Document Version
Final published version

A Vehicle System for Navigating Among Vulnerable Road Users Including Remote Operation

O. de Groot, A. Bertipaglia, H. Boekema, V. Jain, M. Kegl, V. Kotian, T. Lentsch, Y. Lin,
C. Messiou, E. Schippers, F. Tajdari, S. Wang, Z. Xia, M. Zaffar, R. Ensing, M. Garzon, J. Alonso-Mora,
H. Caesar, L. Ferranti, R. Happee, J.F.P. Kooij, G. Papaioannou, B. Shyrokau, and D.M. Gavrila
Department of Cognitive Robotics, Delft University of Technology

Abstract—We present a vehicle system capable of navigating safely and efficiently around Vulnerable Road Users (VRUs), such as pedestrians and cyclists. The system comprises key modules for environment perception, localization and mapping, motion planning, and control, integrated into a prototype vehicle. A key innovation is a motion planner based on Topology-driven Model Predictive Control (T-MPC). The guidance layer generates multiple trajectories in parallel, each representing a distinct strategy for obstacle avoidance or non-passing. The underlying trajectory optimization constrains the joint probability of collision with VRUs under generic uncertainties. To address extraordinary situations ("edge cases") that go beyond the autonomous capabilities — such as construction zones or encounters with emergency responders — the system includes an option for remote human operation, supported by visual and haptic guidance.

In simulation, our motion planner outperforms three baseline approaches in terms of safety and efficiency. We also demonstrate the full system in prototype vehicle tests on a closed track, both in autonomous and remotely operated modes.

Fig. 1: Safe and efficient vehicle navigation among VRUs.

I. INTRODUCTION

Automated driving has made steady progress in recent years. For instance, advanced highway autopilot systems now enable drivers to divert their attention and engage in side activities—until prompted to retake control (i.e., conditional automation). At the same time, driverless vehicles ("robotaxis") have begun offering mobility services within geo-fenced areas in several U.S. cities (high automation).

In this paper, we present a vehicle system whose key feature is the ability to safely and efficiently navigate around Vulnerable Road Users (VRUs), such as pedestrians and cyclists (see Fig. 1). VRUs present persistent challenges for automated driving due to their high maneuverability, unpredictable behavior, and occasional disregard for traffic rules. Our motion planner explicitly accounts for the uncertainty in predicted VRU behavior and evaluates multiple evasive maneuvers in parallel. We focus on this motion planner, without addressing broader mission-level planning. The Operational Design Domain (ODD) is limited to low-speed environments, such as parking lots and traffic-calmed urban areas, where multiple VRUs might interact with the self-driving vehicle.

There may be traffic situations in which autonomous operation would pose a safety risk or cause disruption to other road users. Examples include mismatches between the HD map and the real environment (e.g., due to new construction), the presence of emergency responders following an incident, or scenarios where a police officer is manually directing traffic. To handle such exceptional cases, we implement a mechanism for remote takeover, allowing a human operator to assume control of the vehicle when needed.

In the following sections, we describe related work, our prototype vehicle (an instrumented Toyota Prius), the core system modules—including environment perception, localization and mapping, motion planning, and control—and the overall system architecture.

II. RELATED WORK

There have been a number demonstrations on automated valet parking in academia and industry. For an overview, we refer the reader to [1]. Notable distinctions are in the use of a dedicated infrastructure [2, 3, 4], the use of maps that are generated offline [5, 6], as well as low-cost [7, 8] versus high [9, 10] cost sensors. Many of the previous works focus on the parking aspect and assume that the environment is free of dynamically moving VRUs [9, 10, 11].

When VRUs are present, the vehicle must take evasive action [12]. Trajectory optimization motion planning techniques [13] can consider VRUs as dynamic collision avoidance constraints while optimizing performance criteria (e.g., road-following). This was demonstrated in [14], which incorporated uni-modal uncertainty for VRU predictions to avoid a crossing pedestrian. Several approaches [15, 16] connect optimization-based planners with a high-level planner to concurrently
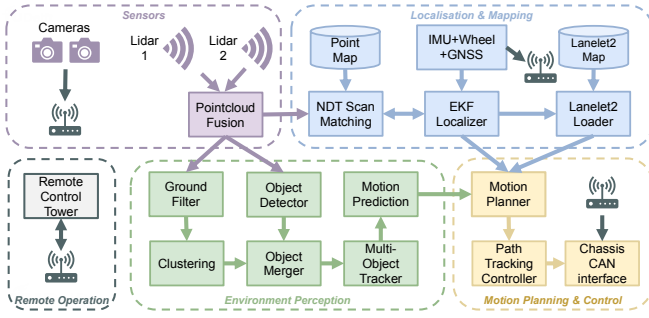
Fig. 2: Overview of the demonstrator's architecture. The dashed-colored regions indicate the main components. The remote transmission symbol indicates data being exchanged between the vehicle and the external Remote Control Tower.

optimize multiple driving maneuvers. They rely however on explicit lane structure and simplify the uncertainty associated with the predicted motion of other traffic participants. Recently [17] used the underlying topology of the collision-free space to plan multiple high-level trajectories without relying on road structure, building on top of existing trajectory optimization methods. Multi-modal uncertainties of multiple VRU motion predictions were considered in [18].

