SafeVRU: A Research Platform for the Interaction of Self-Driving Vehicles with Vulnerable Road Users

L. Ferranti*†, B. Brito†‡, E. Pool*†, Y. Zheng*, R. M. Ensing*, R. Happee*, B. Shyrokau*, J. F. P. Kooij*, J. Alonso-Mora‡, and D. M. Gavrila*

***Abstract*— This paper presents our research platform SafeVRU for the interaction of self-driving vehicles with Vulnerable Road Users (VRUs, i.e., pedestrians and cyclists). The paper details the design (implemented with a modular structure within ROS) of the full stack of vehicle localization, environment perception, motion planning, and control, with emphasis on the environment perception and planning modules. The environment perception detects the VRUs using a stereo camera and predicts their paths with Dynamic Bayesian Networks (DBNs), which can account for switching dynamics. The motion planner is based on model predictive contouring control (MPCC) and takes into account vehicle dynamics, control objectives (e.g., desired speed), and perceived environment (i.e., the predicted VRU paths with behavioral uncertainties) over a certain time horizon. We present simulation and real-world results to illustrate the ability of our vehicle to plan and execute collision-free trajectories in the presence of VRUs.**

Fig. 1: The SafeVRU platform for interaction with VRUs on-board our vehicle demonstrator.

I. INTRODUCTION

Every year between 20 and 50 million people are involved in road accidents, mostly caused by human errors [1]. According to [1], approximately 1.3 million people lost their life in these accidents. Half of the victims are vulnerable road users (VRUs), such as pedestrians and cyclists. Self-driving vehicles can help reduce these fatalities [2].

Active safety features, such as autonomous emergency braking (AEB), are increasingly found on-board vehicles on the market to improve VRUs' safety (see [3] for a recent overview). In addition, some vehicles already automate steering functionality (e.g., [4], [5]), but still require the driver to initiate the maneuver.

Major challenges must be addressed to ensure safety and performance while driving in complex urban environments [6], where VRUs are present. The self-driving vehicle should be aware of the presence of the VRUs and be able to infer their intentions to plan its path accordingly to avoid collisions. In this respect, motion planning methods are required to provide safe (collision-free) and system-compliant performance in complex environments with static and moving obstacles (refer to [7], [8] for an overview).

In real-world applications, the information on the pose (i.e., position and orientation) of other traffic participants comes from a perception module. The perception module should provide to the planner information not only concerning the current position of the other road users, but also their predicted paths (e.g., [9]–[11]). The planner should then account for possible inaccuracy from the perception module to improve the safety of the VRUs.

This paper presents our research platform SafeVRU (Safe interaction with vulnerable road users), that is, a self-driving vehicle (the converted Toyota Prius depicted in Figure 1) able to plan collision-free trajectories in the presence of VRUs. Our platform relies on the interactions between the perception and planning modules, as detailed below. SafeVRU relies on a perception module able to detect and estimate the paths of the VRUs over a prediction horizon, using a Dynamic Bayesian Network (DBN). Then, SafeVRU exploits these paths in the planning module. Our planner relies on model predictive contouring control (MPCC) [12], [13]. MPCC formulates the planning problem as a multi-objective constrained nonconvex optimization problem. Our MPCC plans a collision-free path for vehicle over a predefined time window. In this respect, our planner incorporates time-varying collision-avoidance constraints based on the predicted paths of the VRUs (provided by the perception module). For additional safety (e.g., to account for possible delays in the sensors and for comfort of the VRUs), the planner adds repulsive fields around the predicted paths of the VRUs. We tested the following scenarios:

S1: A cyclist riding along the direction of motion of the vehicle while approaching an intersection (simulation). The desired speed of the car in this scenario is 6 m/s and the road boundaries are symmetric.

S2: Two pedestrians crossing the road in front of the vehicle (simulation). The desired speed of the car in this scenario is 4 m/s and the road boundaries are asymmetric.

†The authors equally contributed to the paper.

*The authors are with the Intelligent Vehicles Group at Delft University of Technology, The Netherlands.

‡The authors are with the Autonomous Multi-Robots Lab at Delft University of Technology, The Netherlands.

S3: One pedestrian standing on the vehicle path (experiment with the real vehicle). The desired speed of the car is 3 m/s and the road boundaries are asymmetric to fit the size of the test track.

II. RELATED WORK

Recently, increasing attention has been dedicated to VRUs safety (e.g., [3], [9]–[11], [14], [15]). In [9], a joint team from Daimler and Karlsruhe Institute of Technology drove an autonomous car on the Bertha Benz Memorial Route, where they had to deal with VRUs. Their planner is divided into a behavior generation and a trajectory planning. The behavior generation decides how to interact with static and dynamic obstacles using a state machine. The trajectory planner computes the desired path (without taking into account the dynamics of the vehicle) and sends it to a path-follower low-level controller. When planning the trajectory decisions concerning the obstacles have already been made.

