technologies

MDPI

Article

Functional Model of a Self-Driving Car Control System

Kirill Sviatov *, Nadejda Yarushkina, Daniil Kanin, Ivan Rubtcov, Roman Jitkov, Vladislav Mikhailov and Pavel Kanin

Faculty of Information Systems and Technologies, Ulyanovsk State Technical University, Severny Venetc, 32, 432027 Ulyanovsk, Russia; jng@ulstu.ru (N.Y.); dan-kan@mail.ru (D.K.); ria0294@mail.ru (I.R.); roman73zh@gmail.com (R.J.); vmixoks@gmail.com (V.M.); kanin.pavel@mail.ru (P.K.)
* Correspondence: k.svyatov@ulstu.ru

Abstract: The article describes a structural and functional model of a self-driving car control system, which generates a wide class of mathematical problems. Currently, control systems for self-driving cars are considered at several levels of abstraction and implementation: Mechanics, electronics, perception, scene recognition, control, security, integration of all subsystems into a solid system. Modern research often considers particular problems to be solved for each of the levels separately. In this paper, a parameterized model of the integration of individual components into a complex control system for a self-driving car is considered. Such a model simplifies the design and development of self-driving control systems with configurable automation tools, taking into account the specifics of the solving problem. The parameterized model can be used for CAD design in the field of self-driving car development. A full cycle of development of a control system for a self-driving truck was implemented, which was rub in the "Robocross 2021" competition. The software solution was tested on more than 40 launches of a self-driving truck. Parameterization made it possible to speed up the development of the control system, expressed in man-hours, by 1.5 times compared to the experience of the authors of the article who participated in the same competition in 2018 and 2019. The proposed parameterization was used in the development of individual CAD elements described in this article. Additionally, the implementation of specific modules and functions is a field for experimental research.

Keywords: self-driving car; visual scene recognition; deep learning; computer aided design

Citation: Sviatov, K.; Yarushkina, N.; Kanin, D.; Rubtcov, I.; Jitkov, R.; Mikhailov, V.; Kanin, P. Functional Model of a Self-Driving Car Control System. *Technologies* **2021**, *9*, 100. https://doi.org/10.3390/technologies9040100

Academic Editor: Vijayakumar Varadarajan

Received: 2 November 2021
Accepted: 7 December 2021
Published: 10 December 2021

1. Introduction

Currently, the market for self-driving cars is actively growing [1], and may reach a value of $1600 billion by 2030 [2] and $7 trillion by 2050 [3].

The automotive industry is confronting a widening and unsustainable gap between software complexity and productivity levels [4]. The only way to deal with the complexity is to automate the design of self-driving cars using the Model Based Systems Engineering approach with the appropriate tooling in the form of computer-aided design, simulation, and evaluation of the quality metrics of the solution being developed [5].

There are studies in which the control function is considered as an end-to-end solution [4] implemented with deep neural networks using reinforcement learning. This approach has a significant drawback—it has a low explanatory feature in decision making [6]. This is a very important aspect of developing an autonomous vehicle control system, because people must understand the criteria for making decisions about traffic and agree with them, and when analyzing an accident, it is important to find out the causes of the accident [7].

An end-to-end solution [8] also seems unlikely due to the complexity of the problem being solved and the low performance of computing systems that can now be used on board, as well as the tendency of neural networks to overfit.

The classical approach [6] involves the creation of a control system based on data flow control with feedback. This approach [9] is even more difficult to implement, but the data generated at each stage of processing can be explained. At the same time, data processing in several blocks is carried out using deep learning neural networks [10]. This article describes a functional model of the control system, developed using the classical approach. The specification of the parameters of each of the functional blocks claims to be scientific novelty, as well as to a high-level functional model of the control system, in which each block is considered as an abstract function, which makes it possible to use the research results to create a CAD system for designing unmanned vehicles: The implementation of an abstract function can be carried out in several ways for formation of a library of software modules.

From the point of view of the control system of a self-driving car, several levels of a functional organization are proposed [6]: sensing, environment perception, environment mapping, motion planning, decision making, and actuation.

Requirements differ in terms of perception, recognition of different types of road scene objects, robot size, robot mechanics and drive mode [7], in terms of decision-making about motion planning, speed of decision-making, as well as in terms of the equipment used at the sensory level [11]. Requirements produce many design specifications, the implementation of which is a laborious process. This process can be automated.

