TRANSPORT PROBLEMS 2024 Volume 19 Issue 1
PROBLEMY TRANSPORTU DOI: 10.20858/tp.2024.19.1.04

Keywords: lane tracking, visual SLAM, neural network application

Maciej KOZŁOWSKI[1]*, Andrzej CZEREPICKI[2], Piotr DZIDO[3]

A LANE TRACKING ALGORITHM FOR LOW-COMPUTATIONAL-POWER MICROCONTROLLER-CONTROLLED AUTONOMOUS VEHICLE MODELS

Summary. At work, three tasks were presented: road lane detection and trajectory estimation, environment mapping, and the application of a neural network. All these tasks are based on the results of the lane detection method. The presented lane detection method stands out due to the execution of an interpolation transformation for all previously detected edge points. This transformation transfers these points to a "bird's-eye" coordinate system and distributes them on a grid. Road lanes are identified by a lane feature filter based on the analysis of the distances between unique points. This allows lane views to be obtained in a coordinate system while preserving the distance condition. The road environment map is constructed from the obtained images using a probabilistic algorithm called Distributed Particle-SLAM (DP-SLAM). Based on the map result, a method for representing characteristic points describing the path of road lanes in each incoming camera image has been developed. These points are then used for training the neural network. The neural network solves a regression task for the coordinates of the points on the road lanes, enabling the identification of coefficients for parabolic fitting. Validation has been performed.

1. INTRODUCTION

Since the second half of the 20th century, we have witnessed an increased interest in mobile robot technology. The foundation of these technologies undoubtedly lies in the theoretical works on robot-agent positioning using optical SLAM (Simultaneous localization and mapping) systems or GPS navigation. These theoretical solutions, supported by the development of sensor technologies such as laser sensors, RGB or RGB-D cameras, and GPS receivers with RTK (Real-time kinematic positioning) corrections, now enable the localization of mobile vehicles in the field with an accuracy of approximately 2 cm. In this context, road lane detection systems play a significant role in the operation of mobile systems, as they determine the lanes on which the robot should navigate.

The systematic development of these initial achievements has enabled the autonomy of mobile robots. In a short period, these technologies have been transferred to automotive applications, and currently, we have what is known as "connected cars" or connected autonomous vehicles [1], which operate at level L4 or L5 according to the SAE (Society of Automotive Engineers) classification [2]. In this class of vehicles, we can distinguish two fundamental technologies: automated vehicles that rely on infrastructure support (e.g., V2I - vehicle-to-infrastructure communication and others) [3] and intelligent vehicles that operate using deep neural networks (end-to-end technologies). Many research institutions are involved in development and industrial work in these knowledge areas. As a result, there is a wide range of educational laboratory robots supported by open-source robot operating system software on

[1] Warsaw University of Technology, Faculty of Transport; Koszykowa 75, 00-662 Warsaw, Poland; e-mail: maciej.kozlowski@pw.edu.pl; orcid.org/0000-0002-1068-8991
[2] Warsaw University of Technology, Faculty of Transport; Koszykowa 75, 00-662 Warsaw, Poland; e-mail: andrzej.czerepicki@pw.edu.pl; orcid.org/0000-0002-8659-5695
[3] Warsaw University of Technology, Faculty of Transport; Koszykowa 75, 00-662 Warsaw, Poland; e-mail: piotr.dzido.stud@pw.edu.pl; orcid.org/0009-0002-9774-3144
* Corresponding author. E-mail: maciej.kozlowski@pw.edu.pl

Linux-based systems (e.g., Ubuntu). There are also proprietary developments, often with significant innovative potential.

The goal of this article is to present a theoretical and system-level solution for a car-robot platform developed with the involvement of supervisors and members of the KNEST (scientific students' club) operating at the Faculty of Transport of the Warsaw University of Technology. In this work, we present our original solutions to three basic problems: course control based on lane detection, road environment mapping (monocular simultaneous localization and mapping (SLAM), and the design of a deep neural network for lane detection. Our solutions are entirely based on our proprietary lane detection algorithm.

Initially, lane detection algorithms were based on tracking methods using sampling and particle filters [4], the B-snake method for energy minimization and spline fitting [5], and edge detection methods using the Canny filter and Hough transform [6]. However, these methods are not suitable for real-time analysis. The first method that enabled real-time lane detection was IPM (Inverse Perspective Mapping) and RANSAC (random sample consensus), which involved transforming the road image into a bird's-eye view [7]. Nowadays, shallow and deep neural networks are also used for lane detection. Deep learning-based methods include semantic segmentation [8], instance segmentation [9], and end-to-end approaches [10]. Fig. 1 presents a block diagram of the lane detection method presented in the article, along with block diagrams of classical methods.