Commercial AEB systems are able to avoid collisions with detected VRUs as long as there is a sufficiently large distance between the vehicle and the VRU. In [3], the authors presented a pedestrian AEB analytical model to calculate the certainty of finding a detected pedestrian in the collision zone, by analysing the pedestrian lateral behavior. Their model can help verify existing AEB systems and design new AEB systems.

If the distance to perform an emergency brake is too small, evasive steering maneuvers are required. Research on evasive steering maneuvers for active pedestrian safety is extremely active. In [10], the authors provide a driver-assistant design to decide whether to brake or evade the crossing pedestrian based on the information provided by the perception module. A situation analysis module automatically evaluate the criticality of the current driving scenario. Then, a decision module decides whether to warn the driver or to trigger the appropriate maneuvers for collision avoidance and mitigation using dedicated controllers. In [11], the authors provide an overview of evasive steering techniques discussing the potential of evasive steering vs. braking. In addition, they also detail the design of the Daimler automatic evasion driver-assistance system for pedestrian protection. Similar to [10], their system also relies on a situation analysis module and a decision module that can take over control of the car to trigger an emergency maneuver. In [14], the authors propose an autonomous lane-keeping evasive maneuver that relies on the road infrastructure (cameras placed at specific hazardous locations). Their method can be used to take over control of the car to avoid collisions with a pedestrian when braking is no longer possible. In [15], the authors present a driver assistance system to help the driver initiate an evasive maneuver with pedestrians. The system is able to take decisions by taking into account upcoming traffic.

Our research platform SafeVRU also allows to avoid collisions with VRUs, but in fully automated mode. Our self-driving vehicle is able to adapt its trajectory based on the predicted paths of the VRUs derived from a DBN modeling approach that captures switching dynamics. The vehicles decides whether to brake or perform an evasive steering maneuver when a VRU is on the path of the car. Hence our system does not need a decision module that waits until the very last moment, as the driver might take control of the vehicle in an emergency after all. Furthermore, SafeVRU is able to handle situations with multiple VRUs (e.g., Scenario *S2*). Safety of the VRUs and smooth driving are achieved in our MPCC framework by solving a nonconvex multi-objective optimization problem. MPCC requires to solve this optimization problem online in real-time. Hence, given that the main modules of SafeVRU are implemented in ROS (Robot Operating System [16]), we integrated ACADO (Automatic Control and Dynamic Optimization Toolkit [17]) in our ROS framework to solve the MPCC problem online in real-time. In addition, we designed a dedicated UDP node to manage the communication from the "ROS PC" (where the perception module and the MPCC module run) to the low-level interface on dSpace Autobox (see Section III-E).

III. SYSTEM ARCHITECTURE

SafeVRU relies on the following modules: *(i)* a route planner, *(ii)* a localization module, *(iii)* a perception module, *(iv)* a local motion planner, and *(v)* a real-time PC. SafeVRU uses these modules to drive in the presence of VRUs. Figure 2 summarizes the overall structure of SafeVRU.

In the reminder of the section, we provide more details on the main components of our system architecture.

A. Route Planner

The route planner provides the global path $\mathbf{p}^{\text{path}}(\phi) \in \mathbb{R}^2$ to the local motion planner. Our current route planner consists in a set of waypoints selected by the user that connects the current position of the vehicle to the desired destination. These waypoints are then converted by the local motion planner into splines that the vehicle can follow. The route planner provides the desired velocity the vehicle should follow along the path (e.g., according to the rules of the road). The global path can contain static and moving obstacles. The local motion planner has the task of planning collision-free trajectories as detailed in Section III-D.

B. Localization

The localization module serves as input to the perception and to the motion planning modules. The perception module needs to correct for the motions of the vehicle to enable tracker-based intent recognition with respect to a world fixed coordinate system. The motion planning module requires the current pose and speed of the vehicle in order to plan the acceleration and steering angle commands.

A nonlinear state estimation through sensor fusion consisting of an unscented Kalman filter is used to estimate the vehicle state, described by the L2-dimensional state vector. The state vector consists of pose, velocity, and angular velocity. This filter is implemented using the ROS robot localization package [18]. For simplicity, the localization module works in 2D, projecting all off-plane values to the ground plane. The following odometry sources serve as

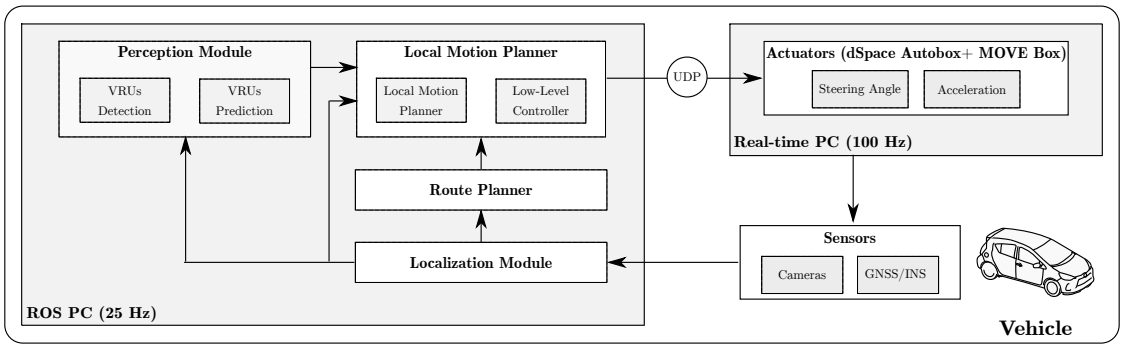


Fig. 2: Overview of our architecture.