The data processing flow has a peculiarity: The way of data processing in each module depends on the peculiarities of data processing at the previous stages. Therefore, an important aspect in the development of modules is to check the solution in a single loop of the recognition and control system, which creates the need for expensive experiments when it comes to a real car.

The entire system must be used to test each module. A good way of rapid prototyping is to use simulation environments for self-driving cars [12]: Gazebo [13], Carla Simulator [14], Webots [15], NVIDIA DRIVE Sim [16], CarSim [17], MATLAB/Simulink with Automated Driving Toolbox [18], PreScan [19], and LGSVL [20]. The use of simulators allows solving the important problem of evaluating (testing) design automation subsystems at various architectural levels, taking into account the fact that a change in each functional block may lead to the need for changes for subsequent functional blocks.

The variability of the specifications due to different technical requirements for the designed self-driving cars suggests the need to create a CAD system, for the creation of which it is necessary to clearly define the design modules, the input and output parameters of the modules transferred from one level to another, as well as the functional dependencies of the modules from each other. This article describes a model of a self-driving car control system that is used to create and prototype control systems in simulation environments and a real car.

The research is based on a model-driven development method, which is based on machine learning models in the field of perception, knowledge base models in the field of the decision-making subsystem, asynchronous programming models in the field of control systems, as well as frameworks. Models, expressed in terms of object-oriented programming, for the programmatic implementation of all APIs and functions. The results were evaluated for each module separately. The purpose of the article is the design, modeling and development of a control system, as well as parameterization and highlighting of common functions for the purpose of further development of the CAD system for the development of self-driving cars. The evaluation was carried out based on the reduction of project labor costs in the design of three control systems: Two without the proposed approach and one using the approach.

The original contribution of the article lies in the formalization and specification of the complete data processing flow of an unmanned vehicle. The presented model is used to design software interfaces of functional blocks of a control system, and can also be used as a basis for CAD system since various CAD components need unified data protocols in

terms of designing a library of typical solutions, which are described in several sections of the article.

2. Parametrized Functional Model of the Control System

A truck-based on the GAZelle Next platform was used as a target self-driving car. This car has a manual transmission and hydraulic power steering. The structural and functional model of the control system is shown in Figure 1. For the robotization of the transmission, a gearshift mechanism with two linear actuators was developed. Linear actuators were installed on the clutch and brake pedals also, and a PWM (pulse-width modulation) signal was generated for the gas pedal. For the steering, an electric power steering was installed, for which an electronic board was developed that allows simulating the signal from the torsion bars, which drives the electric motor, which rotates the steering shaft in the desired direction. All engines and boards are controlled via the CAN bus, to which control signals are transmitted (block i on Figure 1):

$$Cs = (Br,\ Tr,\ Thr,\ Cl,\ St) \tag{1}$$

Cs—control signal for the car controller, *Br*—value for a brake, *Thr*—value for a throttle, *Cl*—value for a clutch, *St*—target angle for the wheel, *Tr*—value for a transmission number.

