Facilitating Autonomous Vehicle Research and Development Using Robot Simulators on the Example of a KAMAZ NEO Truck

Shyngyskhan Abilkassov, Anuar Nurlybayev, Sergey Soltan, Anton Kim,
Elizaveta Shpieva, Nurzhan Yesmagambet, Zhandos Yessenbayev and Almas Shintemirov

***Abstract*— With the widespread of research in the field of autonomous vehicles the value and impact of various simulators increase dramatically as they allow for quick and safe experimentation with the design of a vehicle, environment and driving scenarios. In this paper, the authors demonstrate how autonomous vehicle research and development can be facilitated by open-source robot simulators based on the experience gained from a robotized KAMAZ NEO truck industrial project. In particular, the Webots robot simulator was applied for 3D reconstruction of the experimental test-site for vehicle motion simulation and development of a web-based dashboard for controlling and monitoring the autonomous vehicle both in the simulation and the real-world.**

I. Introduction

Specialized software simulators found wide application in autonomous vehicle research due to their flexibility to model realistic experimental scenarios for testing vehicle motion planning algorithms. Example applications are simulation of traffic in urban regions. Several simulators are being developed for the needs of research such as SUMO (Simulation of Urban Mobility), the open-source simulator for designing communications between platoons of autonomous vehicles [1]. Another example is AORTA (Approximately Orchestrated Routing and Transportation Analyzer) simulator - a traffic simulator with continuous simulation, where autonomous vehicles analyse roads every time-step for traffic jams, intersections to follow the shortest route to goal [2]. The SiVIC (Simulation for Vehicle, Infrastructure, and Sensors) was developed to be able to design a copilot for maneuver-based trajectory planning [3]. Immersive vehicle simulators such as the Stanford Driving Simulator can be used to provide realistic interface to user and receive new velocity and position correction. These simulators can be used to imitate driver distraction emergency situations [4].

This research was funded under the Nazarbayev University industrial project "Development of a Robotized Vehicle on a KAMAZ chassis" supported by VIST (ZYFRA Group) (Russia), and the young researchers' grant project "Development of an Autonomous Skid-Steering Based Mobile Robot-Manipulation System for Automating Warehouse Operations in Kazakhstan" (Project IRN AP08052091), funded by the Ministry of Education and Science of the Republic of Kazakhstan. The work of Z. Yessenbayev was partially supported from the grant No. AP05134272, funded by the Ministry of Education and Science of the Republic of Kazakhstan.

S. Abilkassov, A. Nurlybayev, S. Soltan, A. Kim, N. Yesmagambet, A. Shintemirov are with the Department of Robotics and Mechatronics, Nazarbayev University, Nur-Sultan City, Kazakhstan.

Z. Yessenbayev is with the National Laboratory Astana, Nazarbayev University, Nur-Sultan City, Kazakhstan.

E. Shpieva is with the VIST Robotics (ZYFRA Group), Moscow, Russia.

Corresponding author - A. Shintemirov, ashintemirov@nu.edu.kz

Fig. 1. Experimental KAMAZ NEO 5490 truck redesigned to a robotized vehicle.

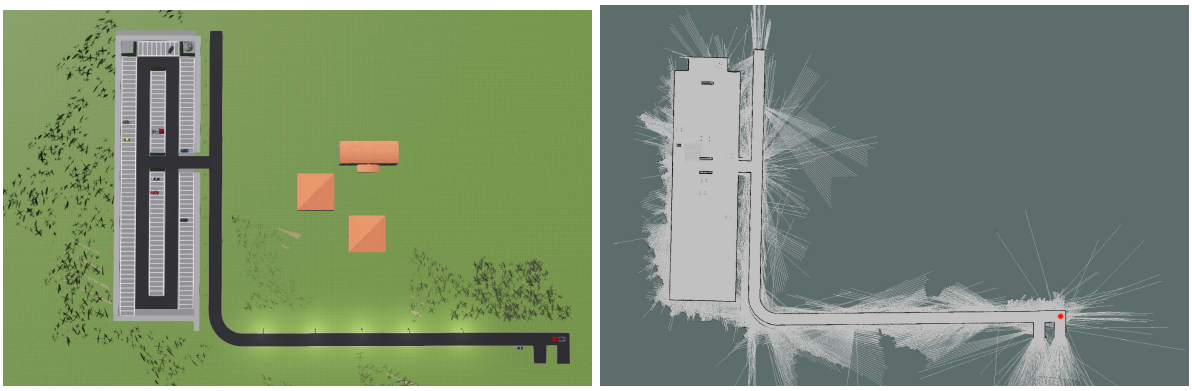
In addition to wide applications in robotics research, open-source mobile robotics simulation software platforms such as Webots [5] have high potential for facilitating autonomous vehicle research. In [6] and [7] the simulator was applied for performance evaluation of vehicular platoons. This paper presents a case study of application of the Webots simulation software for facilitating an industrial project conducted at Nazarbayev University in cooperation with VIST (ZYFRA Group) company [8], the leading developer of self-driving and remotely driven robotized cargo vehicles for mining industry in Russia, focusing on collaborative development of a robotized vehicle on the basis of a novel KAMAZ NEO 5490 truck chassis provided by KAMAZ [9], the largest Russian truck manufacturer.