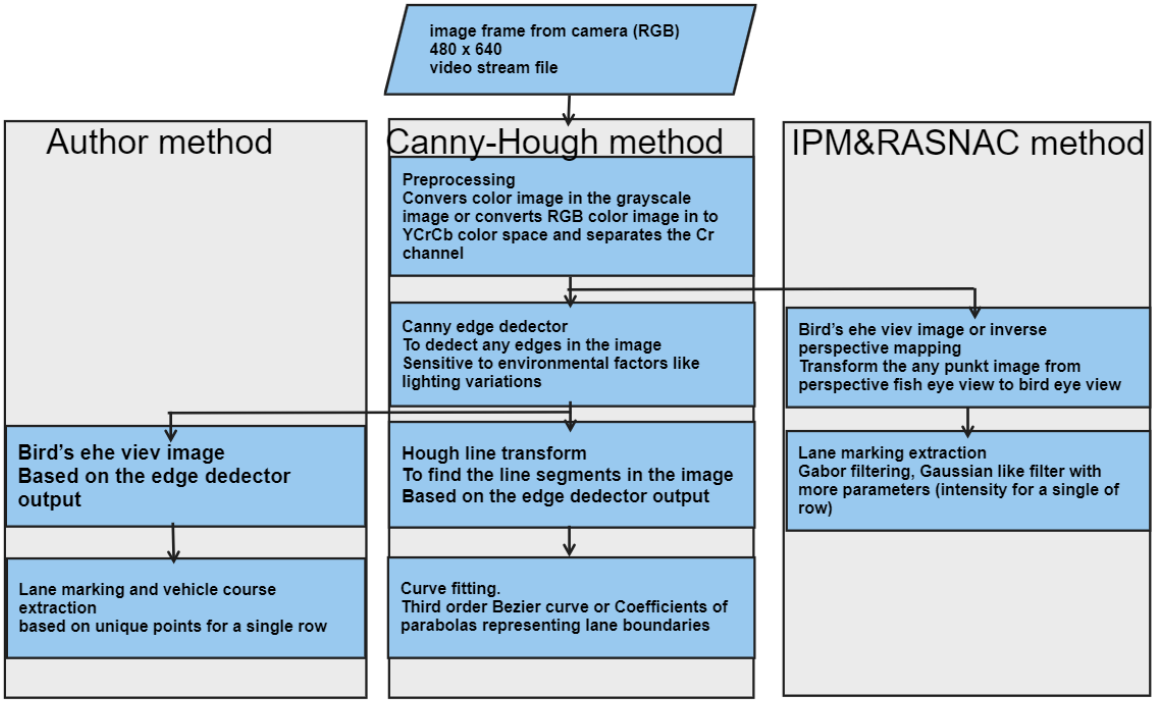


Fig. 1. Block diagrams of the compared lane detection methods

In all algorithms, the camera image is converted to a grayscale image and reduced in resolution. In the IPM & RANSAC algorithm, a perspective transformation is performed to change the "fisheye" perspective to a "bird's-eye" perspective. After this step, edge detection is performed using the Gabor transform [11]. The final step usually involves fitting Bézier curves. In the case of the Canny-Hough algorithm [12], the Canny transform is applied to detect all edges in the image. Then, the Hough transform detects line segments from the detected edges. These line segments are grouped so that they can be assigned to either the left or right lane line. Similar to the previous algorithm, the last step in the custom algorithm involves fitting approximation coefficients.

In the custom algorithm, the reduced grayscale camera image undergoes edge filtering. Then, the edge points are transformed to a top-down view plane in front of the vehicle (referred to as the "bird's-eye" view) and placed on a grid of points (known as discretization). The edges are identified using a simple filter based on characteristic distance features. This filter searches for pairs of points that satisfy

lane marking criteria, such as minimum marking thickness and average lane width. The filter does not perform functional fitting for the detected points but determines a single centreline point representing the road, which is used to determine the desired vehicle trajectory (located on a specific horizontal line in the image). This algorithm is characterized by low computational complexity since the analyzed matrices have reduced dimensions. The matrix of detected edge point coordinates has an average size of 650x2, and the homographic matrix used for image transformation to the bird's-eye view has dimensions of 131x4. Furthermore, typically, only one line of the image needs to be processed to detect lane edge points.

The algorithm was tested on a 1:10 scale model car (shown in Fig. 2). This prototype vehicle is equipped with a control unit consisting of a Raspberry Pi 3 Model B microcomputer (2015) and a subordinate unit - Arduino Uno microcontroller. A wide-angle OV5647 camera with a 1/4" 5 MPx CCD sensor, a resolution of 1080 px, and a viewing angle of 170 degrees (fisheye) was used to capture images. The control circuit diagram is presented in Fig. 3. The chassis of the racing car Maverick Strada served as the platform. A low-speed MM-80 5400 RPM motor, along with an additional planetary reduction gearbox with a gear ratio of 1:3, was employed to reduce the speed. In this case, the speed can be adjusted within the range of 6 cm/sec to 30 km/h.

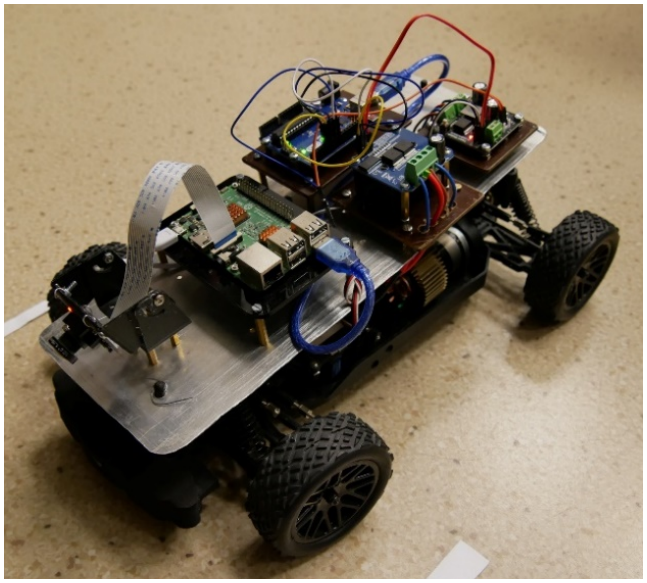


Fig. 2. Car model used in research

As mentioned, the computer system has low computational performance, and this characteristic affects the solutions used in the algorithm for the second task (map construction). To build the map, we only utilize edge images transformed into a bird's eye view. We apply a probabilistic algorithm consisting of two steps: prediction and correction. Prediction involves using the Ackerman model to predict the images seen from the car's successive positions (right propagation of errors - time update), while correction involves the application of Bayesian inference after obtaining a new measurement (image) based on maximum likelihood (measurement update).

In the third task, we determine the prediction of lane lines using a deep neural network. We apply transfer learning, which involves using a pre-trained network. The network is trained with the coordinates of three points for each lane line (a total of six numerical values representing the coordinate values of the left and right lanes). In this solution, we use the "Alex" network, from which we remove the last four layers and replace them with an output layer that solves the regression task. The main part of the article is divided into the following sections: lane detection (where we discuss the author's algorithm), map construction (in which we present the method used to assemble registered lane edge images seen from a bird's eye perspective), and network design (where we present the results of applying transfer learning to the "Alex" network).

