On Robust Context-Aware Navigation for Autonomous Ground Vehicles

Paolo Forte, Himanshu Gupta, Henrik Andreasson, Uwe Köckemann, and Achim J. Lilienthal, *Senior Member, IEEE*

***Abstract*—We propose a context-aware navigation framework designed to support the navigation of autonomous ground vehicles, including articulated ones. The proposed framework employs a behavior tree with novel nodes to manage the navigation tasks: planner and controller selections, path planning, path following, and recovery. It incorporates a weather detection system and configurable global path planning and controller strategy selectors implemented as behavior tree action nodes. These components are integrated into a sub-tree that supervises and manages available options and parameters for global planners and control strategies by evaluating map and real-time sensor data. The proposed approach offers three key benefits: overcoming the limitations of single planner strategies in challenging scenarios; ensuring efficient path planning by balancing between optimization and computational effort; and achieving smoother navigation by reducing path curvature and improving drivability. The performance of the proposed framework is analyzed empirically, and compared against state of the art navigation systems with single path planning strategies.**

***Index Terms*—Autonomous vehicle navigation, motion and path planning, robotics and automation in construction.**

I. Introduction

DEPLOYING autonomous vehicles in highly complex and dynamic environments, such as mines or construction sites, necessitates addressing the path planning problem, that of determining safe and efficient trajectories for robots to follow while performing tasks. Classic solutions for solving the path planning problem include approaches that differ in the planning technique [1], [2], in the kinematics and dynamics model of the vehicle [3], [4], and in the type of environment [5], [6]. Typically, the path planning problem is addressed using a single path planning strategy. However, to achieve long-term autonomous navigation in complex and dynamic environments, exclusively relying on a single planning algorithm is often infeasible or sub-optimal. For instance, a single path planning strategy may fail when transitioning from open spaces to narrow corridors, such as those found in mines. In fact, specialized solutions may be required according to the current task, the current location of the vehicle, the environment, and the weather conditions.

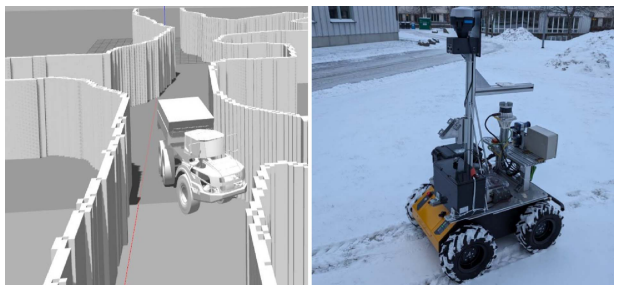

Fig. 1. Vehicles used during the experiments.

To achieve long-term autonomous navigation, we proposed a context-aware navigation framework for autonomous ground vehicles in different weather conditions and in harsh and unstructured environments. The system autonomously selects the best global path planning, control strategies, and navigation parameters based on real-time weather conditions, sensor data, and navigated map areas. This letter has three main contributions: 1) a context-aware navigation system for the autonomous navigation of autonomous ground vehicles utilizing a behavior tree structure with novel nodes; 2) implementation of planning and control strategies for the navigation of articulated vehicles; and 3)a metric and benchmark scenarios involving various contexts, used to conduct an empirical evaluation between the proposed approach and state-of-the-art systems utilizing single-path planning strategies.

The rest of the letter is organized as follows. Relevant literature is analyzed in Section II. Some preliminary concepts are introduced in Section III. The proposed framework for autonomous navigation is described in Section IV and evaluated in Section V using a simulated articulated vehicle and a husky robot, shown in Fig. 1.

II. Related Work

A. Autonomous Navigation

Autonomous driving for car-like vehicles has been an active research topic for the last decades. Proposed solutions have been focused on navigation in dynamic environments [7], on roads [8], and in unknown and obstacle-filled environments [9]. The navigation of articulated vehicles requires specialized techniques

Received 8 July 2024; accepted 22 November 2024. Date of publication 23 December 2024; date of current version 3 January 2025. This article was recommended for publication by Associate Editor Z. Kingston and Editor A. Bera upon evaluation of the reviewers' comments. This work has been partially funded by the Horizon Europe Framework Programme through the euROBIN Grant 101070596. *(Corresponding author: Paolo Forte.)*

Paolo Forte, Himanshu Gupta, Henrik Andreasson, and Uwe Köckemann are with the Centre for Applied Autonomous Sensor Systems, Örebro University, Örebro 70182, Sweden (e-mail: paolo.forte@oru.se; uwe.köckemann@oru.se).

Achim J. Lilienthal is with the Munich Institute of Robotics and Machine Intelligence, Technische Universität München, 80992 Munich, Germany (e-mail: achim.j.lilienthal@tum.de).

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2024.3520920, provided by the authors.

Digital Object Identifier 10.1109/LRA.2024.3520920

that account for the additional maneuverability constraints imposed by the vehicle's articulation and kinematic models [10], and also by the environment [11]. Some works focused on how to generate optimal trajectories for articulated vehicles using a combination of clothoid curves and straight segments [12] or a genetic algorithm [13]. However, these approaches rely on a single planning strategy to compute the path for the vehicle.

B. Context-Aware Navigation

Navigation in complex and dynamic environments requires reasoning about various information and adapting navigation strategy accordingly, namely context-aware navigation. Niemeyer et al. [14] proposed a ROS-based context-aware navigation system for agricultural environments that changes planning and execution strategies based on a semantic map. Galindo et al. [15] proposed an approach to enable a mobile robot to construct a semantic map using sensor data and utilize this semantic information for navigation tasks. Theobalt et al. [16] introduced a combined dialogue and navigation system implemented on a mobile robot that explores the usage of natural language to assist navigation. Qi et al. [17] presented a novel semantic mapping technique for domestic navigation, enhancing occupancy grid maps with object and room semantics to improve robot navigation and environment understanding. However, the proposed approaches lack integration with behavior trees, are not designed for use with articulated vehicles, nor are they implemented in ROS2. Moreover, these works focused only on spatial contexts, neglecting other important contexts such as weather for outdoors environment, human, and time of the day.