II. Hardware and Software Platforms for the Robotized KAMAZ Truck Project

Within the robotized KAMAZ research project tasks were divided as follows: the VIST (ZYFRA Group) partners retrofitted a test KAMAZ vehicle to drive autonomously by equipping it with their patented autopilot hardware/software system, thus, in fact, converting the truck into a mobile robot platform, that can be remotely telecontrolled by a human operator or move autonomously following a high-level mission planner (Fig. 1). For establishing a low-level control of the vehicle, the VIST(ZYFRA Group) team adopted their autonomous software models and algorithms from previous projects in robotization of dump mining trucks such as algo-

(a) (b)

Fig. 2. a) The project test site simulated in the Webots environment; b) A Webots truck model used for autonomous vehicle motion simulation.



(a) (b)

Fig. 3. a) Top view of the Webots simulated project test site; b) Top view of the artificially generated map for ROS-based vehicle motion planning.

rithms for steering wheel control and modules of acceleration and brake controls for specific types of trucks with different physical models. Communication with KAMAZ vehicle's transmission and braking systems was easily set up via CAN and LIN protocols. Due to the vehicle's mechanical steering system, a dedicated mechatronic motion control system for the vehicle's steering wheel was designed by the VIST (ZYFRA Group) team engineers, for steering the vehicle following digital commands. The most complicated part of the project work was to design a hardware controller for the vehicle's acceleration control system due to its atypical PWM signal format. Additional equipment for autonomous driving such as navigation system, network system and set of sensors were also supplied as part of the VIST (ZYFRA Group) standard equipment kit. Moreover, a fully equipped remote control center was installed so that a vehicle operator could efficiently supervise autonomous driving of the robotized KAMAZ truck during experimental testing.

In parallel, the NU project team was tasked to develop software modules for vehicle autonomous motion planning and trajectory following, and machine vision detection and recognition of pedestrians, vehicles and road signs. As part of the project the NU team proposed to develop a 3D visualisation environment for simulation and initial testing of the developed vehicle motion planning software module. More detailed information about the project developments with video demos is available at the project web-page `https://www.alaris.kz/research/robotized-kamaz-truck/`

The Robot Operating System (ROS) [10] was chosen as a software platform for integrating developed modules with the vehicle autopilot system based on the experience of the project partner VIST (ZYFRA Group) [11]. ROS is a de-facto standard software development platform for intelligent robotics research due to its distributed and modular architecture allowing easy integration of custom control and data processing algorithms.

As the work on retrofitting the test KAMAZ vehicle started in parallel with the software module development, we started the project with identifying an experimental test site and creating its 3D simulation environment where the truck model could be placed for simulation analysis of the vehicle motion planning algorithms implemented in ROS. A comparative analysis of the freely available for academic use and ROS-compatible robot simulators including CoppeliaSim (VREP), Gazebo and Webots revealed that the latter is the most suitable option for quick utilization in the project.

Webots was initiated as a dedicated simulation software for mobile robotics research [7]. From 2019 it became an open-source platform for modeling experimental scenarios using a variety of commercial robots and typical vehicle models that can be equipped with virtual models of commercial

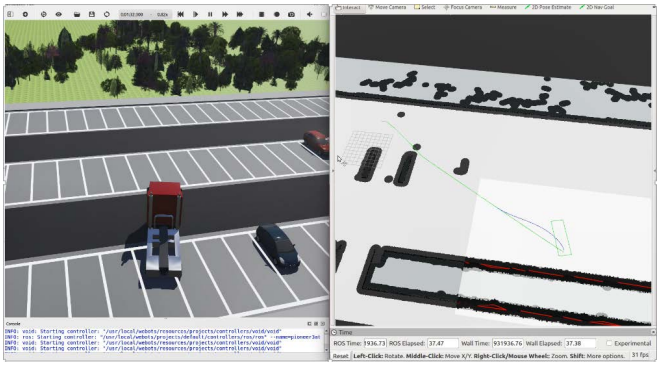
proximity sensors, i.e. LIDARs, GPS, IMU and others. The simulator provides graphical tools for modeling experimental environment. It is interfaced to ROS through standard ROS message passing mechanisms, i.e. topics and services, allowing direct control of the Webots vehicle models from ROS and receiving feedback data from the model sensors. This facilitates fast deployment and realistic testing of custom motion planning and sensor data processing algorithms, implemented externally in ROS as they would be applied on a real experimental robot.

III. Simulation of the Autonomous Vehicle Motion

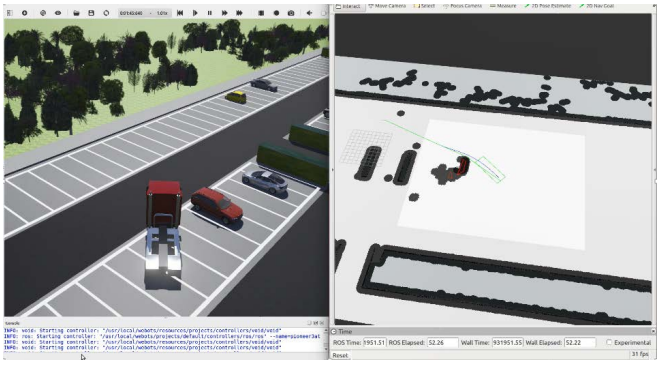
An outdoor open car parking site located within the Nazarbayev University campus was chosen as the project test site. Using the simulator's graphical tools and engineering blueprints of the parking space, its real-size 3D model was created in Webots as demonstrated in Fig. 2(a). The Webots truck vehicle model, shown in Fig. 2(b), was added to the simulated environment. To replicate the real robotized KAMAZ vehicle, a Velodyne VLP-16 LIDAR, IMU and GPS sensor models were added to the Webots vehicle model. The truck model was controlled from ROS through rosservice commands sending linear velocity and wheel rotation angle information and receiving vehicle's current position and orientation data from sensors.

