9

Distributed Data Acquisition and Control Systems for a Sized Autonomous Vehicle

T. Happek, U. Lang, T. Bockmeier, D. Neubauer and A. Kuznietsov

University of Applied Sciences, Friedberg, Germany
Corresponding author: T. Happek <t.Happek@gmx.net>

Abstract

In this paper, we present an autonomous car with distributed data processing. The car is controlled by a multitude of independent sensors. For lane detection, a camera is used, which detects the lane marks using a Hough transformation. Once the camera detects these, one of them is selected to be followed by the car. This lane is verified by the other sensors of the car. These sensors check the route for obstructions or allow the car to scan a parking space and to park on the roadside if the gap is large enough. The car is built on a scale of 1:10 and shows excellent results on a test track.

Keywords: Edge detection, image processing, microcontrollers, camera.

9.1 Introduction

In modern times, the question of safe traveling becomes more and more important. Most accidents are caused by human failure, so that in many sectors of industry the issue of "autonomous driving" is of increasing interest. An autonomous car will not have problems like being in a bad shape that day or tiredness and will suffer less from reduced visibility due to environmental influences. A car with laser sensors to detect objects on the road, sensors that measure the grip of the road, that calculate speed based on the signals of these sensors and with a fixed reaction time will reduce the number of accidents and related costs.

This chapter describes the project of an autonomous vehicle on a scale of 1:10, which was developed based on decentralized signal routing. The objective of the project is to build an autonomous car that is able to drive autonomously on a scaled road, including the detection of stopping lines, finding a parking area and parking autonomously.

The project is divided into three sections. The first section is about the car itself, the platform of the project. This section describes the sensors of the car, the schematic construction and the signal flow of the car.

The second section is about lane detection, the most important part of the vehicle. Utilizing a camera with several image filters, the lane marks of the road can be extracted from the camera image. This section also describes the calculation of the driving lane for the car.

The control of the vehicle is the matter of the third section. The car runs based on a mathematical model of the car, which calculates the speed and the steering angle of the car in real time, based on the driving lane provided by the camera.

The car is tested on a scaled indoor test track.

9.2 The Testing Environment

Since the car is based on a scaled model car, the test track has to be scaled, too. Therefore, the test track has the same dimensions as the scaled road that is used in a competition for cars on a scale of 1:10 that takes place in Germany every year. As you can see in Figure 9.1, the road has fixed lane marks. This is important, because it's about a prototype. On a real road, several

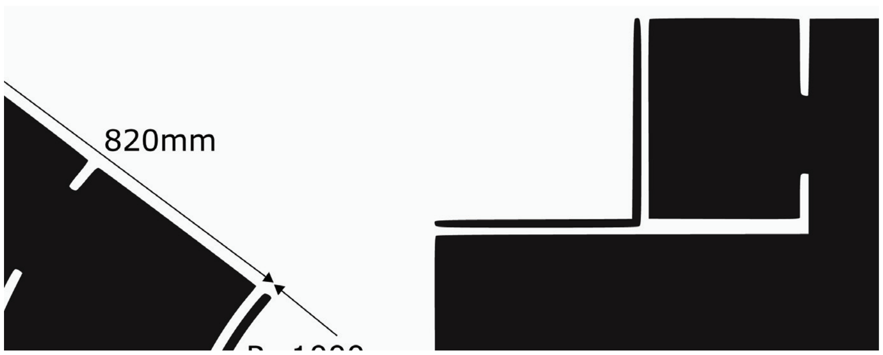

Figure 9.1 Dimensions of the test track.

types of lane marks exist. From white lines, as in the test track, to pillars and a missing center line, every type of lane marks is expected to show up.

In order to simplify the lane detection on the test track, it is assumed, that at all times only one type of lane marks exists. The road has a fixed width and the radius of the curves measure at least 1 meter. The test track has no slope, but a flat surface.

9.3 Description of the System

The basic idea for the vehicle is related to a distributed data acquisition strategy. That means, that all peripherals are not managed by a single microcontroller, but each peripheral has its own microcontroller, which handles the data processing for a specific task. All together are used to analyze the data of the different sensors of the car. Smaller controllers, for instance for the distance sensors or for the camera, are managed by one main controller. The input of the smaller controllers is provided simultaneously via CAN.

The base of the vehicle is a model car scaled 1:10. It includes each mechanical peripheral of a car, like the chassis or the engine. A platform for the control system is added. Figure 9.2 shows the schematic topview of the car with the added platform.

