Sensors & Transducers, Vol. 187, Issue 4, April 2015, pp. 17-22

Sensors & Transducers

http://www.sensorsportal.com

Distributed Data Logging and Intelligent Control Strategies for a Scaled Autonomous Vehicle

Tilman Happek, Uwe Lang, Torben Bockmeier, Dimitrji Neubauer and Alexander Kuznietsov
University of Applied Sciences, Friedberg, Germany
E-mail: tilman.happek@iem.thm.de

Abstract: In this paper we present an autonomous car with distributed data processing. The car is controlled by a multitude of independent sensors. For the lane detection, a camera is used, which detects the lane marks with a Hough transformation. Once the camera detects these, one of them is calculated 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. *Copyright © 2015 IFSA Publishing, S. L.*

Keywords: Edge detection, Image processing, Microcontrollers, Camera.

1. Introduction

In modern times the question of save traveling becomes more and more important. Most accidents are caused by human failure, so that in many sectors of industry the issue "autonomous driving" is of increasing interest.

An autonomous car will not have problems like a bad form on the day or tiredness and will suffer less from a reduced visibility due to environment influences. A car with laser sensors to detect objects on the road, sensors that measure the grip of the road, that calculates speed based on the signals of these sensors and with a fixed reaction time will reduce the number of accidents and related costs.

2. Description of the System

Decentralized signal routing means that all data processing of all peripherals is not only managed by a single microcontroller, but works with the principle "division of labour". Different controllers handle specific tasks for the data processing and send the information simultaneously via a CAN bus to the main controller (in this case STM32F103ZCT6, 144-pin, 72 MHz), which carries out an evaluation of the data.

The vehicle itself is equipped with two front boards, a side board, a rear board and the motherboard (Fig. 1). The front boards consisting of a board with ultrasonic sensors and infrared sensors are for the realization of objects in the distance. The main task of the side board with the infrared sensors is to find a parking space and transmit the information via the CAN bus to the main controller, which undertakes the control for the parking. The rear board, which is equipped with ultrasonic sensors, too, serves the back of the vehicle only. That guarantees a safe distance from all objects in the back. The microcontroller STM32F103C8T6 is responsible for the data processing of each board and sends the information to the main controller via the CAN-BUS. It reacts on the incoming input signals of the corresponding sensors due to its implemented control.

The motherboard with the integrated main controller (STM32F103ZCT6) is the „heart“ of the vehicle and therefore the interface for all other peripheral components. It provides, in addition to the CAN bus connection, the power supply for the other boards and components, for example the camera CMUcam3. In order to detect the speed of the vehicle or the way which the vehicle has traveled, there is an incremental encoder on the rear wheels used to send the actual data for speed and distance of the microcontroller STM32F103C8T6 via CAN-BUS to the main controller. The encoders were mounted on the rear axle because the vehicle has a front-wheel-drive, whereby the traction of the rear wheels is ensured and a spinning is avoided.

The camera is the „eye“ of the vehicle that ensures that the vehicle keeps the track. It is supported by the 9DOF module (consisting of gyroscope and accelerometer) that also has an USART interface connection to the main controller.

In future the focus will be on a control of one or several independent vehicles by radio transmission based on a computer and camera surveillance. The radio transmission is carried out with the industry standard "ZigBee". An XBEE module of the company "Digi" takes over the radio transmission and sends the data to the main controller by a UART interface to control the vehicle.