Implementation of the vehicle motion trajectory planning module in an a-priory known environment requires a 2D occupancy map of the test site which reflects the road and parking site borders and all possible internal stationary obstacles. At the initial stage of the project it was decided to create a 2D map of the Webots simulated test site using the ROS *gmapping* package that implements the Rao-Blackwellized particle filter based algorithm for robot simultaneous localization and mapping (SLAM) [12]. The algorithm processed the LIDAR point cloud sensor measurements artificially generated and collected from the manually controlled truck model across the test site in Webots. Figure 3 presents the Webots simulated test site and its 2D occupancy map, where black and white pixels represent obstacle and free spaces, respectively, while grey pixels denote unexplored area. The generated map was saved using the ROS *map_server* package in the .pgm format for further use.

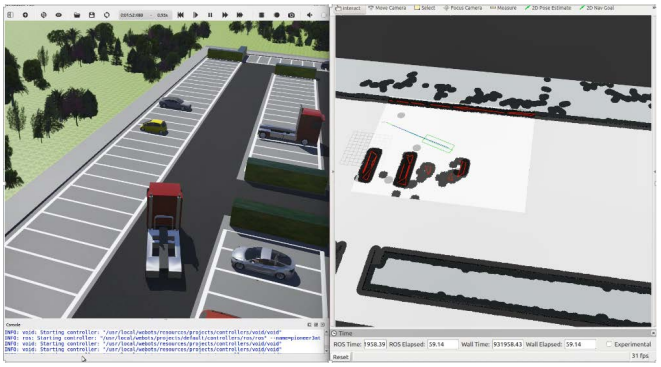
Due to short project duration it was ultimately agreed to utilize the built-in ROS Navigation Stack packages for vehicle trajectory planning and motion command generation. Specifically, we employ the ROS *move_base* package to apply A* global path planning algorithm [13] for vehicle trajectory planning to a dynamically specified target location within the test site. The Timed Elastic Band (TEB) planner (implemented as ROS *teb_local_planner* package) is applied for vehicle local trajectory planning and control command generation taking into account the car-like robot kinematics and dynamically detected obstacles [14]. Specifically, the algorithm finds an optimal solution along the pregenerated A* global trajectory, and creates the closest kinematically realistic local path as sequence of vehicle poses $p_i = (x_i, y_i, \theta_i)^T$.



(a)



(b)



(c)

Fig. 4. Webots simulation of the truck motion in the test site.

Furthermore, the algorithm generates a corresponding set of vehicle linear speed v_i and steering angle control commands within the predefined vehicle velocity and acceleration limits and safe distances to obstacles [15].

Figure 4 presents a sequence of time shots of the Webots simulation run demonstrating the truck motion generation from a starting position in Fig. 4(a) towards its target position in the middle of the test parking site as shown in Figs. 4(b) and 4(c). The left column of each subfigure shows the truck model moving inside the test site in the Webots simulator. The right column presents ROS RViz visualization of the 2D test site occupancy map with generated vehicle global (green) and local (blue) trajectories. The global trajectory defined the vehicle path from the vehicle start point to its target position. The local trajectory of the truck is restricted by the size of

the local costmap, i.e. the white area defining free space and dynamically detected obstacles, e.g. parked cars, in the vicinity of the truck. The detected obstacles are transformed to red colored convex polygons by the TEB planner in the local costmap, while the green rectangle denotes the truck footprint as shown in Fig. 4.

IV. Design of a Web-Based Visualization Dashboard

Direct interaction with ROS is not intuitive enough for non-users, especially, in the context of real-world testing of the autonomous vehicles software packages. As the robotized vehicle completely controls its motion including steering and speed controls, psychological comfort of the vehicle passengers can be ensured by increasing their awareness on vehicle actions through a suitable human-machine interface (HMI) [16]. The minimal setup for such a HMI should contain the vehicle's current status, its intention for the next action and navigation information as well as the basic control buttons [17], [18]. Majority of the HMIs implemented as dashboards for autonomous vehicles are patented and released as proprietary commercial software built by vehicle manufactures or third-parties like Tesla, Waymo, NVIDIA, with very limited configuration options. Existing open-source projects like Udacity Self-Driving Car [19] or OpenPilot [20] are still in their infancy, designed for specific car platforms, and would require additional customization.

As part of the industrial project we have developed a new simplified dashboard for the robotized KAMAZ truck control and state monitoring. The dashboard was implemented as a web application similarly to [21] but using *Flask* and *Vue.js* frameworks for the back- and front-end development, respectively. The web application format provides great flexibility in interface development and data presentation as it can be viewed on either on-board PC, tablet or cell-phone devices.

