Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX; date of current version XX Month, XXXX.

Digital Object Identifier 10.1109/OJITS.2022.1234567

Managing Risk in the Design of Modular Systems for an Autonomous Shuttle

THOMAS DRAGE*, KIERAN QUIRKE-BROWN*, LEMAR HADDAD*, ZHIHUI LAI*, KAI LI LIM†, AND THOMAS BRÄUNL*, SENIOR MEMBER, IEEE

[1]The REV Project, The University of Western Australia, Perth, WA 6009 Australia
[2]Dow Centre for Sustainable Engineering Innovation, The University of Queensland, Brisbane, QLD 4072 Australia
[3]Institute of Transportation Studies, UC Davis, Davis, CA 95616 USA

CORRESPONDING AUTHOR: Kai Li Lim (e-mail: kailim@ucdavis.edu).

ABSTRACT This paper presents an analysis and implementation of a robust autonomous driving system for an electric passenger shuttle in shared spaces. We present results of a risk assessment for our vehicle scenario and develop a flexible architecture that integrates safety features and optimises open-source software, facilitating research and operational functionality. Identifying the Robot Operating System (ROS) framework's limitations, we incorporate our own control measures for autonomous, unsupervised operation with enhanced intelligence. The study emphasises algorithm selection based on application requirements to ensure optimal performance. We discuss system improvements, such as monitoring node implementation and localisation algorithm selection. Future work should explore transitioning to a real-time operating system (RTOS) and establishing standardised software engineering practices for consistent reliability. Our findings contribute to effective autonomous shuttle systems in shared spaces, promoting safer and more reliable transportation solutions.

INDEX TERMS autonomous vehicles, safety systems, functional safety, autonomous shuttle bus

I. INTRODUCTION

THIS paper presents the ongoing research on electric vehicles, vehicle automation, and autonomous driving systems at The Renewable Energy Vehicle (REV) Project, The University of Western Australia (UWA). Our current endeavour involves the development of a highly automated electric passenger shuttle bus for use as a self-driving people mover and last-mile transport solution on the university campus. This bus is designed to adapt and plan its route dynamically, with all sensory and navigation processing on-board, eliminating the need for external communication systems or remote servers. To achieve this, we have integrated and enhanced both hardware and software components to ensure reliable and safe operation [1].

Mass produced passenger cars have incorporated levels of automation since the introduction of adaptive cruise control in the 1990s and advances in cost of sensors and computational abilities have driven widespread adoption of advanced driver assistance systems (ADAS) in the last decade. Indeed, such features have tended towards being labelled as autonomous driving however they all require the presence of an alert supervising human driver and are incapable of executing a high level journey goal [2]. The Society of Automotive Engineers (SAE) defines "Levels of Driving Automation", from zero to five, with the lower three being considered driver support and the upper automated driving [17]. The Autopilot and Full Self-Driving offerings from American car manufacturer Tesla achieve just SAE Level 2, whilst Level 4/5 systems, which will lead the way to making human drivers redundant may become publicly available to the market in the late 2020s [3].

One significantly desirable aspect of automation of the road transport system is the improvement in safety by means of eliminating the opportunity for human error which results from a variety of factors including fatigue, intoxication, distraction and skill [4]. The basis for the presumption that an autonomous car is safer is consistency of performance and unwavering compliance with rules. However, in reality the dynamically changing nature of public roads, for example by unexpected ingress of a pedestrian to the roadway or the bursting of a tyre on another vehicle, means it is not possible to avoid all incidents and making an automated

decision which is minimises harm is legally, ethically and practically complex [5]. Indeed, we must instead utilise layers of automation which not only perform their objective function but seek to minimise or control risk to the vehicle and other traffic [6].