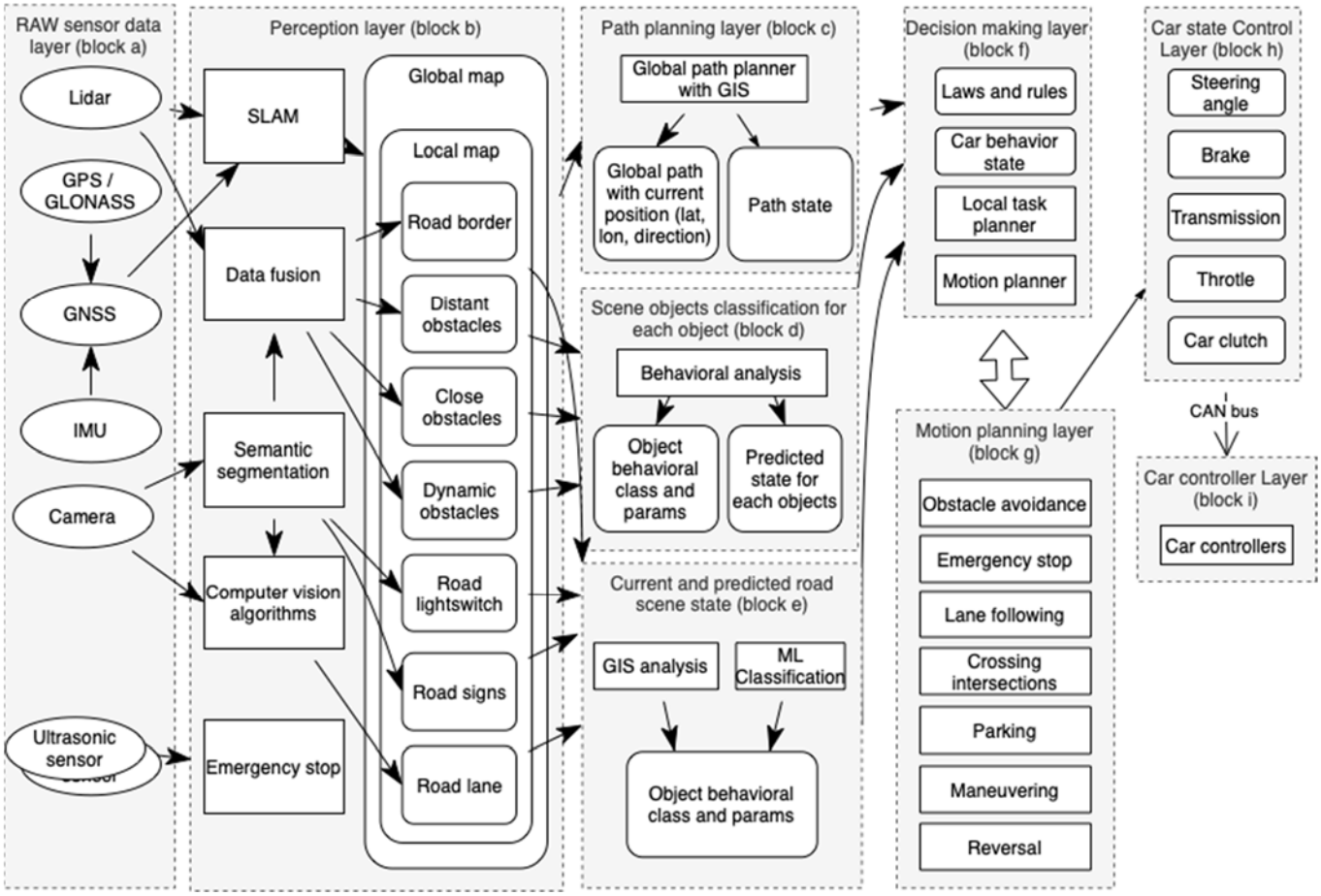

Figure 1. Structural and functional model of the control system of a self-driving car.

The function of controlling a self-driving vehicle is as follows (block h on Figure 1):

$$Cs_t = \text{control}(Tw_t) \tag{2}$$

Tw—is a tuple characterizing the parameters of the vehicle's movement. a—target acceleration, α—desired angular velocity.

At the same time, the control function checks the current state of the vehicle, the ability to set target values, and generates specific control actions for the engine controller and control modules.

The structural and functional model of a self-driving car control system can be represented as follows (Figure 1).

The data and the following expressions are intended to be used as formalized specifications when creating a behavioral (imitation) model for the operation of a self-driving car. The use of specification of abstract machines networks simplifies the further implementation of an intelligent behavioral model of a discrete-event system based on known programming languages.

Generation of a values is a task of continuous decision making (block f on Figure 1) and motion planning (block g on Figure 1) module for the vehicle (i.e., the generation of a sequence of desired accelerations $a_0 = (a(t=0), a(t=1), a(t=2), \cdots)$, for example, for traversing an unsignalized intersection with an arbitrary layout and a variable number of other traffic participants with unknown intentions.

Path planning (block c on Figure 1) is performed in two stages [21]. In the first, the path of the vehicle r_0 must be free of collisions with static obstacles and is either generated by the path planner or given by the geometry of the road of the given map. At the second stage, the linear velocity is planned along r_0. This practice is called in the literature [22] path-velocity decomposition and reduces the problem to the problem of planning a trajectory in one-dimensional space.