input to the ROS localization module: Inertial Measurement Unit (IMU) and stereo odometry. The IMU is part of the Global Navigation Satellite System/Inertial Navigation System (GNSS/INS) Combination Advanced Navigation Spatial Dual and provides to our module only the orientation and angular velocity data. The GNSS/INS device is configured to measure the car heading using the velocity of the vehicle. Although a GNSS/INS system is used to calculate the heading of the vehicle, the position of the vehicle is only computed in a locally accurate coordinate system. The stereo odometry (pose) is calculated using Libviso2 [19]. The stereo camera setup mounted on our vehicle consists of two ueye camera's (model: UI-3060CP-C-HQ R2).

C. Perception Module

The perception module provides to the local motion planner a probabilistic prediction of the future location of the VRUs in the world-fixed coordinate frame (according to the localization module). Our VRU prediction module relies on the DBN described in [20].

a) Prediction: At time t, the goal is to create a distribution over the VRU position y_{t+n} for all $n \in [1 \dots N]$ time steps into the future. This is done with a DBN. Loosely speaking, a DBN is a Switching Linear Dynamical System (SLDS) consisting of two Linear Dynamical Systems (LDSs). The probability of switching between the two LDSs is governed by additional discrete context variables.

The linear switching models are defined as follows [20]:

$$x_t = A^{(M_t)} x_{t-1} + B\epsilon_t \qquad \epsilon_t \sim \mathcal{N}(0, Q) \tag{1a}$$

$$y_t = C x_t + \eta_t \qquad \eta_t \sim \mathcal{N}(0, R), \tag{1b}$$

where, x_t is the state of the model at time t, and $A^{(M_t)}$ indicates that at any time step the state propagation is done with the state matrix A of LDS model M. The probability model M_t is the current model is governed by four binary latent variables, as Figure 3 depicts. These latent variables are: Z_t^{STAT} and and Z_t^{ACT}/Z_t^{ACTED}. In particular, Z_t^{STAT} indicates the spatial context (e.g., is a cyclist at an intersection?), and Z_t^{ACT}/Z_t^{ACTED} indicates whether the VRU is acting/has acted, respectively (e.g., is a cyclist putting out an arm/has a cyclist put out an arm, indicating the desire to turn the upcoming intersection?).

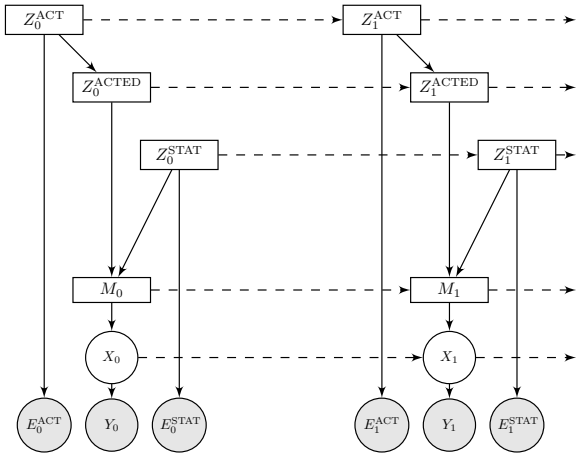


Fig. 3: The DBN with context cues shown for two consecutive time steps, adapted from [20]. The discrete, continuous and observed nodes are rectangular, circular and shaded, respectively. The binary context nodes represent the relation to the static environment Z_t^{STAT}, and object behavior (i.e. how VRU acts, Z_t^{ACT}, or has acted, Z_t^{ACTED}).

The latent variables Z_t^{STAT} and Z_t^{ACT} can be observed through E_t^{STAT} and E_t^{ACT} respectively. The relationship between the latent variables and the observations is modeled by a predefined probability distribution.

The resulting distribution over a future position y_{t+n} is a Mixture of k Gaussians (MoG) $\gamma \in \mathcal{I}^{\Gamma} := \{1, ..., n_{\text{MoG}}\}$ (n_{MoG} is the total number of Gaussians) consisting of a mean position x_k and a covariance matrix $\Sigma_k^{(\gamma)}$, one for each model, and can be computed by integrating over all latent variables, as detailed in [20].

Remark 1. For Scenario *S2* and the experiments (Section IV), the perception module uses a single LDS, which has a larger uncertainty region (leading to more conservative, but robust predictions), for safety reasons.

b) Detection: A Single Shot Detector (SSD) [21] trained on our Eurocities Persons dataset [22] detects the VRU in front of the vehicle. Using the stereo-camera setup and the absolute location of the ego vehicle (Section III-B), the location of the VRU is transformed to a temporally consistent reference frame. Consider for example the cyclist

in Scenario *S1*. In this reference frame, the position of the intersection is known, as well as his distance to the intersection. In addition, the probability of the left arm being raised is detected through a pipeline built on top of the SSD. A crop around the bounding box that is found by the SSD is fed to a ROS implementation of OpenPose [23] to retrieve the 2D skeleton of the cyclist. Finally, a Support Vector Machine computes the probability the cyclist has a raised arm, based on the keypoints from the 2D skeleton.