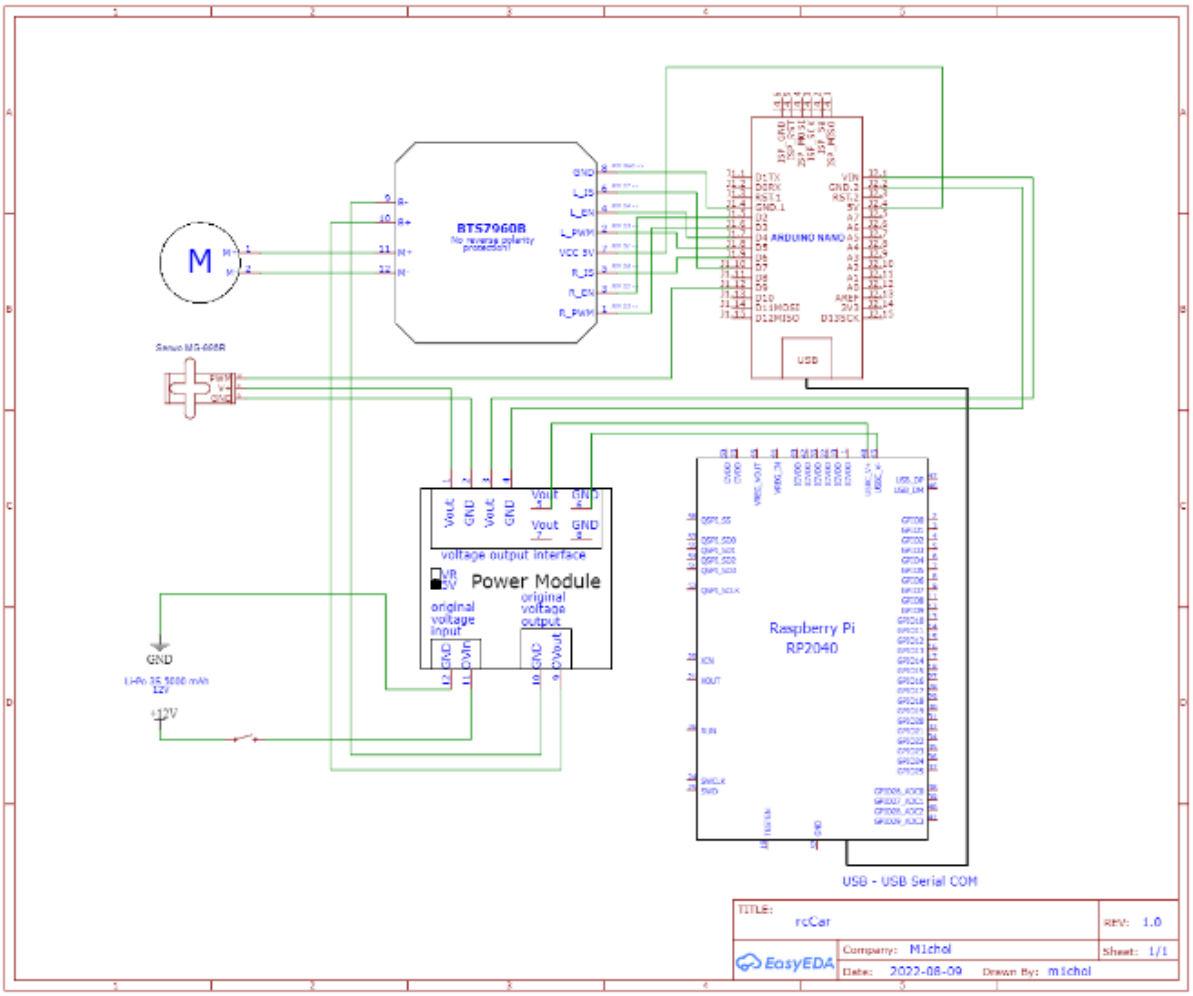


Fig. 3. The control diagram with the master microcomputer (Raspberry Pi) and the slave microcontroller (Arduino Uno) and peripherals: servo, motor, camera

2. LANE DETECTION

2.1. Transformation of detected image edges from a “fisheye” to a “bird’s eye” perspective

In the method we have developed for lane detection, obtaining a top-down view (also known as a “bird’s-eye view”) is particularly crucial. Fig. 4 illustrates three different perspectives (presented as point grids): a) a “fisheye” perspective (barrel distortion projected onto an inclined plane), b) a classical perspective, and c) a “bird’s-eye” perspective.

When the perspective is converted from configuration (a) to (b), a well-known mathematical relationship is commonly applied to represent the fisheye camera distortion effect [13-15]. Implementations of source code for Python are available in the OpenCV library [16-18], as well as in Matlab [19]. The operation can be expressed using the following formula [13-15]:

$$x = KX_c = K(R| - Rc)X = HX \tag{1}$$

$$X = H^{-1}x \tag{2}$$

where: x – image plane, K – camera calibration matrix, X_c – camera coordinate system, R – rotation, c – translation, H - homography matrix, X – Bird’s plane.

a) b) c)

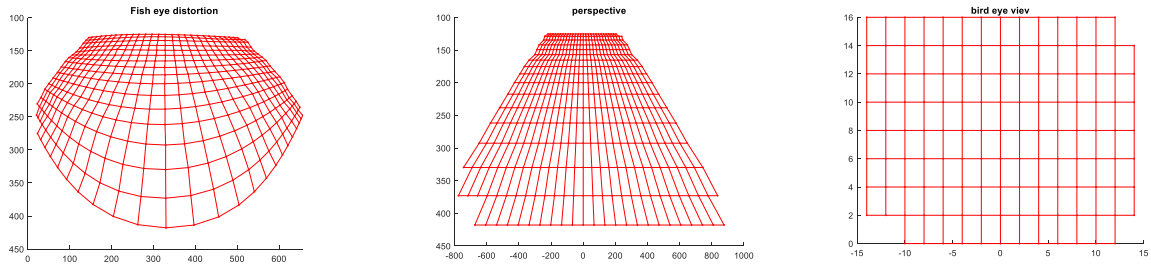


Fig. 4. Perspective grids: a) "fisheye," b) normal, and c) "bird's eye"

However, in our work, we change the perspective from configuration (a) to (c) using a projection method [20]. With the help of an interpolation function, this principle can be expressed using two matrices: a pixel coordinate matrix (Fig. 4a) and a road point coordinate matrix (Fig. 4c). The method of displacing points on the grids is illustrated in Fig. 5.

If we apply the aforementioned principle only to the points that describe the edges of the image detected using the Canny filter, we can reduce computational overhead because the matrices have smaller dimensions compared to the original images. The computation process is illustrated in Fig. 6.

As a result of the performed operations, we obtain a scaled plot of the image edges (in centimeters); the camera position point is located at the origin of the coordinate system.