For safe vehicle remote driving, various challenges have to be addressed, such as the remote driver's situational awareness [19] and embodiment. A multitude of efforts have focused on overcoming this bottleneck, such as assisting drivers in locating vehicles [20], and recognising speed [21] and distance [22]. Recent research findings [23, 24] have shown that vibrational feedback can increase their situational awareness and embodiment of the remote operator.

The main contributions of this system paper are:

- We present a vehicle system for navigating among VRUs, which includes a novel motion planner based on [18, 17] to plan several distinct probabilistic safe vehicle trajectories in parallel. Our high-level planner does not rely on an explicit lane structure, which enables its application in more unstructured urban environments. The underlying trajectory optimization constrains the joint probability of collision with VRUs under generic uncertainties.
- We furthermore integrate remote operation as a fall-back to autonomous driving. A Remote Control Tower, placed on a motion platform, provides vibrational feedback to the remote operator, in addition to video and audio, enhancing situational awareness [23, 24].

III. Vehicle System

The high-level system architecture is shown in Fig. 2. This section discusses the vehicle's main components: hardware and sensors (Section III-A), perception (Section III-B), localization and mapping (Section III-C), motion planning and control (Sections III-D and III-E). Section III-F presents the remote operation.

A. Prototype Vehicle

Our vehicle prototype is a Toyota Prius Hybrid (cf. Fig. 1). It features a custom drive-by-wire interface *"Movebox"* that allows lateral and longitudinal vehicle control. On-board processing uses a PC with an AMD EPYC 7313 16-core processor, 256 GB RAM, and two NVIDIA RTX A6000 GPUs.

The vehicle is equipped with two 128-beam rotating Li-DARs, a Robosense Ruby and an Ouster OS1-Rev 7 both on the vehicle roof. The Robosense LiDAR, which has a larger detection range, is placed on the front at the roof, whereas the Ouster LiDAR, which has a larger vertical Field of View ($45°$), is placed at an additionally elevated position, more centrally on the roof. It also has eight Surrounding Lucid Triton Cameras and an OXTS3000 Dual antenna GNSS/INS unit.

The prototype vehicle's software stack is built on the ROS 2 Autoware[1] framework. We use official open-source ROS 2 drivers for the LiDARs[2][3], cameras[4] and GNSS/INS[5]. For the *"Movebox"* interface, we developed our own ROS 2 driver, which translates Autoware control commands (ROS 2 messages) to CAN-BCM messages and sends them to the Electronic Control Unit (ECU) of the vehicle. It also reads the state of the car (*i.e.* Control Mode, Gear, Steering angle, etc.) from the ECU and translates these signals back to high-level ROS 2 messages. For the motion planner (see Subsection III-D) and vehicle controller (see Subsection III-E) new ROS 2 packages were developed, which will be publicly released[6].

B. Perception

We employ a modular LiDAR perception pipeline — based on Autoware [25] — to detect and track objects and predict their future trajectories. Specifically, we perform ego-motion compensation on the LiDAR point cloud that covers the $360°$ surroundings of the vehicle. To detect objects from a set of predefined classes, as well as generic objects, we use two detection components. A deep learning-based detector detects objects from a set of predefined classes, e.g. car, bike, and pedestrian. We use a traditional LiDAR clustering pipeline to segment generic objects that are not part of the above set of classes. These object candidates are merged, tracked across time, and a motion prediction component predicts the future path of each tracked object.

Object Detection. We employ the learnable 3D object detector CenterPoint [26] due to its strong performance and real-time capabilities. CenterPoint is trained on the nuScenes [27] dataset. An example of the resulting detections is shown in Fig. 3. In future work, auto-labeling methods such as UNION [28] may be used to retrain the detector on newly collected data, eliminating the need for manual annotation.

---

[1]https://github.com/autowarefoundation/autoware

[2]https://github.com/ouster-lidar/ouster-ros

[3]https://github.com/RoboSense-LiDAR/rslidar_sdk

[4]https://github.com/lucidvisionlabs/arena_camera_ros2

[5]https://github.com/OxfordTechnicalSolutions/oxts_ros2_driver

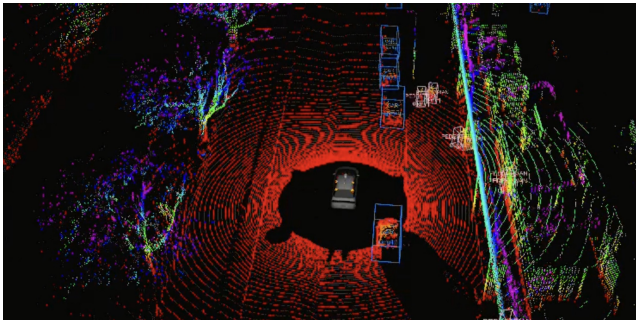[6]The planner is available at: https://github.com/tud-amr/mpc_planner

Fig. 3: Visualization of the object detections. The detector detects a predefined set of object classes (e.g. car and pedestrian).

Fig. 5: The point cloud map used for map-based localization in our demo. A point's hue indicates its height in the z-direction.
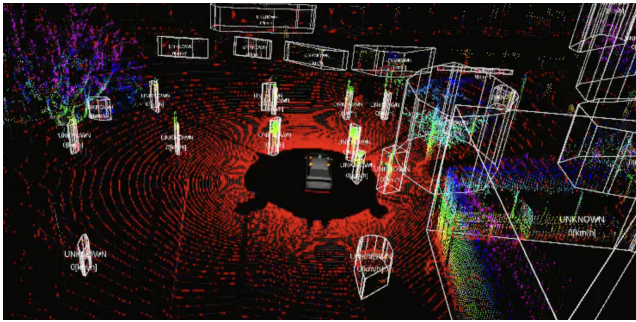

Fig. 4: LiDAR clustering visualized. It segments any object protruding from the ground plane, including generic objects.