D. Local Motion Planner

The local motion planner exploits the information provided by the route planner, localization module, and perception module. Our local motion planner has two tasks: *(i)* compute a collision-free trajectory for the vehicle, *(ii)* directly control the car, by sending acceleration and steering commands to the vehicle through the real-time PC. To achieve these objectives we rely on a MPCC formulation for the following reasons. First, MPCC allows one to plan safe trajectories and compute control commands for the vehicle, that is, MPCC incorporates in one module a local motion planner and a path-following controller. Second, MPCC allows us to take into account the predicted paths of the VRUs provided by the perception module. Our local motion planner builds on the approach of [13] with some modifications as detailed below.

At time t, our planner solves the following model predictive contouring control problem:

$$\min_{\mathbf{x},\mathbf{u},\boldsymbol{\phi}} \sum_{k=0}^{N} J(\mathbf{x}(t+k),\mathbf{u}(t+k),\boldsymbol{\phi}(t+k)) \tag{2a}$$

$$\text{s.t.}: \mathbf{x}(t+k+1) = f(\mathbf{x}(t+k),\mathbf{u}(t+k)), \tag{2b}$$

$$\boldsymbol{\phi}(t+k+1) = \boldsymbol{\phi}(t+k) + v(t+k)\Delta t_k \tag{2c}$$

$$\mathbf{x}(t) = \mathbf{x}^{\text{init}}, \tag{2d}$$

$$G(\mathbf{x}(t+k),\mathbf{u}(t+k)) \leq \mathbf{g}, \tag{2e}$$

$$\mathbf{c}_j^{h(\gamma)}(t+k) > 1, j \in \mathcal{V}, h \in \mathcal{I}^{\text{discs}}, \gamma \in \mathcal{I}^{\Gamma}, \tag{2f}$$

where constraints (2b)-(2f) are for $k = 1,\dots,N$.

The planner takes into account the model of the vehicle (2b), which is a discretized (at 25 Hz) kinematic bicycle model [24]. The kinematic bicycle model is consistent for the maneuvers we consider at this stage (i.e., maneuvers with a low lateral acceleration) [25]. The problem above is solved in a receding horizon fashion. At time t, the planner acquires the state vector $\mathbf{x} := [x,\, y,\, \theta,\, v]^{\mathrm{T}}$ (current pose and speed) of the car from the localization module and the approximated path parameter ϕ (Eq. (2c)) according to [13]. Then, the planner solves the optimization problem above to obtain a sequence of commands $[\mathbf{u}]_{k=t}^{k=t+N} := \{\mathbf{u}(t),...,\mathbf{u}(t+N)\}$. Then, the planner sends the first control command $\mathbf{u}(t) := [a,\, \delta]^{\mathrm{T}}$ (i.e., acceleration a and front steering δ) through the UDP node to the real-time PC (as Figure 2 shows) and discards the other elements of the sequence.

The cost J defines the objectives of the planner[1]:

$$J := J_v + J_a + J_\delta + J_e + J_{\text{rep}}. \tag{3}$$

[1]We omit the time dependency when it is clear from the context.

$J_v := \|v^{\text{ref}} - v\|_{Q_v}^2$ is a quadratic penalty (with weight Q_v) on the deviation from the desired speed v. $J_a := \|a\|_{Q_a}^2$ and $J_\delta := \|\delta\|_{Q_\delta}^2$ impose a quadratic penalty (with weights Q_a and Q_δ, respectively) on acceleration and steering commands, respectively. $J_e := \mathbf{e}^{\mathrm{T}} Q_e \mathbf{e}$, where $\mathbf{e} \in \mathbb{R}^2$ is the error with respect to the path provided by the route planner module $\mathbf{p}^{\text{path}}(\phi) \in \mathbb{R}^2$ in the path's tangential and normal directions (see [13] for details). Finally, J_{rep} is a repulsive penalty function. Its definition is related to the collision avoidance constraints and the road boundaries as detailed below.