The vehicle itself is equipped with two front boards, a side board, a rear board and the motherboard. All of these boards have one microcontroller for

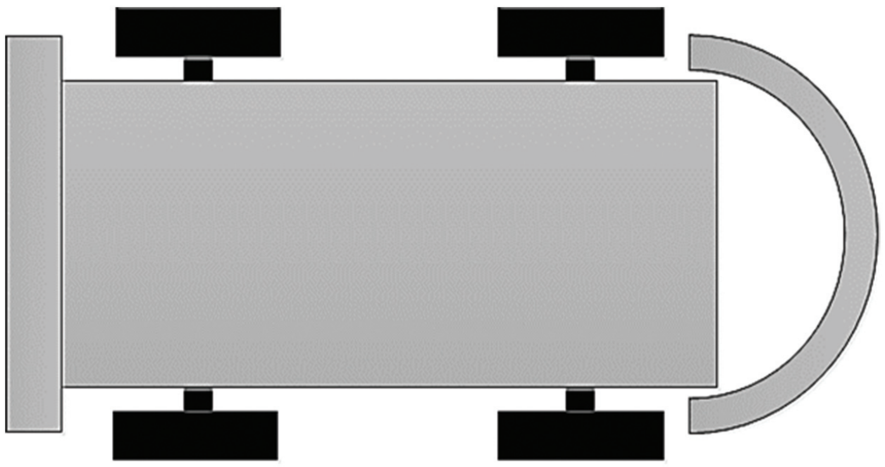

Figure 9.2 Schematic base of the model car.

data analysis that is positioned next to the sensors. The front boards provide infrared sensors for object tracking in the distance.

The infrared sensors of the side board have the task of finding a parking space and transmitting the information via CAN bus to the main controller, which undertakes the control of parking supported by the information it gets from the sensors in the front and back of the car.

The rear board is equipped with infrared sensors, too. It serves the back of the vehicle only. That guarantees a safe distance to all objects in the back. The microcontrollers are responsible for the data processing of each board and send the information to the main controller via CAN bus. Each of the microcontrollers reacts on the incoming input signals of the corresponding sensors according to its implemented control. The infrared sensors are distributed alongside the car as you can see in Figure 9.3.

The motherboard with the integrated main controller is the main access point of the car. It provides the CAN bus connection, the power supply for the other boards and external sensors of the car. But primarily it’s the central communications point of the car and manages the information that comes from the peripheral boards, including the data from the external sensors, the control signals for the engine and the servo for the starring angle.

The motherboard gets its power supply from three 5 V batteries. With these three batteries, the model car is able to drive about one hour autonomously.

The main task for the main controller is the control of the vehicle. It calculates the speed of the car and the starring angle based on a mathematical model of the car and the information of the sensors. The external engine driver sets the speed via PWM. The starring angle of the car is adjusted by the front wheels. An additional servo controls the wheel’s angle.

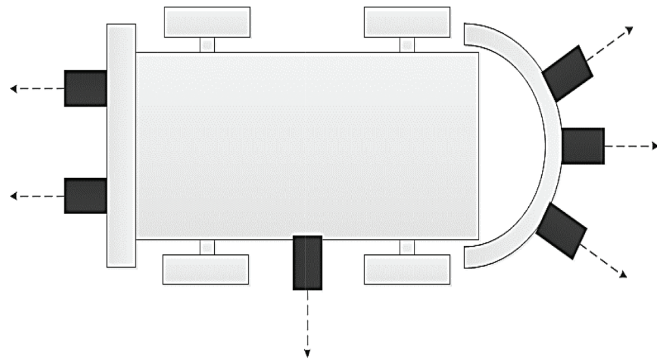
Figure 9.3 The infrared sensors distributed alongside the car.

The camera and its lane detection is the most important component of the vehicle. It is installed in the middle of the front of the car, see Figure 9.4. The viewing angle is important for the position of the camera. If the viewing angle is too small, the pictures of the camera show a near area in front of the car only, but not the area in the middle distance. If the viewing angle is too big, the camera shows a big area in front of the car indicating near and far distances, but the information of the road is so condensed, that an efficient lane detection isn't possible. The angle depends also on the height of the camera and the numerical aperture of the lens. The higher the camera is positioned, the smaller the viewing angle. For this project, the camera has a height of 30 cm and a viewing angle of 35 degrees. The height and the angle of the camera are based on experimental research.