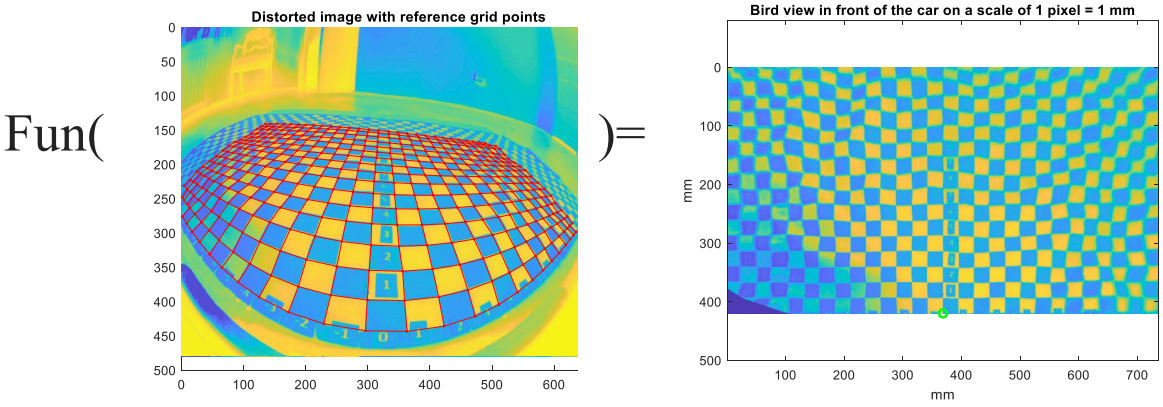


Fig. 5. Principle of transformation of image perspective

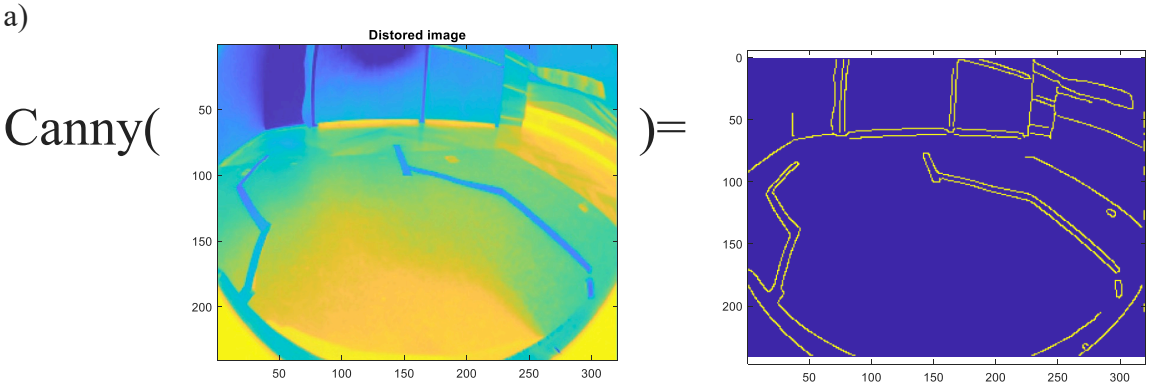
2.2. Road lane edge filter and course setting

The detected edge points of the image provide information about the edges of all objects seen by the camera, including the edges of the traffic lanes. The task of the lane edge filter is to determine which edges should be assigned to the traffic lanes. This task is performed by a feature filter. The developed filter operates in two steps. First, the coordinates of the edges are rounded to a specific format, which corresponds to moving the points to the nodes of a defined grid (discretization). In the second step, the filter examines the distribution properties of the points based on the criterion of mutual distances. The input parameters include the X and Y coordinates of the object edges in the image, the distance of the points from the camera for which the analysis will be conducted, the lane width, and the width of the road between the lanes. Computational overhead is reduced by conducting the described analysis until the points on a given level meet the conditions for belonging to the lane edges. The result of the calculations is the midpoint of the road at a specified distance from the origin of the coordinate system (the center of the front wheel axis in the so-called "motorcycle model"). The calculation result is presented in Fig. 7. The points are marked as follows: O – the origin of the coordinate system, A – the detected center of the lane at the analyzed distance from the front of the vehicle, L – the detected location point of the left lane, R – the detected location point of the right lane. In this case, the steering angle can be estimated from the following relationship:

$$\varphi = \operatorname{atan}\frac{A_Y}{A_X} \tag{3}$$

where: φ – the steering angle, A_Y – the Y-coordinate of point A, A_X – the X-coordinate of point A.

a)



b)

Fun(

	edge X	edge Y
1	-32.4040	10.0230
2	-31.8563	9.7235
3	-31.3694	9.4487
4	-33.0416	10.8633
5	-32.3623	10.4771
	(…)	(…)

)=

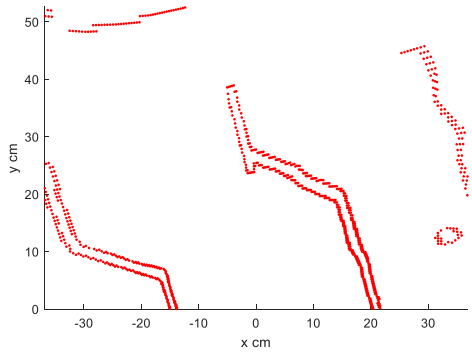


Fig. 6. Image transformations: a) edge detection using a Canny filter and b) the interpolation transformation of image edge points to a bird's eye view coordinate system (131x4 matrix)

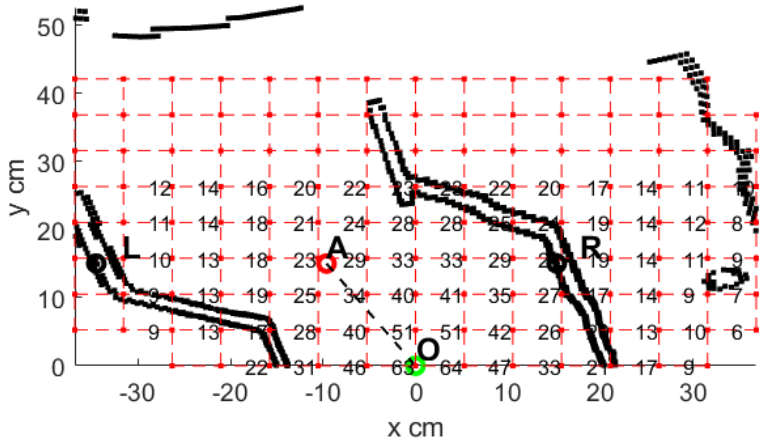


Fig. 7. Vehicle traffic control principle: a flat image with the grid of the interpolation matrix overlaid with pixel density in the grid cells. The camera resolution is 240 x 320 pixels, with a viewing angle of 170 degrees

In the discussed control system, the obtained heading value is treated as the desired steering angle of the car's wheels. Considering control principles, the presented control system operates as a proportional (P) steering controller with an adjustable value of the front wheel steering angle.

2.3. Analysis of the control error using the vehicle model

The intended result of control should be the movement of the vehicle model such that point O, the center of the front wheel axis, travels along the centerline of the road. Consequently, the control error is defined as the deviation of the executed trajectory of the model from the points of the geometric center of the road. In the presented method, among the most significant factors influencing the value of the control error are the conversion error from the camera image to a flat image, the discretization error of the lane edge filter, the error in determining the center of the road based on the position of detected lanes, the error in determining the steering angle of the wheels, and the error in setting the desired value by the executive system (actuator).

The most significant error is the error that can occur when converting the camera image to a flat image. The process depends on the camera's resolution and field of view. The conversion method is determined by the interpolation matrix H. Fig. 7 shows pixel density in selected windows of this matrix grid. Points L and R in Fig. 7 are used to determine the position of point A and are located in areas with a density of 10 and 24 pixels. Since the side of the interpolation grid cell is 52.6 mm, the accuracy of the position of point A can be determined at a level of 6 mm/pixel. Also, considering that the edge filter operates on a discretization grid of 2 mm, the total error in the location of the center of the road point A does not exceed 10 mm. Since the position of point A is determined at a distance of 150 mm, the maximum steering angle error is 3.8 degrees. A significant improvement in accuracy will be achieved for cameras with a higher resolution than the adopted one (240 x 320 pixels).