Following [13] our vehicle is represented in the MPCC problem as n_{discs} discs of radius r centered in $\mathbf{p}^h$ (where we used $\mathbf{p}$ to indicate the position on the (x, y) plane in the body frame, $h \in \mathcal{I}^{\text{discs}} := \{1, 2, \dots, n_{\text{discs}}\}$, and n_{disc} is the number of discs used to describe the vehicle). From the perspective of vehicle, VRU j ($j \in \mathcal{V} := \{1,\dots,n_{\text{VRUs}}\}$, n_{VRUs} is the number of VRUs seen by the car) is represented as an ellipse centered in $\mathbf{p}_j$ with orientation R_j and a (longitudinal direction) and b (lateral direction) as semi-major and semi-minor axis, respectively. For our design, the position of the VRU at prediction step k is provided by the perception module as a mean position. Then, recall that for each Gaussian $\gamma \in \mathcal{I}^{\Gamma}$, the perception module provides also $\Sigma^{(\gamma)}$ (capturing perception inaccuracy and behavioral variance) that is used to derive a, b, and R_j. Let $\Sigma^{(\gamma)}$ be defined as follows:

$$\Sigma^{(\gamma)} := \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix} \tag{4}$$

Then, by using the singular value decomposition (SVD) of $\Sigma^{(\gamma)}$ we can obtain the values of a, b, and the orientation of the obstacle, as follows:

$$R_j^{(\gamma)} \begin{bmatrix} {a^{(\gamma)}}^2 & 0 \\ 0 & {b^{(\gamma)}}^2 \end{bmatrix} {R_j^{(\gamma)}}^{\mathrm{T}} = \text{SVD}\left(\Sigma^{(\gamma)}\right), \tag{5}$$

where $R_j^{(\gamma)}, a^{(\gamma)}, b^{(\gamma)}$ represent the values of R_j, a, and b, respectively, according to the Gaussian γ. Finally, the collision avoidance constraints at time t for $h \in \mathcal{I}^{\text{discs}}$, can be approximated as follows[2]:

$$\underbrace{\left(R_j(\mathbf{p}^h - \mathbf{p}_j)\right)^{\mathrm{T}} \begin{bmatrix} \frac{1}{(a+r)^2} & 0 \\ 0 & \frac{1}{(b+r)^2} \end{bmatrix} R_j(\mathbf{p}^h - \mathbf{p}_j)}_{\mathbf{c}_j^{h(\gamma)}} > 1, \tag{6}$$

Loosely speaking, along the prediction horizon, the collision avoidance constraints above impose that each circle representing the vehicle and the ellipse representing VRU j do not intersect (if they intersect it means that a collision occurs). We repeat the argument above for each Gaussian provided by the perception module (for one obstacle we have as many collision avoidance constraints as the number of Gaussians).

Constraints (6) require $\mathbf{p}_j$ and η_j, that is, the pose (i.e., position and orientation) of VRU j, over the prediction horizon of the planner N. The perception module provides this information with some uncertainty. This uncertainty grows with time due to the behavioral variance of the VRU

[2]We omit the dependency on γ to simplify the notation.

(see for example Figure 5). Hence, when choosing N for the planner, we take into account *(i)* the quality of the predictions of the VRUs' intentions provided by the perception module (if the prediction horizon is too long the uncertainties in the predicted position of the VRU will be too large), *(ii)* safety margins to react to the obstacles (if the prediction horizon is too short the vehicle does not have enough time to react), *(iii)* real-time performance. If the MPCC plans too far in the future, it will have to deal with larger uncertainties on the predicted position of the VRUs (leading to more conservative trajectories), but if the horizon is too short the vehicle might not have enough time to react to the presence of the VRU. In addition, increasing the length of the prediction horizon leads to a larger number of decision variables and constraints that might compromise the real-time performance. Real-time performance is required given that the planner does not only provide a feasible trajectory for the car, but also the direct commands to the actuators (acceleration and steering angle). For our experiment, we selected $N = 3$ s, which offers a good compromise between the quality of the prediction provided by the perception, the safety margins to react to nearby obstacles, and real-time performance.

Our planner also takes into account the road boundaries, which are defined with respect to the global path as follows:

$$r_b := \left[-\sin\bar{\theta}(\phi)\ \cos\bar{\theta}(\phi)\right]\left(\mathbf{p}-\mathbf{p}^{\text{path}}\right) \in \left[r_b^r,\, r_b^l\right], \quad (7)$$

where $\bar{\theta}(\phi)$, r_b^r, and r_b^l represent the heading of the path, the allowed offset to the right and to the left of the global path, respectively. Constraints (7) appear in the MPCC formulation as (2e), which includes convex constraints on states and actuators (such as actuator limitations).

Using the definition of the collision avoidance constraints (2f) and of the road boundaries (7), we can define the cost associated with the repulsive fields as follows:

$$J_{\text{rep}} := \sum_{j\in\mathcal{V}}\sum_{h\in I^{\text{discs}}} \frac{Q_{\text{c.a.}}}{\|1-c_j^{h(\gamma)}\|^2} + Q_{\text{r.b.}}\left(e^{r_b-r_b^r}+e^{-r_b+r_b^l}\right), \quad (8)$$

where $Q_{\text{c.a.}}$ and $Q_{\text{r.b.}}$ are tuning parameters. The repulsive cost is an additional safety measure (we already impose collision avoidance constraints and road boundary constraints in (2)). Loosely speaking, the repulsive cost is such that the closer the vehicle gets to the VRU j (according to the information provided by the perception module) or to the road boundaries, the higher is the cost. Hence, the planner has the incentive to keep a *safe* distance from the VRUs and to stay within the road boundaries.

