tracking, mainly thanks to lidar, radar, and camera based sensors but also because of algorithmic advances, e.g., how to model uncertainties [Thrun (2005)] and how to apply these methods to sensor fusion [Schnuermacher (2013)], or how to train object classifiers using machine learning techniques [Mitchell (1997)]. In the past, acoustic sensors have played a minor part in robotics, especially in autonomous driving or for outdoor robotics in general only. One reason for this might be the omnipresent noise in most city road traffic and outdoor scenarios and the domination of other sensors like lidar, camera, or radar. In this paper we want to present how an autonomous vehicle can localize other vehicles in a real-world road-traffic environment. For this task we wanted to use low-cost off-the-shelve microphone arrays like the ones provided in a Microsoft Kinect camera. Since it is usually hard to determine the euclidic distance to an object with acoustic data, we will to focus on angular direction approximation. This data can still be very helpful, especially when combined with data from other sensors, e.g., lidar data from laser scanners. One possible scenario, even though not pursued in this work, would be to localize the direction at which an emergency vehicle was detected and then to assign this direction to a tracked object using lidar data. Another challenge in our scenario are the moving sound sources and comparably high velocities of other vehicles, in addition to temporarily occluded, emerging and disappearing vehicles. The presented solution was implementated on a real autonomous car using the OROCOS realtime robotics framework. For evaluation of the algorithm the acoustic data were synchronized and evaluated with lidar objects from Ibeo Lux sensors.

## 2. RELATED WORK

A lot of progress for sound source localization has been achieved in the speech and language processing community, as in [Benesty (2007)] on beam-forming methods, or for dialog management [Frechette (2012)].

In the robotics community and especially for indoor robots there are a variety of publications on sound source local-


Fig. 1. Test Car MadeInGermany from Freie Universität Berlin, the Kinect devices were placed on top of the roof, in front of the Velodyne HDL 64 lidar sensor.

⋆ Part of this work has been funded by DAAD and DFG; in addition the authors would like to thank Prof. Dr. Raúl Rojas (Freie Universität Berlin, Germany) and Dr. Gerald Friedland (International Computer Science Institute, Berkeley, CA, USA) for their support.

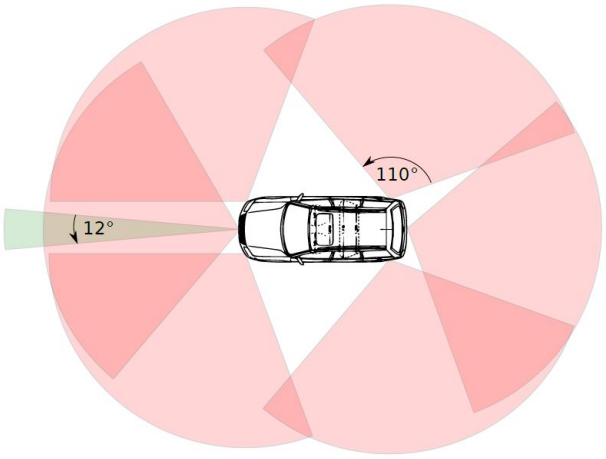



Fig. 2. The six fields of view for the six lidar sensors Lux from Ibeo and one of the radar sensors are shown, facing to the front.

ization available. The interaural time difference method (IDT) has been widely applied, as in [Liua (2010)]. In [Liu (2010)] and in [Li (2012)] the generalized cross-correlation function (GCC) is extended to localize different sound sources using an array of four different microphones. A further approach unsing a four-microphone-array in a room and time-delay estimates is provided by [Pineda (2010)], with focus on a geometric analysis and under optimization criteria. In [Valin (2003)] and in [Valin (2004)], a robot with 8 microphones was used to localize moving sound sources. The work of [Markowitz (2014)] gives a broader perspective on how people can interact with robots by using speech.

This paper is structured as follows: Section 3 will introduce the accoustic sensor setup and setup of lidar sensors, which will be used to evaluate the presented approach. Section 4 will describe the applied and implemented algorithms with an emphasis towards the sensor fusion method in this approach. In Section 5 we will perform experiments and present the results. Section 6 will summarize the approach and will give an outlook for future work.

## 3. SENSOR SETUP

As a test platform, we used the autonomous car named "MadeInGermany" from Freie Universität Berlin, cf. Fig. 1. The car is fully equipped with a combined lidar system from Ibeo, including 6 laser scanners, as shown in Fig. 2, a second 64 ray lidar sensor from Velodyne, in addition 7 radar sensors for long and short distance perception, at least 5 different cameras for lane marking and traffic light detection, including a stereo camera system for visual 3D algorithms, and a highly precise GPS unit. The car can be operated via a CAN-bus interface, thus, no further actuators are necessary to operate the throttle or brake pedals.

Different configurations were tried for the Kinect camera devices. To be independent from rain or snow and also to avoid wind noise while driving, we would have preferred to put the acoustic sensors inside the car. Unfortunately, the disadvantage of this configuration would have been the weaker signal strengths as well as signal reflections inside the vehicle. Therefore, we decided to mount both Kinect devices outside on the roof of the test platform, see Fig. 3.

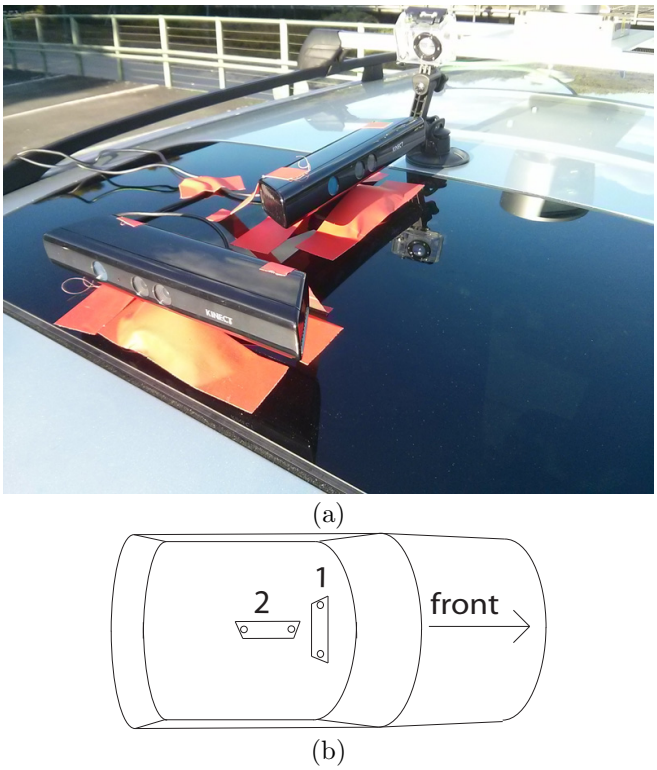



Fig. 3. Kinect sensor setup. (a) the device in the lower left corner is facing to the front of the car, the other one to the left. (b) View from above.

### *3.1 Kinect Sensor*

Each Kinect sensor is equipped with four different, non-equally spaced microphones which are aligned in line, cf. Fig. 4. As a result of this configuration, only pairs of microphones are linearly independent. To achive the highest precision for an angle estimation, we decided to use the two microphones with the largest distance to each other, i.e., the two outer microphones on the left and right side of the Kinect device, depicted in Fig. 4. Another advantage is that the signal strength for those microphones is almost equal. This is not necessarily true for the inner two microphones which are located more inside the kinect case.

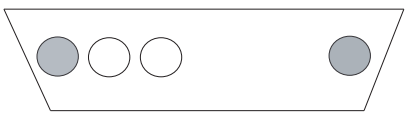


Fig. 4. Kinect microphone configuration, 4 mics are aligned in a line, we used to two outer mics (gray).

In the next section we want to describe how the sound source estimation and sensor fusion of the two Kinect devices was implemented.