Figure 5 demonstrates the dashboard user interface consisting of two parts: a control panel and the Google Maps visualization of the project test site for real-time vehicle's position tracking along with its global and local trajectory display. During the experimental testing, the global and local path visualizations proved to be the most essential factors for ensuring truck passengers' awareness about the vehicle's intended actions, and, thus, overall driving comfort and safety. The control panel allows to establish connection with ROS-based truck control modules for setting and publishing vehicle trajectory waypoints and final vehicle pose on the test-site map, and, finally, to start and stop the vehicle in the autonomous mode. Additionally, there is also a second tab for vehicle telemetry information.

The dashboard can operate in two modes - *testing* in the Webots simulator or *production* when connected to the real vehicle. In either mode the ROS interface is established using the *roslibjs* library [22] via WebSockets [23] connecting to *rosbridge* for topic publishing and subscription, service calls and other ROS functionalities. The dashboard can be accessed remotely over the Internet or locally within the *roscore* based network.

The dashboard development was done using the Webots truck motion simulations. To monitor the position and state of the vehicle model in Webots, special conversion method was developed to convert data between the Webots local coordinate frame (shown in Fig. 5) and the GPS coordinate system used by Google Maps module in the web interface. The GPS uses the World Geodetic System (WGS84) as its reference coordinate system. Although Webots internally supports it as well, subsequent dashboard interfacing with the experimental robotized KAMAZ truck also required transformation of the truck coordinates defined in the local RTK (real-time kinematic) frame of the VIST (ZYFRA Group) autopilot system (the frame is defined as in Fig. 5). This allowed to use the same coordinate transformation from both the Webots simulated and real RTK local frames to the GPS and vice versa.

Given the vehicle's position with coordinates (x,y), specified in metres in a local frame, the corresponding GPS coordinates, i.e. latitude φ and longitude λ, are computed as follows [24]:

$$\theta = atan2(x,y)\cdot\frac{180}{\pi}; \tag{1}$$

$$d = \sqrt{x^2+y^2}; \tag{2}$$

$$\delta = \frac{d}{R}; \tag{3}$$

$$\varphi = \varphi_0 + \delta\cdot\cos\theta; \tag{4}$$

$$\Delta\psi = \ln\left(\frac{\tan(0.25\,\pi+0.5\,\varphi)}{\tan(0.25\,\pi+0.5\,\varphi_0)}\right); \tag{5}$$

$$q = \frac{\Delta\varphi}{\Delta\psi}; \tag{6}$$

$$\Delta\lambda = \delta\cdot\frac{\sin\theta}{q}; \tag{7}$$

$$\lambda = \lambda_0 + \Delta\lambda, \tag{8}$$

where θ and d denote bearing and distance , φ_0 and λ_0 are the GPS coordinates of the local frame and $R = 6,356,356.7$ m is the Earth's radius.

On the other hand, given the vehicle's goal position set in the dashboard in the GPS latitude and longitude coordinates, the conversion to a local frame is done as follows. Firstly, a distance and a rhumb line bearing are computed as below:

$$\theta = atan2(\Delta\lambda,\Delta\psi); \tag{9}$$

$$d = \arccos(\sin\varphi_0\cdot\sin\varphi + \cos\varphi_0\cdot\cos\varphi\cdot\cos\Delta\lambda)\cdot R, \tag{10}$$

where $\Delta\psi$ and $\Delta\lambda$ and are found using (5) and (7), respectively. Finally, the corresponding vertical and horizontal components of d are used as new equivalent x and y local coordinates.

Although, this coordinate transformation technique is not physically exact, the dashboard testing with the Webots simulations confirmed its sufficient accuracy comparable with the centimeter's scale precision of the RTK system used with the experimental KAMAZ vehicle. In addition, the low