3. MAP CONSTRUCTION

3.1. Principle

The main problem in aligning images captured by a car during drives on specified road segments arises from the lack of sensors (encoders) to measure the actual values of vehicle speed and front wheel steering angle. The only quantities that can be utilized are the set parameters of these quantities that are sent to the Raspberry Pi, which acts as the control system. Therefore, it is reasonable to construct a map using a probabilistic algorithm that reflects the structure of the Kalman filter. The schematic of the calculation method in a fixed time step is presented in Fig. 8.

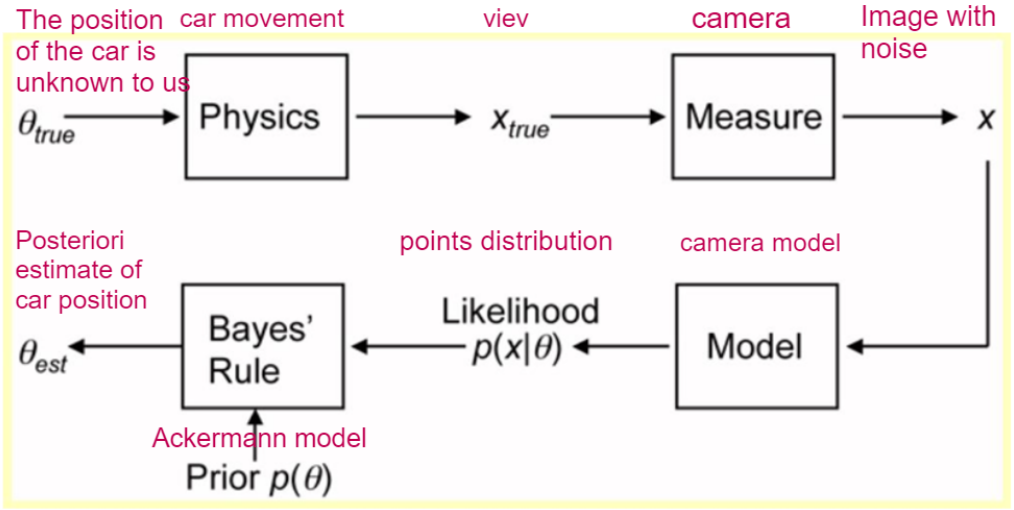


Fig. 8. Flow chart of the map construction method (based on [21])

The schematic refers to the physics of the motion process in which the vehicle changes its position. The actual coordinates of the vehicle's position are unknown. However, we can acquire views of the road environment from these positions. The views are obtained using a camera that captures images with

added noise. We estimate the probability distributions of the image coordinates using a camera model. On the other hand, an estimate of the vehicle's position from the previous iteration and the parameters of the applied control allows us to obtain probability distributions of hypothetical vehicle positions. From these positions, we can predict the images that would be captured by the camera mounted on the vehicle. The choice of which view to select (and the associated predicted position) is determined by the Bayesian inference rule - maximum likelihood.

3.2. The Ackerman model: Law of error propagation and time update

The Ackermann model is a simple vehicle kinematics model that allows for the description of motion. The iterative form of the model equations enables easy application for predicting the position in the next time step based on known control inputs. The time update equations of the model take the following form [22]:

$$x_k = x_{k-1} + V_k\,\Delta T\,cos\theta_k \tag{4}$$

$$y_k = y_{k-1} + V_k\,\Delta T\,sin\theta_k \tag{5}$$

$$\theta_k = \theta_{k-1} + \frac{V_k\,\Delta T}{L}\,tg\emptyset_k \tag{6}$$

where: Φ_k – steering angle of the front wheels, Θ_k – rear axle alignment angle relative to the X-axis of the absolute reference frame (turning angle, orientation), V_k – linear velocity of the vehicle, ΔT – time increment, x_k – X-coordinate of the absolute coordinate system's X-axis, y_k – coordinate of the absolute coordinate system's Y-axis.

We make the simplifying assumption that the probability distributions of the control uncertainties (the set values of velocity and steering angle) follow normal distributions and are characterized by their variances as follows: ΔV_k^2 – variance of the velocity distribution, $\Delta\,\Phi_k^2$ – variance of the front wheel steering angle distributions.

3.3. Application of the dispersed particles method (DP-SLAM) for image prediction

This method is based on the concept of particle filtering, by which particles represent the trajectory of the model's motion under uncertain conditions. It allows us to generate hypothetical camera positions consistent with the assumed distribution of position and control uncertainties. As a result, we obtain a set of hypothetical scans that could be observed from the generated positions. These scans are treated as conditionally independent, and they indicate potential "candidates" for the map [23]. The optimal scan is chosen based on the principle of maximum likelihood, according to which the number of matched points serves as a measure of matching quality [24]. The process of estimating the new position based on this algorithm is illustrated in Figs. 9 and 10. Fig. 9 depicts the situation of the best match, while Fig. 10 represents the worst match. Figs. 9 and 10, labeled with index "1," illustrate the camera motion prediction using the Ackermann model. The black points represent the scan view from the previous iteration ("prior" – Fig. 8). The green point denotes the camera position in the current iteration, and the blue point represents the position predicted under hypothetical motion conditions. Figs. 9 and 10, labeled with index "2," show the matching result, where the red color indicates the prediction, and the green color represents the new scan obtained from the subsequent observation. The better matching of scans presented in Fig. 9 supports the selection of the trajectory point as the posterior estimate of the position (Fig. 8).

3.4. Resulting map

After the algorithm described above was applied, the process of image stitching resulted in a map of road lane edges. Fig. 11 illustrates the road image and the generated map. The utilized microcomputer system is capable of controlling and generating the map at a sampling frequency of 6 frames per second. In this case, the lowest achievable driving speed is 6 cm/sec, which corresponds to 1 image captured per 1 cm of road distance.