C. Behavior Tree

Behavior trees, due to their significant advantages in terms of safety and efficiency [18], have been used as an architecture for the navigation of cars [19] and aerial vehicles [20]. Macenski et al. [21] proposed NAV2, a ROS2 framework that relies behavior trees to handle robot navigation for different types of robots. Several frameworks have been based on NAV2 for the navigation of mobile robots in uneven terrain [22] or based on lingustic instruction [23]. However, the current NAV2 frameworks have two main limitations: (1) to the best of the authors' knowledge, they currently lack an implementation for articulated vehicles, and (2) they do not perform context-aware navigation.

III. Preliminary Concepts

A. Formulation for Car-Like and Articulated Vehicles

The kinematic equations for a car-like vehicle are derived from the bicycle model and are as follows:

$$\begin{cases} \dot{x} = V\cos(\theta) \\ \dot{y} = V\sin(\theta) \\ \dot{\theta} = \frac{V}{L_c}\tan(\delta) \end{cases} \tag{1}$$

Where L_c denotes the distance between front and rear axles of the car; and δ is the steering angle of the car front wheels. An articulated vehicle comprises two bodies, called front and rear, connected by an articulation hitch. As each body has only one axle and all wheels are non-steerable, the vehicle steers at the joint, utilizing hydraulic actuators to adjust the angle between the two bodies. The mathematical model for an articulated vehicle, derived by Altafini [24], is reported in (2).

$$\begin{cases} \dot{x} = V\cos,(\theta) \\ \dot{y} = V\sin(\theta) \\ \dot{\theta}_r = \frac{V\sin(\varphi)}{L_f + L_r\cos(\varphi)} - \dot{\varphi}\frac{L_f}{L_f + L_r\cos(\varphi)} \\ \dot{\theta}_f = \frac{V\sin(\varphi)}{L_f + L_r\cos(\varphi)} + \dot{\varphi}\frac{L_f\cos(\varphi)}{L_f + L_r\cos(\varphi)} \end{cases} \tag{2}$$

where: $(x,\ y,\ \theta) \in$ SE(2) and V denote the pose and linear velocity of the vehicle; φ is the steering angle between the front and rear bodies; L_j is the length from the wheel axle of a body and the hinge; j $\in$ {r,f} are the indexes referring to the rear and front body respectively.

B. Path Planning

Path planning can be divided into two complementary parts: local (online) and global (offline) path planning [25]. In global path planning approaches, the robot assumes complete knowledge of the environment and uses this information to compute an initial path that minimizes some target metric, e.g., the traveled distance [26]. For the local path planning, the robot relies on partial knowledge of the environment, usually retrieved using single or multiple sensors [27].

Global path planning approaches include sampling-based algorithms, such as rapidly exploring random tree (RRT) [28] and lattice-based [29] planners, and grid-based algorithms, like Dijkstra [2] and A* [30].

1) Grid-based methods: Search algorithms designed to find the shortest path from a starting node to a goal node within a graph. They use heuristic information to guide exploration efficiently and guarantee an optimal path when an admissible heuristic is applied. These methods are widely used in robotics due to their efficiency and ability to ensure optimality. While they excel in structured environments, they can be computationally demanding in high-dimensional or dynamic spaces.

2) Sampling-based methods: Sampling-based methods explore a robot's configuration space by incrementally sampling points and connecting them to form feasible paths. An example is the lattice-based approach, which is based on two main aspects: (1) systematically sampling the state space; (2) constraining the motions of the vehicle to a lattice graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ embedded in a Euclidean space (i.e., $\mathbb{R}^n$), where each vertex $v \in \mathcal{V}$ represents a valid configuration of the vehicle, and each edge $e \in \mathcal{E}$ denotes a viable motion between two configurations which respects the constraints of the vehicle. Let $\mathcal{P}$ be a set of primitives. Each primitive p $\in \mathcal{P}$ is a motion (i.e., a set of poses) between two configurations that comply with the kinematic constraints of the vehicle. A solution $\pi = (p_0, \ldots, p_n)$ consists of a set of collision-free primitives that connect the start and the goal configurations. Classical graph-search algorithms can be used to explore the state space and find a solution. Unlike grid-based approaches, these approaches do not require an explicit representation of the entire space, making them efficient for high-dimensional and complex environments. However, they struggle with narrow passages and cannot guarantee optimal paths. Approaches to efficiently target sampling in narrow passages include the Bridge Test algorithm [31] and a sampling-based method that learns from configuration space graphs [32].

TABLE I
DATASET COMPOSITION FOR WEATHER CLASSIFICATION

	Clear	Rain	Fog	Snow
Train	4201	3125	3705	3417
Test	1800	1338	1587	1464
Total	6001	4463	5292	4881

C. Image-Based Weather Detection

Various image-based datasets [33], [34] are available for single weather labels, but none can be used to train a single model for multi weather prediction. Additionally, no pre-trained models are available. Thus, we combined several datasets to create a well-balanced adverse weather recognition dataset and trained various deep neural network models to find the best model for our context-aware navigation system.

1) Dataset: The dataset used is a combination of various weather-based image datasets: the DAWN dataset [35], AAU RainSnow Traffic Surveillance dataset [36], Multi-Label Weather Recognition dataset [37], Rain in Surveillance dataset [36], and Weather Phenomenon dataset [38]. Additional foggy weather images were scrapped from internet to mitigate the datasets imbalance. Table I provides the dataset composition used for training and testing the weather classification models.