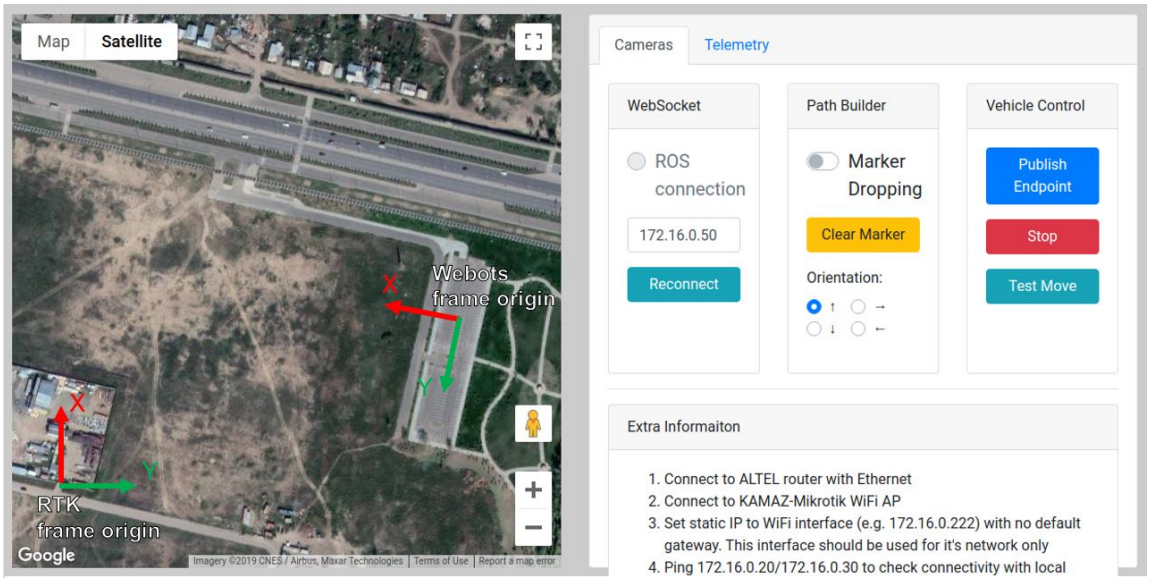


Fig. 5. Web-based dashboard interface for remote control and monitoring of the vehicle movement in real-time. Left side shows the KAMAZ RTK and the Webots frames in the global GPS frame.

computational power demands allowed remapping hundreds of points representing current and target positions of the truck model and its local and global trajectories from the Webots's local frame to the GPS in real-time.

V. Experimental Results

A. Generation of the Project Test Site Map

Integration of the adopted ROS-based motion planning module to the experimental robotized KAMAZ NEO truck started from preparing a realistic 2D map of the physical test site, i.e. the car parking site with connected road segments. The vehicle tracking was conducted in the local RTK frame, positioned such that it would cover the whole test site within the positive directions of x and y frame axes as shown in Fig. 5 for error-less coordinate transformations.

In the process of initial testing of the robotized KAMAZ truck sensor measurements were recorded from the vehicle Velodyne VLP-16 LIDAR, IMU, and RTK coordinates into ROSBAG files while the truck was manually driven across the test site, i.e. the parking site, until each obstacle was scanned at least ones. Using the collected sensor data a 3D point cloud of the test site was restored using the ROS-integrated Point Cloud Library (PCL) tools and then transformed into a 2D occupancy map as follows.

First, the point cloud data was processed with a Voxel Grid Filtering algorithm to reduce the point cloud size by grouping points into 3D voxels of a prespecified size equal to 0.2 m and approximating them by computing centroids in each voxel. Next, all points belonging to the asphalt covered surface of the test site were excluded. Due to the uneven nature of the test site surface plane it was experimentally determined to filter out all points that lie below a 0.09 m threshold level from it. The remaining points corresponded to boundaries (obstacles) of the test site. The final 3D point cloud representation of the test site obstacle map is presented

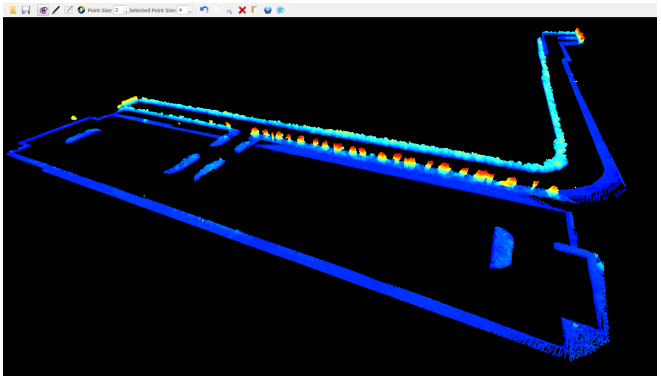


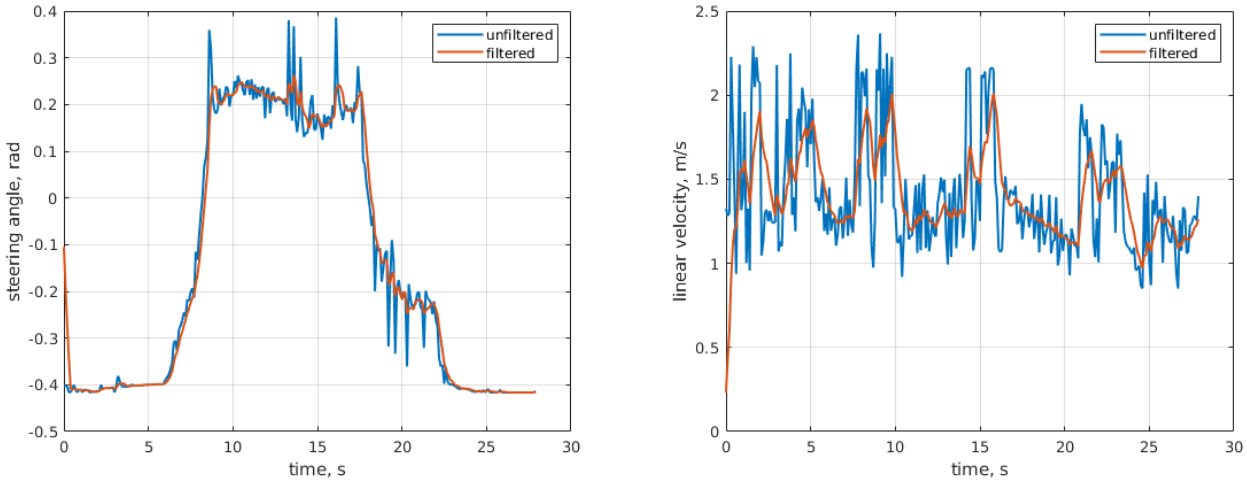
Fig. 6. Constructed 3D obstacle map of the project test site.

in Fig. 6. Subsequently, a 2D occupancy map was generated in the ROS .pgm format similarly to the artificial one in Fig. 3(b). The map covers the square area 500 x 500 m with the precision set to 0.4 m per pixel.