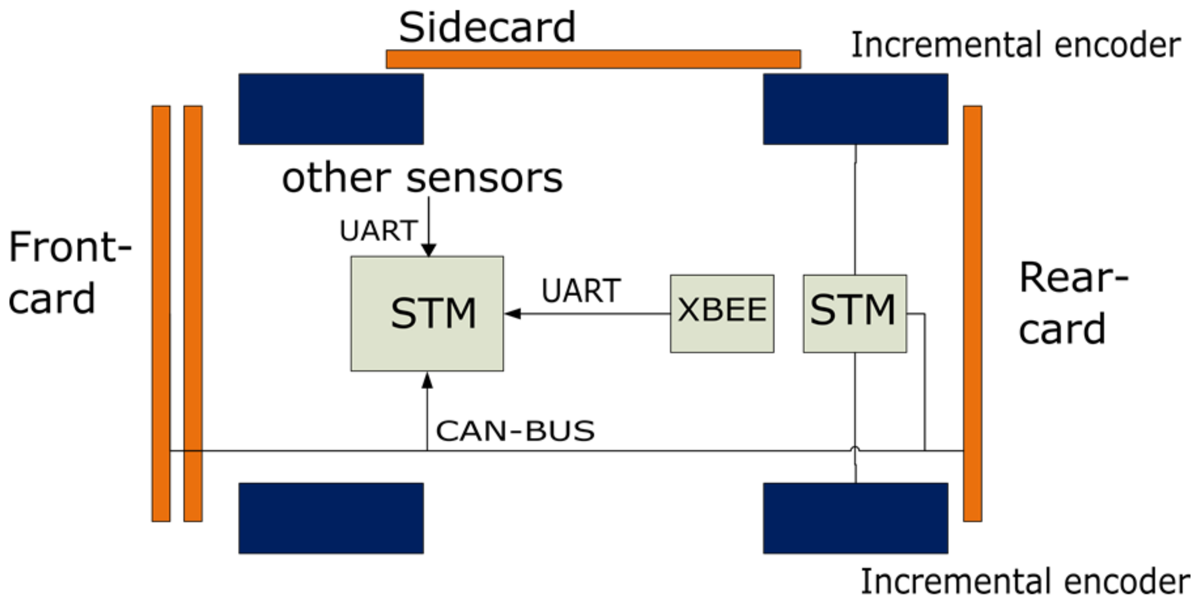


Fig. 1. Overview of the data processing system in vehicle.

3. 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 of 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.

3.1. 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.

Fig. 2 shows the result of the In-Range filter.

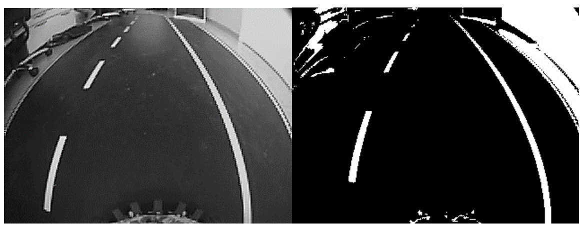


Fig. 2. Comparison between original and In-range image.

3.2. Hough-transformation

The Hough-transformation is an algorithm to detect lines or circles in images, that means in this case, 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 formula 1., utilizing the most probably angles:

$$r = x * \cos(a) + y * \sin(a), \qquad (1)$$

The intersection of the sinuses 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 Fig. 3 shows, the lines of the Hough-transformation are precisely mapped on the lane marks of the road.

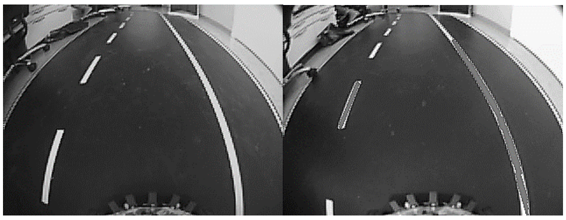


Fig. 3. Original image without and with Hough-lines.

3.3. 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 first criterion for the sort, 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 uncritical information only, only points in the middle area of the image are used. Fig. 4 shows the sorted points on the right and the corresponding Hough-lines on the left side.

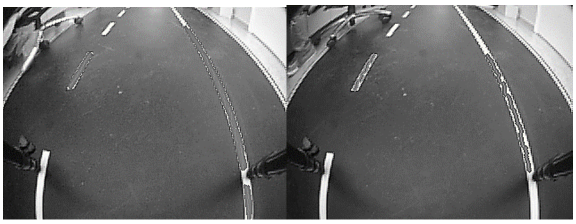


Fig. 4. Hough-lines and sorted points along the Hough-lines.

3.4. Polynomial

To describe the lane marks more efficiently, a second-degree polynomial is used. The coefficients of the parable are derived from the least-square 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 typically form of a third-degree polynomial.

As visible in Fig. 5 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.

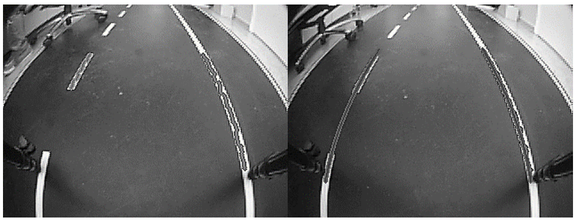


Fig. 5. Sorted points and least-square parable.

3.5. 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 of two opponent points of the two parables is taken. According to 2 the average for the x- and y-coordinates is calculated.

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

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 stabilization the several 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 changings 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 curves the critical values are higher. If not, only the three last 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 lower.

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.

Fig. 6. shows the driving lane in red color. The four edge points mark the edge points of the rectangle.

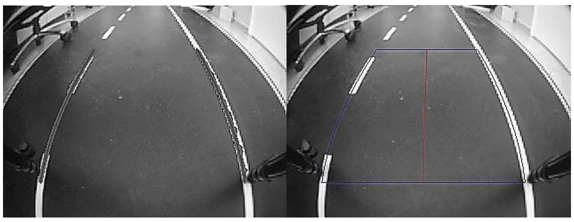


Fig. 6. Parables and driving lane.