Figure 9.5 shows the reduced signal flow of the vehicle. The information from the infrared sensors is sent to a small microcontroller, as it is visualized by the spotted lines. In reality, each sensor has its own microcontroller, but to reduce the complexity of the graphic, they were shown as one. The camera has its own microcontroller. This controller must be able to accomplish the necessary calculations for lane detection in time. For the control of the vehicle by the main controller, it is necessary that all information from all other controllers are actualized in one calculation step, this is needed for the mathematical model of the car. The main controller gathers the analyzed data provided by the smaller microcontrollers, the data from the camera

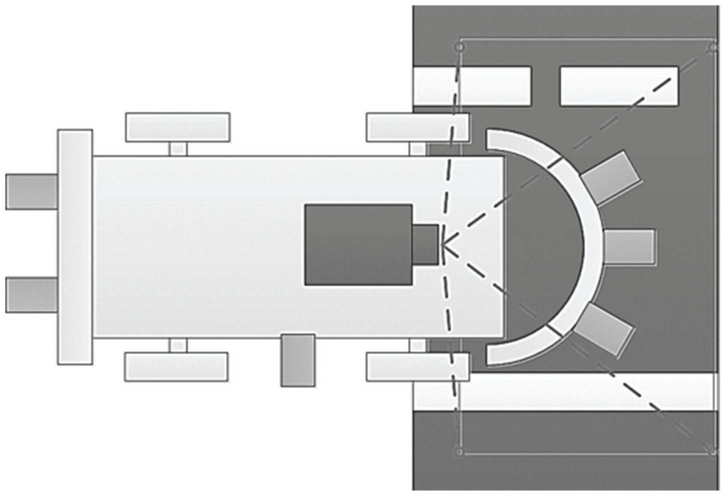
Figure 9.4 Position of the camera.

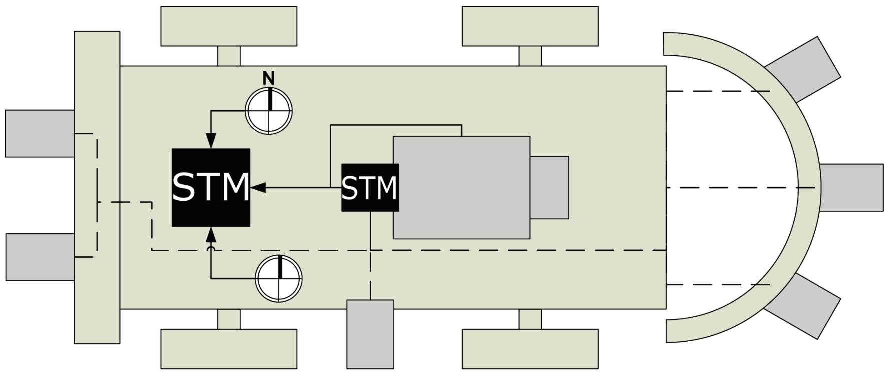

Figure 9.5 Schematic signal flow of the vehicle.

about the driving lane and the information from other sensors like gyroscope and accelerometer for its calculation. The essential signal flow of all these components to the main controller is visualized by the solid lines in Figure 9.5. After its calculation, the main controller sends control signals to the engine and the servo, which controls the starring angle of the car.

Incremental encoders on the rear wheels detect the actual speed and calculate the path the vehicle has traveled during the last calculation step of the mathematical model. The sensors send the data via CAN bus to the main controller. The vehicle is front-engined, so traction of the rear wheels is ensured. Potential error in measurement through spinning is avoided.

There are two modules that do not communicate via CAN bus with the main controller: the first one is the camera, ensuring that the vehicle keeps the track, the second is a sensor module, which includes the gyroscope and accelerometer. Both modules do not have a CAN interface, but they communicate via an USART interface with the main microcontroller.

In the future, the focus will be on an interactive network of several independent vehicles based on radio transmission. This will allow all vehicles to communicate with each other and share information like traction and behavior of the road, actual position from GPS, or speed. The radio transmission is carried out with the industry standard called "Zigbee". An XBEE module of the company "Digi" undertakes the radio transmission. The module uses an UART interface for the communication with the main microcontroller on the vehicle. Via this interface, the car will get information from other cars nearby. A detailed overview of the data processing system, including the XBEE module, is shown in Figure 9.6.

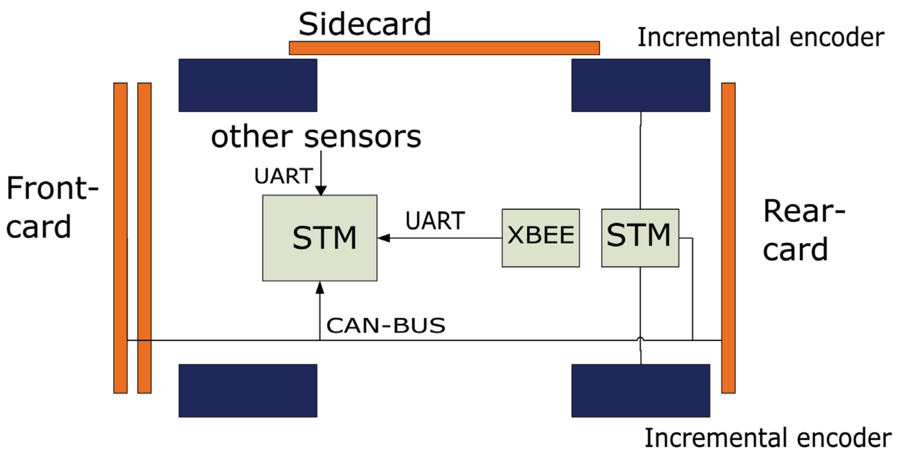

Figure 9.6 Overview of the data processing system.