B. Interfacing of the Motion Planning Module

The adopted ROS-built-in vehicle motion planning modules, described in Section III, were interfaced with the real KAMAZ vehicle's autopilot system. Although simulation tests in Webots were accurate enough, during the initial experimental tests on the real vehicle it was observed that input velocity and steering angle commands to the vehicle autopilot system were rapidly changing that caused abrupt and fluctuated vehicle motion. This was attributed to high latency of the vehicle autopilot hardware that forced the planner algorithm to overcompensate and enforce control commands in subsequent actions.

To avoid abrupt changes in the velocity and steering angle, which could damage mechanical systems of the vehicle, a simple data smoothing was integrated adopting a Finite



(a) (b)

Fig. 7. (a) KAMAZ truck steering angle and (b) linear velocity command signals before and after filtering.

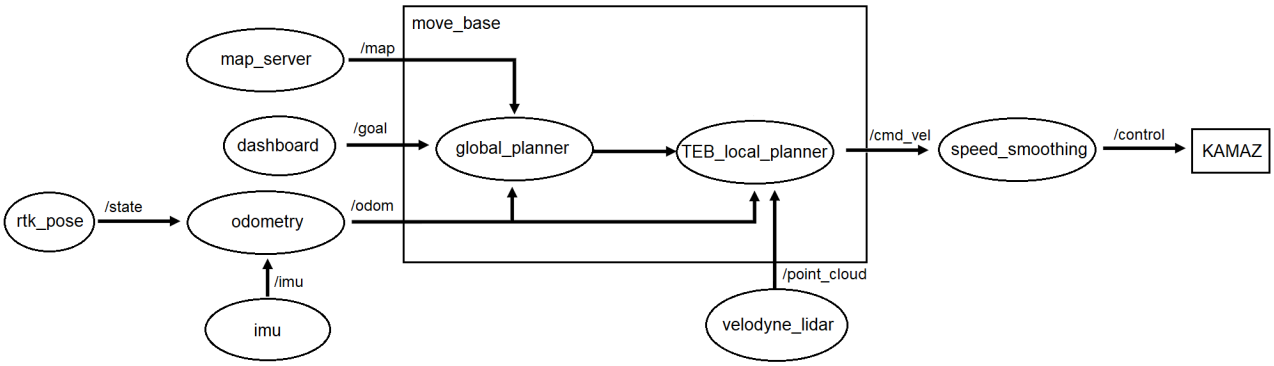


Fig. 8. Graphical representation of nodes and topics of the ROS-based motion planning module of the robotized KAMAZ truck.

Impulse Response Filter algorithm defined as

$$y[n] = \sum_{i=0}^{k} b_i x[n-i]. \tag{11}$$

After several experiments, the best performance was observed during 10-order discrete convolution of the steering angle. The filter weights were computed using an exponent function and were set as $b = \{4.1727,\ 3.6173,\ 3.1357,\ 2.7183,\ 2.3564,\ 2.0427,\ 1.7708,\ 1.5351,\ 1.3307,\ 1.1536\}$. In this configuration, the last and previous ROS-generated steering angle signals contribute 17.5% and 15% respectively in the filtered angle control command send to the vehicle. The velocity commands were processed with 5-order filter with $b = \{1.5169, 1.3956, 1.2840, 1.1814, 1.0869\}$. In this case, the last and previous ROS-generated velocity signals contribute 23% and 22% to the processed velocity command. Figure 7 shows the results of the integrated smoothing filter with command values from the TEB planner shown in blue and red before and after filtering.

Figure 8 presents a graphical representation of nodes and topics of the ROS-based motion planning module interfaced with the experimental robotized KAMAZ truck. The vehicle position and orientation are published by *rtk_pos* and *imu* nodes into */state* and */imu* topics, respectively, that are, in turn, connected to *odometry* node for odometry computation. The A* trajectory planner in *global_planner* node is subscribed to the */odom*, */goal* and */map* topics and based on the received data generates the truck global trajectory, that is then sent to *TEB_local_planner* along with the Velodyne LIDAR sensor measurements and the vehicle odometry for generating an obstacle-aware local path of the truck. The output linear velocity and steering angle control messages from *TEB_local_planner* node are passed via */cmd_vel* topic to *speed_smoothing* node for filtering, and are then ultimately sent to the robotized KAMAZ autopilot system for execution via */control* topic.