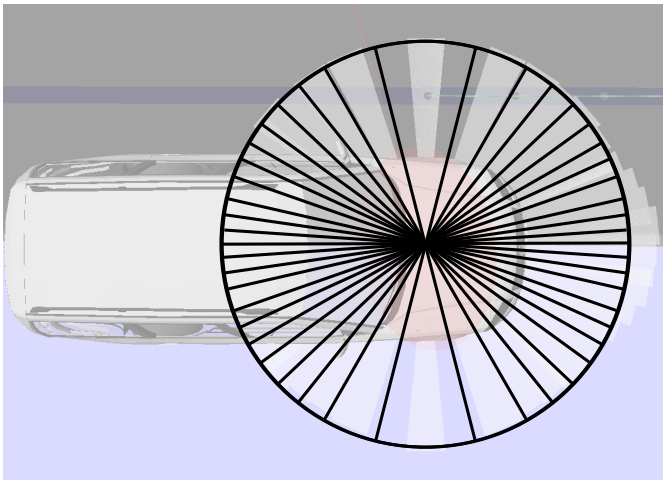


Fig. 5. Each shift between the two microphone signals corresponds to a segment (an interval) of possible angles, given that shifts can take only integer values.

## 4. SOUND SOURCE LOCALIZATION

In this section we are going to show how to localize an object using two microphones only. Furthermore we will focus on the direction accuracy given all possible directions. In the second part we will show how the resulting angular probabilistic distribution functions of two Kinect devices can be combined. One advantage of this method will be to constrain the set of possible solutions.

### *4.1 Calculation for one Kinect with two microphones*

Estimation of the sound source using two microphones was designed straightforward using a cross-correlation function over the two microphone signals. Given the signal of the left microphone $f$ and the right one $g$, for a continuous signal the cross-correlation function $f \star g$ with respect to the shift $\tau$ can be calculated as:

$$(f \star g)(\tau) = \int_{-\infty}^{\infty} f(t) \cdot g(t+\tau) \quad dt \tag{1}$$

Since we handle digital signals, for discrete functions the cross-correlation is calculated similarly with respect to a given shift $n$ between the two signals $f$ and $g$:

$$(f \star g)[n] = \sum_{-\infty}^{\infty} f[m] \cdot g[m+n] \tag{2}$$

Now we want to take a look at the real Kinect audio signals. Both Kinect microphones were sampled with 16800 Hz. For every calculation step we compared 216 data points from the two signals with a shift $n$ ranging from -20 to +20. These 216 data points (provided by a module including the open source libFreenect library) showed to be sufficient for the cross-correlation calculation and allowed us to estimate the sound direction with more than 70 Hz. Each shift between the two signals would result in a certain direction. Regarding the number of possible shifts between the two signals, the two outer microphones of the Kinect are about 22 cm apart, we therefore assumed a base distance of $b = 0.22m$. With the speed of sound at $v_s = 340\frac{m}{s}$ at sea level and with a sampling rate for each microphone of $f_k = 16800Hz$, there is a maximum and a minimum value for possible shifts. These two boundaries correspond to the sound source being perfectly on the left or on the right side of the device. The maximum and minimum shift can be calculated as:

$$\begin{aligned} n_{max} &= b \cdot f_k \cdot v_s^{-1} && (3)\\ &= \frac{0.22m \cdot 16.8kHz}{340ms^{-1}} && (4)\\ &\approx \quad 11 && (5)\\ n_{min} &= -b \cdot f_k \cdot v_s^{-1} && (6)\\ &\approx -11 && (7) \end{aligned}$$

, resulting in approx. 22 possible values for shifts, making it sufficient to check these 22 possible shifts. As we will see later, on a planar surface with two microphones there are ususally two solutions for each signal shift (except for $n_{min} = -11$ and $n_{max} = 11$). Thus, we can map each shift $n$ to two angular segments (angular intervals) which are symmetrically located with respect to the connecting line between the two microphones. The angular segments (or intervals) are depicted in Fig. 5.

The calculation of the corresponding angle for a given signal shift is straightforward, too. Given the speed of sound $v_s$ we can translate each shift $n$ into a distance $n \cdot v_s$. Now we have a triangle with a base length of $b = 0.22m$ and a known difference of the two other sides of $n \cdot v_s$ towards each other. Since the real distance to the sound source is unknown, we have to make an assumption, e.g., 25 m (the result of the calculation converges for higher distances) and can solve the angle to the object for each microphone using the Law of Cosines. A geometric sketch of the triangle is shown in Fig. 6.

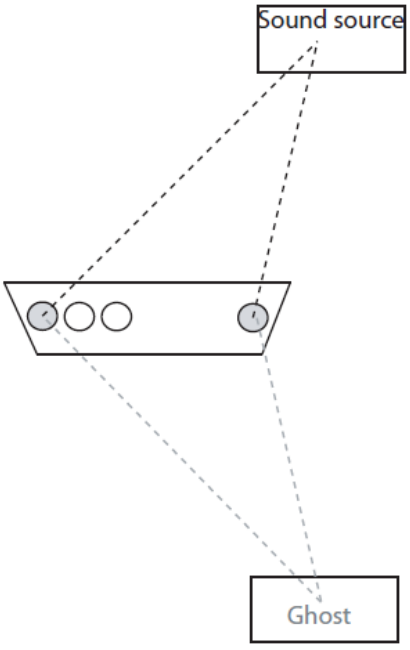



Fig. 6. Given the base distance of the triangle, the difference of the two sides and an assumed far distance (for drawing reasons the distance here is very close) to the object, the angles of the object to each microphone can be calculated - and should converge with increasing distance. Two solutions remain.

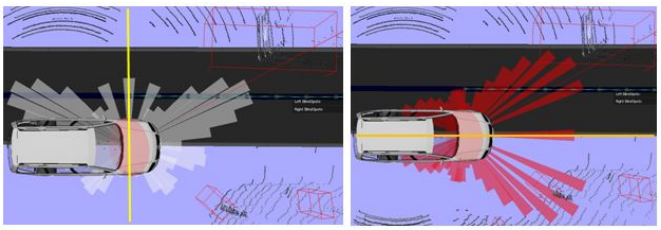


Fig. 7. Symmetry of angular distribution for front facing Kinect (left) and sideways facing Kinect (right). Symmetry axis depicted in yellow.

### *4.2 Sensor fusion of two Kinect devices*

Since the two Kinect devices are not synchronized, we cannot just combine the data of the four outer microphones for triangulation. Moreover, we decided to combine the resulting probability distributions, cf. Fig. 5 of the Kinect devices with each other. As mentioned earlier, the probability of each segment containing the angle to the sound source is calculated from the cross-correlation function. Since both Kinect devices are rotated to each other by 90 degrees, the segment sizes do not match and thus cannot be combined directly. To overcome this problem, we subsample the angular segments for each Kinect with 64 equally-spaced angular segments. In a next step, after we generate the two equally spaced angular segment sets, we can combine them by pairwise multiplication of the probabilities of two corresponding segments, i.e., segments that contain the same angles. As a result of this combination via multiplication, we get a final segment set which represents the resulting probability distribution for both Kinect sensors (belief distribution). While each Kinect device alone cannot distinguish between objects in front and objects behind (see symmetry depictions in Fig. 7), after combination with the second sensor, those symmetries vanish. We show the calculation schematically in Fig. 8 and a step by step calculation with experimental data in a real traffic scenario in Fig. 9.

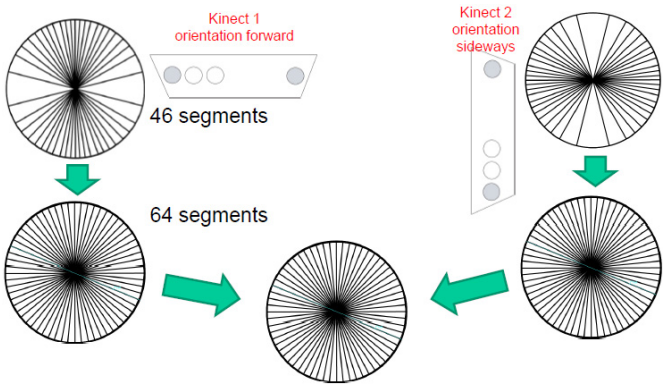



Fig. 8. Schemtatic calculation. The upper two segment sets result from the two different Kinect sensors. Since the segment sizes of the two sets are not equally aligned with respect to each other, we need to subsample them seperately into two segment sets with 64 equally sized segments. In a next step, they can be combined via pair-wise multiplication into a final segment set.