The environment (block d on Figure 1) contains a set of agents $A = A_0, \cdots, A_K$ with $K \in N$, where A_0—target vehicle with the designed control system. Every other agent A_k, with $k \in \{1, \cdots, K\}$ has a set of future path hypotheses. The path of the vehicle, $r^{(0)}$, and all other agents path hypotheses are defined within the topological map $R = r^{(0)}, r^{(1)}, \cdots, r^{(I)}$, with $I \in N_0$ $\mathrm{r}(i) = \left\{q_{i,0}q_{i,1}, \cdots, q_{i,J-1}q_{i,J}\right\}$ for $i \in 0, \cdots, I, j \in 0, \cdots, J$ and $J \in N_0$, and $q_{i,j} \in R^2$ being the position of waypoint j of route i. Every agent A_k is mapped onto a corresponding path $r_k : A_k \to r \in R$ on which it moves with velocity $v_k(t) \in [0, v_{max}]$ for time $t \in [0, \infty)$. Every path has a set of following path hypotheses $M^{(i)}$, with $M^{(i)} = succ(r^{(i)})$. An agent traverses from its current to its next path, r'_k, with the unknown probability $P(r'_k|r_k)$.

A path intersection function between two routes $c(ri,\ rj)$ is defined as

$$c(r_i, r_j) = \begin{cases} 1, \text{ if } r_i \cap r_j \neq \varnothing \\ 0, \quad \text{otherwise} \end{cases}, \forall i, j \in \{0, \cdots, I\}, i \neq j \tag{3}$$

Taking into account the uncertainty in the movement of other vehicles, the autonomous vehicle must constantly choose the optimal acceleration a^* to maximize the expected accumulated reward in the future.

$$a^* := \underset{a(t=0)}{\operatorname{argmax}}\, E\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | a(t=0)\right] \tag{4}$$

This reward should be inversely proportional to the time of crossing the intersection, the total acceleration and speed deviation from the curvature, and the initial speed based on the law of the path following, taking into account the limited possible acceleration and avoidance of collisions. There are several path planning methods, the most promising are LQR [23], ILQR [24], as well as algorithms based on inference rules. Such algorithms should be based on the rules of the road. Because the implementation of a knowledge base (block f on Figure 1) of such rules is a laborious process, then develop the expert system with an internal language for describing the rules is promising [25].

$$Tw_t = \mathrm{MotionPlanning}(Mn_t,\ Sc_t) \tag{5}$$

Mn—nearest maneuver defines the current local travel task to be performed by the vehicle,
Sc—current scene.

$$Mn_t \in \left\{ \begin{array}{c} \text{LaneFollowing, LeftTurn,} \\ \text{RightTurn, Front, Parking, Reversal,} \\ \text{EmergencyStop, Roundabout Circulation} \end{array} \right\} \quad (6)$$

$$Mn_t = \text{GISLocalPlanner}(\text{carPositionState}_t, \text{p}_t) \quad (7)$$

The GISLocalPlanner function (block c on Figure 1) selects the closest local maneuver to follow the intended path. This task is solved in GIS systems such as Yandex Maps, Google Maps, OpenStreetMap and allows you to drive a car with a change of modes. Modern GIS systems do not provide access to high-precision maps, but there are many services that have solved this problem for self-driving (TomTom [26], DeepMap (Nvidia [21]), Ushr [27], and others). This method is based on the use of open APIs of the specified map services.

Sc_t—is the description of the current scene, identified based on the analysis of data from sensors. Understanding scenes (block c on Figure 1) is one of the key challenges in self-driving car design. In this task, it is extremely important to build several levels of abstraction in scene decomposition, from raw sensor data to a high level of abstraction using complex static and behavioral models. In general, the scene can be represented as a set of values:

$$Sc_t = (A_t, m_t, Rs_t, Rd_t, Tls_t) \quad (8)$$

A—agents,
$A = \{A_0, \cdots, A_K\}$ with $K \in N$, $A_{k,t} = (latitude_{k,t}, longitude_{k,t}, Cb_{k,t})$
Cb—class for a behavioral characteristic of the agents [28]. It has one of the following values:

'vehicle overtaking from left',
'vehicle overtaking from right',
'vehicle driving away to left',
'vehicle driving away to right',
'vehicle driving in from left',
'vehicle driving in from right',
'vehicle parallel driving in left',
'vehicle parallel driving in right',
'vehicle straight accelerating',
'vehicle straight decelerating',
'vehicle uniformly straight driving',
'vehicle stopping',
'walking (riding) away and getting closer',
'walking (riding) up and getting closer',
'riding away and getting farther',
'pedestrian (rider) crossing slowly from right',
'pedestrian (rider) crossing slowly from left',
'pedestrian crossing quickly from right',
'pedestrian crossing quickly from left',
'pedestrian (rider) stopping',
'others (vehicles, pedestrians, riders)'

m_t is a two-dimensional occupancy grid generated by SLAM module (block b on Figure 1), it represents a fine-grained grid over the continuous space of locations in the environment. $p_{i,j,t}$ represents the probability that cell with index (i, j) is occupied. Let $z_{1:t}$ is the set of measurements from time 1 to t, and $x_{1:t}$ is the set of robot poses from time 1 to t. Then the posterior of a map is approximated by factoring it into

$$p(m_i|z_{1:t}, x_{1:t}) = \prod_i p\left(m_{(i)} \middle| z_{1:t}, x_{1:t}\right) \quad (9)$$