The developed motion planning module also implements a watchdog timer for stopping the vehicle in the case of significant delay or absence of ROS control messages. In addition, it is also able to integrate with a vehicle computer vision module, that reduces the truck velocity control signal in 80% from the original ROS-generated command upon

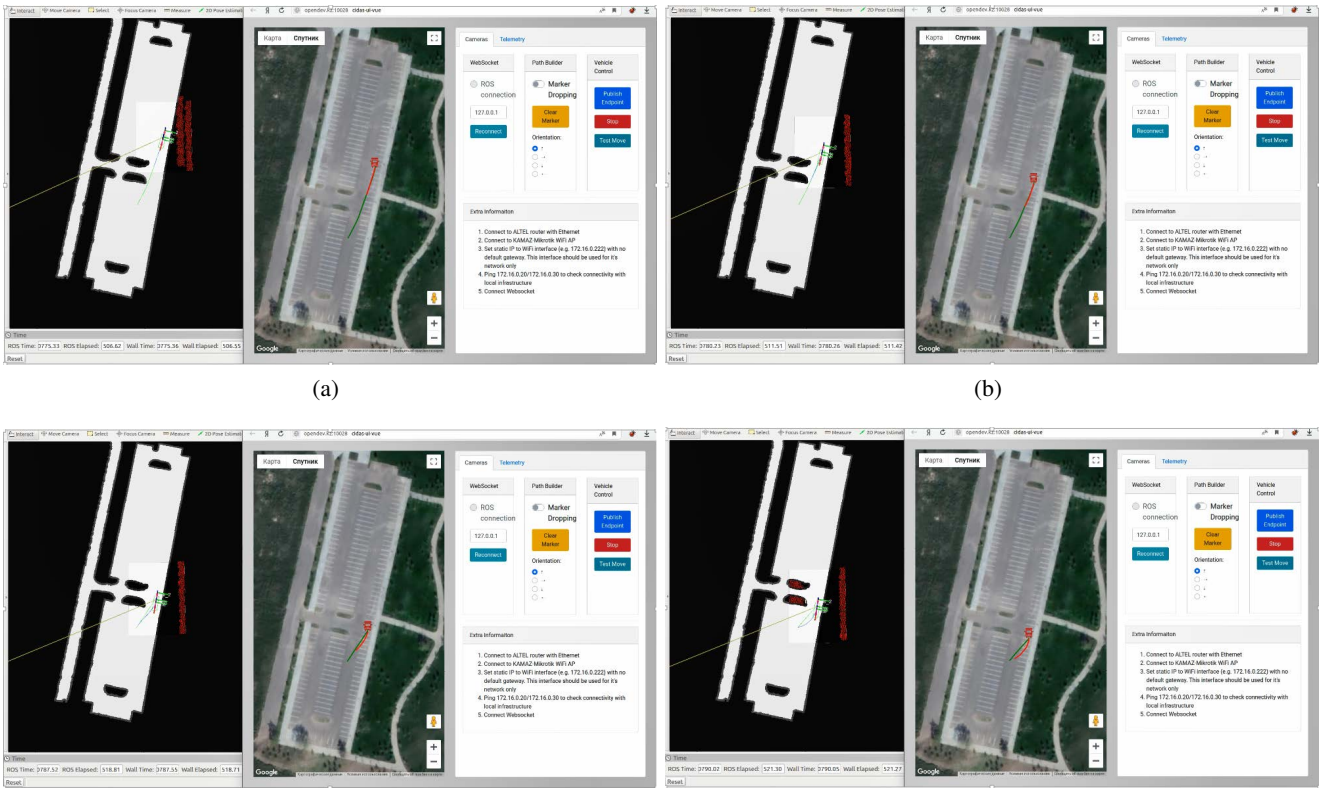


Fig. 9. Dashboard and ROS RVIS visualizations of autonomous motion of the robotized KAMAZ NEO truck.

detection of a human in the vicinity of the vehicle.

C. Visualization Dashboard Integration and Testing

Integration of the developed visualization dashboard into the experimental KAMAZ truck was done using the dashboard WebSockets interface. The truck real-time location with ROS-generated trajectories given in the RTK local frame were converted to the GPS coordinates of the dashboard following the coordinate transformation approach presented in Section IV.

On the other hand, after the vehicle target pose (position and orientation) is set in the dashboard in the GPS frame, it is converted to the local RTK frame and constantly published to the ROS-based motion planning module via */goal* topic as shown in Fig. 8.

Figure 9 shows one of test launches of the robotized KAMAZ truck on the test site visualized in the dashboard and the ROS RViz visualization on background. The red automobile marker in the dashboard's Google Maps window denotes the experimental KAMAZ truck while green and red curves visualize the global and local trajectories, respectively. After setting the truck target pose in the dashboard, the truck starts moving along the generated global and local trajectories as visualized in real-time in the dashboard in Figs. 9(a) and 9(b). However, as shown in Figs. 9(c) and 9(d) when the vehicle reaches the point on the global path that is too close to the road borders, the local path is reconstructed for maneuvering around the borders keeping save distance.

The tests were conducted with a driver and several team members present in the truck, where they monitoring the vehicle status via the web-based dashboard open on a vehicle onboard computer. The driver did not control the vehicle and was needed in case of a potential emergency situation for stopping the truck by switching off the autopilot system and getting manual control of the truck. Other team members were observing outside with the dashboard application run on handheld tablets. In overall, in all devices the dashboard application clearly demonstrated the vehicle status in real-time. Moreover, it was found out the along with the global vehicle trajectory, visualization of the local path is very valuable during experimental testing of the developed vehicle modules. This was especially useful for the test driver who could evaluate the truck behavior and intervene accordingly in case of n emergency and/or significant deviation of the vehicle from its generated trajectory.