After sensor fusion, the resulting segment set corresponds to a probability distribution (belief distribution) of possible directions, i.e., where the sound source is located. To calculate a discrete value for the most likely direction, we selected the segement with the highest probability value assigned and took the mean value of that particular segment as the resulting angle. There would have been more sophisticated methods, e.g., integrating over different segments; also we thought about how to calculate directions to multiple sound sources but left this open to future research work.

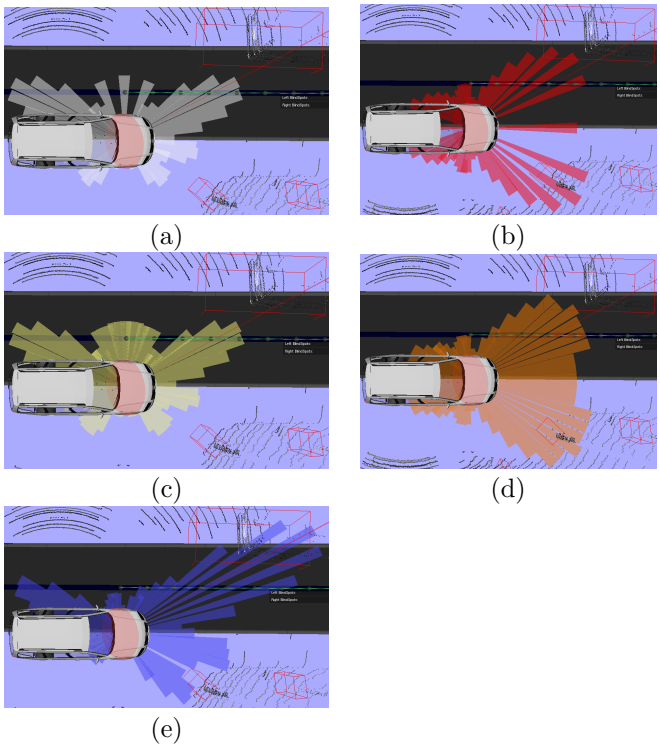



Fig. 9. Illustration of the approach, sound source vehicle in the upper right corner. Segment lengths correspond to cross-correlation amounts of the underlying signal shift and can be interpreted as a probability for the sound source lying in that angular interval. (a) Non-uniform segments for front facing Kinect and (b) left facing Kinect; (c) uniform (equally spaced) segments for front facing Kinect after subsampling, (d) uniform segments for left facing kinect; (e) uniform segments after combining (c) and (d), the resulting probability distribution (belief) for the sound source direction.

## 5. EXPERIMENTAL EVALUATION

As mentioned above, the proposed algorithm was implemented for our autonomous vehicle and tested in a real traffic scenario. The algorithms were tested within a modular robotics framework, the Open Robot Control Software Project Orocos (2011) under an Ubuntu 12.4. 64bit operating system. The data from both Kinect sensors were integrated into our AutoNOMOS software project and time stamped to compare them with our lidar sensory data. The six lidar Lux sensors from Ibeo run with a frequency of 12.5 Hz, the Kinect sensors ran with 70 Hz.

### 5.1 Test scenario

We tested our approach in a Berlin traffic scenario, close to the campus of the Freie Universität Berlin. Because driving the car was causing a lot of wind noise, we decided to park the car on the road side of the Englerallee, a medium-sized traffic road with trees, parked cars and houses on the side. Vehicles on the street were maintaining a velocity of 50-60 km/h (approx. 35 mph). Since there are trees on the middle strip seperating the two road lanes, cars of the more distant lane were partially occluded by trees while passing. We were interested in the angular accuracy of our approach in comparison to object angles from the lidar sensor. Therefore, data from the lidar sensor (a point cloud) was clustered into 3d-objects and tracked over time, resulting in a position and velocity vector for all clustered lidar objects. Since we were interested in moving (non-static) objects only, we compared the calculated angle from audio data to the closest angle of a moving object (from lidar).

### 5.2 Experimental results

In Fig. 10 we evaluated the angular error over time. We therefore took more than 5000 measurements, the resulting error-over-time function is depicted in Fig. 10.

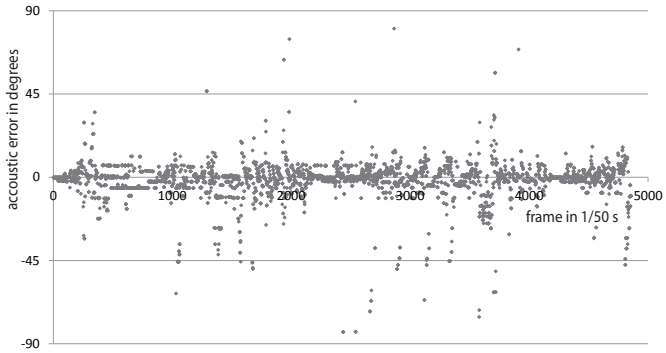



Fig. 10. Experimental evaluation: Difference between the angle from Kinect data to lidar data over time, 5000 data points were recorded. The standard deviation for the difference is $\sigma = 10.3$ degrees.

We also plotted the angular errors over all distances to the objects, as can be seen in Fig. 11. What is interesting, the highest angular errors occurred not for the farest objects but for objects within medium distances. One explanation could be that objects very far away would occupy a very small angular segment in the laser scanner, while objects closer occupy larger angular segments. Since the laser scanner always takes the center point of the detected object as a reference, and since the Kinect sensor will receive the loudest noise from the closest part of the vehicle, which is usually not the center of a car but the approaching front or leaving rear, this might be one reason for an increased detection error. Another reason could be increased reflection of noise on houses or trees for certain distances, which need further analysis.

In Fig. 12 we plotted the standard deviation of the angular error for different distance intervals, which showed the same result in terms that medium distances generated the highest error rates. The calculation time of the algorithm

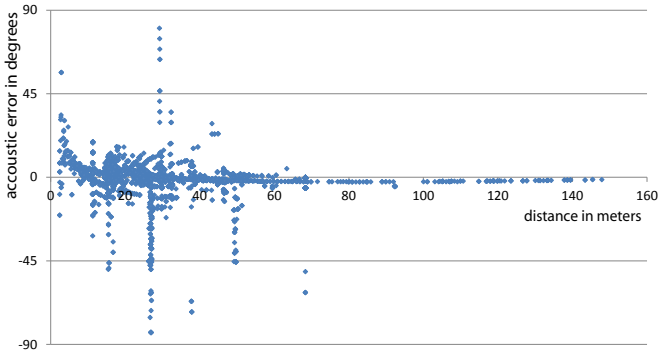



Fig. 11. The angular error over different object distances (measured by lidar). Higher error rates occured for medium distanced objects.

was negligible so that all experiments were performed under realtime constraints

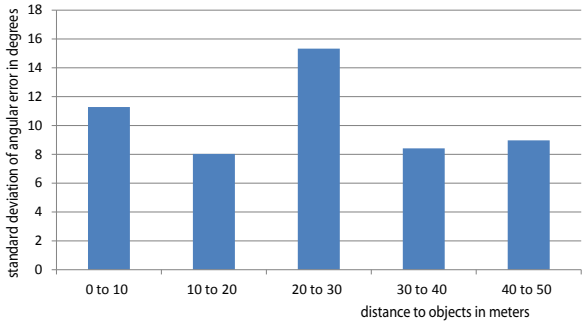



Fig. 12. Angular detection error for different distance intervals. While for high distances the angular error standard deviation was about 9 degrees, for medium distances it was approx. 15 degrees.

## 6. CONCLUSION

We presented, implemented and tested an approach which allows a road vechicle, equipped with to off-the-shelve Kinect cameras to localize objects in a distance of up to 50 meters and with a velocity of 50-60 km/h. We showed how to combine probabilistic density functions from two Kinect microphone devices using equally spaced angular interval segment sets, which helped to disambiguate possible angular locations while keeping the whole belief distribution. The algorithm can easily perform under real-time constraints with a frequency of 70 Hz. We also showed how the acoustically derived angle to the sound source could be assigned to moving objects from lidar sensors.