2) Training Procedure: We trained state-of-the-art models and selected the model with the highest validation accuracy to test the proposed weather-based navigation system. The investigated models are: residual network (ResNet18), EfficientNet, YOLOv5, vision transformer (ViT), and Swin transformer. Each model was trained for 20 epochs on 224 $\times$ 224pixel images using a batch size of 32 and stochastic gradient descent optimizers with a learning rate that decayed from 0.001 to 0.0001. Image augmentations used are random cropping of images at different scales (0.08–1.0) and aspect ratio (0.75–1.33), resizing to desired image size, random horizontal flipping, AugMix, and random erasing implemented in Torchvision library. The augmented image were normalized using Imagenet statistics. During testing, the short side of the image is scaled to the desired shape, maintaining the aspect ratio, centered crop to have the same height and width, and then normalized using the Imagenet statistics.

IV. PROPOSED SYSTEM

A. Context-Aware Navigation

Assuming that the map is completely or partially known and the vehicle is equipped with various sensors (e.g., camera and lidar), the objective is to develop a framework that selects the most suitable global planner $P \in \mathcal{P}$ and control strategy $L \in \mathcal{L}$, along with their parameters, based on a set of navigation-related contexts $\mathcal{C}$, defined by sensor data $\mathcal{S}$ and the map $\mathcal{M}$.

$$F_C : \mathcal{S} \times \mathcal{M} \longrightarrow \mathcal{C} \quad F_P : \mathcal{C} \longrightarrow \mathcal{P} \quad F_L : \mathcal{C} \longrightarrow \mathcal{L} \qquad (3)$$

Examples of mapping from the sensor set $\mathcal{S}$ and the map $\mathcal{M}$ to set of context $\mathcal{C}$ are listed in Table II. In this work, we focus on two contexts: spatial and weather conditions.[1]

[1]Additional sensors or context can be added based on user specification.

TABLE II
EXAMPLES OF MAPPING FROM SENSOR SET $\mathcal{S}$ AND MAP $\mathcal{M}$ TO CONTEXT SET $\mathcal{C}$. * ARE USED IN THIS WORK

Source	Abstract Context	Context
*Lidar (S)	Spatial	Corridor, Open Space
*Camera (S)	Weather	Rain, Snow, Sun, Fog
Camera (S)	Human	Human(s) Presence
Map (M)	Spatial	Occupancy Levels, Rooms
Map (M)	Semantic	Different Zones, Terrain status

TABLE III
PROPOSED NOVEL ACTION (A), CONDITION (C), AND DECORATOR (D) BEHAVIOR TREE NODES

Add Planners (A)	Adds a set of planners to the list of selectable planners
Controller Selector (A)	Selects the most feasible control strategy
Planner Selector (A)	Selects the most feasible global path planner
Remove Planner (A)	Remove a planner from the list of selectable planners
Check Path (C)	Checks if the path has been computed with a specific strategy
Check Planner (C)	Verify if the selected global planner has changed
Area Controller (D)	Ticks its child node when entering a new area of the map
Waypoint Controller (D)	Ticks its child node when near a waypoint

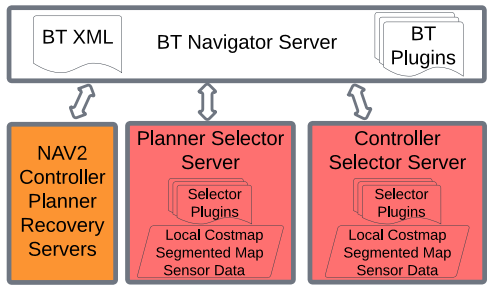

Fig. 2. Architecture of the overall system. The system expands the NAV2 framework with novel behavior tree nodes (red boxes) and introduces new plugins tailored to articulated vehicles (orange box).

B. Context-Aware Navigation Framework

To achieve context-aware autonomous navigation of autonomous ground vehicles, we (1) designed a set of novel behavior tree actions, decorators, and condition nodes, summarized in Table III, and (2) implemented global planning algorithms, controller strategies, and recovery strategies explicitly tailored to meet the requirements of car-like and articulated vehicles. The proposed system is shown in Fig. 2. The key feature of the proposed system lies in its ability to adaptively choose and switch between different global planners and control strategies during run-time based on a fusion of data from robot constraints (e.g., geometry, configuration, and current operation), as well as inputs derived from sensors (e.g., objects positions and weather conditions) and the map (e.g., vehicle location and obstacles). Indeed, different areas of the map can possess distinct attributes that require specific navigation techniques or parameterizations, such as velocity reduction or safety distance augmentation.

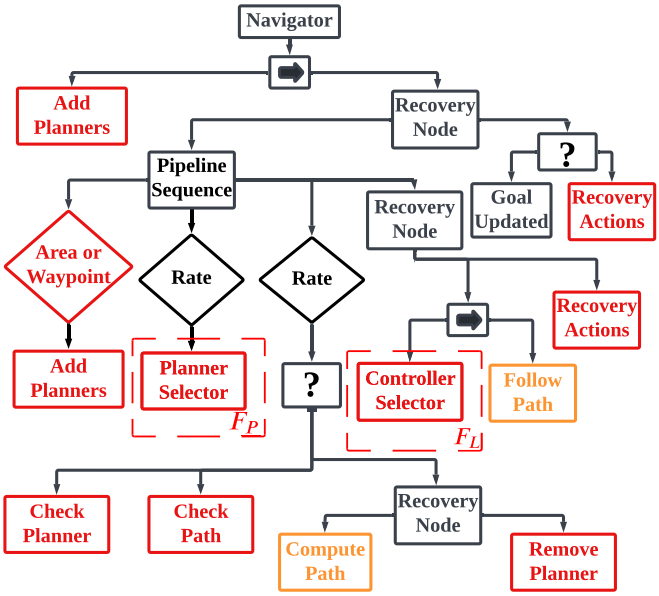

Fig. 3. Proposed behavior tree framework for context-aware navigation. The formulation is based on [18]. Nodes highlighted in red indicate the contributions of the letter. Nodes highlighted in orange indicate NAV2 nodes with new custom plugins for car-like and articulated vehicles. The remaining nodes remain unchanged from the current NAV2 framework. Either the area or waypoint controller is used to detect when a new area is entered.