VI. CONCLUSIONS

In this paper, we presented the case study of application of ROS and the Webots robot simulator for developing a web-based dashboard to support the development of an autonomous KAMAZ NEO truck. Using the desctribed in the paper simulation tools, we were able to implement additional autonomous functionally to the remotely controlled robotized truck prototype within a short period of time.

The presented work shows that open-source robot simulators can be used well in advance before a test autonomous

vehicle platform is ready for experimentation for solving various project tasks such as 3D modeling of the vehicle and the environment, 2D occupancy map generation, testing developed software modules and various driving scenarios.

It was experimentally confirmed that the vehicle dashboard helps vehicle passengers to feel more comfortable and safe during driving in an autonomous car in case of proper information visualization.

As future work we plan to apply the gained in this project experience and robot simulators for developing new robot motion planning algorithms based on reinforcement learning and social awareness.

ACKNOWLEDGMENTS

The authors would like to thank Mr. Artemiy Oleinikov for his contribution to the ROS TEB planner simulations along with other NU and NURIS team members for their valuable inputs to the overall project realization. Special thanks go to the VIST (ZYFRA Group) project team led by Elizaveta Shpieva and Artem Fedotov for their constant support and experience sharing.

REFERENCES

[1] P. Fernandes and U. Nunes, "Platooning of autonomous vehicles with intervehicle communications in SUMO traffic simulator," in *13th International IEEE Conference on Intelligent Transportation Systems (ITSC 2010)*, 2010, pp. 1313–1318.

[2] D. Carlino, S. D. Boyles, and P. Stone, "Auction-based autonomous intersection management," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, 2013, pp. 529–534.

[3] S. Glaser, B. Vanholme, S. Mammar, D. Gruyer, and L. Nouvelière, "Maneuver-based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 589–606, 2010.

[4] B. Mok, M. Johns, K. Lee, D. Miller, D. Sirkin, P. Ive, and W. Ju, "Emergency, automation off: Unstructured transition timing for distracted drivers of automated vehicles," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, 2015, pp. 2458–2464.

[5] "Webots: Open Source Robot Simulator," https://cyberbotics.com/, accessed: 2020-02-27.

[6] O. Karoui, E. Guerfala, A. Koubaa, M. Khalgui, E. Tovard, N. Wu, A. Al-Ahmari, and Z. Li, "Performance evaluation of vehicular platoons using Webots," *IET Intelligent Transport Systems*, vol. 11, no. 8, pp. 441–449, 2017.

[7] O. Michel, "Cyberbotics ltd. webots™: professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 5, 2004.

[8] "Zyfra Mining: Robotic vehicles for mining and industial use," http://vistgroup.ru/en/solutions/robotizirovannaya-tekhnika/, accessed: 2020-02-15.

[9] "KAMAZ PTC," https://kamaz.ru/en/, accessed: 2020-02-25.

[10] "ROS: Robot Operating System," https://www.ros.org/, accessed: 2020-02-22.

[11] E. Shpieva, "BELAZ on ROS: How do we in VIST Group create solutions for mining (in Russian)," https://habr.com/ru/post/476436/, accessed: 2019-09-10.

[12] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, pp. 34–46, 2007.

[13] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. The MIT Press, 2005.

[14] C. Rösmann, F. Hoffmann, and T. Bertram, "Online Trajectory Planning in ROS Under Kinodynamic Constraints with Timed-Elastic-Bands," in *Robot Operating System (ROS). Studies in Computational Intelligence. Vol. 707*, A. Koubaa, Ed. Springer, 2017, pp. 231–261.

[15] P. Marin-Plaza, A. Hussein, D. Martin, and A. de la Escalera, "Global and local path planning study in a ROS-based research platform for autonomous vehicles," *Journal of Advanced Transportation*, pp. 1–10, 2018.

[16] M. Elbanhawi, M. Simic, and R. Jazar, "In the Passenger Seat: Investigating Ride Comfort Measures in Autonomous Cars," *IEEE Intelligent Transportation Systems Magazine*, pp. 4–17, Fall 2015.

[17] N. Gowda, D. Sirkin, W. Ju, and M. Baltzer, "Tutorial on Prototyping the HMI for Autonomous Vehicles: A Human Centered Design Approach," in *Adjunct Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 2016, p. 229–231.

[18] O. Benderius, C. Berger, and V. Malmsten Lundgren, "The best rated human–machine interface design for autonomous vehicles in the 2016 grand cooperative driving challenge," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1302–1307, 2018.

[19] "Challenge 4: Self-Driving Car Android Dashboard," https://medium.com/udacity/challenge-4-self-driving-car-android-dashboard-83a2a5c8b29e, accessed: 2020-02-29.

[20] "Openpilot," https://github.com/commaai/openpilot, accessed: 2020-02-29.

[21] L. Marques, V. Vasconcelos, P. Pedreiras, and L. Almeida, "A flexible dashboard panel for a small electric vehicle," in *6th Iberian Conference on Information Systems and Technologies (CISTI 2011)*, 2011, pp. 1–4.

[22] "The Standard ROS JavaScript Library," http://wiki.ros.org/roslibjs, accessed: 2020-02-23.

[23] I. Fette and A. Melnikov, "The websocket protocol," 2011.

[24] "Calculate distance, bearing and more between Latitude/Longitude points," http://www.movable-type.co.uk/scripts/latlong.html, accessed: 2019-02-26.