### 6.1 Future work

Future work needs to focus on localization and tracking of multiple objects, since in real traffic scenarios there are usually multiple vehicles in close proximity. Handling wind noise will be a crucial and challenging task for sound localization while in motion. Noise reflections on trees,

buildings and cars provide another challenge. Distance estimation, at least to some extend could support the data fusion problem with objects from other sensors. Band pass filters, e.g., application of Fast Fourier Transformation (FFT) shall be considered in future works. FFT can help to select specific signals, e.g. emergency vehicles with certain signal horn frequencies and signal patterns. Here the detection of alternating sound frequencies, as for emergency horns, would be helpful, too. Another research path worth following could be acoustic object tracking and velocity estimation, taking advantage of the doppler effect, i.e., the change of a frequency spectrum for an approaching or leaving vehicle.

## REFERENCES


J. Benesty, J. Chen, Y. Huang, J. Dmochowski: On microphone-array beamforming from a mimo acoustic signal processing perspective. *In: IEEE Transactions on Audio, Speech and Language Processing*, 2007.

M. Frechette, D. Letourneau, J.-M. Valin, F. Michaud: Integration of Sound Source Localization and Separation to Improve Dialog Management on a Robot. *In: Proceedings of the IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, 2012.

Xiaofei Li, Maio Shen, Wenmin Wang, Hong Liu: Real-time Sound Source Localization for a Mobile Robot Based on the Guided Spectral-Temporal Position Method, *In: International Journal of Advanced Robotic Systems*, 2012.

Hong Liu: Continuous sound source localization based on microphone array for mobile robots, *In: Proceedings of the IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, 2010.

Rong Liua and Yongxuan Wanga: Azimuthal source localization using interaural coherence in a robotic dog, *In: modeling and application, Robotica / Volume 28 / Issue 07, Cambridge University Press*, December 2010.

Markowitz: Robots that Talk and Listen, *Technology and Social Impact*, 2014.

Tom Mitchell: Machine Learning, *McGraw Hill*, 1997.

J. Murray, S. Wermter, H. Erwin: Auditory robotic tracking of sound sources using hybrid cross-correlation and recurrent networks, *In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2005.

Open Robot Control Software Project *http://www.orocos.org/*, 2011.

Xavier Alameda Pineda, Radu Horaud: A Geometric Approach to Sound Source Localization from Time-Delay Estimates, *In: Robotica*, 12/2010.

M. Schnrmacher, D. Ghring, M. Wang, T. Ganjineh: High level sensor data fusion of radar and lidar for carfollowing on highways, *In: Recent Advances in Robotics and Automation*, 2013.

S. Thrun, W. Burgard, D. Fox: Probabilistic Robotics, *MIT Press*, 2005.

J.-M. Valin, F. Michaud, J. Rouat, D. Letourneau: Robust Sound Source Localization Using a Microphone Array on a Mobile Robot, *In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2003.

J.-M. Valin, F. Michaud, B. Hadjou, J. Rouat: Localization of Simultaneous Moving Sound Sources, *In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

# Online Vehicle Detection using Deep Neural Networks and Lidar based Preselected Image Patches

Stefan Lange, Fritz Ulbrich, Daniel Goehring



DOI: 10.1109/IVS.2016.7535503
URL: https://doi.org/10.1109/IVS.2016.7535503

# Traffic Awareness Driver Assistance based on Stereovision, Eye-tracking, and Head-Up Display

Tobias Langner, Daniel Seifert, Bennet Fischer, Daniel Goehring, Tinosch Ganjineh, Raul Rojas



DOI: 10.1109/ICRA.2016.7487485
URL: https://doi.org/10.1109/ICRA.2016.7487485

# Pole-Based Localization for Autonomous Vehicles in Urban Scenarios

Robert Spangenberg, Daniel Goehring, Raul Rojas



DOI: 10.1109/IROS.2016.7759339
URL: https://doi.org/10.1109/IROS.2016.7759339

# Extracting Path Graphs from Vehicle Trajectories

Fritz Ulbrich, Simon Rotter, Daniel Goehring, Raul Rojas



DOI: 10.1109/IVS.2016.7535552
URL: https://doi.org/10.1109/IVS.2016.7535552

# Online Vehicle Detection using Haar-like, LBP and HOG Feature based Image Classifiers with Stereo Vision Preselection

Daniel Neumann, Tobias Langner, Fritz Ulbrich, Dorothee Spitta, Daniel Goehring



DOI: 10.1109/IVS.2017.7995810
URL: https://doi.org/10.1109/IVS.2017.7995810

# Stable Timed Elastic Bands with Loose Ends

Fritz Ulbrich, Daniel Goehring, Tobias Langner, Zahra Boroujeni, Raul Rojas



DOI: 10.1109/IVS.2017.7995718
URL: https://doi.org/10.1109/IVS.2017.7995718

# Flexible Unit A-star Trajectory Planning for Autonomous Vehicles on Structured Road Maps

Zahra Boroujeni, Daniel Goehring, Fritz Ulbrich, Daniel Neumann, Raul Rojas



DOI: 10.1109/ICVES.2017.7991893
URL: https://doi.org/10.1109/ICVES.2017.7991893

# Traffic Mapping for Autonomous Cars

Nicolai Steinke, Fritz Ulbrich, Daniel Goehring, Raul Rojas



DOI: 10.1109/IVS.2018.8500601
URL: https://doi.org/10.1109/IVS.2018.8500601

# Autonomous Car Navigation using Vector Fields

Zahra Boroujeni, Mostafa Mohammadi, Daniel Neumann, Daniel Goehring, Raul Rojas



DOI: 10.1109/IVS.2018.8500446
URL: https://doi.org/10.1109/IVS.2018.8500446

# Following Cars With Elastic Bands

Fritz Ulbrich, Tobias Langner, Stephan Sundermann, Daniel Goehring, Raul Rojas



DOI: 10.1109/IVS.2018.8500481
URL: https://doi.org/10.1109/IVS.2018.8500481

# Application of Open-Source Deep Neural Networks for Object Detection in Industrial Environments

Christian Poss, Olimjon Ibraginov, Anoshan Indreswaran, Nils Gutsche, Thomas Irrenhauser, Marco Prueglmeier, Daniel Goehring



DOI: 10.1109/ICMLA.2018.00041
URL: https://doi.org/10.1109/ICMLA.2018.00041

# Analytic Collision Risk Calculation for Autonomous Vehicle Navigation

Andreas Phillipp, Daniel Goehring



DOI: 10.1109/ICRA.2019.8793264
URL: https://doi.org/10.1109/ICRA.2019.8793264

# Pedestrian Head and Body Pose Estimation with CNN in the Context of Automated Driving

Michaela Steinhoff[1], Daniel Göhring[2]
[1]*Business Area Intelligent Driving Functions, IAV GmbH, Rockwellstr. 3, 38518 Gifhorn, Germany*
[2]*Institute of Computer Science, Freie Universität Berlin, Arnimallee 7, 14195 Berlin, Germany*
*michaela.steinhoff@iav.de, daniel.goehring@fu-berlin.de*

Keywords: Automated Driving, Convolutional Neural Network, Headpose, Pedestrian Intention, Semi-Supervision

Abstract: The challenge of determining pedestrians head poses in camera images is a topic that has already been researched extensively. With the ever-increasing level of automation in the field of Advanced Driver Assistance Systems, a robust head orientation detection is becoming more and more important for pedestrian safety. The fact that this topic is still relevant, however, indicates the complexity of this task. Recently, trained classifiers for discretized head poses have recorded the best results. But large databases, which are essential for an appropriate training of neural networks meeting the special requirements of automatic driving, can hardly be found. Therefore, this paper presents a framework with which reference measurements of head and upper body poses for the generation of training data can be carried out. This data is used to train a convolutional neural network for classifying head and upper body poses. The result is extended in a semi-supervised manner which optimizes and generalizes the detector, so that it is applicable to the prediction of pedestrian intention.