Therefore, a given planner might encounter problems in generating a feasible path within a specific map area, such as narrow corridors, whereas alternative planners could outperform.

The proposed *Planner Selector* and *Controller Selector* nodes implement the F_P and F_L functions, respectively, and are designed with a standard plugin interface, simplifying the implementation and integration of new selection strategies. Users can add new contexts and implement custom selection plugins[2] to choose the optimal planner, control strategies, and navigation parameters based on their specific requirements and preferences. This structure also enables dynamic switching between selection methods during runtime. We implemented a selection strategy plugin (see Section IV-D) based on lidar and camera data.

C. Behavior Tree System

The architecture of the proposed holistic framework is shown in Fig. 3. The overall system works as follows. Upon receiving a goal, the system is initialized with an initial set of global planners through the *"Add planners"* node. Then, the "Recovery Node" executes the "Pipeline Sequence" which trigger the *"Area controller"* decorator which activates another *"Add planners"* node only when a new area is entered, triggered by either providing a set of waypoints or a segmented map. This situation leads to the restoration of the set of available planners or the addition of new planners.

Subsequently, the framework examines and selects the best global planner based on F_P and F_L. The selector will choose from a user-defined list of path planning and control strategies, making the system versatile for any path planning algorithm and control technique. The system continuously performs selection

[2]https://github.com/PaoloForte95/navigo_ros2.

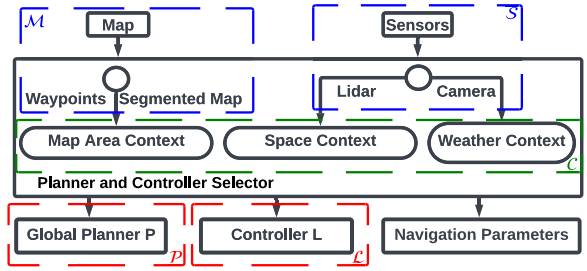

Fig. 4. Overview of the implemented selection procedure.

during navigation with a customizable control rate. The control rate of the selector node is a configurable parameter within the behavior tree, allowing for flexibility in adjusting the frequency of planning strategy updates during navigation. Once the optimal global planner has been selected, the system will compute a new global path from the current robot position to the goal location. The global path is recomputed with a frequency defined by a *Rate controller* node. However, we introduce the *"Check planner"* and *"Check path"* nodes, which verify whether the optimal global planner has been changed and whether a path has already been computed using a specific strategy, respectively. This avoids the recomputation of the global path for specific planners (e.g., lattice planners) that may have large computationally expensive and introduce problems like path discretization, oscillations, and curvature issues during path evaluation. If the plan computation is successful, the global path is provided to the local controller server, which then computes the control actions to follow it based on the selected controller strategy. Instead, if the plan computation fails, the *"Remove Planner"* node removes the currently selected planner from the list of available global planners. Subsequently, a new global planner is selected from the list, and the process is repeated. If none of the planners can find a feasible path, the system fails as it exhausts all available planning options and proceeds with executing recovery actions.

D. Planner Selector Node

This node implements the function F_P to choose the most appropriate planner. We developed a selector plugin that determines the optimal global planner by utilizing data from a semantic map, camera images, and lidar point clouds (see Fig. 4). This strategy is divided into three main steps.

Firstly, the selector creates a configurable bounding box around the vehicle. This box represents the selector's obstacle detection range. We created four spatial contexts: (1) open space, (2) obstacle on the left, (3) obstacle on the right, and (4) corridor (see Fig. 5), and we associate each of them with a specific global planner strategy.

The selector examines the data gathered through the lidar sensor and identifies its spatial context by considering the quantity and position (whether on the left or right side of the vehicle) of obstacles within the designated area, i.e., F_C. Additional contexts, such as front obstacles and dead ends, could be incorporated if a specialized planner is better suited for those situations. If the values on both sides exceed a configurable threshold, the system identifies itself as being in a corridor. Secondly, the selector acquires information regarding the current area of the map in which the vehicle is situated. This information can be obtained by two approaches: (1) defining a set of waypoints and

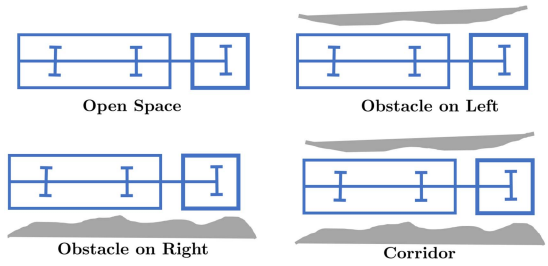

Fig. 5. Four spatial contexts in $\mathcal{C}$.

evaluating the distance by each of them; (2) employing a map filter that integrates a segmented map and the current location of the vehicle to identify the map area where the vehicle is situated. Lastly, a trained neural network is used to identify the current weather conditions. Using the identified weather condition, the selector fine-tunes the vehicle's parameters, such as maximum velocity and acceleration. For instance, in the case of snow or fog, the vehicle's velocity is reduced.