9.4 Lane Detection

There are several steps needed to accomplish the lane detection.

First, the image has to be analyzed with an In-Range filter. In the second step, the points that the Hough-transformation has identified as lane marks, are divided into left and right lane marks. Next, the least squares method is used to transform the lane marks into a second-degree polynomial, thus providing the base to calculate the driving lane. Subsequently, the points of the driving lane are transformed into world coordinates.

Two types of filters are used to get the needed information from the image. Both are functions from the OpenCV-library. An In-Range filter is used to detect the white lane marks on the defined test track. The Hough-transformation calculates the exact position of the lane marks preparing them for the next steps.

9.4.1 In-Range Filter

The In-Range filter transforms an RGB-image into an 8-bit binary image. It's made for the detection of pixels in a variable color range. The transformed picture has the same resolution as the original picture. Pixels belonging to the chosen color range are white. All other pixels in the image are black. The function works with the individual values of the RGB format. The chosen color is defined by two critical values of this format.

Figure 9.7 shows the result of the In-Range filter.

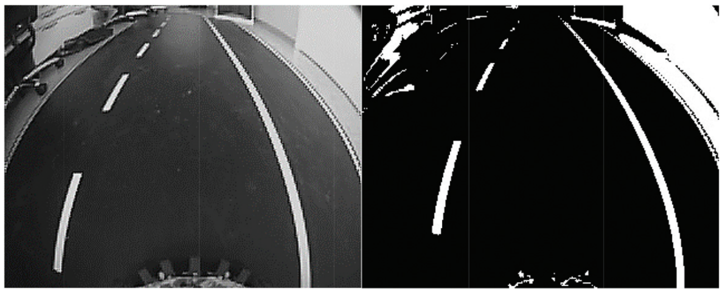

Figure 9.7 Comparison between original and in-range image.

9.4.2 Hough-Transformation

The Hough-transformation is an algorithm to detect lines or circles in images, which in this case means that it investigates the binary image from the In-Range filter in order to find the lane marks.

The Hessian normal form converts individual pixels, so that they can be recognized as lines in the Hough space. In this state, space lines are expressed by the distance to the point of origin and the angle to one of the axes. Due to the fact that the exact angle of the marks is unknown, the distance to the point of origin is calculated based on Equation (9.1), utilizing the most probable angles:

$$r = x \cdot \cos(a) + y \cdot \sin(a)\,. \tag{9.1}$$

The intersection of the sinusoidals provides an angle and the distance of the straight line from the origin of coordinates. These parameters create a new line, so that the majority of the pixels can be detected. Furthermore, the function from the OpenCV-library returns the start and the endpoint of each Hough-line. As Figure 9.8 shows, the lines of the Hough-transformation are precisely mapped on the lane marks of the road.

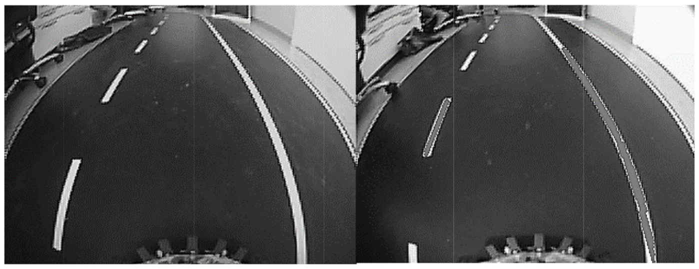

Figure 9.8 Original image without and with Hough-lines.

9.4.3 Lane Marks

To provide a more precise calculation, all points along the line are included. These points are stored in two arrays and then sorted. As a first sorting criterion, the position of the last driving lane is used. The second criterion for sorting derives from their position in the image.

As mentioned before, the information in the image regarding long distances can be critical depending on the viewing angle and height of the camera. In order to concentrate on noncritical information only, points in the middle area of the image are used. Figure 9.9 shows the sorted points on the right and the corresponding Hough-lines on the left side.

9.4.4 Polynomial

To describe the lane marks more efficiently, a second-degree polynomial is used. The coefficients of the parable are derived by the least-squares method. A polynomial of a higher degree isn't needed, because the effort to calculate the coefficients is too high to make sense in this context, for the speed of the image processing is one of the critical points of the project. Furthermore, the area of the road, which is pictured by the camera, is too small. The road is unable to clone the typical form of a third-degree polynomial.