## 1 INTRODUCTION

The research on automated driving is more relevant than ever. Semi-automated functions such as automatic parking or driving in stop-and-go traffic have long been available in the form of assistance systems (parking and traffic jam assistant). Even fully automated driving is no longer limited to motorway scenarios. Many projects, like Stimulate[1] in Berlin, master the challenges of urban traffic already completely autonomous, although limited in speed. This work is part of a project, which is contributing to the ongoing development of self-driving cars.

One of the biggest challenges in urban scenarios is the robust prediction of pedestrians. Simple tracking and adapted motion models are not sufficient to map the highly dynamic behaviour of humans. Therefore, countless research groups try to extract more information from the human posture. In addition to a more precise analysis of the leg positions, many researchers also focus on the head pose. Kloeden et al. (2014) have already shown that the head pose is suitable as a characteristic for predicting the movements of pedestrians. They proved that pedestrians show a protection behaviour particularly before crossing the road, which can be attached to the increased head movement. For many decades, classical machine learning has been used to extract this orientation of the head. A common methodology is the quantification of the angular ranges, and thus the declaration of a classification problem (Schulz and Stiefelhagen, 2012). In that work, the authors scan the upper part of a pedestrian image, assuming the head to be there. Eight classifiers are used to locate the head within this part and estimate an initial pose. These classifiers are trained for eight different head pose classes, each with a range of 45°. For the continuous estimation of poses, regression is the preferred method. Lee et al. (2015) and Chen et al. (2016) extract gradient based characteristics like HOG (Histogram of Oriented Gradients) features and then use a SVR (Support Vector Regressor) to estimate the head pose.

All these methods only consider the yaw angle of the head. Contrary to this, the approaches of Rehder et al. (2014), Chen et al. (2011) and Fanelli et al. (2011) take additional orientation directions into account. The latter receives 3D data from a depth camera and uses it to find the position of the tip of the nose as well as the yaw, pitch and roll angles of the head using a Random Regression Forest. A disadvantage of this method is therefore that only a limited area of the possible head poses, namely the one with a visible

[1]https://www.wir-fahren-zukunft.de

nose, can be determined. It is furthermore assumed that the head was detected in the image in advance. Conversely, in Reh-der et al. (2014) monocular RGB images serve as input data, which do not have to be restricted to the head section, but can contain complete as well as covered pedestrians. The head localization is done within the algorithm via HOG/SVM and a part-based detector proposed by Felzenszwalb et al. (2010). Proceeding from this, four discrete classes are defined for the pose estimation, for each of which a classifier is trained using logistic regression with LBP (Local Binary Pattern) features. By integrating the discrete orientation estimates using a HMM (Hidden Markov Model), they obtain continuous head poses. This approach is particularly interesting in that the head poses are plausibilized and impossible poses are discarded with the help of the upper body pose and the motion direction. Chen et al. (2011) go one step further and estimate both the head and body poses in pedestrian images. Therefore, the orientation of the body is divided into eight discrete direction classes and multi-level HOG features are extracted. Furthermore, the yaw angle range of the head is divided into twelve classes and the pitch angle range of the head into three classes. After localizing the head, texture and color features are extracted by another multi-level HOG descriptor and histogram-based color detector. A particle filter framework is subsequently used to estimate the body and head poses. The dependency between the poses as well as the temporal relationship are taken into account. Another approach also estimates both the head and body pose (Flohr et al., 2015). For both poses, eight orientation-specific detectors are trained, whose class centers are shifted by $45^\circ$ each. To locate the exact body and head position in the image, they make use of disparity information obtained from the stereo input data. Based on this, a DBN (Dynamic Bayesian Network) is used to get the current orientation states. Thereby the current head pose depends on the previous head pose and also on the current body pose.

Recently, (deep) neural networks have become increasingly important and their application also aims for an improvement of the head pose detection. Latest nets as presented in (Patacchiola and Cangelosi, 2017) or (Ruiz et al., 2017) predict yaw, pitch and roll angles in a continuously manner and achieve great accuracy. The input, however, is also here only the head section, which must be available in relatively high resolution. If these approaches are to be used in the context of automatic driving, pedestrians and their head positions must be recognized early, i.e., from a great distance, so that the poor quality of the input data does not fulfill the requirements of the mentioned methods. The present work, therefore, presents a neural network that recognizes head poses from images with the quality of cameras commonly used in vehicles. Not only the head but the entire pedestrian's image section serves as input, since the head pose in relation to the upper body provides further important information. From this it can be deduced, for example, whether a pedestrian shows a safety behaviour, which is a clear indication of the intention to cross the road. For the training of this head and upper body pose detector, commonly available data sets for head poses and pedestrians in general like Human3.6m[2], PETA[3] or INRIA[4] cannot be used, because the reference to the upper body alignment is missing. In addition, most researchers only consider yaw angles in the range of $-90^\circ$ to $90^\circ$, i.e., the frontal view of the pedestrian heads. In the present project, however, it is just as important whether a passer-by perceives oncoming traffic or the automated driving vehicle. Therefore, a framework for the generation of a "full-range" data set will be briefly presented here. Using a semi-supervised approach, a trained convolutional neural network (CNN) is extended so that the comparatively small amount of self-generated annotated data is enriched by many unlabeled data from real test drives within the project and the detector achieves more accurate results.

The contributions of this work can be summarized in the following key points:

- a framework for generating a data set with head and upper body poses,
- training and evaluation of a network (CNN) using the data set,
- enhancement of training data with real driving data,
- evaluation of an approach to semi-supervised learning and improvement of the network.

The paper is structured as follows:

After the second chapter presented the framework for data set generation, Chapter 3 gives a detailed description of our detector design. The obtained results are illustrated in the following chapter. Chapter 5 draws a conclusion and presents an outlook for future work.

[2]http://vision.imar.ro/human3.6m/description.php
[3]http://mmlab.ie.cuhk.edu.hk/projects/PETA.html
[4]http://pascal.inrialpes.fr/data/human

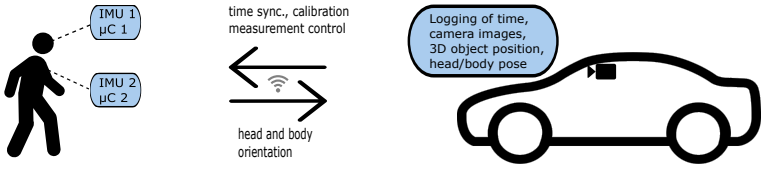


Figure 1: The used framework setup for data set generation.

# 2 DATA SET GENERATION

As already mentioned, this work requires data that is not annotated in the common pedestrian data sets. The added value lies in the fact that not only the pose of the head, but also that of the upper body in relation to the head is considered. In order to annotate such data automatically, a data generation setup and software framework was developed which can be applied to capture images of pedestrians together with the corresponding head and upper body poses. Therefore this section first explains the experimental setup and the processing infrastructure after that. An overview of the framework setup is given in Figure 1.

## 2.1 Experimental Environment

The images were taken by a camera installed in a test vehicle alongside other sensors such as Lidar. The vehicle was also equipped with an object tracking module, which outputs 3D positions of the pedestrians in vehicle coordinates. Two inertial sensors (MPU6050) with 6 degrees of freedom each were used to measure the exact head and upper body orientations. Together with one microcontroller with integrated WiFi module (ESP8266 – 12F) each, these were placed on the head and upper body of the test persons. Since the position and orientation of the MPU6050s on the head and body depend a lot on the probands and the upcoming measurement, an online calibration was performed at the beginning of each exposure and the sensor values were transformed into quaternions relative to the corresponding initial pose. In addition, IMU (inertial measurement unit) drift compensation was carried out beforehand and the drift behavior was analyzed in the following. With an average duration of the measurement sequences of 2 minutes, the drift of $0.5^\circ$ per minute was negligible.

## 2.2 Processing Infrastructure

The control of the IMU, the online calibration and the time synchronization via ntp server were realized in Arduino on the microcontrollers. The measured poses and the related timestamps were sent via TCP to a logging computer where they were processed and added to the data set. A single date then consists of the timestamp with corresponding image, the 3D object position and yaw, pitch and roll angles of either head and the upper body. A total of 2500 test and training data was annotated, including recordings of 20 different people at different times of the day and year.