After analyzing the spatial context, weather condition and map position, the selector chooses the best global planning, control strategy, and navigation parameters for the current navigation task. The selection of the most suitable planner used in the experiments is based on our current knowledge and experimental findings. Gridmap-based solutions, like A*, encounter challenges when applied to ackermann or articulated vehicles in open space due to their complex kinematic models and limited maneuverability. In contrast, lattice-based techniques are well-suited for ackermann or articulated vehicles, as they accommodate intricate constraints, such as curvature, velocity, and acceleration. Gridmap methods offer speed, simplicity, and the assurance of finding the shortest path. When applied to ackermann or articulated vehicles, they exhibit particular strengths in narrow corridors, where paths are typically centralized and consistent across different planning strategies. In contrast, a lattice-based motion planner might struggle to compute a path in a narrow corridor. This is because its effectiveness relies on the granularity of motion primitives. To navigate successfully in such constrained spaces, the lattice planner would need to generate a finer set of motion primitives. However, this augmentation inevitably leads to longer computation time and memory demand for finding a solution, potentially rendering online planning impractical. Generally, due to the complex kinematics of ackermann and articulated vehicles, lattice-based motion planners are preferred in open spaces, while grid-based approaches are suitable for narrow environments.

V. Experimental Validation

We evaluate our framework from three points of view: (a) weather identification, (b) the navigation ability in complex environments, and (c) the solution quality in maps with different spatial and weather contexts.

a) Weather Identification: Table IV reports the top-1 accuracy on the test set for weather classification using deep learning models. The SWIN-T model, a vision transformer, achieves 94% accuracy on the test set, so we have chosen it for our navigation framework. The model is converted into the ONNX format to use as a cross-platform model with support in both C++ and Python programming languages.

TABLE IV
Weather Classification Accuracy (%)

	Clear	Fog	Rain	Snow	Overall
EfficientNet	91	90	**94**	95	92.5
ResNet18	95	89	91	95	92.5
ResNext50	96	90	92	96	93.5
SWIN-T	**97**	**92**	92	**97**	**94.5**
VIT-B-16	96	91	91	96	93.5
YOLO-v5-S	92	88	90	95	91.25

TABLE V
Simulation Parameters

Planners: $\{LatticeBased, A^*\}$
Controllers: $\{SplineFollowing, LocalController\}$
Selector: $\{CostmapSelector\}$
Planning Params: ct_{max} = 50 (s), oe_{max}= 0.3 (rad)
Robot params (Eqs. 1 and 2): L_c, L_f, L_R: = 0.5, 1.2, 3.8 m, φ_{max} : ±0.7rad, v_fwd_max, v_rev_max : 7, 2 m/s.

b) Navigation ability: We compared the proposed framework (P) against two single path planning navigation systems implemented using the state of the art NAV2 framework: the first utilizes a gridmap-based approach A* (G), and the second uses a lattice-based approach (L). We examine navigation ability in Test 1 (see Section VI-B), featuring diverse starting points and multiple destination locations for each navigation problem in different scenarios. Test 3 (see Section VI-D) investigates the robustness and adaptability of the proposed system in partially known maps. The aim is to investigate how changing spatial context can impact navigation ability.

c) Solution quality: Solution quality ζ_q is evaluated as a linear combination of four normalized metrics: path curvature[3] (p_c), traveled distance (t_d), orientation error on goal pose (o_e), and computational time (c_t):.

$$\zeta_{\text{oe}},\ \zeta_{\text{pc}},\ \zeta_{\text{ct}} = 1 - \frac{x}{x_{\max}} \qquad \zeta_{\text{td}} = \frac{x_{\min}}{x} \tag{4}$$

$$\zeta_q = \alpha_1 * \zeta_{\text{oe}} + \alpha_2 * \zeta_{\text{pc}} + \alpha_3 * \zeta_{\text{ct}} + \alpha_4 * \zeta_{\text{td}} \tag{5}$$

with weights $\alpha_k \in [0, 1]$ for each factor such that $\sum_{k=1}^{4} \alpha_k = 1$. In this work, we consider $\alpha_k = 0.25$. The max values for the orientation error, path curvature, and computation time are retrieved from vehicles specifications and reported in Table V. The minimum value for the travelled distance is computed using the A* planner for a single grid-cell. Test 2 (see Section VI-C) investigates the ability of the proposed solution to dynamically adjust planning strategies in a mine and office maps (based on changing spatial context), with the goal of improving solution quality.

d) Experimental Setup: We validate the proposed system in both real-world and simulated environments. Due to the unavailability of a real articulated vehicle, real-world experiments were conducted using a Husky robot. For simulated experiments, we used ROS2 Humble and tested both a car-like vehicle and an articulated truck in the GAZEBO simulator. Table V lists simulation parameters common to all tests. The motion primitives necessary for the lattice-based approach were generated offline using the model in (2) with a resolution of $1m$. Both vehicles

[3]Curvature refers to the degree of turning required for the vehicle to align with the path.

TABLE VI
MAPPING FROM THE CONTEXTS SET $\mathcal{C}$ TO THE GLOBAL PLANNER SET $\mathcal{P}$, THE CONTROL STRATEGY SET $\mathcal{L}$ AND THEIR MAX VALUES

Context	Planner	Controller	Params
Open Space (Spatial)	Lattice	Spline	-
Obstacle Left/Right (Spatial)	Lattice	Spline	-
Corridor (Spatial)	A*	Spline	-
Snow, Fog, Rain (Weather)	-	-	$V, A = 3, 1$
Sun (Weather)	-	-	$V, A = 7, 2$
White (Semantic)	-	-	$V, A = 7, 2$
Grey (Semantic)	-	-	$V, A = 4, 1$

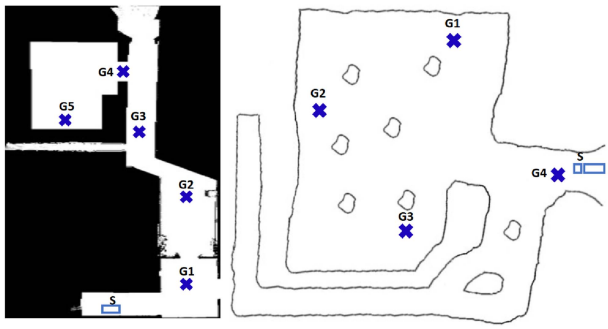

Fig. 6. Office and mine scenarios used to evaluate the solution quality (Test 2).