Each measurement is a data obtained from lidar. $z_i = (d_{i,j})$, $j = [1, \mathrm{V}]$ lidar beam number up to viewing angle V, $d_{i,j} = [0, \infty)$—distance from vehicle to obstacle for each beam.

Promising technology is a fusion method [29] that allows using data from lidar and camera simultaneously.

$$z_i = (d_{i,j}, c_{i,j}) \tag{10}$$

$c_{i,j} \in \{0, \cdots, \mathrm{S}\}$, S is a segmentation class number. Currently we recognize the following semantic classes:

0—Road
1—Car
2—Truck
3—Pedestrian
4—Bicycle
5—Motorcycle
6—Road signs
7—Road light switch
8—Sidewalk

Classes 1–5 are agents with specific behavior, 6, 7 influences decision making when driving, and 0 is used to control the car whilst moving.

The classical broadly accepted formal definition of image segmentation is as follows. If $P(o)$ is a homogeneity predicate defined on groups of connected pixels, then the segmentation is a partition of the set I into connected components or regions $C = \{C^1, \cdots, C^n\}$, such that

$$\bigcup_{i=1}^{n} C^i \; with \; C^i \cup \; C^j = \varnothing, \; \forall \; i \neq j \tag{11}$$

The uniformity predicate $P(C_i)$ is true for all regions C^i and $P(C^i \cup \; C^j)$ is false when $i \neq j$ and sets C^i and C^j are neighbors. The result of the segmentation module is the image with same size as original image where each pixel is labeled with semantic class described before. Semantic segmentation is realized with deep convolutional network U-Net with attention models.

This segmentation mask can be used for map generation and for the behavioral analysis of agents (interactive event recognition and intention prediction [30]):

$$A_t = \mathrm{AgentsRecognition}(C_t) \tag{12}$$

$$Rs_t = \mathrm{TrafficSignRecognition}(C_t) \tag{13}$$

$$Rd_t = \mathrm{RoadBoundariesRecognition}(C_t) \tag{14}$$

$$Tls_t = \mathrm{TrafficLightSwitchRecognition}(C_t) \tag{15}$$

Additional recognition modules use the result of semantic segmentation and perform a classification task with convolutional neural networks:

Rs_t is a traffic sign recognition module that allow to recognize more than 100 sign classes [31]. It uses the convolutional neural network and open dataset.

Tls_t is a traffic light switch detection module based on the computer vision algorithm.

Rd_t is a road boundaries detection module that uses the computer vision algorithm to detect image area that corresponds to a road.

A_t is an agent recognition module.

3. Results

Following the functional model described above, a prototype of a self-driving car based on the production truck GAZelle Next was developed. The car was developed for participation in the "Robocross 2021" competition. The task was the following: It was necessary to drive a part of the road bounded by bumpers, on which barrels obstacles were

placed, recognize all road signs, make a U-turn at the place indicated on the map, and return to the finish zone. All tasks were to be completed in a fully autonomous way. RoboLife's teams of authors took second place overall and first place among autonomous trucks. To develop the control system, the structural and functional model presented in this article was used. The implementation of modules related to the perception layer is described in articles [31,32]. The Decision Making and Motion Planning layer is implemented using a rule-based approach, implemented as a program code in the python language.

The car (Figure 2) is equipped with the following sensors:

- Lidar Ouster OS-1, 16-channel.
- Stereo Camera Stereolabs ZED.
- GPS sensor U-blox GY-GPSV3-NEO-M8N.
- Eight ultrasonic parking sensors.
- Limit switches.

Figure 2. Self-driving truck.

For control are used:

- For gear shifting—2 Linak LA15 linear actuators, self-developed mechanical parts for connecting actuators.
- Linak LA36 actuators for brake and clutch pedals.
- To rotate the steering wheel—electric power steering from the Lada Kalina car.
- An additional rotation sensor was installed on the steering shaft with the transmission of the shaft rotation angle without the need for synchronization.