LiDAR clustering. Given a LiDAR point cloud, we remove the ground plane [29] and group the remaining points into several class-agnostic clusters [30]. Then, we fit a bounding box for each cluster using [31]. We tune the clustering hyperparameters qualitatively on a small set of self-collected data. Fig. 4 shows the clustering results for a scene.

Object Merging. To avoid duplicate detections from the object detector and the LiDAR clustering step, we merge their outputs, which can be regarded as a minimum-cost flow problem (MCFP). We solve the MCFP using the successive shortest path algorithm [32]. Matched objects are assigned labels derived from the CenterPoint detection pipeline. Detections that do not have a corresponding match are labeled as "unknown objects".

Multi-Object Tracking. We apply a multi-object tracker on the merged objects to smooth tracks, infer the track identities, and estimate the velocity of each object. The data association module performs maximum score matching using muSSP [33] as a solver, to associate objects from neighboring frames. We build a separate Extended Kalman Filter (EKF) tracker for each learned object class, as well as the unknown object class.

Motion Prediction. We use the EKF proposed in the previous step and propagate the uncertainty 4 seconds into the future. These unimodal distributions represent the future locations of all actors in the scene. The predicted distributions are fed to the motion planner (see Subsection III-D).
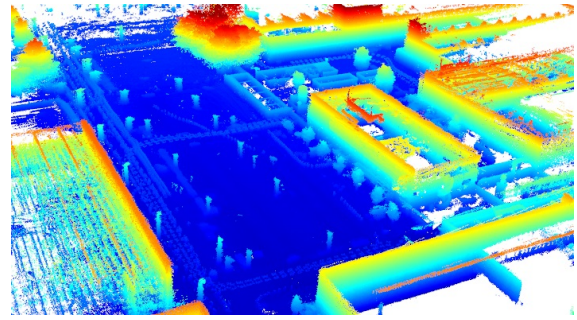
C. Localization and Mapping

GNSS localization alone is not always reliable when driving in areas surrounded by buildings as these obstruct the satellite reception and produce multipath reflections that yield inaccurate location estimates. Our Autoware-based localization stack is therefore configured to also integrate the vehicle's LiDAR scans, wheel odometry and IMU to determine its position with respect to a geo-localized 3D reference map.

Offline Map Creation. The map is created offline prior to driving. First, a 3D point cloud map of the environment is built with the open-source ROS 2 LiDAR SLAM package[7], as shown in Fig. 5. Afterwards, the 3D map is manually annotated with semantic information using Autoware's online *Vector Map Builder* tool in the Lanelet2 [34] format. The annotations consist of polylines indicating the driving lanes, sidewalks, and parking areas required for planning the vehicle's trajectory.

Online Localization. To produce a smooth trajectory for downstream motion planning, an EKF temporally integrates the relevant sensor measurements. At each timestamp, the EKF first predicts the current vehicle state based on a predefined kinematics model and previous state estimate, and then updates the predicted state using the vehicle's measured *pose* and *twist* (i.e., linear and angular velocities). To measure pose, the vehicle's most recent LiDAR scan is compared to the fixed 3D map using NDT LiDAR scan matching [35] and the pose estimate is adjusted iteratively to maximize the matching score. The twist is measured using Eagleye [36], which combines GNSS, wheel odometry, and IMU measurements. We ensure that the GNSS has a clear reception during system start-up, and use the GNSS/INS-based pose estimate to initialize the EKF.

D. Motion Planning

The motion planner processes the information provided by localization and perception systems and outputs a trajectory to be tracked by the vehicle controller. The objective is to achieve safe, comfortable, and time-efficient automated driving in a complex urban environment while engaging with VRUs such as pedestrians and cyclists.

Fig. 6 shows the structure of the motion planner that is based on Topology-driven Model Predictive Control (T-MPC) [17].

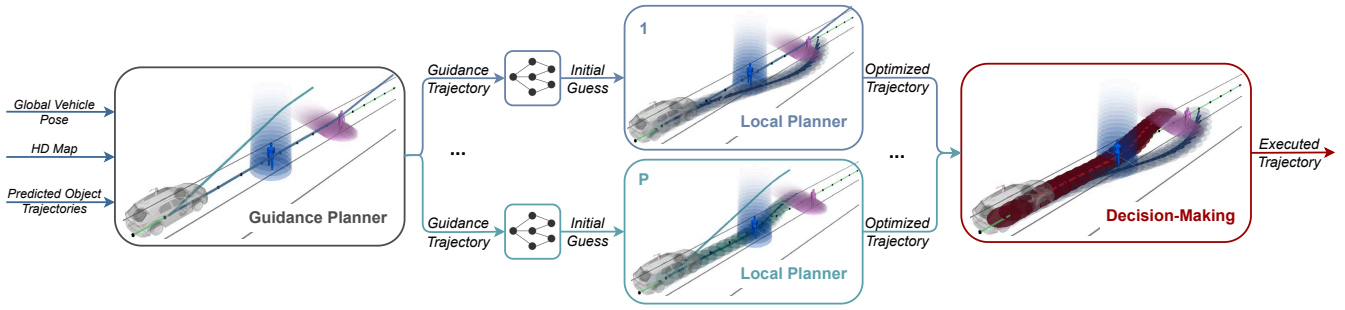[7]https://github.com/rsasaki0109/lidarslam_ros2

2484

Fig. 6: An overview of the motion planner (Sec. III-D). The **Guidance Planner** plans several guidance trajectories that each pass the dynamic obstacles differently. Each guidance trajectory initializes a **Local Planner** that, considering the uncertainty associated with the predicted motion of VRUs, locally optimizes the trajectory. The **Decision-Making** component selects one of the optimized trajectories as output of the motion planner.