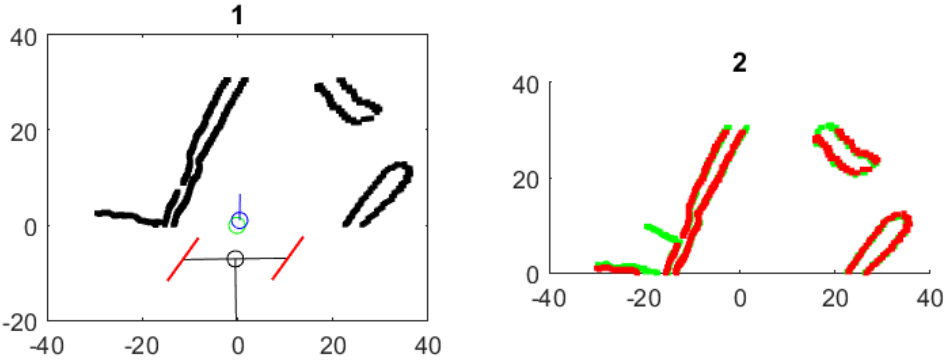


Fig. 9. Principle for determining the estimate of the new position (best fit): (1) − prediction of the new position of the camera point relative to the road environment, (2) − prediction of the road scan relative to the measured image

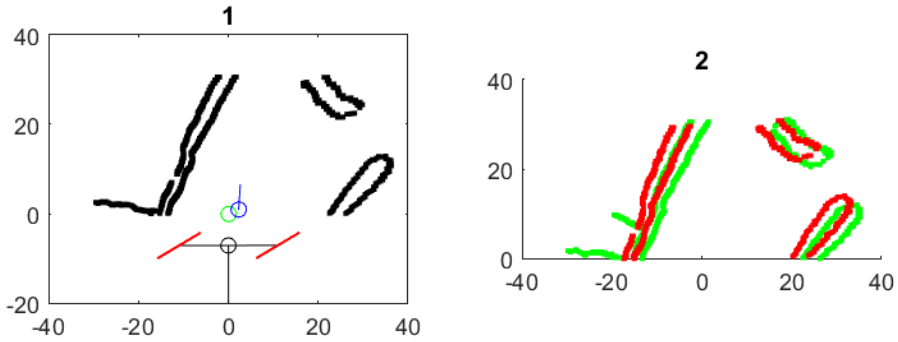


Fig. 10. Principle for determining the estimate of the new position (worst fit): (1) − prediction of the new position of the camera point relative to the road environment, (2) − prediction of the road scan relative to the measured image

a)

b)

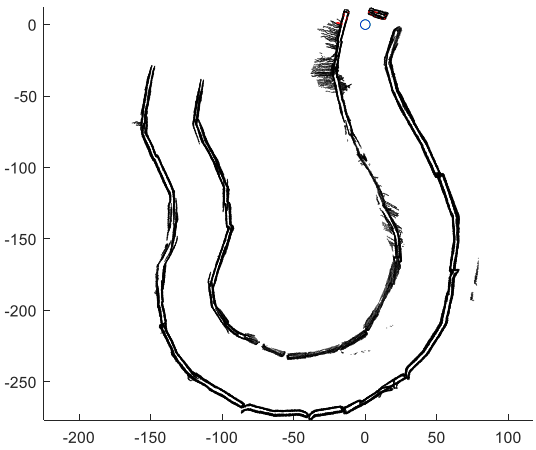


Fig. 11. Photograph of the route and results of the calculations: a) top view and b) the resulting map

4. APPLICATION OF NEURAL NETWORKS

4.1. Transfer learning and network task

Transfer learning involves utilizing a pre-trained deep neural network for new tasks. Typically, this is achieved by replacing the last few decision layers of the network with new ones tailored to the specific

project [25-27]. In our project, we use the "Alex Network" and prepare it to solve the task of lane detection based on the procedure outlined by [28]. This means that we modify the last four layers of this network in the described manner so that it can tackle regression tasks.

In contrast to classical solutions, our neural network will utilize scaled edge images of the road foreground presented in a "bird's eye" perspective while preserving the distance function between points. The objective of applying the network is to determine three characteristic points that approximate the parabolic line of the road lane associated with a specific relationship:

$$x = f(y) = ay^2 + by + c \tag{7}$$

$$[x_1, x_2, x_3] = f(y_1, y_2, y_3) \ , \tag{8}$$

where: x - represents the width, y - represents the distance from the camera position (in our method, it is the origin point of the coordinate system), [a, b, c] are the coefficients of the parabolic fit of the road lane edges, [y1, y2, y3] are parameters representing distances with predefined constant values: y1 = 0, y2 = 15, y3 = 30 [cm].

The situation is illustrated in Fig. 12. For two lanes of traffic, it is necessary to determine six coordinates. Therefore, the trained network will perform the following function, which uniquely defines the parabolic fit:

$$NETWORK(IMAGE) = COORDINATES\ OF\ CHARACTERISTIC\ POINTS \tag{9}$$

This formulation distinguishes our method from others in which the network directly determines coefficients based on camera images rather than the representation of lane edges.

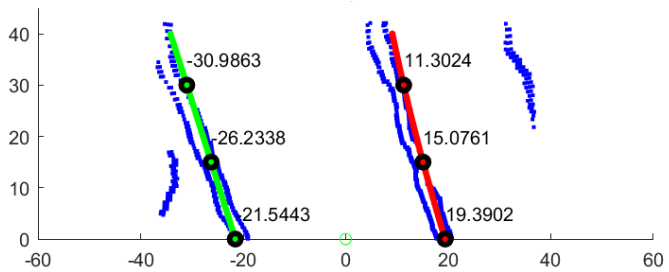


Fig. 12. The detected edges of the camera image, approximations of the road lanes, and the coordinates of the characteristic points

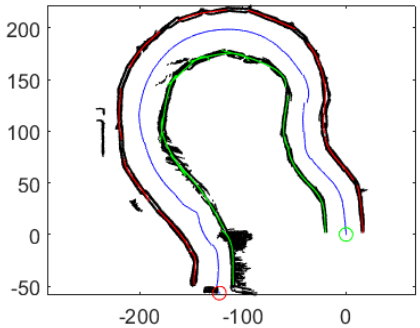


Fig. 13. The approximation of road lane edges on the map is achieved by using piecewise polynomial fitting of third-degree (cubic) polynomials

4.2. Preparation of training and validation data

Preparing data for network training can be a time-consuming task. The Automated Driving Toolbox™ includes an application called the Ground Truth Labeler app [29] that can be used for this purpose. However, it cannot be directly applied in cases in which images describe only the lane edges. Therefore, in our work, the traffic lanes were separated and approximated using polynomial functions after assembling the map. Fig. 13 illustrates the results of the fitting process. Subsequently, these results were automatically transferred to individual images, resulting in the effect shown earlier in Fig. 12. In this way, we obtained a total of 4503 training samples from seven test drives conducted on different

road segments (in two directions of travel). The training images presented to the network during training were not rotated or additionally distorted. The remaining 1502 training samples obtained from two other drives were used for network validation.