The planner requires a solver to find a solution for (2) in real time. We integrated ACADO [17] in our ROS framework. ACADO uses a direct multiple shooting method to solve (2), which is nonconvex and nonlinear. For $N = 3$ s, the optimization problem has 200 decision variables (control and state variables) and 250 constraints (road boundaries, collision avoidance, control, and velocity constraints).

E. Low-level Control System

Our platform is equipped with a MOVE Box, a device developed by TNO (the Netherlands Organisation for Applied Scientific Research). The MOVE Box allows us to remove the driver from the loop. It enables the longitudinal control by exploiting the existing adaptive cruise control and the lateral control by exploiting the electric power steering system.

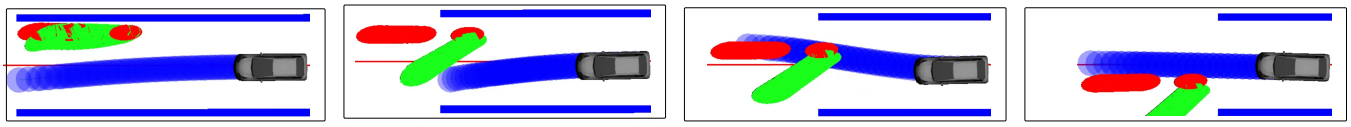
A dSpace AutoBox bridges the communication between the ROS PC (Ethernet-UDP node) and the MOVE Box (CAN). dSpace forwards the control commands provided by the MPCC to the MOVE Box. In addition, dSpace implements low-level safety measures (e.g., monitoring maximum acceleration and steering angle) and handle communication loss (e.g., sending a back-up command that combines neutral steering and slight braking is sent to the MOVE Box in case of loss of communication).

IV. EXPERIMENTS

We tested our proposed design in different scenarios with extensive simulations and experiments. Our simulation setup uses a 9 DoF non-linear car model (three rigid bodies representing the sprung body, front, and rear axles) developed in MATLAB/SimMechanics [26]. The tire dynamics are modelled using Delft-Tyre 6.2 with a Magic Formula steady-state slip model describing nonlinear slip forces and moments [27]. The car model runs on a Windows PC with an Intel Xeon CPU running at 3.60 GHz. The car model is simulated at 100 Hz, while the localisation, perception, and control on the ROS machine (running Ubuntu 18.04.1 LTS) send messages at 25 Hz (as in the setup we have on the real vehicle). Our modules in our experimental setup are implemented on a PC (mounted on board of the car) running Ubuntu 18.04.1 LTS with an Intel(R) Core(TM) i7-6900K CPU at 3.20GHz. The PC has 64GB memory. In addition, the PC contains two Titan X (Pascal) GPUs for stereo matching and VRU detection. The CUDA version used is 10.0.

For safety reasons, we tested the interactions of the vehicle with a cyclist and two pedestrians in simulation. Furthermore, we tested the interactions between the real vehicle and a pedestrian dummy during our experiments. Our design was able to adapt the vehicle behavior to different initial configurations (e.g., different reference velocities for the vehicle and different behavior of the pedestrians and cyclist). In all the cases studied, the MPCC provided suitable paths for the vehicle to follow to ensure the safety of the VRUs by taking into account their predicted paths (with behavioral uncertainties) provided by the perception module. If sufficient space was available, the vehicle passed the VRUs, planning agile maneuvers when needed (e.g., Scenario *S2*). If passing was unsafe the vehicle reduced its speed or stopped (e.g., Scenario *S1*). We detail the scenarios described in Section I.

S1: Figure 4 shows the simulation results obtained from the interaction with a cyclist and shows the benefits of using the estimated paths of the cyclist in the planner. The cyclist decides to turn at an upcoming intersection. Thanks to the perception module that predicts the path (Figure 4a), the car starts to brake (notice that the length of the blue path shrinks) before the cyclist starts to turn (Figure 4b) and adapts its path to prevent a possible collision (Figure 4c), while remaining within the road boundaries. Without the



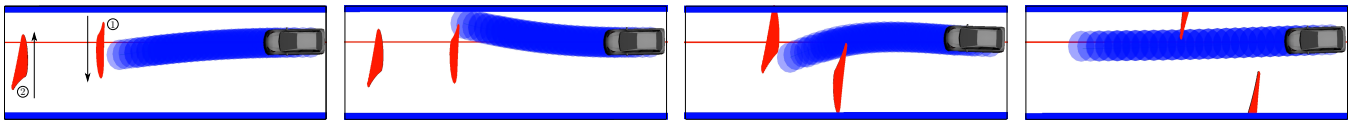
(a) The vehicle plans to overtake the cyclist at the left.

(b) The vehicle brakes based on the cyclist's estimated path.

(c) The vehicle plans based on cyclist's estimated path.

(d) The vehicle returns to its path.

Fig. 4: *S1: turning cyclist.* The vehicle adapts its predicted path based on the two estimated paths of the cyclist. The blue lines are the road boundaries, the red line is the global path, the blue path (circles) is the predicted trajectory of the car, the green and red paths (ellipses) represent the predicted trajectory of the cyclist provided by the perception module (as a mixture of two Gaussians). The red path is associated with the prediction that the cyclist will go straight at the intersection, while the green path is associated with the prediction that the cyclist will turn at the intersection.