As visible in Figure 9.10, the parables derived from the sorted points are mapped precisely on the lane marks of the road. The algorithm to calculate the coefficients derived from the points of the lane marks is handwritten.

9.4.5 Driving Lane

The driving lane for the car lies between the parables mentioned in the last chapter. To calculate the position of the points of the driving lane, the average

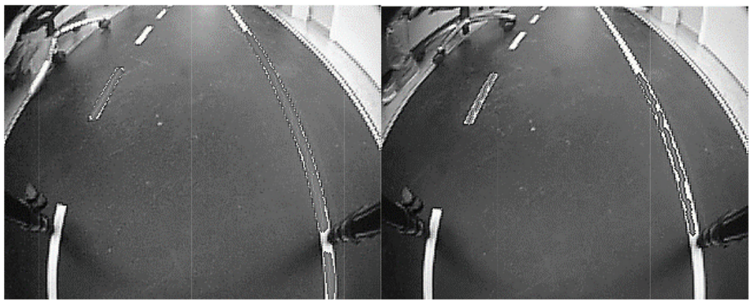

Figure 9.9 Hough-Lines and sorted points along the Hough-Lines.

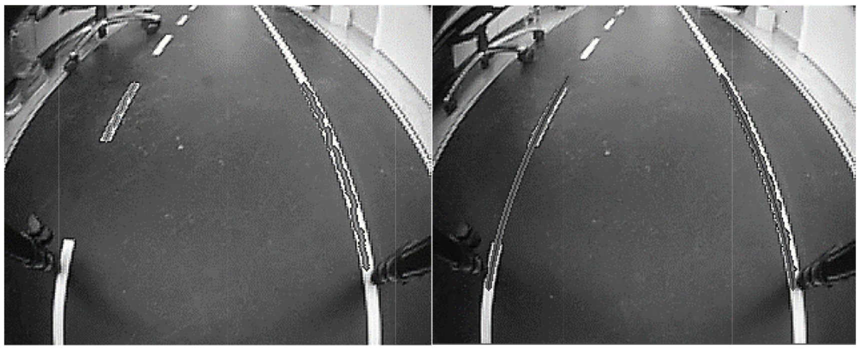

Figure 9.10 Sorted Points and Least-Square Parable.

of two opponent points of the two parables is taken. According to 9.2, the average for the x- and y-coordinates is calculated.

$$\begin{pmatrix} x_m \\ y_m \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \frac{1}{2}\left(\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}\right). \tag{9.2}$$

In order to simplify the transformation from pixel-coordinates to world-coordinates, the driving lane is described by a fixed number of points in the image. The essential feature of these points is that they lie in predefined rows in the image. So, there is only the need to calculate the horizontal position of the parable for these points.

Theoretically it is possible that the program delivers an incorrect driving lane. Mistakes can occur because of flash lights, reflections on the road, missing lane marks due to different reasons or extreme light conditions, which are much faster than the auto white balance of the camera can bear. So in order to avoid mistakes that occur within a short time period, some kind of stabilization is required. Short time in this case means shorter than one second.

For the purpose of stabilization, the different driving points are stored. The stabilization works with these stored points in combination with four defined edge points in the image. First, the algorithm checks if the edge points of the new image differ from the edge points in the old image.

If the difference between the old points and the new points is low, the driving lane is calculated and the driving points are stored. In case that the difference between the points is too big, the driving lane is not updated and the driving lane is calculated by using the stored points. The algorithm works with the changes of the stored points. The new points are calculated by using the difference between the last image and the current one. This difference is derived from the change of the difference between the third and second image, that have been taken before the current one, and the difference between the second and the first image before the current one.

The critical values for the difference also depend on this calculation. That means that in curvas, the critical values are higher. If not, only the last three images are used for the calculation, in order to reduce the noise of the driving lane. However, in this case, the reaction time of the algorithm is lesser.

The reaction time also depends on the fps (frames per second) of the camera. For this project, a camera with 100 fps is used and the last fifteen driving lanes are stored. The number of stored driving lanes for 100 fps is based on experimental research.

Figure 9.10 shows the driving lane in red color. The four edge points mark the edge points of the rectangle.

9.4.6 Stop Line

One of the main tasks of the camera is to detect stop lines. Figure 9.11 shows the dimensions of the stop lines for this test track.

In order to detect stop lines, the algorithm is searching for the main characteristics. First, the stop line is a horizontal line in the image. If the angle of a vertical line in the image is defined as zero degrees, that means, that the perfect stop line has an angle of 90 degree. The algorithm not only searches for 90 degree lines. The angle for a potential stop line is smaller than –75 degree and bigger than +75 degree.