4.3. Training and validation

The network was trained on a desktop computer using Matlab software with standard settings: a learning rate of 10^{-5} and a 10-fold increase in the learning rate of the last layers' coefficients. The training process was manually terminated after approximately 15 epochs due to minimal changes in the RMSE error metric, which reached a value of 5 cm (for six coefficients determined in a single iteration). Fig. 14 presents two scans with training result plots; the detected edges are marked with yellow lines. Fig. 15 shows two scans illustrating the validation results. In Fig. 15a, only the left lane boundary is visible, and there is a disturbing element present in the road's illumination. Another disturbing element with a different shape is visible in Fig. 15b. Despite these disturbances, the network accurately determined the road lanes. Overall, during validation, a similar RMSE value was achieved at a comparable level of 5 cm per iteration.

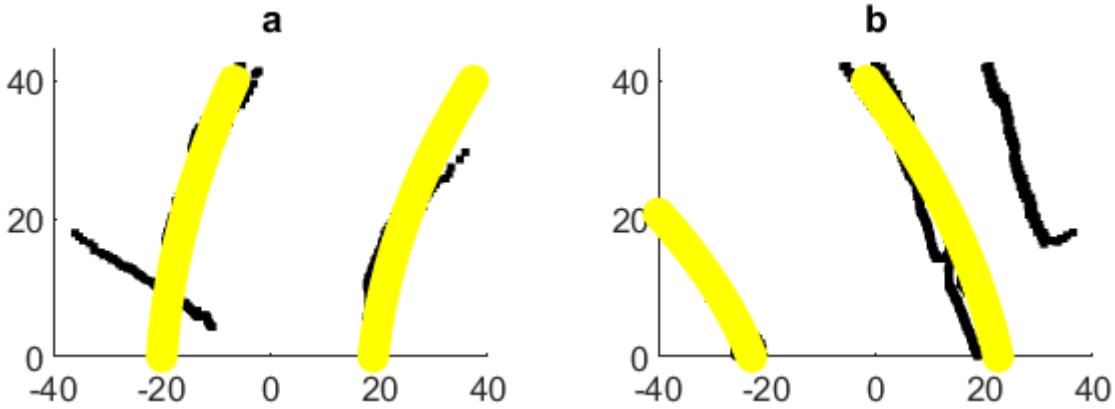


Fig. 14. Scans with training result graphs

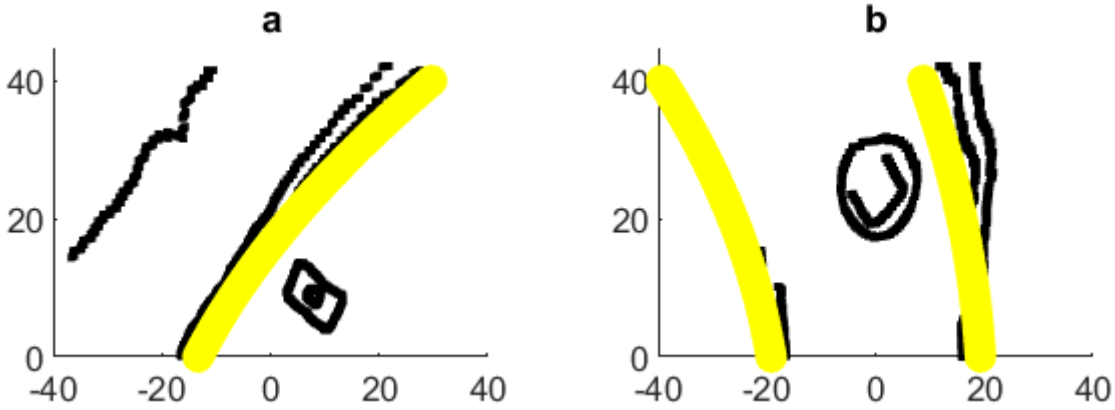


Fig. 15. Scans with validation results

5. CONCLUSIONS

The paper presented novel solutions to three problems related to the task of lane detection for autonomous vehicle models: control, map construction, and the utilization of neural networks.

The presented vehicle control method is a vision-based approach designed for systems with low computational capabilities. Computational overhead reduction was achieved through edge analysis, which was performed not as typical image analysis but as simple algebraic matrix transformations. After edge detection, the detected points were transformed into a "bird's eye" coordinate system using

interpolation. This operation contributed to computational reduction since it involved transforming only around 600 points and an interpolation transformation described by a 131x4 matrix. Subsequently, a simple filter was applied to identify lane edges based on measurable morphological features. Lane detection allowed us to determine the coordinates of the road center. The vehicle moved toward the center using a "pursuit curve" approach by employing a proportional controller that maintained the course (always toward the center of the road) by adjusting the appropriate steering angle. This straightforward approach to lane detection and course-keeping sets our method apart from other solutions. Calculations of the theoretical control error, taking into account the conversion principle from a "fisheye" image to a "bird's-eye" image, showed that the maximum error in the steering angle of the model was 3.8 degrees. However, in practice, the deviation of the trajectory of the center of the front wheel axis from the geometric center of the road was examined using a road map generated after a specified route was driven. The results of this analysis reveal that the vehicle model stays within the road area.

The second problem addressed was the construction of a road environment map. The map was composed of road edge points detected at each iteration of the system's operation. This solution also contributed to computational reduction as it involved a smaller number of points compared to raw camera frames. A classical algorithm called DP-SLAM was employed to solve this problem.

The third problem addressed was the application of a neural network for lane detection on edge images. In contrast to classical approaches, our neural network was trained on scaled edge images represented from a "bird's eye" perspective, preserving the distance function between points. Furthermore, the objective of the network was not to directly determine the coefficients of the parabolic fitting. Instead, our network identified characteristic points of the approximation located at standardized distances. This formulation sets our method apart from others in which the network's task is to directly estimate coefficients based on camera images rather than edge representations. During validation, it was found that our trained network performed well in handling unforeseen issues, such as the absence of lane edge points in one lane or the presence of additional unknown objects in the road's field of view.

This outcome demonstrates the robustness of our trained network in dealing with unexpected challenges, providing reliable lane detection even in complex scenarios. By training the network on perspective-transformed and standardized edge images, we enhanced its ability to generalize and adapt to varying road conditions.

Below, we provide several points regarding further directions for our work:

1. Refining network training: We can experiment with simpler network architectures than the currently chosen "Alex Network." We believe that our requirements are small, and the current network may be too large for our needs. Utilizing a more optimized network architecture may yield better results and improve the system's efficiency.
2. Expanding the training dataset: To improve network fitting, we can expand the training dataset by rotating, adding noise, or introducing edges of unknown objects to the images. Training on diverse and varied data can help the network better cope with unpredictable situations on the road.
3. Detecting changes in road structure: By leveraging a neural network, we can attempt to detect changes in the road structure, such as intersections and parking lots. This additional functionality will enable our system to recognize more advanced road infrastructure elements and contribute to further enhancing autonomous vehicle management.