The guidance planner computes several trajectories that each pass obstacles in distinct ways. It applies a Visibility Probabilistic Roadmap (PRM) [37] planner in x, y-position and time, filtering out trajectories that pass obstacles in the same way. Existing trajectories are tracked by propagating the planning graph over consecutive iterations. The piecewise-linear trajectories are smoothed by fitting cubic splines yielding several guidance trajectories (for more details, see [17]). In this work, we strengthen the filter in the visibility-PRM by differentiating non-passing trajectories in addition to left-right passing trajectories. We compute the difference between left, right and non-passing using winding numbers [38]. The absolute value of the winding number for each agent indicates the ego-vehicle's passing progress, that we use to detect passing behaviors. The sign of the winding number indicates passing direction.

T-MPC optimizes the guidance trajectories by several local planners in parallel, ensuring that it is dynamically feasible and collision-free. A limitation of prior work [17] is that optimization of new guidance trajectories may be computationally demanding, leading to less feasible trajectories in practice. In this work, we learn a mapping from guidance trajectory to optimized trajectory. The network follows an encoder-decoder structure with dynamics incorporated in the decoder. The input of this network considers:

- Discrete-time positions of one of the guidance trajectories over the time horizon,
- Initial ego-vehicle state,
- Predicted obstacle positions over the horizon each encoded with the same Long Short-Term Memory (LSTM)
- Reference path defined by x and y components, each as a sequence of cubic splines.

The decoder outputs jerk and steering rate control signals over the time horizon that are Euler integrated to retrieve trajectories. The loss of the network penalizes the deviation of state and input trajectories from the optimized ground truth obtained by running a local planner. The mapping is trained in simulation, repeatedly collecting the above inputs together with the local planner outputs over time horizon N.

In a simulation example, the learned model, trained with simulation data, reduced the infeasibility of guided trajectories

from 26% to 10%, greatly improving the performance of the planner. T-MPC passes the lowest-cost feasible trajectory to the vehicle control module.

In the local planners, we incorporate Gaussian uncertainty associated with the predicted motion of obstacles by using Chance Constrained MPC (CC-MPC) [39]. CC-MPC bounds the probability of collision with each obstacle at each time step by a risk ϵ. It solves the following trajectory optimization, online:

$$\min_{\boldsymbol{u} \in \mathbb{U}, \boldsymbol{x} \in \mathbb{X}} \quad \sum_{k=0}^{N} J(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{1a}$$

$$\text{s.t.} \quad \boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}} \tag{1b}$$

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \ k = 0, \dots, N-1 \tag{1c}$$

$$g(\boldsymbol{x}_k, \boldsymbol{o}_k^j) \leq 0, \ \forall k, j. \tag{1d}$$

The objective J includes contouring and lag costs J_c and J_l, respectively, that track the reference path [13] (constructed from lanelet information), a path preview cost J_p [40] to align the vehicle with the reference path at the end of the horizon, a term $||v - v_{\text{ref}}||_2^2$ to track a reference velocity, and penalties $||a||_2^2$ and $||\omega||_2^2$ on the commanded input and steering rate, respectively. Constraints (1b) and (1c) constrain the initial state and vehicle dynamics. Probabilistic collision avoidance with obstacles is specified through (1d).

We solve the optimization problem in (1a)-(1d) using the Sequential Quadratic Programming (SQP) solver provided by FORCES Pro[41], replanning the vehicle trajectory at $10\,\text{Hz}$. Each planning cycle operates over a $7\,\text{s}$ time horizon, discretized into $N = 35$ steps with $0.2\,\text{s}$ intervals. The vehicle dynamics follow a kinematic bicycle model.

E. Control

The vehicle controller computes acceleration and steering commands from the motion planner's optimised path to ensure safe trajectory tracking. A Model Predictive Contouring Control (MPCC) is proposed to consider the coupled longitudinal and lateral dynamics [42]. Its prediction model relies on a nonlinear single-track vehicle with a Fiala tyre model, parametrised with experimental data to capture also the

nonlinear behaviour. The MPCC computes optimal steering and longitudinal forces to minimise tracking and velocity errors while enforcing stability. The cost function is defined as follows:

$$J_{pf} = \sum_{i=1}^{N} \left(q_{e_{\mathrm{Con}}} e_{\mathrm{Con},i}^2 + q_{e_{\mathrm{Lag}}} e_{\mathrm{Lag},i}^2 + q_{e_{\mathrm{Vel}}} \left(e_{\mathrm{Vel}}\right)^2 \right.$$
$$\left. + q_{\dot{\delta}} \dot{\delta}_i^{\,2} + q_{\dot{F}_x} \dot{F}_{x_i}^{\,2} \right), \quad (2)$$

where N is the length of the prediction horizon, the contouring error e_{Con}, the lag error e_{Lag}, the difference between the vehicle's velocity and the desired one e_{Vel}, the steering rate $\dot{\delta}$, the rate of longitudinal force $\dot{F}_x$, and parameters q_* are the weights of the respective quadratic errors. The weights are fine-tuned to optimize controller performance by minimizing longitudinal velocity error, and trajectory offset [43], The cost function is constrained based on actuator limits, and to take into account diverse weather conditions, the total tire force at each axle is constrained within the tire friction circle to ensure vehicle stability.