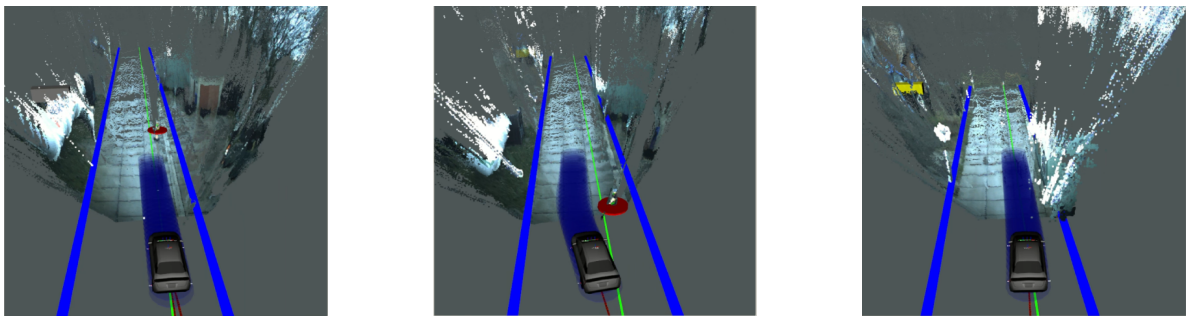
(a) The vehicle plans to pass at left the first pedestrian.

(b) The vehicle plans to pass at right the first pedestrian.

(c) The vehicle encounters the second pedestrian and plans a path to avoid both pedestrians.

(d) The vehicle returns to its path and the pedestrians safely cross the road.

Fig. 5: *S2: two pedestrians*. The vehicle adapts its predicted path based on the estimated path (red ellipses) of each pedestrian. Notice that the size of each ellipses grows over the horizon due to the uncertainties on the pedestrian positions over time.



(a) Start of the overtaking.

(b) During the overtaking.

(c) End of the overtaking.

Fig. 6: *S3: experimental results with a pedestrian dummy*. Trajectory of the vehicle during one of our experiments with our research platform. The blue lines depict the road boundaries, the green line is the global path, the blue circles depict the trajectory planned by the local planner, the red ellipses represent the dummy's predicted position.

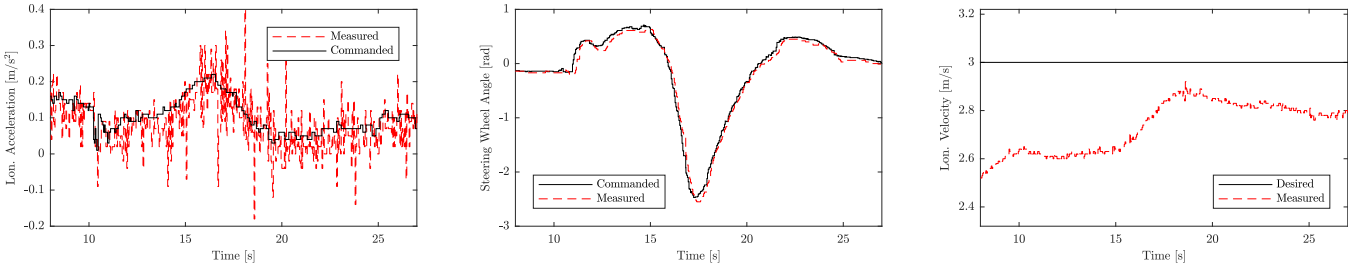


Fig. 7: *S3: experimental results with a pedestrian dummy.* Acceleration, steering wheel angle, and longitudinal velocity of the vehicle.

prediction the cyclist will turn represented in green (e.g., with just a constant velocity model) the car would not have enough time to react to the turning cyclist. Notice that during the maneuver the planner commands the car to brake (reducing its speed) for the safety of the cyclist (second plot from the left in Figure 4b). This is possible thanks to the MPCC formulation that, compared to classical path-following approaches, allows the controller more flexibility to determine the state trajectories.

S2: Figures 5 shows simulation results with two pedestrians crossing in front of the car. This scenario shows how our vehicle handles multiple VRUs. The vehicle starts to

pass at left the first pedestrian (Figure 5a). Then, given that the first pedestrian continues to cross the street (from top to bottom) the vehicle plans to pass at right (Figure 5b). During the maneuver, the vehicle encounters the second pedestrian (crossing the road from bottom to top), and plans a path to avoid both pedestrians (Figure 5c). The two pedestrians cross the road safely and the car returns to its path.

S3: Figures 6 and 7 show the experimental results. As Figure 6 depicts, the vehicle is able to overtake the pedestrian dummy by taking into account its predicted position. At the same time, the car is also able to increase its speed to reach its desired speed (3 m/s), as Figure 7 shows. Figure 7 shows that the measured vehicle motion, closely follows the desired motion, with some noise in the acceleration and a small delay in steering (approx 0.2 s) caused by physical steering-wheel limitations. Nevertheless, the vehicle is able to safely pass the pedestrian dummy.