are equipped with an Intel Realsense camera and an Ouster OS1 lidar sensor. The scenarios were created using examples of real-world maps from industrial partners and the university campus[2]. Table VI describes the mapping from $\mathcal{C}$ to $\mathcal{P}$ and $\mathcal{L}$. The selection of the two global planners is based on the authors' experience and considerations of articulated vehicles' requirements (see Section IV-D). The framework allows flexibility in choosing planners, enabling users to customize the planner list and associate the planners with specific selection criteria. The proposed system[2] is implemented in C++ and runs on an Intel Core i9-10885H CPU 2.40 GHz $\times$ 16 processor with 32 GiB of memory. Selected moments during all experiments are shown in the video attachment.

VI. DISCUSSION

A. Weather Identification

Fig. 7 showcases moments from both a simulated experiment using an articulated vehicle in a small city map and a real-world experiment involving a vehicle in a mine. In the simulation (see Fig. 7(Bottom)), the articulated vehicle was subjected to different weather scenarios to test the robustness and responsiveness of the weather detection system. Similarly, the real vehicle was deployed in real-world conditions to validate the system's performance in real-world applications (see Fig. 7(Top)). Results confirm that the system can efficiently detect various weather conditions and make real-time adjustments to the navigation parameters, thereby improving the overall robustness and safety of the autonomous navigation system. For instance, upon detecting adverse weather such as snow, fog, or rain, the system automatically decreases the maximum velocity of the vehicle. This capability ensures that the robot can maintain optimal performance while minimizing risks associated with adverse weather conditions.

Fig. 7. Snapshots of the weather identification during real **(Top)** and simulated **(Bottom)** experiments.

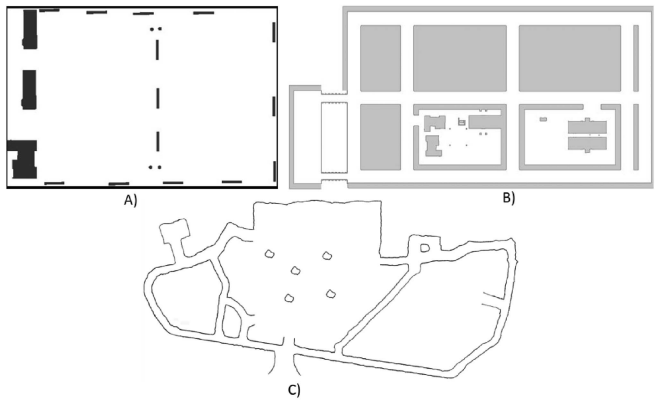

Fig. 8. Maps used to evaluate the navigation ability (Test 1). A) Field, B) Small City, and C) Mine.

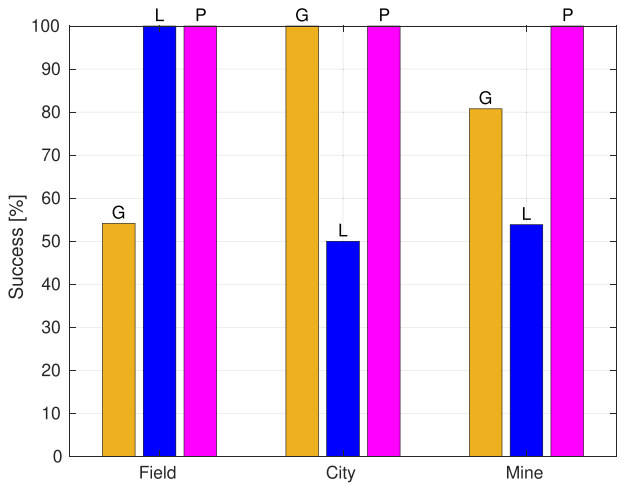

Fig. 9. Success rates of the navigation task across 30 instances for the proposed system (P), NAV2 grid-based framework (G), and NAV2 lattice-based framework (L).

B. Test 1: Navigation Ability

The navigation ability of each approach $\{\mathrm{P}, \mathrm{L}, \mathrm{G}\}$ is evaluated in three benchmark scenarios shown in Fig. 8. For each navigation task, the vehicle is required to reach a set of positions located in various regions of the map. The outcomes

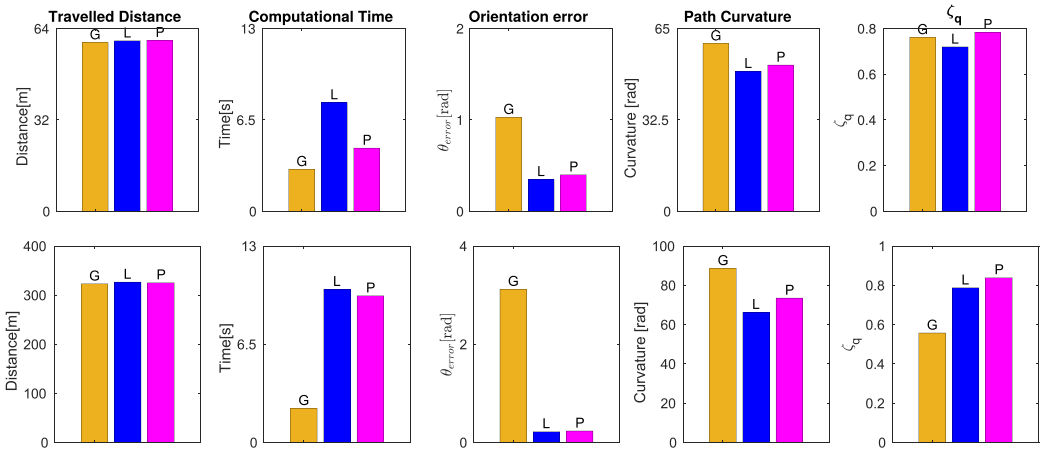

Fig. 10. Test 2: Solution quality for navigation using the NAV2 grid-based framework (G), the NAV2 lattice framework (L), and the proposed framework (P) in the office **(Top)** and mine **(Bottom)** scenarios.

are detailed in Fig. 9 and validate the efficacy of our proposed approach. Indeed, the proposed system (P) achieves success across all scenarios, deftly adapting its planning and control strategies in response to real-time map data and the failure of a specific planner. In contrast, both the single NAV2 navigation frameworks (G and L) encountered failures in some navigation tasks, highlighting the limitations of relying on a single planner in complex scenarios.