The steering angle is sent directly to the logic of the column-assist electric motor (function of lane-keeping assist) as a reference via CAN. For acceleration control, the low-level controller consists of a feedforward component considering powertrain steady-state characteristics and braking dynamics and a feedback component to compensate for the difference between the commanded and actual accelerations.

F. Remote Operation

The remote operation takes over vehicle control in edge cases, where the vehicle cannot correctly match the perceived information with the known datasets (e.g., if the epistemic uncertainty of the perception component becomes too large [44]).

To that end, a Remote Control Tower (RCT) is connected with the prototype vehicle, and receives information provided by the experimental vehicle (Fig. 2). The RCT is comprised of several components: a motion platform, three high-resolution monitors (curved 32 inch, 60Hz), a steering wheel, pedals, and a racing seat. Each of these elements contributes to creating an immersive and realistic driving experience for the remote operators. The motion platform is a Gforcefactory EDGE 6D motion simulator[8] (Fig. 7), running with a UDP protocol at $200\,\mathrm{Hz}$. Three low-latency AOC monitors ($1\,\mathrm{ms}$) are used to provide video feedback. For controlling the steering and throttle of the vehicle, we have used the Logitech G920 steering wheel and pedals[9]. For a more realistic steering feel, force feedback is implemented in the form of a spring and auto centering. The steering wheel has a small motor which can generate different types of feedback. This motor can be activated using ff_{effect} from the force feedback framework for Linux [10].
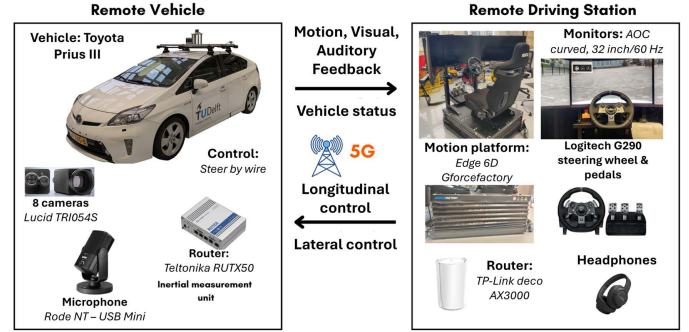

Fig. 7: Remote Control Tower (RCT) with a Gforcefactory Edge 6D motion simulator equipped with three monitors.

Within the RCT, auditory feedback is provided to the operator via noise-canceling headphones. The headphones receive the sound from the car captured by a RODE NT-USB Mini microphone [11] which is located directly next to the driver. In the experimental vehicle, multiple cameras are used to collect high-resolution real-world video. To further enhance the situational awareness of the remote driver, the motion platform also provides the operator with vibrational feedback from the experimental vehicle [23, 24].

The RCT and the vehicle maintain connectivity through a 5G network, using Hyperpath for low-latency communication [12]. Hyperpath creates a multi-connectivity peer-to-peer mesh VPN (Virtual Private Network) between the vehicle and the RCT. The protocol used is the User Datagram Protocol (UDP). For real-time streaming of camera footage, WebRTC is used [45]. In this way, all required signals are transmitted i.e., actuation inputs (steering wheel and throttle input) and feedback (video, audio, and vibrational).

IV. EXPERIMENTS

To evaluate overall driving performance, we compare our proposed motion planner in simulation to three baselines:

- **Autoware Planner**[13] [25]: The base planner included in Autoware. Its default behavior brakes for any nearby obstacle and is not competitive with other methods. To make the planner more competitive, we enabled dynamic collision avoidance and tuned the planner (lower safety margins, prefer overtaking over braking) to avoid the pedestrians as much as possible in our simulation environment.
- **LMPCC** [14]: A non parallelized MPC implementation.
- **T-MPC++ (w/o fallback)** [17]: Parallelized MPC without fallback trajectory classes and learned initial guesses.

All optimization-based methods have identical objectives and collision avoidance constraints. We compare these methods

---

[8]https://www.gforcefactory.com/edge-6d

[9]https://www.logitechg.com/en-us/products/driving/driving-force-racing-wheel.html