Working on these aspects can refine our system, increase the accuracy of lane detection, and expand its ability to recognize and respond to changing road conditions.

Acknowledgments

This article was funded by the funds of IDUB PW (Inicjatywa Doskonałości – Uczelnia Badawcza − Initiative of Excellence − Research University) No. 1820/274/Z16/2022 entitled "Building an Environment − Scale Models for Testing the Functionality of Autonomous Car Software" − Rector's grant for the KNEST scientific club at the Faculty of Transport, Warsaw University of Technology.

References

1. Choromański, W. & Grabarek, I. & Kozłowski, M. & Czerepicki, A. & Marczuk, K. *Pojazdy autonomiczne i systemy transportu autonomicznego.* Wydawnictwo Naukowe PWN. 2020. [In Polish: *Autonomous vehicles and autonomous transport systems*. PWN Scientific Publishing House].
2. *SAE Levels of Driving Automation™ Refined for Clarity and International Audience.* Available at: https://www.sae.org/blog/sae-j3016-update.
3. Dey, K. & Rayamajhi, A. & Chowdhury, M. & Bhavsar, P. & Martin, J. Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication in a heterogeneous wireless network – Performance evaluation. *Transportation Research Part C: Emerging Technologies*. 2016. Vol. 68. P. 168-184. DOI: 10.1016/j.trc.2016.03.008.
4. Apostoloff, N. & Zelinsky A. Robust vision based lane tracking using multiple cues and particle filtering. *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings (Cat. No.03TH8683).* Columbus, OH, USA. 2003. P. 558-563. DOI: 10.1109/IVS.2003.1212973.
5. Wang, Y. & Teoh, E. & Shen, D. Lane detection and tracking using B-Snake. *Image and Vision Computing*. 2004. Vol. 22. P. 269-280. DOI: 10.1016/j.imavis.2003.10.003.
6. Forogh, P. *Line Detection in Python OpenCV with HoughLines.* Available at: https://www.youtube.com/watch?v=OchCsSiffeE.
7. Aly, M. Real time detection of lane markers in urban streets. *2008 IEEE Intelligent Vehicles Symposium*. Eindhoven, Netherlands. 2008. P. 7-12. DOI: 10.1109/IVS.2008.4621152.
8. Ding, L. & Zhang, H. & Xiao, J. & Shu, C., & Lu, S. A lane detection method based on semantic segmentation. *Comput. Model. Eng. Sci*. 2020. Vol. 122(3). P. 1039-1053.
9. Ko, Y. & Lee, Y. & Azam, S. & Munir, F. & Jeon, M. & Pedrycz, W. Key Points Estimation and Point Instance Segmentation Approach for Lane Detection. In: *IEEE Transactions on Intelligent Transportation Systems.* 2022. Vol. 23. No. 7. P. 8949-8958. DOI: 10.1109/TITS.2021.3088488.
10. Neven, D. & De Brabandere, B. & Georgoulis, S. & Proesmans, M. & Van Gool, L. Towards end-to-end lane detection: an instance segmentation approach. In: *2018 IEEE intelligent vehicles symposium*. 2018. P. 286-291.
11. Getahun, T. *Lane Detection for Autonomous Driving: Conventional and CNN approaches*. Access Laboratory. Available at: https://www.youtube.com/watch?v=xIRT3rgWrFQ.
12. ChanHee, J. *Curved Lane Detection_Algorithm Explanation in English.* Available at: https://www.youtube.com/watch?v=0RAijzUnQAU.
13. Weng, J. & Cohen, P. & Herniou, M. Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1992. Vol. 14(10). P. 965-980.
14. Dulari, B. *A comprehensive guide for Camera calibration in computer vision*. Available at: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-for-camera-calibration-in-computer-vision/.
15. Kutila, M. & Korpinen, J. & Viitanen, J. Camera calibration in machine automation. *Human Friendly Mechatronics.* Elsevier Science. 2001. P. 211-216. ISBN: 9780444506498. Available at: 10.1016/B978-044450649-8/50036-X.
16. Jiang, K. *Calibrate fisheye lens using OpenCV* - part 1, part 2. 2017. Available at: https://medium.com/@kennethjiang/calibrate-fisheye-lens-using-opencv-333b05afa0b0. https://medium.com/@kennethjiang/calibrate-fisheye-lens-using-opencv-part-2-13990f1b157f.
17. *OpenCV documentation for camera calibration.* Available at: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html.
18. Kausthub Sadekar. *Understanding Lens Distortion.* Available at: https://learnopencv.com/understanding-lens-distortion/.
19. MathWorks. *What Is Camera Calibration?* Available at: https://in.mathworks.com/help/vision/ug/camera-calibration.html.

20. Lenton, D. *Part I & II: Projective Geometry in 2D.* Available at: https://medium.com/@unifyai/part-i-projective-geometry-in-2d-b1ca26d5fa2a. https://medium.com/@unifyai/part-ii-projective-transformations-in-2d-2e99ac9c7e9f.
21. Stone, J.V. *Bayes' Rule: A Tutorial Introduction to Bayesian Analysis*. Available at: http://jimstone.staff.shef.ac.uk/.
22. Mogensen, L. & Andersen, N.A. & Ravn, O. & Poulsen, N. *Using Kalmtool in Navigation of Mobile Robots*. 2023. Available at: https://www.researchgate.net/publication/269262121_Kalmtool_Used_for_Mobile_Robot_Navigation.
23. Eliazar, A. & Parr, R. *DP-SLAM: Fast, Robust Simultaneous Localization and Mapping Without Predetermined Landmarks*. Available at: http://people.ee.duke.edu/~lcarin/Lihan4.21.06a.pdf.
24. Biber, P. & Strasser, W. The normal distributions transform: A new approach to laser scan matching. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2003. P. 2743-2748.
25. Tensorflow.org. *Transfer learning and fine-tuning*. TensorFlow Core. Available at: https://www.tensorflow.org/tutorials/images/transfer_learning.
26. Baheti, P. *A Newbie-Friendly Guide to Transfer Learning*. Available at: https://www.v7labs.com/blog/transfer-learning-guide.
27. MathWorks. *Transfer Learning for Training Deep Learning Models*. Available at: https://www.mathworks.com/discovery/transfer-learning.html.
28. Nehemiah, A. *Deep Learning for Automated Driving (Part 2) – Lane Detection*. Available at: https://blogs.mathworks.com/deep-learning/2017/11/17/deep-learning-for-automated-driving-part-2-lane-detection/.
29. MathWorks. *Automated Driving Toolbox, Design, simulate, and test ADAS and autonomous driving systems.* Available at: https://www.mathworks.com/products/automated-driving.html.

Received 12.10.2022; accepted in revised form 05.03.2024