C. Test 2: Solution Quality

We evaluated the capability of the proposed system to autonomously choose the most suitable planner and control strategy based on the current data for both a car-like vehicle and an articulate vehicle in scenario involving a change in spatial context (see Fig. 6). Results shown in Fig. 10 confirm our expectations: the system selects the planner that produces the optimal path in terms of path curvature and with minimal computational time. Indeed, it navigates in open space using the lattice planner while it uses A* to navigate into narrow spaces. For the office scenario (Fig. 10(Top)), the A* algorithm and the lattice planner produce solutions with similar path curvature. However, the A* approach slightly outperforms the lattice approach regarding computation time (Fig. 10 (Top)), but results in a higher final orientation error. In this case, the proposed system uses the lattice planner to reach the corridor, switches to A* for navigating through the corridor to the next room, and then switches back to the lattice planner to navigate within the room. This strategy generates an optimal path that combines the computational efficiency of A* with the orientation accuracy and path curvature characteristics of the lattice approach.

Results show that for the mine scenario (Fig. 10 (Bottom)), navigating using A* demands a higher degree of curvature, resulting in difficulties in achieving the correct final orientation for each goal. Conversely, employing a lattice-based motion planner reduces the required curvature substantially, leading to minimal orientation errors. However, this advantage comes at the cost of increased computation time. In this scenario, the proposed system mostly utilizes a lattice-based approach for navigation. It switches to a grid-based approach when exiting and entering the parking area. This strategy results in a traversed trajectory that closely resembles one computed using a single lattice approach, albeit with slightly reduced computational time.

D. Additional Experiments

We finally demonstrate the capability of the proposed system to autonomously select and navigate in partially known maps using the benchmark scenario shown in Fig. 6. However, in this instance, the complete map is intentionally withheld from the planner. In such situations, preselecting an offline planner can lead to suboptimal or impractical outcomes, emphasizing the significance of adaptable planning strategies for the long-term navigation of autonomous vehicles.

To validate the system's ability to switch planning and control strategies based on map and sensor data in real scenarios, we conducted real-world experiments using an Husky robot. The scenario involved transitions between open spaces and narrow corridors. The results demonstrate that the proposed system can accurately detect the local map scenario and switch to the optimal planner and controller strategies. These experiments are documented in the attached video but are omitted here for brevity.

VII. Conclusion and Future Work

We presented a context-aware autonomous navigation framework for autonomous ground vehicles in complex environments and with different spatial contexts and weather conditions. The proposed approach integrates autonomous and real-time decision-making processes using behavior trees to select the optimal planner, control strategies, and navigation parameters. Selections are based on a combination of desired criteria (e.g., distance minimization), data sourced from the global map and sensors, and information about weather conditions. The experiments demonstrate the capability of the proposed system to select and switch navigation strategies based on real-time data in both simulation and real-world applications. The proposed approach exhibits three primary capabilities. Firstly, it can find a path if any implemented global path algorithms can. Secondly, it computes a path by balancing optimization and computational effort. Finally, it can achieve smoother navigation by reducing path curvature. For future work, we are working on integrating additional context related to human presence and terrain conditions. Additionally, we plan to introduce machine learning approaches to simplify the process of planner specification in certain areas of the map. Then, we plan to

deploy the proposed framework in a construction site application, which involves multiple heterogeneous vehicles and a dynamic environment influenced by the actions performed by the vehicles.

REFERENCES

[1] M. Cirillo, T. Uras, and S. Koenig, "A lattice-based approach to multi-robot motion planning for non-holonomic vehicles," in *Proc. 2014 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 232–239.

[2] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[3] J.-P. Laumond, P. E. Jacobs, M. Taïx, and R. M. Murray, "A motion planner for nonholonomic mobile robots," *IEEE Trans. Robot. Automat.*, vol. 10, no. 5, pp. 577–593, Oct. 1994.

[4] B. Hong and X. Ma, "Path planning for wheel loaders: A discrete optimization approach," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst.*, 2017, pp. 1–6.

[5] R. Kala, A. Shukla, and R. Tiwari, "Robotic path planning in static environment using hierarchical multi-neuron heuristic search and probability based fitness," *Neurocomputing*, vol. 74, pp. 2314–2335, 2011.

[6] V. Narayanan, M. Phillips, and M. Likhachev, "Anytime safe interval path planning for dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 4708–4715.

[7] S. Petti and T. Fraichard, "Safe navigation of a car-like robot in a dynamic environment," in *Proc. Eur. Conf. Mobile Robots*, 2005.

[8] R. Hazra, A. Sygkounas, A. Persson, A. Loutfi, and P. Z. D. Martires, "REvolve: Reward evolution with large language models for autonomous driving," 2024, *arXiv:2406.01309*.

[9] S. Sivashangaran and M. Zheng, "Intelligent autonomous navigagtion car-like unmanned ground vehicle via deep reinforcement learn," *IFAC-LettersOnLine*, vol. 54, no. 20, pp. 218–225, 2021.