A single-board computer Nvidia Jetson Xavier is used for control, and STM32 family controllers are used for motors controlling.

The Robotics Operating System (ROS) is used at the software control level. ROS is a software on top of the Linux operating system, which makes it easy enough to develop robot control systems through the use of standard solutions:

- Drivers for manipulators and sensors;
- The platform of client-server distributed architecture using message transfer between different objects;
- The ability to implement modules in popular programming languages (including C ++ and Python).

At the abstract level, the architecture of the solution in terms of ROS nodes is as follows (Figure 3).

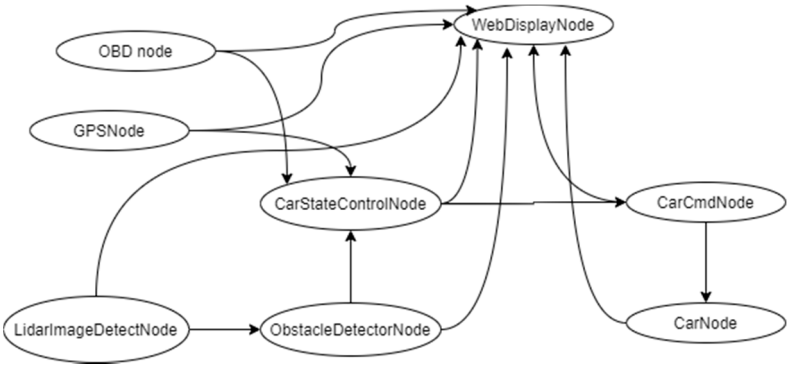

Figure 3. Nodes in ROS solution.

This diagram shows the software components and the order in which messages are passed between them. These software modules correspond to the structural and functional diagram of the control system of a self-driving car:

GPSNode—GPS data processing with NMEA format parser.

CarStateControlNode—Decision Making Layer and Path Planning Layer. Uses rule-based control algorithm with detection of near and far obstacles. It is necessary to take into account the dimensions of the machine when avoiding obstacles. Additionally, it uses PID regulation for the wheel control to follow the way.

ObstacleDetectorNode—Scene Recognition Layer. This node recognizes obstacles of two types: near and far. For obstacle recognition, lidar data is converted from a PointCloud array to a frontal projection image, which is a depth map, where the value of each pixel corresponds to the distance to the object.

CarCmdNode and CarNode—Car Controller Layer. Use car state, path state to generate control messages for throttle, transmission, wheel, brake, and clutch.

LidarImageDetectNode—Perception Layer. Lidar and camera used for scene image segmentation and object detection. It realizes deep learning methods to detect road signs [26], road boundaries [27].

OBDNode—node that collects data from CAN bus of the car through OBD connector (engine speed, speed, brake state).

WebDisplayNode—map path constructor and visualization tool (Figure 4). Tool based on OpenStreetMap (offline maps) for planning a route and visualizing the current state of the car (point on the map, distance from key points of start, turn and finish, speed, recognized sign, lidar data, near and far obstacles, and planned path).

Figure 4. Mapping tool and car state visualization.

The use of a real car seems to be extremely costly [28] for testing all launch scenarios, as well as testing prototypes of control algorithms. To debug the components of the solution, the following tools were used, developed by the authors of the article when creating a self-driving truck:

- Simulation of moving and control of path following with GPS coordinates on a pre-built map.
- Simulation of PointCloud obtained from lidar using recorded data in rosbag format.
- Simulation of recognition of road signs and traffic light switch from a recorded video file with a data augmentation subsystem.

The experimental results are presented in [31,32].

To debug a full solution, 40 races were performed in a real car in order to adjust the parameters of the algorithms. As a result, 15 races with the configured parameters were completed, all 15 were successful—the car successfully coped with the task at the competition, taking second place in the test races.

From the point of view of developing a parametrized model of the control system, parametrization and using of elements of CAD design system (mapping, control data visualization) made it possible to reduce labor costs for development by about 1.5 times. The comparison was carried out on projects to develop a control system for a self-driving car for the Robocross competitions in 2018, 2019, 2021. A new control system was developed every year. Labor costs in 2021—480 man-hours approximately, in 2019—710 man-hours approximately.