The development of Level 3+ Autonomous Driving Systems (ADS) poses significant risks to both people and infrastructure due to the need for intricate software-driven electromechanical systems to ensure safe driving behaviour in complex and ever-changing environments [7]. Although autonomous vehicles are often touted as having the potential to reduce collision rates and improve traffic safety, current technology has yet to deliver on these promises [8]. At a more basic level the automation itself must be sufficiently reliable that it doesn't make an error under reasonably expected conditions or fail to operate during navigation in a way which in fact causes an accident or near miss. This requires research and modelling of failure modes in sensors [9] and algorithms, particularly those involving machine learning methods [10]. Data collected from various autonomous driving trials has shown that this has not yet been achieved and an autonomous "driver" is many times more dangerous than the average human [11]. Thus significant progress is needed in the areas of safety and reliability, particularly in relation to vehicle automation systems and human intervention when necessary.

Historically, the automotive industry has relied on the ISO 26262 standard for ensuring the functional safety of electronic and software systems in vehicles [12]. However, it is known that this standard does not extend in scope to fully satisfy the design requirements for highly autonomou vehicles due to the changed potential impacts of a component failure and the limited availability or applicability of human intervention as a control [13]. This standard, derived from IEC 61508 [14], has recently been supplemented by ISO/PAS 21448:2019, which provides guidelines for the design and validation of Level 1 and 2 Advanced Driver Assistance Systems (ADAS) and ADS, with scope for application to higher levels of automation [15]. The latter emphasises not only hardware failure, but also functional insufficiencies and reasonably foreseeable misuse of ADS. International regulations have also been published recently, including UN-R157 [16] on the basis of which Mercedes-Benz has been granted approval for their now commercially available SAE Level 3 DRIVE PILOT system [17]. Mercedes-Benz publicise their application of standards [18] including ISO 26262 and ISO/PAS 21448 to facilitate the safety assessment of their self driving platform, as do other car manufacturers such as Ford [19] who seek to develop autonomous driving systems (ADS) which are able to be validated and approved when their products are ready for release to the consumer market.

In this paper, we describe a comprehensive system architecture based on fail-safe design principles, drawing from established industrial practices which target compliance with e.g. IEC 61508/61511. Our system includes control and safeguarding hardware and software components, and utilises the Robot Operating System (ROS) middleware for automation and AI implementation [20], [21]. We also discuss measures taken to ensure that our system meets the required reliability targets.

The main contributions of this paper are:

- Development of a flexible and modular system architecture for an autonomous shuttle bus that integrates safety features and optimises open-source software. This facilitates research while ensuring reliable and safe operation.
- Implementation of additional control measures within the ROS framework to enhance system robustness and meet safety and reliability criteria for autonomous operation without a human supervisor. This includes modules like a monitoring node for system health checks.
- Selection, tuning and improvement of open-source ROS packages for key functions like navigation, localisation and path planning to optimise performance and reliability. This involved testing and comparing different algorithms.
- Analysis of system failures and reliability before and after improvements to the architecture. Quantitative metrics like mean time between failures were used to evaluate contributions to system robustness. Areas optimised included system initialisation, localisation, and fault tolerance.

II. BACKGROUND

The REV Project acquired a pre-existing electric shuttle bus equipped with drive-by-wire hardware, but lacking any software. To repurpose this vehicle, a new compute node (Jetson AGX Xavier) and additional sensors (SBG Ellipse-D RTK-GNSS with IMU) were installed alongside the existing eight LIDARs (4 SICK, 2 Velodyne and 2 ibeo Lux). A new hardware subsystem was implemented to send drive commands to the shuttle's motor controllers. We opted for the Linux operating system and ROS robotics framework [22], selecting and modifying various ROS packages to enable autonomous driving on the university campus (Fig. 1). This autonomous shuttle bus project builds on our previous work developing an autonomous Formula-SAE car capable of detecting and navigating a racecourse marked by traffic cones [23] (Fig. 2).

It is important to note that our current focus is on developing a shuttle bus for use in pedestrian areas. Our target is to not require a human monitor to achieve the required risk reduction; allowing fully autonomous operation. Instead, we identify risks associated with component or system failure and implement appropriate safeguards, such as emergency stop procedures. However, further research is needed to validate this approach in more complex environments, such as road traffic systems. Moreover, we also consider and address potential accidents resulting from the application of

FIGURE 1: The nUWAy autonomous shuttle bus, designed for on-campus use.