[10] U. Larsson, C. Zell, K. Hyyppä, and Å. Wernersson, "Navigating an articulated vehicle and reversing with a trailer," in *Proc. 1994 IEEE Int. Conf. Robot. Automat.*, 1994, pp. 2398–2404.

[11] B. Alshaer, T. Darabseh, and M. Alhanouti, "Path planning, modeling and simulation of an autonomous articulated heavy construction machine performing a loading cycle," *Appl. Math. Modelling*, vol. 37, pp. 5315–5325, 2013.

[12] S. Sarata, N. Koyachi, T. Tubouchi, H. Osumi, M. Kurisu, and K. Sugawara, "Development of autonomous system for loading operation by wheel loader," in *Proc. 23rd Int. Symp. Automat. Robot. Construction*, 2006, pp. 466–471.

[13] T. Takei, K. Ichikawa, K. Okawa, S. Sarata, T. Tsubouchi, and A. Torige, "Path planning of wheel loader type robot for scooping and loading operation by genetic algorithm," in *Proc. 13th Int. Conf. Control, Automat. Syst.*, 2013, pp. 1494–1499.

[14] M. Niemeyer et al., "Towards context-aware navigation for long-term autonomy in agricultural environments," in *Proc. 12th IROS Workshop Plan., Percep., Navigation Intell. Veh.*, 2020.

[15] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J.-A. Fernández-Madrigal, and J. González, "Multi-hierarchical semantic maps for mobile robotics," in *Proc. 20005 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2005, pp. 2278–2283.

[16] C. Theobalt et al., "Talking to Godot: Dialogue with a mobile robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2002, pp. 1338–1343.

[17] X. Qi et al., "Building semantic grid maps for domestic robot navigation," *Int. J. Adv. Robot. Syst.*, vol. 17, 2020, Art. no. 172988141990006.

[18] M. Colledanchise and P. Ögren, "Behavior trees in robotics and AI: An introduction," Jul. 2018.

[19] M. Colledanchise and P. Ögren, "How behavior trees modularize robustness and safety in hybrid systems," in *Proc. 2014 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 1482–1488.

[20] K. Y. W. Scheper, S. Tijmons, C. C. de Visser, and G. C. de Croon, "Behavior trees for evolutionary robotics," *Artif. Life*, vol. 22, pp. 23–48, 2014.

[21] S. Macenski, F. Martin, R. White, and J. Ginés Clavero, "The marathon 2: A navigation system," in *Proc. 2020 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 2718–2725.

[22] S. Dergachev, K. Muravyev, and K. S. Yakovlev, "2.5D mapping, pathfinding and path following for navigation of a differential drive robot in uneven terrain," *IFAC-LettersOnLine*, vol. 55, pp. 80–85, 2022.

[23] J. Zhang et al., "NaVid: Video-based VLM plans the next step for vision-and-language navigation," *Robot. Sci. Syst.*, 2024.

[24] C. Altafini, "A path-tracking criterion for an LHD articulated vehicle," *Int. J. Robot. Res.*, vol. 18, pp. 435–441, 1999.

[25] H. ye Zhang, W. ming Lin, and A. xia Chen, "Path planning for the mobile robot: A. review," *Symmetry*, vol. 10, 2018, Art. no. 450.

[26] M. Arzamendia, D. Gregor, D. Gutiérrez-Reina, and S. L. T. Marín, "An evolutionary approach to constrained path planning of an autonomous surface vehicle for maximizing the covered area of Ypacarai Lake," *Soft Comput.*, vol. 23, pp. 1723–1734, 2019.

[27] S. Thrun et al., "A system for volumetric robotic mapping of abandoned mines," in *Proc. 2003 IEEE Int. Conf. Robot. Automat.*, 2003, vol. 3, pp. 4270–4275.

[28] S. M. LaValle, "Rapidly-exploring random trees : A new tool for path planning," The Annual Research Report, Tech. Rep. 98-11, 1998.

[29] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *J. Field Robot.*, vol. 26, no. 3, pp. 308–333, Mar., 2009.

[30] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. TSSC-4, no. 2, pp. 100–107, Jul. 1968.

[31] W. Wang, Y. Li, X. Xu, and S. X. Yang, "An adaptive roadmap guided multi-RRTs strategy for single query path planning," in *Proc. 2010 IEEE Int. Conf. Robot. Automat.*, 2010, pp. 2871–2876.

[32] S. Li and N. T. Dantam, "Sample-driven connectivity learning for motion planning in narrow passages," in *Proc. 2023 IEEE Int. Conf. Robot. Automat.*, 2023, pp. 5681–5687.

[33] J.-P. Tarel, N. Hautiere, A. Cord, D. Gruyer, and H. Halmaoui, "Improved visibility of road scene images under heterogeneous fog," in *Proc. 2010 IEEE Intell. Vehicles Symp.*, 2010, pp. 478–485.

[34] T. Wang, X. Yang, K. Xu, S. Chen, Q. Zhang, and R. W. Lau, "Spatial attentive single-image deraining with a high quality real rain dataset," in *Proc. 2019 IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 12262–12271.

[35] M. A. Kenk and M. Hassaballah, "DAWN: Vehicle detection in adverse weather nature," 2020.

[36] C. H. Bahnsen and T. B. Moeslund, "Rain removal in traffic surveillance: Does it matter?," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 8, pp. 2802–2819, Aug. 2019.

[37] B. Zhao, X. Li, X. Lu, and Z. Wang, "A CNN–RNN architecture for multi-label weather recognition," *Neurocomputing*, vol. 322, pp. 47–57, 2018.

[38] H. Xiao, F. Zhang, Z. Shen, K. Wu, and J. Zhang, "Classification of weather phenomenon from images by using deep convolutional neural network," *Earth Space Sci.*, vol. 8, 2021, Art. no. e2020EA001604.