[10]https://www.kernel.org/doc/Documentation/input/ff.txt

[11]https://rode.com/de/microphones/usb/nt-usb-mini

[12]Hyperpath - https://www.hyperpath.ie/
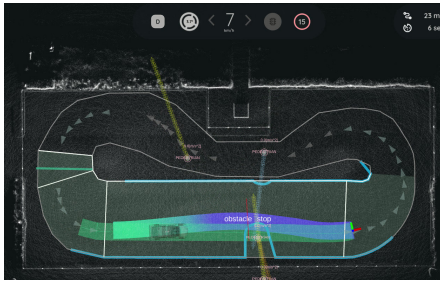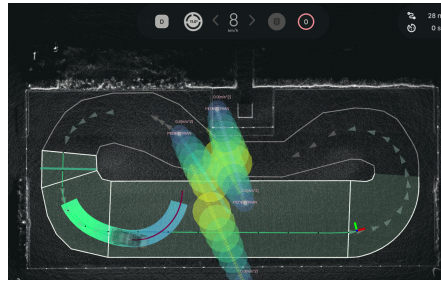
[13]see https://autowarefoundation.github.io/autoware-documentation/main/design/autoware-architecture/planning/
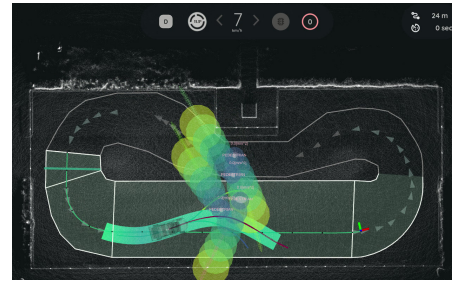
(a) The Autoware planner brakes for the obstacles.

(b) LMPCC finds a poor local optimal trajectory that brakes for the obstacles.

(c) T-MPC++ steers to evade the obstacles efficiently.

Fig. 8: Screenshot of the simulations where the vehicle is following annotated lanelets (green solid line) in the vicinity of pedestrians (marked pink). Pedestrian predictions (sequence of circles) are visualized including level sets of the associated uncertainty in (b, c). Vehicle planned trajectory visualized in blue (slow) to green (fast). Alternative topological trajectories in (c) are shown with solid colored lines.

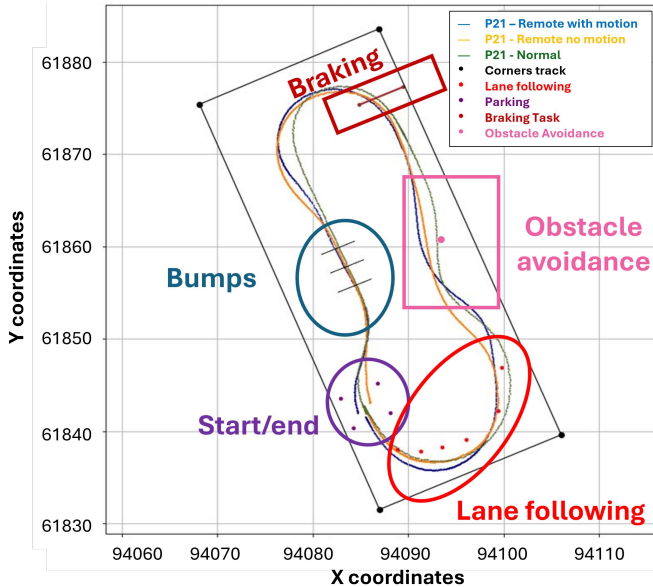Fig. 9: A snapshot of the vehicle tests.

Fig. 10: GNSS data of one participant, lap 4 of all conditions, including locations of tasks.

to our proposed method in scenarios with $0, 2$ and 4 pedestrians to study how well these methods cope with increasingly crowded dynamic environments. Quantitative results are shown in Table I and screenshots are shown in Fig. 8.

The Autoware baseline is the slowest planner and often collides. LMPCC (i.e., single trajectory MPC) collides less and drives faster on average. It is outperformed however by both versions of T-MPC++, in particular by comparing the task durations with an increasing number of obstacles. With 4 obstacles, on average LMPCC takes 7.3s longer, while T-MPC++ takes only 4.1s longer (i.e., T-MPC++ is more than 50% faster in dealing with 4 obstacles in this environment). T-MPC++ therefore passes the obstacles more efficiently.

We deployed our motion planner on our prototype vehicle to navigate around a static obstacles and a moving dummy cyclist on our (closed) test track. The vehicle's task is to follow encoded lanelets, while avoiding the obstacles. Fig. 9 shows snapshot from one of the many runs. Our vehicle was able to safely and efficiently avoid the obstacles.

To evaluate the remote driving performance, we considered three conditions. Participants control the vehicle: (a) from the vehicle driver's position; (b) from the RCT receiving motion feedback (c) from the RCT without any motion feedback, relying solely on visual and auditory feedback. The participants drove five laps around the test track for each condition. The paths driven by one participant are illustrated in Fig. 10. The values measured for the velocity, steering position, longitudinal and lateral acceleration were significantly higher (p < 0.004) in both conditions of remote driving (motion/no motion

TABLE I: Comparison of navigation performance on a scenario with a varying number of randomized crossing pedestrians over 25 experiments. Reported is the task duration, number of collisions, time-outs (vehicle did not reach the goal in time) and average velocity.