4. Conclusions

The functional model described in the article contains a description of the control modules for a self-driving car, their functional relationship and briefly outlines the approaches used in the implementation of each of the modules. An important aspect is the parameterization of each module, which is important in the development of interfaces for the interaction of modules. This scheme was used in the design of a self-driving truck for the Robocross 2021 competition [33], and can also be used in the design of self-driving cars. Its peculiarity is that it can be used to build a specialized CAD system based on a library of related components with functional dependencies and parameters. These components can be used according to the communication protocols presented in this article. The implementation of each of the components depends on the requirements for the problem being solved, on the specific design specifications.

An example of using the presented functional model is the self-driving truck built on the basis of ROS, developed by the team of authors of this article, which took second place in the overall standings and first place among trucks. In general, the proposed models, parameters, and individual elements of the CAD system made it possible to speed up the process of developing a control system by almost one and a half times, expressed in man-hours.

Author Contributions: Conceptualization, K.S.; methodology, K.S.; software, P.K., R.J., V.M.; validation, P.K., I.R.; resources, I.R.; data curation, D.K.; writing—original draft preparation, K.S.; writing—review and editing, N.Y.; supervision, N.Y.; project administration, K.S.; funding acquisition, N.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This study was funded by Ministry of Education and Science of Russia in framework of project No. 075-00233-20-05 from 3 November 2020 «Research of intelligent predictive multimodal analysis of big data, and the extraction of knowledge from different sources».