FIGURE 2: The REV Formula-SAE Autonomous vehicle, a precursor to the nUWAy autonomous shuttle bus project.

safety measures themselves, which are analogous to cases of slow driver reaction to disengagement during high-speed freeway driving.

Our research aims to contribute to the advancement of autonomous vehicle technology by improving safety and reliability. By integrating state-of-the-art hardware and software components, we strive to develop a robust, fail-safe system architecture that meets industry standards while providing a practical solution for last-mile transportation needs on university campuses.

III. PROBLEM DESCRIPTION

The design of a systems architecture for special-purpose autonomous research vehicles which operate in a shared space presents unique challenges. Firstly, the vehicle must achieve its desired operational targets safely and reliably. Secondly, the development process must be accessible to multiple researchers and re-usable for different applications of specific deployments of such low-volume systems. REV has developed multiple such systems, which have supported research for over ten years, with evolution enabled by the increasing availability of compact, high-performance computing hardware [20], [24], [25]. As ROS has evolved as the de-facto standard for such projects [26], we leverage the ecosystem, but in doing so, introduce potential safety and reliability problems managed by the techniques described in this paper.

ROS [22] is a convenient middle-ware framework for robot and autonomous vehicle development. It provides a publish/subscribe type messaging system with many packages, drivers and additional fast prototyping and development tools. These tools provide an excellent avenue to produce research as one can focus on a single aspect without concern for other areas. However, this system has several drawbacks that make it less reliable for real-world applications, particularly for non-technical users. Being a loosely coupled framework gives rise to a significant scope of outcomes in terms of software quality, particularly as projects become complex. Thus it becomes necessary to enforce architectural guidelines [27] or add additional software to improve robustness.

ROS provides a software framework that uses nodes to run different aspects of the navigation stack. These nodes communicate over the ROS backbone, allowing the user to focus on their primary area of research. However, ROS packages do not implement any reliability standards. In addition, many of the open-source ROS packages can cause unexpected errors when applied in different scenarios. One major issue with ROS is its typical process node starting sequence; in larger systems, manually starting up each node would be too time-consuming, so launch files are provided for an automated system start. These launch files will start each node in the file with the given parameters; however, little attention is given to the order in which nodes are started up and whether or not those nodes are started correctly. A common system failure is due to critical nodes starting out of sequence.

Another concern for ROS is the lack of system health monitoring. Nodes may fail silently in the background leading to unusual and often undesirable behaviour. For instance, the waypoint manager node may terminate abnormally, which leaves the system stranded, as no further waypoints will be added to a path. Nodes that fail silently may have other system nodes that depend on them. This failure can lead to a knock-on effect of node failures due to missing dependencies. A system designed without fault tolerance will not be able to restart or return to service once a failure has occurred.

In critical systems, such as driving algorithms for autonomous vehicles, these errors can result in catastrophic failures, from unintended property damage to injury and loss of life. Therefore we have analysed the ROS based system and implemented methods that make the system as a whole more robust and reliable. The scope includes correct configuration of the required nodes, detection of errors or stopped nodes and reporting of faults to the users.

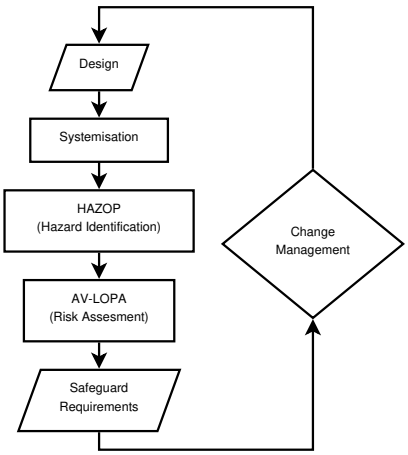


FIGURE 3: HARA process flow.