#	Method	Dur. [s]	Collisions	Time-outs	Avg. Vel. [m/s]
0	Autoware	19.5 (0.1)	**0**	**0**	**1.87**
	T-MPC++	**19.4** (0.3)	**0**	**0**	1.86
2	Autoware	24.0 (4.2)	1	**0**	1.54
	LMPCC	22.4 (3.3)	**0**	**0**	1.65
	T-MPC++ (w/o fallback)	**21.5** (1.6)	**0**	**0**	**1.70**
	T-MPC++	21.6 (1.8)	**0**	**0**	**1.70**
4	Autoware	27.9 (9.7)	2	3	1.22
	LMPCC	26.7 (6.6)	1	2	1.43
	T-MPC++ (w/o fallback)	24.0 (3.3)	**0**	**0**	1.57
	T-MPC++	**23.5** (3.8)	**0**	**0**	**1.61**

feedback) compared with in-vehicle driving. No significant difference was identified between the remote driving conditions (motion/no motion feedback) for any of the metrics stated above. Regarding the driving experience, all items included in our questionnaire were significantly higher ($p<0.03$) for in-vehicle driving compared to both remote driving conditions. The findings are further detailed in [46], and suggest that motion feedback, in its current form, may not be necessary for low-velocity scenarios. Thus a simplified RCT without motion feedback could suffice for safe driving in these scenarios. Additionally, the results show that there is still a large gap between remote driving and in-vehicle driving both in the overall driving performance and experience, but also in the successful completion of the specific tasks.

V. Conclusions

We presented a vehicle system that is able to navigate safely and efficiently around Vulnerable Road Users (VRUs). One distinguishing aspect is a novel motion planner, based on Topology-driven Model Predictive Control (T-MPC). The guidance planner computes several trajectories in parallel that each pass obstacles in distinct way, or are non- passing. The underlying trajectory optimization constrains the joint probability of collision with VRUs under generic uncertainties. To account for "edge" cases that the vehicle might not be able to handle autonomously, the system included remote operation.

In simulation, we showed that our motion planner outperforms three baselines in terms of safety and efficiency. Furthermore, we described tests with the prototype vehicle on a test track in self-driving mode and with remote operation. Future work includes more sophisticated trajectory prediction, a complete mission planner, improved situational awareness of the remote operator at higher speeds, and a self-assessment method for switching from autonomous to remote driving.

Acknowledgments

References

[1] H. Banzhaf, D. Nienhüser, S. Knoop, and J. Zöllner. The future of parking: A survey on automated valet parking with an outlook on high density parking. In *Intelligent Vehicles Symposium (IV)*, pages 1827–1834, 2017.

[2] K. Min and J. Choi. Design and implementation of autonomous vehicle valet parking system. In *International Conference on Intelligent Transportation Systems (ITSC)*, pages 2082–2087, 2013.

[3] A. Ibisch, S. Stümper, H. Altinger, M. Neuhausen, et al. Towards autonomous driving in a parking garage: Vehicle localization and tracking using environment-embedded LiDAR sensors. In *Intelligent Vehicles Symposium (IV)*, pages 829–834, 2013.

[4] C. Löper, C. Brunken, G. Thomaidis, S. Lapoehn, et al. Automated valet parking as part of an integrated travel assistance. In *International Conference on Intelligent Transportation Systems (ITSC)*, pages 2341–2348, 2013.

[5] P. Jeevan, F. Harchut, B. Mueller-Bessler, and B. Huhnke. Realizing autonomous valet parking with automotive grade sensors. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 3824–3829, 2010.

[6] G. Stanek, D. Langer, B. Müller-Bessler, and B. Huhnke. Junior 3: A test platform for advanced driver assistance systems. In *Intelligent Vehicles Symposium (IV)*, pages 143–149, 2010.

[7] P. Furgale, U. Schwesinger, M. Rufli, W. Derendarz, et al. Toward automated driving in cities using close-to-market sensors: An overview of the V-charge project. In *Intelligent Vehicles Symposium (IV)*, pages 809–816, 2013.

[8] U. Schwesinger, M. Bürki, J. Timpner, S. Rottmann, et al. Automated valet parking and charging for e-mobility. In *Intelligent Vehicles Symposium (IV)*, pages 157–164, 2016.

[9] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of field Robotics (JFR)*, 25(8):425–466, 2008.

[10] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics (JFR)*, 25(9):569–597, 2008.

[11] R. Kummerle, D. Hahnel, D. Dolgov, S. Thrun, et al. Autonomous driving in a multi-level parking structure. In *International Conference on Robotics (ICRA)*, pages 3395–3400, 2009.

[12] T. Dang, J. Desens, U. Franke, D. Gavrila, et al. Steering and evasion assist. *Handbook of Intelligent Vehicles*, pages 759–782, 2012.

[13] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, et al. Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model. *Transactions on Intelligent Transportation Systems (T-ITS)*, 19(9):2994–3008, 2017.

[14] L. Ferranti, B. Brito, E. Pool, Y. Zheng, et al. SafeVRU: A research platform for the interaction of self-driving vehicles with vulnerable road users. In *Intelligent Vehicles Symposium (IV)*, pages 1660–1666, 2019.

[15] F. Eiras, M. Hawasly, S. Albrecht, and S. Ramamoorthy. A two-stage optimization-based motion planner for safe urban driving. *Transactions on Robotics (T-RO)*, 38(2):822–834, 2021.

[16] W. Ding, L. Zhang, J. Chen, and S. Shen. EPSILON: An efficient planning system for automated vehicles in highly interactive environments. *Transactions on Robotics (T-RO)*, 38(2):1118–1138, 2021.

[17] O. de Groot, L. Ferranti, D. Gavrila, and J. Alonso-Mora. Topology-driven parallel trajectory optimization

in dynamic environments. *Transactions on Robotics (T-RO)*, 2024.

[18] O. de Groot, L. Ferranti, D. Gavrila, and J. Alonso-Mora. Scenario-based motion planning with bounded probability of collision. *arXiv preprint*, 2023.