3.6. 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 lateral deviation is meter and degree 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 meter, 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 pixel into meter. Pixels can be converted via formula 3 and 4.

$$x = \frac{h * \cos\left[y - a + u \frac{2a}{n-1} \right]}{\tan\left[\Theta - a + u \frac{2a}{n-1} \right]} + l \,, \tag{3}$$

$$y = \frac{h * \sin\left[y - a + u \frac{2a}{n-1} \right]}{\tan\left[\Theta - a + u \frac{2a}{n-1} \right]} + d \,, \tag{4}$$

In the formulas x and y are the coordinates in meter. $\bar{\gamma}$ stands for the drift angle of the camera in the plane area and $\bar{\theta}$ 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 formula 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 formula 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 meter 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.

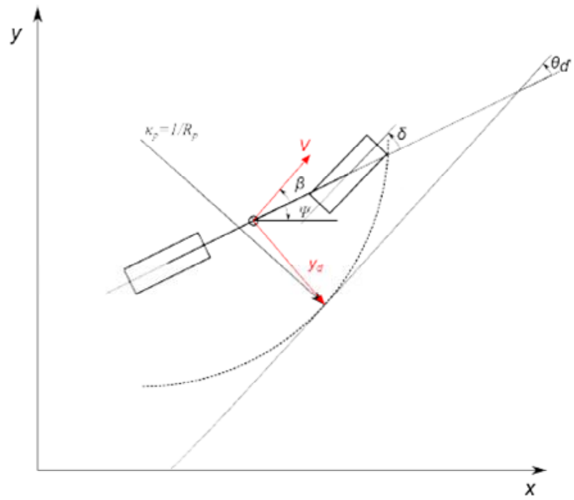
The method to prefer depends on the project requirements on accuracy and the projects hardware. For this project the second method is used. To meet the demands on accuracy, for each tenth pixel of the camera a reference is stored.

4. Control of the Vehicle

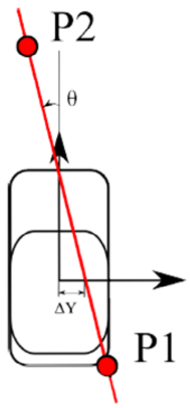
The driving dynamic of the vehicle is characterized by the linear track model of Ackermann. As Fig. 7 (a) shows, the model is composed of a rear and front wheel which is connected by an axe. In order to rotate the vehicle on its main axe, you can set the steering angle with the front wheel.

To reduce the complexity of vehicle dynamics, 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.



(a) track model



(b) lateral deviation and heading angle

Fig. 4. Driving along a set path.

5. Conclusions

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:

$$\begin{bmatrix} \ddot{\psi}(t) \\ \dot{\theta}_{\Delta}(t) \\ \dot{\gamma}(t) \end{bmatrix} = \begin{bmatrix} a_{22} & 0 & 0 \\ -1 & 0 & 0 \\ 0 & V & 0 \end{bmatrix} * \begin{bmatrix} \dot{\psi}(t) \\ \theta_{\Delta}(t) \\ \gamma(t) \end{bmatrix} + \begin{bmatrix} b_2 \\ 0 \\ 0 \end{bmatrix} * \delta(t), \quad (5)$$

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

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 data. The camera sends the pixel coordinates that the vehicle is intended to drive. To transform the pixel coordinates to world coordinates, the pixel coordinates of the camera are stored in a table in reference to the measured world coordinates. 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. Fig. 7 (b) illustrates the relationship of lateral deviation (ΔY) and heading angle (α).

An autonomous vehicle with distributed data and acquisition and control systems has been presented in this paper. For controlling, the vehicle has an amount of independent sensors. The main sensor is a camera with 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's planned to build up a superordinate control system, which leads an amount of autonomous vehicles over a wireless communication protocol. All algorithms for the image processing and the control of the car are handwritten. The algorithms don't contain software modules from third persons

References

[1]. L. Bergen, H. Burkhardt, Morphological image processing, *Institute for Information*, *University of Freiburg* (in German).

[2]. 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 (in German).

[3]. B. Hulin, Video-based obstacle detection clearance of the pantograph electric railways, Fakultät für Elektrotechnik und Informationstechnik, *Technische Universität München*, 2003 (in German).

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

[5]. Tutorial Filter, *Vision&Control System Lighting Optics*, 2007.

[6]. Habil G. Linß, Object detection with Hough transform, *Technische Universität Ilmenau, Department of Mechanical Engineering, Department of Quality Assurance* 2010 (in German).

[7]. C. Wagner, Edge extraction, Classical methods, *Seminar Presentation on Image Segmentation and Computer Vision*, 2006.

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

[9]. V. Schaefer, A. Zipser, Algorithmic Applications Hough transform, *Institute for Information*, *University of Applied Sciences Köln*, 2006 (in German).

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

[11]. 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.

[12]. D. Berger, Lane inference system for lane algorithm Three Feature Based Lane Detection Algorithm

(TFALDA), *Hamburg University of Applied Sciences*, 200.

[13]. T. Happek, Distributed data acquisition and control systems for a sized autonomous vehicle, in *Proceedings of the 7th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS, 2013)*, 12-14 September 2013, Berlin, Germany, pp. 437-440.