# 3 HEAD AND UPPER BODY POSE DETECTOR

This section introduces the developed head pose detector. First of all, the definition of the individual classes is discussed. The training process is divided into the two parts supervised and its unsupervised extension, which are explained in the following two subchapters.

## 3.1 Class Definition

The detector presented in this paper is intended to detect the yaw angles of the head and upper body. Since we want to address a classification problem, the annotated data has to be mapped to classes. Therefore, the possible head poses in the range $[-105^\circ, 105^\circ]$ are quantized in $\alpha_H = 30^\circ$ steps, whereby a yaw angle of $0^\circ$ implies the head pointing directly towards the camera. All following angles are specified in this definition of coordinate system.

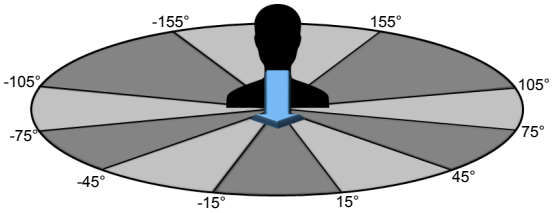


Figure 2: The head pose range is divided into 10 classes.

The range $]105°,180°] \cup [-180°,-105°[$ (i.e., facing away from the camera) is divided into three parts á $\alpha_{H2} = 50°$. Accordingly there are $n_{hc} = 10$ head classes in total (see Figure 2). For anatomical reasons, the deviation of the upper body pose from the head pose is limited to a range of $-90°$ to $+90°$. This area is divided into $n_{bc} = 3$ body classes $C_B$ depending on the head pose. The body either points left ($C_B = 0$), right ($C_B = 2$) or in the same direction as the head ($C_B = 1$). This results in an overall number of 30 classes for the detector. The output class $C_{out}$ resulting from the head ($C_H$) and upper body ($C_B$) class is calculated according to Eq. 1. The head and upper body class are derived from the respective yaw angles $\psi_H$ and $\psi_B$.

$$C_{out} = C_B \cdot n_{hc} + C_H \tag{1}$$

with

$$C_H = \begin{cases} \lfloor \frac{\psi_H - \frac{1}{2}\alpha_H}{\alpha_H} \rceil + \frac{n_{hc}}{2} & , \text{if } -105° \leq \psi_H \leq \frac{\alpha_H}{2} \\ \lfloor \frac{\psi_H - \frac{1}{2}\alpha_H}{\alpha_H} \rceil + \frac{n_{hc}}{2} + 1 & , \text{if } \frac{\alpha_H}{2} < \psi_H \leq 105° \\ \lfloor \frac{\psi_H - \frac{1}{2}\alpha_{H2}}{\alpha_{H2}} \rceil + \frac{n_{hc}}{2} - 1 & , \text{if } \psi_H < -105° \\ \lfloor \frac{\psi_H - \frac{1}{2}\alpha_{H2}}{\alpha_{H2}} \rceil + \frac{n_{hc}}{2} + 1 & , \text{if } 105° < \psi_H \leq 155° \\ 0 & , \text{otherwise} \end{cases} \tag{2}$$

and

$$C_B = \lfloor \frac{\delta\psi}{\alpha_B} + \frac{1}{2} \rceil + \lfloor \frac{n_{bc}}{2} \rceil \tag{3}$$

where

$$\delta\psi = \begin{cases} \psi_H - \psi_B + 360° & , \text{if } \psi_H - \psi_B < -180° \\ \psi_H - \psi_B - 360° & , \text{if } \psi_H - \psi_B > +180° \\ \psi_H - \psi_B & , \text{otherwise} \end{cases} \tag{4}$$

## 3.2 Semi-Supervised Learning with CNN

In the domain of neural networks, CNN have been established to handle classification tasks. Most of the best-known classification networks are trained in a supervised manner with a large amount of annotated data. Since the present use case makes different demands on the annotation, only the few self-generated data are available in comparison. The idea to train a reliable classification network from it nevertheless is based on a semi-supervised approach.

Figure 3: Samples of unlabeled data, the first row shows unsorted, the second and third row clustered samples.

**Supervised Learning**

As mentioned above, CNN are very well suited to solving classification problems and there are many proven network architectures. Hence, a CNN is also used here and the layer topology is oriented to these architectures. Figure 4 shows a schematic representation of the underlying network structure.

The input data is scaled to a fixed size (128*x*128) and converted to grayscale values. They subsequently pass through three consecutive blocks each with three convolutional and one maxpooling layer until a fully connected layer maps them to an embedding vector with size 64, which is transformed to logit class scores by a final dense layer. During training, a dropout layer located between the last two fully-connected layers was used with a dropout rate of 0.5 in order to generalize the learning result. To find the best hyper parameters for the training, a grid search was applied. Accordingly, the following parameter configuration provides the best performance and has been used further:

Table 1: Best parameters found by grid search.

| | |
|---|---|
| batchsize | 50 |
| initial learning rate | 0.0001 |
| learning rate decay | 0.33 |
| decay steps | 10000 |
| optimizer | Adam |
| loss function | Cross Entropy |

The loss function of the supervised part with labels $\lambda$ and predicted outputs $y$ is given by

$$loss_{logit} = -\sum_x \lambda \cdot \log(y + 1e^{-8}). \tag{5}$$

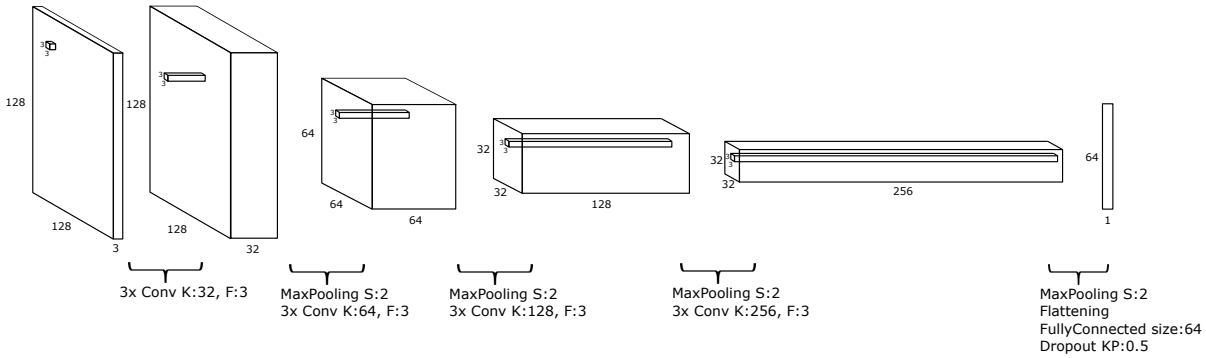



Figure 4: Schematic representation of the used network architecture.

**Semi-Supervised Extension**

Due to the comparably small variance of the training data, the network performed well on similar testing data. With the aim of optimizing and generalizing the network, it was extended to include unsupervised learning. In general, as with most semi-supervised methods, the results of supervised learning are improved by clustering the unlabeled data and then assigning them to already trained classes. Inspired by the concept of Haeusser et al. (2017), the assignment is not based on the last but the penultimate layer. In this so-called "embedding" level, the similarity of all unlabeled data to the labeled data is determined. An actual assignment of the unlabeled data only takes place if it has been attributed to the same class label twice due to the highest similarity. To find a suitable scale for this similarity, different metrics were tested and compared to each other. The following two metrics have emerged as the ones with the best-performing results.

The *cosine similarity* describes the correspondence of the orientations of two vectors to be compared. For this purpose the cosine of the angle between them is determined according to Eq. 6.

$$\cos(\theta) = \frac{a \cdot b}{\|a\| \, \|b\|} \tag{6}$$

The resulting value range for this scale is therefore limited to [-1, 1], where '1' means that the orientation of both vectors is identical ($\theta = 0^\circ$). '-1' however denotes an opposite orientation ($\theta = 180^\circ$) and '0' signifies the vectors are orthogonal to each other ($\theta = 90^\circ$). Aside from being independent of the vectors magnitudes, this metric has the advantage that it is very computation-performant, since only the dot product has to be calculated. The loss is determined analogue to Haeusser et al. (2017) by comparing the resulting association probability with the expected probability distribution using cross entropy.