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Authors are grateful for a SimbirSoft, LLC and RITG, LLC companies.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Stoma, M. The Future of Autonomous Vehicles in the Opinion of Automotive Market Users. Available online: https://www.mdpi.com/1996-1073/14/16/4777/pdf (accessed on 7 August 2021).
2. Baltic, T.; Cappy, A.; Hensley, R.; Pfaff, N. The Future of Mobility Is at Our Doorstep. Available online: https://www.mckinsey.com/~{}/media/McKinsey/Industries/Automotive%20and%20Assembly/Our%20Insights/The%20future%20of%20mobility%20is%20at%20our%20doorstep/The-future-of-mobility-is-at-our-doorstep.ashx (accessed on 8 August 2021).
3. Using a Powerful Integrated Digital Solution to Develop Autonomous Vehicles. Available online: https://www.plm.automation.siemens.com/global/ru/resource/autonomous-vehicle-development-software/93559 (accessed on 7 August 2021).
4. Fletcher, R.; Mahindroo, A.; Santhanam, N.; Tschiesner, A. The Case for an End-To-End Automotive Software Platform. Available online: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/the-case-for-an-end-to-end-automotive-software-platform (accessed on 8 August 2021).
5. Model-based Systems Engineering for Autonomous Vehicle Development. Available online: https://www.plm.automation.siemens.com/media/global/ko/Siemens-PLM-Model-based-systems-engineering-for-autonomous-vehicle-development-wp-31354-A3_tcm72-52979.pdf (accessed on 9 August 2021).
6. Liu, L.; Lu, S.; Zhong, R.; Wu, B.; Yao, Y.; Zhang, Q.; Shi, W. Computing Systems for Autonomous Driving: State-of-the-Art and Challenges. Available online: https://arxiv.org/pdf/2009.14349.pdf (accessed on 9 August 2021).
7. Kuutti, S.; Bowden, R.; Jin, Y.; Barber, P.; Fallah, S. A Survey of Deep Learning Applications to Autonomous Vehicle Control. Available online: https://arxiv.org/pdf/1912.10773.pdf (accessed on 2 November 2021).
8. Haavaldsen, H.; Aasboe, M.; Lindseth, F. Autonomous Vehicle Control: End-to-End Learning in Simulated Urban Environments. Available online: https://arxiv.org/abs/1905.06712 (accessed on 20 October 2021).
9. Yurtsever, E.; Lambert, J.; Carballo, A.; Takeda, K. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. Available online: https://arxiv.org/pdf/1906.05113.pdf (accessed on 30 October 2021).
10. Omeiza, D.; Webb, H.; Jirotka, M.; Kunze, L. Explanations in Autonomous Driving: A Survey. Available online: https://arxiv.org/pdf/2103.05154.pdf (accessed on 1 November 2021).
11. Lee, D.-H.; Chen, K.-L.; Liu, K.-H.; Liu, C.-L.; Liu, J.-L. Deep Learning and Control Algorithms of Direct Perception for Autonomous Driving. Available online: https://arxiv.org/pdf/1910.12031.pdf (accessed on 30 October 2021).
12. Kaur, P.; Taghavi, S.; Tian, Z.; Shi, W. A Survey on Simulators for Testing Self-Driving Cars. Available online: https://arxiv.org/pdf/2101.05337.pdf (accessed on 8 August 2021).
13. Gazebo. Available online: http://gazebosim.org/ (accessed on 17 August 2021).
14. Carla Simulator. Available online: https://carla.org/ (accessed on 17 August 2021).
15. Webots Simulator. Available online: https://cyberbotics.com/ (accessed on 17 August 2021).
16. NVidia Autonomous Vehicle Simulation. Available online: https://www.nvidia.com/en-us/self-driving-cars/simulation/ (accessed on 17 August 2021).
17. CarSim Mechanical Simulation. Available online: https://www.carsim.com/products/carsim/index.php (accessed on 17 August 2021).
18. Matlab Automated Driving Toolbox. Available online: https://www.mathworks.com/products/automated-driving.html (accessed on 17 August 2021).
19. Siemens PreScan Simulation Platform. Available online: https://tass.plm.automation.siemens.com/prescan-overview (accessed on 15 August 2021).
20. SVL Simulator. Available online: https://www.svlsimulator.com/ (accessed on 16 September 2021).
21. NVidia Mapping for Self-Driving Cars. Available online: https://www.nvidia.com/en-us/self-driving-cars/hd-mapping/ (accessed on 8 September 2021).
22. Kant, K.; Zucker, S.W. Toward efficient trajectory planning: The path-velocity decomposition. *Int. J. Robot. Res.* **1986**, *5*, 72–89. [CrossRef]
23. Chen, J.; Zhan, W.; Tomizuka, M. Autonomous Driving Motion Planning with Constrained Iterative LQR. *IEEE Trans. Intell. Veh.* **2019**, *4*, 244–254. [CrossRef]
24. Pan, Y.; Lin, Q.; Shah, H.; Dolan, J.M. Safe Planning for Self-Driving via Adaptive Constrained ILQR. Available online: https://arxiv.org/abs/2003.02757 (accessed on 1 September 2021).
25. Zhang, Z.; Jiang, Q.; Sun, B.; Song, L.; Wang, R.; Mei, T.; Huang, X.; Wang, L.; Yu, B. Researches on Expert System for Automatic Driving Traffic Rules of Unmanned Vehicle. *J. Phys. Conf. Ser.* **2018**, *1069*, 012016. [CrossRef]
26. TomTom HD Maps. Available online: https://www.tomtom.com/products/hd-map/ (accessed on 8 September 2021).
27. Ushrauto Maps. Available online: https://www.ushrauto.com/ (accessed on 8 September 2021).
28. Xue, J.; Fang, J.; Li, T.; Zhang, B.; Zhang, P.; Ye, Z.; Dou, J. BLVD: Building A Large-Scale 5D Semantics Benchmark for Autonomous Driving. Available online: https://arxiv.org/pdf/1903.06405.pdf (accessed on 1 October 2021).
29. De Silva, V.; Roche, J.; Kondoz, A. Robust Fusion of LiDAR and Wide-Angle Camera Data for Autonomous Mobile Robots. Available online: https://arxiv.org/pdf/1710.06230.pdf (accessed on 5 September 2021).
30. Siemens-Microsoft Enabling Simulation at Scale. Available online: https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RWwzpZ (accessed on 10 September 2021).

31. Sviatov, K.; Miheev, A.; Kanin, D.; Sukhov, S.; Tronin, V. Scenes Segmentation in Self-driving Car Navigation System Using Neural Network Models with Attention. In *Lecture Notes in Computer Science, Proceedings of the Computational Science and Its Applications—ICCSA 2019, Saint Petersburg, Russia, 1–4 July 2019*; Misra, S., Ed.; Springer: Cham, Switzerland, 2019; Volume 11623.
32. Sviatov, K.; Miheev, A.; Lapshov, Y.; Shishkin, V.; Sukhov, S. Detection of Scenes Features for Path Following on a Local Map of a Mobile Robot Using Neural Networks. In *Recent Research in Control Engineering and Decision Making, Proceedings of the ICIT 2020, Saratov, Russia, 3–4 December 2020*; Dolinina, O., Ed.; Studies in Systems, Decision and Control; Springer: Cham, Switzerland, 2021; Volume 337. [CrossRef]
33. Robocross Competitions. Available online: https://www.russianrobotics.ru/competition/robocros/ (accessed on 3 September 2021).