V. CONCLUSIONS

We presented our research platform SafeVRU for the interaction of self-driving vehicles with VRUs. Our self-driving vehicle relies on a local motion planner that incorporates the predicted VRUs paths provided by stereo vision-based perception module to safely navigate in the presence of VRUs. The MPCC creates suitable trajectories to safely interacts with multiple VRUs taking into account the predicted paths (with behavioral uncertainties) provided by the perception module. The platform is implemented in ROS and runs in real time. We showed promising results with our platform both in simulation and in real-world experiments. The modular structure of our planner allows one to adjust the driving style of the vehicle and the comfort levels of the VRUs. Our design is robust to different scenarios and allows the vehicle to interact with different VRUs (cyclists and pedestrians) at different speeds.

SafeVRU is an on-going research effort. We aim to increase the number and complexity of scenarios addressed. This will lead to several design and implementation challenges to guarantee, for example, real-time performance.

ACKNOWLEDGMENT

This work received support from the Dutch Science Foundation NWO-TTW Foundation, within the SafeVRU project (nr. 14667) and Veni award nr. 15916.

REFERENCES

[1] World Health Organization, "Road traffic injuries," 2018, https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries.

[2] D. J. Fagnant and K. Kockelman, "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations," *Transportation Research Part A: Policy and Practice*.

[3] A. L. Rosado, S. Chien, L. Li, Q. Yi, Y. Chen, and R. Sherony, "Certainty and critical speed for decision making in tests of pedestrian automatic emergency braking systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1358–1370, 2017.

[4] Daimler AG, "On the road in self-driving vehicles," 2018, https://www.daimler.com/innovation/autonomous-driving/special/changes.html.

[5] A. Davies, "Americans can't have AUDI's super capable self-driving system," 2018, https://www.wired.com/story/audi-self-driving-traffic-jam-pilot-a8-2019-availablility/.

[6] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[7] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416–442, 2015.

[8] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[9] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller *et al.*, "Making Bertha Drive–An Autonomous Journey on a Historic Route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[10] C. G. Keller, T. Dang, H. Fritz, A. Joos, C. Rabe, and D. M. Gavrila, "Active pedestrian safety by automatic braking and evasive steering," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1292–1304, 2011.

[11] T. Dang, J. Desens, U. Franke, D. Gavrila, L. Schäfers, and W. Ziegler, "Steering and evasion assist," in *Handbook of intelligent vehicles*. Springer, 2012, pp. 759–782.

[12] T. Faulwasser, B. Kern, and R. Findeisen, "Model predictive path-following for constrained nonlinear systems," in *Proc. of CDC/CCC 2009*, 2009, pp. 8642–8647.

[13] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model," *IEEE Transactions on Intelligent Transportation Systems*, no. 99, pp. 1–15, 2017.

[14] S. Köhler, B. Schreiner, S. Ronalter, K. Doll, U. Brunsmann, and K. Zindler, "Autonomous evasive maneuvers triggered by infrastructure-based detection of pedestrian intentions," in *IV Symposium*, 2013, pp. 519–526.

[15] T. Grußner, L. Bürkle, and C. Marberger, "Erweiterung Aktiver Fußgängerschutzsysteme Durch Eine Fahrerinitierte Ausweichunterstützung," in *Uni-DAS e.V. Workshop Fahrerassistenzsysteme*, 2015, pp. 19–28.

[16] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Kobe, Japan, May 2009.

[17] B. Houska, H. Ferreau, and M. Diehl, "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization," *OCAM*, vol. 32, no. 3, pp. 298–312, 2011.

[18] T. Moore and D. Stouch, "A generalized extended kalman filter implementation for the robot operating system," in *Proceedings of IAS-13*, July 2014.

[19] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3D reconstruction in real-time," in *IV Symposium*, 2011.

[20] J. F. Kooij, F. Flohr, E. A. Pool, and D. M. Gavrila, "Context-Based Path Prediction for Targets with Switching Dynamics," *International Journal of Computer Vision*, pp. 1–24, 2018.

[21] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, *Lecture Notes in Computer Science*, vol. 9905 LNCS, pp. 21–37, 2016.

[22] M. Braun, S. Krebs, F. Flohr, and D. M. Gavrila, "EuroCity Persons: A Novel Benchmark for Person Detection in Traffic Scenes," *IEEE Trans. on Pattern Analysis Machine Intelligence*, 2019, DOI: 10.1109/TPAMI.2019.2897684.

[23] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *2017 CVPR*, 2017, pp. 1302–1310.

[24] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IV Symposium*, June 2015, pp. 1094–1099.

[25] P. Polack, F. Altch, B. d'Andra-Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" in *IV Symposium*, 2017, pp. 812–818.

[26] Z. Lu, B. Shyrokau, B. Boulkroune, S. van Aalst, and R. Happee, "Performance benchmark of state-of-the-art lateral path-following controllers," in *15th IEEE International Workshop on AMC*, 2018, pp. 541–546.

[27] Tass International, "Delft-tyre 6.2."