The *Mahalanobis distance* indicates the distance of a data point to the mean of a point distribution of one class in multiples of the standard deviation. Thus, in contrast to the *Euclidean distance*, the correlation between the data points is taken into account and the assignment to individual clusters of data (classes) becomes more accurate.

If $\vec{x}$ is a data point to be assigned and $\vec{\mu}$ is the mean value of the data set of a class with covariance matrix $C$, the Mahalanobis distance is given by:

$$D_M(\vec{x}) = \sqrt{(\vec{x}-\vec{\mu})^T C^{-1} (\vec{x}-\vec{\mu})} \tag{7}$$

The result initially expresses the dissimilarity of the sample to the data set. By scaling to the value range $D_{Ms} = [0, 1]$, reverting the range and normalizing the multiplication of this association probability with its transposed the following probability distribution is obtained stating that several unlabeled data points are associated with the respective classes:

$$p = ||(p_A \cdot p_A^T)||_2 \tag{8}$$

with

$$p_A = 1 - D_{Ms} \tag{9}$$

The expected probability distribution $p_E$ in this case is equal to the unit matrix with rank $n_C$ (number of classes), since a sample is to be assigned uniquely to one class. For this purpose it must be ensured that each class is represented with at least one sample per batch in the set of unlabeled data. This is achieved by adding one labeled sample for each class to the batch with unlabeled samples. The total loss is finally calculated by applying cross entropy on these probabilities and adding the result to the logit loss from the supervised part (see Eq.5).

$$loss = -\sum_x p_E \cdot \log(p_A + 1e^{-8}) + loss_{logit} \tag{10}$$

Table 2: Train error, test error, average precision (AP), average recall (AR) and test error including 'adjacent classes' (TE*) of SUP, COS and MAHA in [%]. The last two columns declare the distribution of train and test samples within the supervised and the semi-supervised methods.

| | train err | test err | AP | AR | TE* | train samples | test samples |
|---|---|---|---|---|---|---|---|
| SUP | 0.17 | 82.19 | 16.96 | 18.27 | 54.86 | 2000 | 500 |
| COS | 30.59 | 66.55 | 32.57 | 25.17 | 43.52 | 12000 | 500 |
| MAHA | 0.41 | 58.53 | 32.26 | 30.21 | 32.15 | 12000 | 500 |

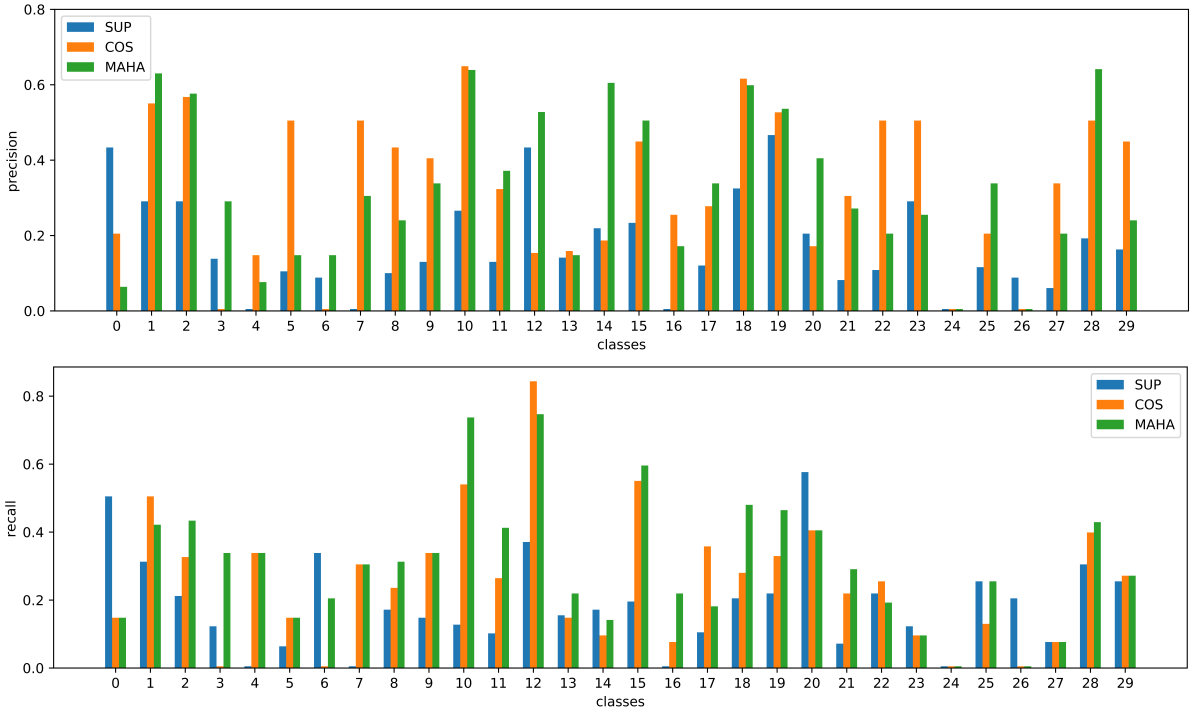



Figure 5: Precision and recall of the trained networks, the results of the semi-supervised approaches (COS and MAHA) improve the supervised one (SUP).

# 4 EXPERIMENTS

For the training of the head pose detector 2500 labeled and 10000 unlabeled data were used. It was performed on a computer with four GTX 2080 Ti with 12 GB memory each. Because of the high imbalance of the class distribution in the labeled training data set, the maximum number of samples used per class in all three trainings was limited to avoid overfitting of more frequent classes. This was already recommended by Weiss and Provost (2001), who showed that an unequal distribution does not usually lead to the best performance. In the following, the results of the purely supervised trained network (further referred to as SUP) and the two different methods for estimating similarity within the semi-supervised trained network (MAHA for the one using the mahalanobis distance, COS for the cosine similarity) are compared. Due to the small number of labeled samples, SUP converged comparatively quickly after about 500 epochs. With the best parameters found by the grid search, a training error of 0.17% was achieved. But the evaluation with test data confirmed that the network specialized in the training data. The error rate for the randomly distributed test data was 82.19% at best (see Table 2).

In the approach of association using cosine similarity, it was necessary that each class is represented in each batch of labeled data so that each unlabeled sample can be assigned properly as well. Depending on the number of samples used per class per batch, this results in very large batch sizes for 30 classes, which caused memory issues. But with 10 samples per class per batch a suitable compromise between training efficiency and executability was found. This of course led to a declining of the obtained network accuracy, resulting in a training error of about 30%. Nevertheless, this as well as the second semi-supervised