The next criterion of a stop line is that it lies on the car traffic lane. So, the algorithm does not need to search in the complete image for stop lines, but only in the area of the cars traffic lane. This area is marked by the four edge points of the rectangle mentioned in the last chapter. Once the algorithm

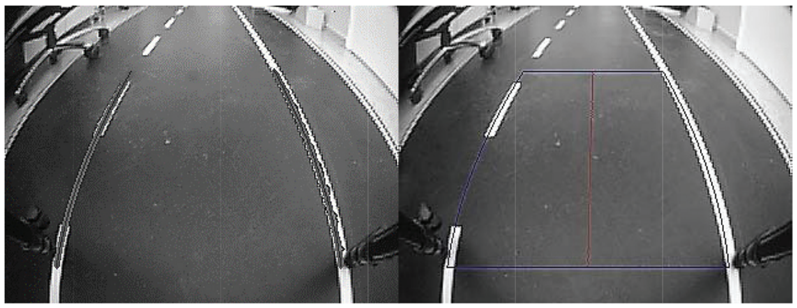

Figure 9.11 Parables and driving lane.

finds a potential stop line in the right area with a correct angle, the algorithm checks the next two characteristics of a stop line: the length and the width of the stop line.

The length of the stop line is easy to check. The stop line must be as long as the road is wide, so the algorithm only needs to check the endpoints of the line. On the left side, the endpoint of the stop line must lie on the middle road marking. On the right side, the stop line borders on the left road marking from the crossing road. The stop line and the road marking differ in just one point: the width.

Since it is not possible to perceive the differences of the width in each situation, the stop line has no defined end point on this side. So, the algorithm checks if the end point of the potential stop line lies on or above the right road marking. It is hard to measure the width of a line in an image that has constant width and length in reality. The width of the line in the image in pixels depends on the camera position in relation to the line, the numerical aperture of the camera lens and the resolution of the camera. So, because in this project the position of the camera changes from time to time, measuring the width is not reliable to perceive the stop line. Therefore, the width is not used as a criterion for stop lines.

Figure 9.12 shows a typical crossing situation. The left image visualizes the basic situation and the middle image shows the search area as a rectangle. Here you can see that the stop line on the left side is not covered by the research area so the algorithm doesn't recognize the line as a stop line. On the right image, the stop line ends correctly on the middle road marking. The line in the image shows that the algorithm has found a stop line. Due to the left road marking from the crossing road, the line ends outside the real stop line.

9.4.7 Coordinate Transformation

To control the car, the lateral deviation and the course angle are needed. Both are calculated by the controller of the camera. The scale unit for the

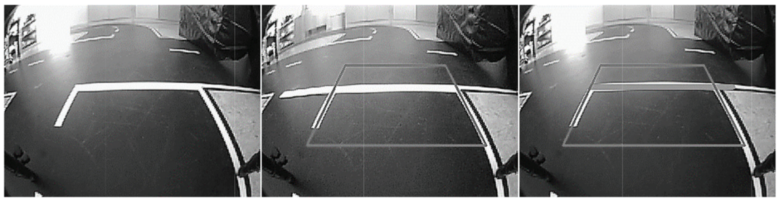

Figure 9.12 Detection of stop lines.

lateral deviation is meters and degrees for the course angle. Course angle means the angle of the driving lane which is calculated by the camera. The lateral deviation is the distance of the car's center of gravity to the driving lane when they are at the same level. Since the lateral deviation is needed in meters, the algorithm has to convert the pixel coordinates from the image into meters in the real world. The course angle can be calculated from the pixel coordinates in the image, but this method is error-prone.

There are two different methods to convert the pixels into meters.

Pixels can be converted via Equations (9.3) and (9.4).

$$x(u,v) = \frac{h}{\tan\left[(\bar{\theta}-\alpha)+u\frac{2\alpha}{n-1}\right]} \cdot \cos\left[(\bar{\gamma}-\alpha)+u\frac{2\alpha}{n-1}\right]+l, \tag{9.3}$$

$$x(u,v) = \frac{h}{\tan\left[(\bar{\theta}-\alpha)+u\frac{2\alpha}{n-1}\right]} \cdot \cos\left[(\bar{\gamma}-\alpha)+u\frac{2\alpha}{n-1}\right]+l. \tag{9.4}$$

In the equations, *x* and *y* are the coordinates in meters. $\overline{\gamma}$ stands for the drift angle of the camera in the plane area and $\overline{\theta}$ stands for the pitch angle of the camera. α is the numerical aperture of the camera, *u* and *v* are the coordinates of one pixel in the image.

Using this equation, the complete image can be converted into real-world coordinates. The drawback of this method is that all parameters of the camera have to be known exactly; every difference between the numerical aperture in the equation and the exact physical aperture of the camera lens can cause massive failure in the calculation. Furthermore, this method needs more calculation time on the target hardware. A big plus of this method is that the camera can be re-positioned during experimental research.