[19] L. Zhao, M. Nybacka, and M. Rothhämel. A survey of teleoperation: Driving feedback. In *Intelligent Vehicles Symposium (IV)*, 2023.

[20] S. Vozar and D. Tilbury. Improving UGV teleoperation performance using novel visualization techniques and manual interfaces. In *Unmanned Systems Technology XIV*, volume 8387, pages 448–455, 2012.

[21] T. Tang, J. Kurkowski, and M. Lienkamp. Teleoperated road vehicles: A novel study on the effect of blur on speed perception. *International Journal of Advanced Robotic Systems (IJARS)*, 10(9):333, 2013.

[22] D. Tyczka, R. Wright, B. Janiszewski, M. Chatten, et al. Study of high-definition and stereoscopic head-aimed vision for improved teleoperation of an unmanned ground vehicle. In *Unmanned Systems Technology XIV*, volume 8387, pages 226–243, 2012.

[23] G. Papaioannou, L. Zhao, M. Nybacka, J. Jerrelind, et al. Motion comfort and driver feel: An explorative study about their relation in remote driving. *arXiv preprint*, 2023.

[24] L. Zhao, M. Nybacka, M. Rothhämel, A. Habibovic, et al. Driving experience and behavior change in remote driving-an explorative experimental study. *Transactions on Intelligent Vehicles (T-IV)*, 2023.

[25] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, et al. Autoware on board: Enabling autonomous vehicles with embedded systems. In *ICCPS*, pages 287–296, 2018.

[26] T. Yin, X. Zhou, and P. Krahenbuhl. Center-based 3D object detection and tracking. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11784–11793, 2021.

[27] H. Caesar, V. Bankiti, A. Lang, S. Vora, et al. nuScenes: A multimodal dataset for autonomous driving. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11621–11631, 2020.

[28] T. Lentsch, H. Caesar, and D.M. Gavrila. UNION: Unsupervised 3D object detection using object appearance-based pseudo-classes. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 22028–22046, 2024.

[29] Z. Shen, H. Liang, L. Lin, Z. Wang, et al. Fast ground segmentation for 3D LiDAR point cloud based on jump-convolution-process. *Remote Sensing*, 13(16):3239, 2021.

[30] R. Rusu. Semantic 3D object maps for everyday manipulation in human living environments. *KI-Künstliche Intelligenz*, 24:345–348, 2010.

[31] X. Zhang, W. Xu, C. Dong, and J. Dolan. Efficient L-shape fitting for vehicle detection using laser scanners. In *Intelligent Vehicles Symposium (IV)*, pages 54–59, 2017.

[32] J. Orlin. A polynomial time primal network simplex algorithm for minimum cost flows. *Mathematical Programming*, 78:109–129, 1997.

[33] C. Wang, Y. Wang, Y. Wang, C. Wu, et al. muSSP: Efficient min-cost flow algorithm for multi-object tracking. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[34] F. Poggenhans, J. Pauls, J. Janosovits, S. Orf, et al. Lanelet2: A high-definition map framework for the future of automated driving. In *International Conference on Intelligent Transportation Systems (ITSC)*, pages 1672–1679, 2018.

[35] P. Biber and W. Straßer. The normal distributions transform: A new approach to laser scan matching. In *International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2743–2748, 2003.

[36] A. Takanose, Y. Kitsukawa, J. Megruo, E. Takeuchi, et al. Eagleye: A lane-level localization using low-cost GNSS/IMU. In *Intelligent Vehicles Symposium Workshops (IV Workshops)*, pages 319–326, 2021.

[37] T. Siméon, J.-P. Laumond, and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics*, 14(6):477–493, January 2000.

[38] M. Berger. Topological Invariants in Braid Theory. *Letters in Mathematical Physics*, 55(3):181–192, March 2001.

[39] H. Zhu and J. Alonso-Mora. Chance-Constrained Collision Avoidance for MAVs in Dynamic Environments. *Robotics and Autom. Letters (RA-L)*, 4(2):776–783, 2019.

[40] G. Chen, J. Yao, H. Hu, Z. Gao, et al. Design and experimental evaluation of an efficient MPC-based lateral motion controller considering path preview for autonomous vehicles. *Control Engineering Practice*, 123:105164, 2022.

[41] A. Domahidi and J. Jerez. *FORCES Professional*. Embotech AG, July 2014.

[42] A. Bertipaglia, M. Alirezaei, R. Happee, and B. Shyrokau. Model predictive contouring control for vehicle obstacle avoidance at the limit of handling. In *International Symposium on Dynamics of Vehicles on Roads and Tracks (IAVSD)*, pages 132–142, 2023.

[43] A. Bertipaglia, B. Shyrokau, M. Alirezaei, and R. Happee. A two-stage Bayesian optimisation for automatic tuning of an unscented Kalman filter for vehicle sideslip angle estimation. In *Intelligent Vehicles Symposium (IV)*, 2022.

[44] M. Itkina and M. Kochenderfer. Interpretable self-aware neural networks for robust trajectory prediction. In *Conf. on Robot Learning (CoRL)*, pages 606–617, 2023.

[45] RobotWebTools. GitHub - RobotWebTools/webrtc_ros: Streaming of ROS Image Topics using WebRTC, 2020.

[46] E. Schippers, A. Schrank, V. Kotian, C. Messiou, et al. A motion for no motion: The redundancy of motion feedback in low-velocity remote driving of a real vehicle. *Available at SSRN 5065264*, 2024.