solution MAHA reduces the test error by a relevant amount, which is also reflected in Figure 5, depicting the precision and recall per class of all three approaches. Even if individual classes perform worse for COS and MAHA than for SUP, the average precision (AP) and recall (AR) noticeably are higher as can be seen in Table 2. And although MAHA also required restrictions in parameter selection due to the limited memory capacity, this approach yielded the lowest test error of 59.17%. The fact that the semi-supervised approaches generalize the training result and thus enhance it is particularly evident when the confusion matrix is analyzed more closely. Therefore Figure 6 illustrates the confusion matrix of MAHA. For comparison, those of SUP and COS can be found in the appendix. First of all, it is conspicuous that test samples of other classes are assigned more often to the columns 10 to 19, which correspond to the classes with the same alignment of head and upper body. This is probably due to the fact that this natural human pose occurs more frequently in the unlabeled data set and thus their training was more effective. Furthermore, the principal diagonal is highlighted in dark blue as these cells map the amount of true positives. According to the class definition in Section 3, the classes ending with the same number (e.g. 3, 13 and 23) represent the same head class. These cells are also shaded light blue. The remaining cells are marked darker gray the higher their cell value is. Mainly in contrast to the confusion matrix of SUP (see Figure 8), whose predictions are highly scattered, an orientation of the predicted classes to the principal diagonal as well as partially to the secondary diagonals of the same head classes can be observed here.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 14 | 42 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 14 | 0 | 14 | 0 | 0 | 0 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 7 | 53 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 5 | 33 | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 5 | 0 | 22 | 5 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 9 | 36 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 9 | 0 | 27 | 0 | 0 | 0 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 9 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 25 | 25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 25 |
| 5 | 16 | 0 | 0 | 16 | 16 | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 66 | 0 | 0 | 0 | 33 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 20 | 0 | 20 | 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 40 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 33 | 33 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 0 |
| 9 | 0 | 0 | 0 | 20 | 0 | 0 | 0 | 0 | 0 | 40 | 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 0 |
| 10 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 67 | 1 | 1 | 0 | 0 | 5 | 0 | 1 | 7 | 0 | 1 | 2 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 11 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 13 | 41 | 10 | 6 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 3 | 6 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 3 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 9 | 46 | 12 | 3 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 0 |
| 13 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 12 | 31 | 18 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 12 | 6 |
| 14 | 0 | 0 | 7 | 7 | 0 | 0 | 0 | 0 | 0 | 7 | 0 | 7 | 7 | 23 | 7 | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 0 | 0 | 7 | 0 | 0 | 0 |
| 15 | 0 | 0 | 13 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 6 | 0 | 0 | 0 | 50 | 3 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 3 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 8 | 8 | 16 | 33 | 8 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 |
| 17 | 0 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 18 | 27 | 36 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 9 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 5 | 2 | 0 | 2 | 0 | 0 | 0 | 11 | 0 | 0 | 2 | 2 | 0 | 61 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 0 | 11 | 7 | 3 | 3 | 0 | 0 | 0 | 0 | 7 | 42 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 0 | 10 | 10 | 0 | 0 | 0 | 0 | 10 | 0 | 0 | 10 | 30 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 21 | 10 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 0 | 30 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 33 | 33 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 16 | 0 | 0 | 16 | 0 | 0 | 0 | 0 |
| 23 | 0 | 0 | 25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 50 | 25 | 0 | 0 | 0 | 0 | 0 |
| 24 | 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 20 | 20 | 0 | 0 | 0 | 0 |
| 25 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 14 | 0 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 14 | 0 | 14 | 14 | 0 | 0 | 0 |
| 26 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 25 | 25 | 0 | 25 |
| 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 23 | 7 | 23 | 7 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 15 | 0 | 7 | 0 |
| 28 | 0 | 0 | 7 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 11 | 3 | 3 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 53 | 0 |
| 29 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 20 | 20 | 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 |

Figure 6: Confusion matrix of MAHA, rows index the predicted and columns the actual classes.

In addition, cells of *adjacent classes*, i.e., those which differ only in the head pose by a maximum of 30°, were colored green. It becomes plausible that a high percentage of test samples are associated with these cells regarding the fact that these small differences are difficult to detect, as can be seen in Figure 7. Considering this in the error calculation and including the *adjacent classes* in the set of correct predictions, results in the test error listed in Table 2 under TE*, which for MAHA is only 32.15%.

Figure 7: Prediction example, Left: a sample incorrectly predicted as class 14, Right: an actual sample of class 14.

## 5 CONCLUSIONS

In this paper, a head pose detector was presented that meets the special requirements of automated driving. Since the relative pose of the upper body was of importance in the project within which the work was developed, in addition to the pure yaw angle of the head, a new data set was generated. Conceived for this purpose, a reference data measuring setup with software framework was used to generate data for training and evaluating a neural network. Due to the relatively small amount of data, the performance of this purely supervised trained classifier was, as expected, poorly. Therefore, the data set was enriched by the numerous unlabeled data available from test drives in the project and an approach of semi-supervised learning was developed and optimized. The test result was thus improved by almost 25%. Furthermore, it was found that many of the misclassifications were associated with the so-called *adjacent classes*. In the context of automated driving, one of the strongest motivations for the detection of head poses is the assessment of whether a pedestrian has perceived the driving vehicle or not. It could be demonstrated that the small pose differences between two adjacent classes are often very difficult to identify and have little influence on the determination of whether the vehicle was seen or not. An adjusted test error of only about 32% could be reported.

Overall, it was found that the semi-supervised method used is very well suited to improve the performance of the head pose detector despite a small data set. In the future, further result optimizations can be achieved by more labeled training data. In addition, better performance will be obtained by upgrading the computer's performance and memory capacity.

## REFERENCES


Chen, C., Heili, A., and Odobez, J. (2011). A joint estimation of head and body orientation cues in surveillance video. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 860–867.

Chen, J., Wu, J., Richter, K., Konrad, J., and Ishwar, P. (2016). Estimating head pose orientation using extremely low resolution images. In *2016 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, number 1, pages 65–68. IEEE.

Fanelli, G., Gall, J., and Van Gool, L. (2011). Real time head pose estimation with random regression forests. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 617–624. IEEE.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.

Flohr, F., Dumitru-Guzu, M., Kooij, J. F. P., and Gavrila, D. M. (2015). A probabilistic framework for joint pedestrian head and body orientation estimation. *IEEE Transactions on Intelligent Transportation Systems*, 16(4):1872–1882.

Haeusser, P., Mordvintsev, A., and Cremers, D. (2017). Learning by association - a versatile semi-supervised training method for neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kloeden, H., Brouwer, N., Ries, S., and Rasshofer, R. (2014). Potenzial der Kopfposenerkennung zur Absichtsvorhersage von Fußgängern im urbanen Verkehr. In *FAS Workshop Fahrerassistenzsysteme, Walting, Germany*.

Lee, D., Yang, M.-H., and Oh, S. (2015). Fast and accurate head pose estimation via random projection forests. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1958–1966.

Patacchiola, M. and Cangelosi, A. (2017). Head pose estimation in the wild using convolutional neural networks and adaptive gradient methods. *Pattern Recognition*, 71:132 – 143.

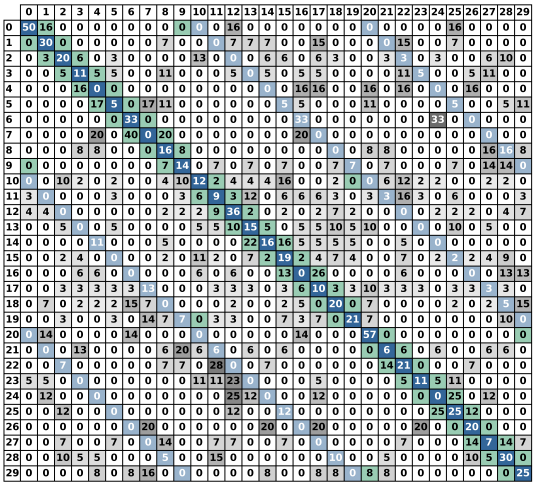
Reh-der, E., Kloeden, H., and Stiller, C. (2014). Head detection and orientation estimation for pedestrian safety. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 2292–2297. IEEE.

Ruiz, N., Chong, E., and Rehg, J. M. (2017). Fine-grained head pose estimation without keypoints. *CoRR*, abs/1710.00925.

Schulz, A. and Stiefelhagen, R. (2012). Video-based pedestrian head pose estimation for risk assessment. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 1771–1776.

Weiss, G. and Provost, F. (2001). The effect of class distribution on classifier learning: An empirical study. Technical report.


# 6 APPENDIX



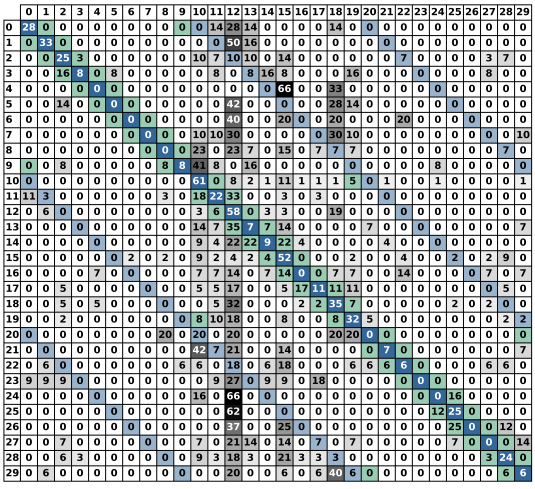


Figure 8: Confusion matrices of the supervised (top) and the cosine (bottom) approach.