The second method is to store references to some pixels in lookup tables. For these pixels, the corresponding values in meters can be calculated or can be measured. This method expends much less calculation time but is also much less precise. With this method, the camera cannot be re-positioned during experiment research. Every time the camera is re-positioned the reference tables must be re-calculated.

The method to prefer depends on the project requirements regarding accuracy and the projects hardware. For this project, the second method is used. To meet the demands on accuracy, for each tenth pixel of the camera, a reference is stored.

9.5 Control of the Vehicle

The driving dynamic of the vehicle is characterized by the linear track model of Ackermann. As Figure 9.13(a) shows, the model is composed of a rear and front wheel which are connected by an axe. In order to rotate the vehicle on its main axe, the steering angle can be set with the front wheel.

To reduce the complexity of vehicle dynamics, three simplifications are made.

These are:

- Neglect the air resistance, because the vehicle speed is very low;
- Lateral forces on the wheels are linearized;
- No roll of the vehicle about the x and y axis.

Using these simplifications, the created model should differ only marginally from reality. Linking the transverse dynamics of the vehicle with the driving dynamics, you can derive the following relation:

$$r \begin{bmatrix} \ddot{\psi}(t) \\ \dot{\theta}_{\Delta}(t) \\ \dot{\gamma}(t) \end{bmatrix} = \begin{bmatrix} a_{22} & 0 & 0 \\ -1 & 0 & 0 \\ 0 & V & 0 \end{bmatrix} \cdot \begin{bmatrix} \dot{\psi}(t) \\ \theta_{\Delta}(t) \\ \gamma(t) \end{bmatrix} + \begin{bmatrix} b_2 \\ 0 \\ 0 \end{bmatrix} \cdot \delta(t). \qquad (9.5)$$

Because of the equation in the state space, a controller can be designed using tools such as Matlab Simulink or Scilab X-cos.

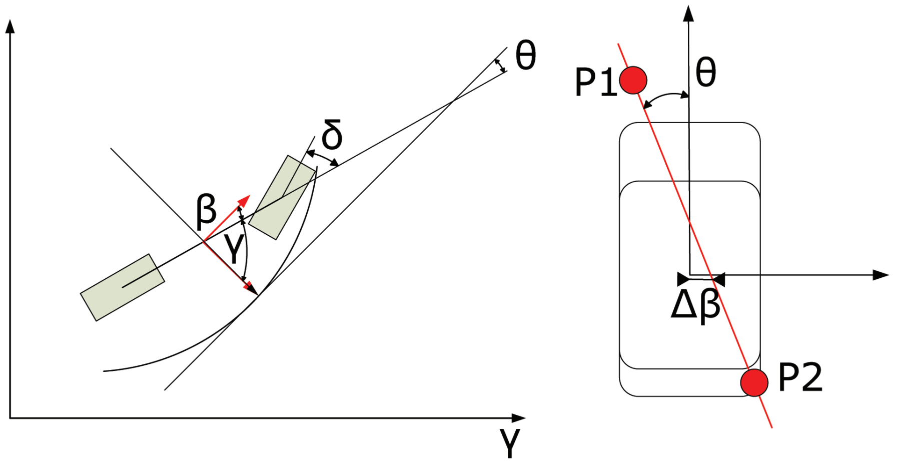

Figure 9.13 Driving along a set path: Track model (a); Lateral deviation and heading angle (b).

In order to keep the vehicle on the track, the conditions such as heading angle, the yaw rate and lateral deviation must be known. A gyroscope is used to detect the yaw rate of the vehicle. The lateral deviation and the course angle are calculated from the camera. The camera sends the lateral deviation and the curse angle. Until the next image is analyzed, the coordinates on the microcontroller stay the same as before. Between two pictures the lock angle and the transverse deviation were recalculated after each motion. This is possible because the velocity and yaw rate are known at any time. Figure 9.13(b) illustrates the relationship of lateral deviation (ΔY) and heading angle (α).

9.6 Results

This section gives an overview of the project results.

The autonomous car was built with the hardware suggested before. Experiments on scaled test roads show that the car can drive autonomously. However, the tests also showed the limitations of this prototype. The effort for the image processing was undervalued. The on-board processor of the camera isn't able to accomplish the necessary calculations in time. In terms of reaction to this fact, the maximum speed of the car has to be very slow. If it isn't, the control of the vehicle gets unstable with a more or less random driving path. In addition, the car has problems with too sharp curves. The process to divide the image isn't dynamic, so in curves the preset section becomes incorrect and the algorithm isn't able to calculate a correct driving path. Thanks to its laser sensors, the car is able to avoid collisions with baffles.

To improve the performance of the car, the hardware for the image processing has to be improved. The image processing works stable. Problems derive from the calculation algorithm of the driving path. At this point in time, the algorithm doesn't contain the necessary interrupts for every situation on the road, but this drawback will be corrected in the second prototype.