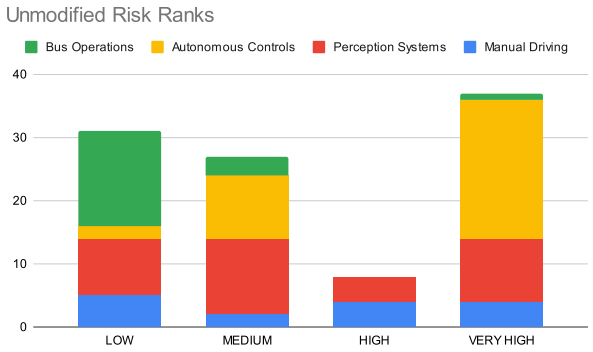


FIGURE 4: Count of hazards by risk rank.

IV. HAZARD AND RISK ANALYSIS

For risk assessment of our autonomous driving projects, we utilise a three-step process shown in Fig. 3, consisting of vehicle and ADS systemisation; then, HAZOP-style hazard identification followed by a semi-quantitative risk assessment (AV-LOPA) which is used to drive specification of safeguarding requirements. Due to the experimental nature of our vehicle systems and iterative design approach, this process is repeated as features and modifications occur, including the additions of safeguards themselves. These activities are performed as a cross-functional team in a workshop environment to provide breadths of experience and knowledge, particularly during hazard identification. Systematic application minimises the failure to identify hazards created by new functionality. This paper presents the results of the methodology described in [1] as applied to the nUWAy shuttle bus and extends to the methods used to achieve the safety requirements through the automation system architecture and software framework design.

A series of five risk assessment workshops of approximately 2.5 hours each were held for the nUWAy shuttle bus, examining four scenarios:

1) Manual (human) driving of the shuttle
2) Instrumented perception systems
3) Autonomous driving controls
4) Shuttle bus passenger operations

A total of 20 subsystems were examined across the scenarios and a total of 103 hazards risk assessed. Hazards were identified by the application of HAZOP guidewords defined in [1] and the University's corporate risk matrix was applied to calibrate the risk tolerance of the assessment. The outcomes are shown in Fig. 4, below, with the implication that all risk ranks must be reduced to the ranking *LOW* by application of appropriate safeguards.

Applying AV-LOPA per [1], we identified Independent Protection Layers (IPLs) of sufficient risk reduction to manage the hazards identified. These safeguards were defined to be:

1) **Independent**: it should not have failure modes (e.g. shared sensors) common to other safeguards used in the scenario.
2) **Effective**: each IPL must be able to fully mitigate the hazard.
3) **Validatable**: each IPL's functionality must be able to be assured and maintained.

In all cases except two, the risk was able to be reduced to *LOW*. The most commonly applied safeguard related to the nUWAy shuttle's safety LIDAR curtain system which utilises a safety PLC and four independent LIDAR sensors to prevent collision. However, in the case of 7 hazards of *VERY HIGH* risk ranking, the presence of an alert safety-driver, equipped with an emergency stop switch was required in order to rationalise the risk during autonomous driving operations. These were all related to perception system failure modes and required additional IPLs to be defined in order to eliminate the requirement for a human supervisor in the nUWAy shuttle.

Additionally, two cases of *MEDIUM* risk ranking were identified for which insufficient IPLs were available and related to a false emergency stop occurring whilst at speed. The ranking of *MEDIUM* was obtained due to the high likelihood of this occurring during testing and development of the research vehicle and must be managed by a reduction of likelihood through improvement of reliability of the vehicle's systems.

V. Safety and Reliability Requirements

In order to address the findings of Section IV, in particular for operation without an alert safety-driver we require to implement additional or alternative IPLs within the scope of our control and automation architecture - specifically within the high-level automation system. These IPLs are able to be independent by virtue of separate sensors (to e.g. the LIDAR safety curtain), a separate logic solver (a general purpose computer instead of a safety PLC) and separate outputs

(active driving controls and alert systems). However, the challenge is in ensuring that they are effective and able to be validated, given the non-deterministic, non-realtime nature of the ROS control framework employed. If an advanced safeguard is implemented within a ROS node, and required to act as an IPL for the purposes of AV-LOPA, we must guarantee that:

1) The vehicle cannot proceed under automatic control unless that node is active and fault-free.
2) That the node is continuously monitored for operation according to its intended functionality and appropriate actions, which are not unsafe in themselves, are taken on detection of failure.
3) That monitoring functions cannot be disabled and have full coverage of the faults and errors which could occur within the target node.
4) That the node is sufficiently reliable that it performs its function without introducing further risk.
5) Testing is performed to assess the performance and correctness of any algorithms employed.