9.7 Conclusions

In this chapter, an autonomous vehicle with distributed data acquisition and control systems has been presented. For control, the vehicle has a number of independent sensors. The main sensor is a camera with a lane tracking algorithm, which contains edge detection and Hough transformation. The lane is verified by laser sensors in the front and side of the vehicle. It is planned to build a superordinate control system, which leads a group of autonomous vehicles using a wireless communication protocol.

References

[1] J. Canny, 'A Computational Approach to Edge Detection', IEEE 1986.

[2] A. Erhardt, 'Einführung in die Digitale Bildverarbeitung', Offenburg 2008.

[3] W. Heiden, 'Kanten in Bildern - Filterung und Kantenerkennung', St. Augustin 2009.

[4] B. Jähne, 'Digitale Bildverarbeitung und Bildgewinnung', Berlin 2013.

[5] M. Jauernig, 'Einsatz von Algorithmen der Photogrammmetrie und Bildverarbeitung zur Einblendung spezifischer Lichtraumprofile in Videosequenzen', Hochschule Leipzig 2006.

[6] A. Kant, 'Bildverarbeitungsmodul zur Fahrspurerkennung für ein autonomes Fahrzeug', Hochschule Hamburg 2007.

[7] L. F. Kirk, 'Schätzung der Brennweite mit Hilfe der Hough-Transformation', Fachhochschule, Köln 2011.

[8] N. Kruse, 'Kameragestützte Fahrspurerkennung für autonome Modellfahrzeuge', Universität Hamburg 2008.

[9] G. Linß, 'Praktische Ausbildung und Training Qualitätsmanagement Objekterkennung mit Hough-Transformation', Technische Universität Ilmenau.

[10] R. Maini, H. Aggarwal, 'Study and Comparison of Various Image Edge Detection Techniques', Punjabi University, India.

[11] P. Schöley, Kantendetektoren, Technische Universität Dresden, 2011.

[12] J. Unger, 'Untersuchung von Linien und Kantenextraktionsalgorithmen im Rahmen der Verifikation von Ackerlandobjekten', Universität Hannover, 2009.

[13] C. Wagner, Kantenextraktion – Klassische Verfahren, Universität, Ulm, 2006.

[14] G. M. Wagner, G. M., 'Bestimmung der Kameraverzerrung mit Hilfe der Hough-Transformation', Fachhochschule Köln, 2011.

[15] L. Bergen, H. Burkhardt, Morphological image processing.

[16] J. Wohlfeil, 'Detection and tracking of vehicles with a moving camera' Humbolt-Universität zu Berlin Institut für Informatik; Deutsches Zentrum für Luft- und Raumfahrt Institut für Verkehrsforschung, 2006.

[17] B. Hulin 'Video-based obstacle detection clearance of the pantograph electric railways', Fakultät für Elektrotechnik und Informationstechnick, Technische Universität München, 2003.

[18] J. Alberts, 'Vision for the project FAUST, Focus of Expansion/optical flow', Hamburg University of Applied Sciences, 2007.

[19] Tutorial Filter; Vision&Control System Lighting Optics, 2007.

[20] Prof. Dr. –Ing. Habil G. Linß, 'Practical education and training quality management, object detection with Hough transform', Technische Universität Ilmenau, Department of Mechanical Engineering, Department of Quality Assurance.

[21] C. Wagner, Edge extraction, Classical methods; Seminar presentation on "Image segmentation and computer Vision", 2006.

[22] I. Nikolov, 'Adaptive camera parameters for optimum lane detection and tracking', Hamburg University of Applied Sciences, 2007.

[23] V. Schaefer, A. Zipser, 'Algorithmic Applications Hough transform', 2006.

[24] C.Rathemacher, 'GPU-based detection of algebraic structures writable', Fachhochschule Wiesbaden, Fachbereich Design Informatik Medien, 2007.

[25] M. Pelkofer, 'Behavioral decision for autonomous vehicles with gaze control', Universität der Bundeswehr München, Fakultät für Luft- und Raumfahrttechnik, Institut für Systemdynamik und Flugmechanik, 2003.

[26] D.Berger, 'Lane inference system for lane algorithm Three Feature Based Lane Detection Algorithm (TFALDA)', Hamburg University of Applied Sciences, 2008.

[27] Alberto Broggi, 'A Massively Parallel Approach to Real-Time Vision-Based Road Markings Detection', Dipartimento di ingengneria dell'Information Universita di Parma.

[28] Andreas Weide, 'Entwicklung einer Kameragestützten Fahrspurerkennung für ein autonomes Fahrzeug', University of Applied Sciences (Friedberg, Germany), 2013.