To achieve this within ROS we must implement the monitoring and performance evaluation functionality in conjunction with implementing the safeguards themselves. We must also provide means to validate that the ROS middleware and the operating system itself remain functional at all times, or at least that failures are detected. Given that such advanced safeguards may have multiple or unexpected failure modes, the concept of proving their effectiveness through use is expected, which can be accomplished through simulation and field trials (IX). At the same time, we must consider that nodes involved in continuous control must not introduce additional risk than was additionally assumed in the HARA through unreliability and thus apply the same constraints to their implementation; the concepts of safety and reliability in general are inextricably linked in a continuously controlled intelligent system such as this.

VI. Control and Automation Architecture

The nUWAy shuttle bus system architecture follows that of the prior Formula-SAE vehicle project, modified to suit the architecture of the commercial vehicle. In addition to the ECUs used in the drive control system, the shuttle bus features an industrial safety PLC and a computer-based high-level navigation system connected to a single CAN network. The safety PLC provided by the shuttle manufacturer handles basic sequence-control of the vehicle (e.g., interlocking doors and brakes) and provides action for fault conditions and detections from the perimeter LIDARs, which implement a simple distance-based safety curtain. The safety PLC contains software, however it is a simple logic system and operates independently of any other computer-based systems in the shuttle bus.

The high-level automation systems in our shuttle bus comprise two main computers; an industrial x86 PC for handling IO and core navigational functionality, and an Nvidia Jetson AGX Xavier, which provides accelerated execution of deep learning based algorithms used for interpreting sensor data. Load is distributed across the two computers based on interfacing and computational requirements to ensure maximal reliability. Both computers implement the Linux operating system along with the ROS software framework. All functionality is contained within ROS nodes, including sensor interfacing, localisation, drive control and control of outputs.

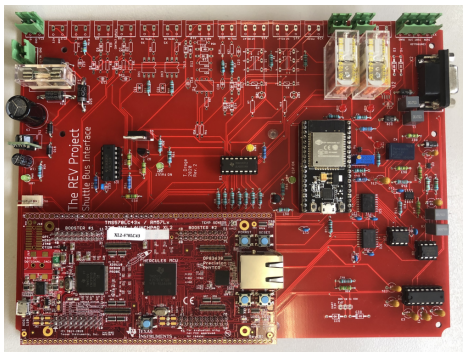


FIGURE 5: REV project nUWAy system interface controller.

As the vehicle's CAN bus specification is proprietary, for driving and steering we developed an interface system based on a TI Hercules automotive ARM microcontroller, certified for ASIL applications (Fig. 5). This development also allows us to implement additional drive-control and safety features as part of our automation system outside of the "black box" shuttle bus ECU. Additionally, it provides a hard-wired interface to the safety systems, providing redundancy and the opportunity to embed independent fault detection monitoring for the high-level systems in reliable hardware. Critical interfaces are implemented using isolated buses with loopback circuits for constant error checking, ensuring that any abnormal condition will stop the vehicle. A driver for the interface system is implemented in ROS, allowing control of the vehicle's driving hardware systems and activation of the safety system.

The nUWAy shuttle bus implements several layers of safety systems shown in Fig. 6 designed to ensure that fully-autonomous operation in an environment shared with pedestrians is safe and efficient at all times. Our project has extended the safety functionality beyond low-level hazard mitigation systems and addresses the SOTIF and SOTAI regimes.

The electric shuttle bus features some built-in low-level safety features [28], which were assessed and credited as applicable in our HARA process, including:

1) **Obstacle detection**: Four single-beam LIDAR sensors form a detection area at a 30 cm height above ground level and secure a 2 m collision-free safety zone around

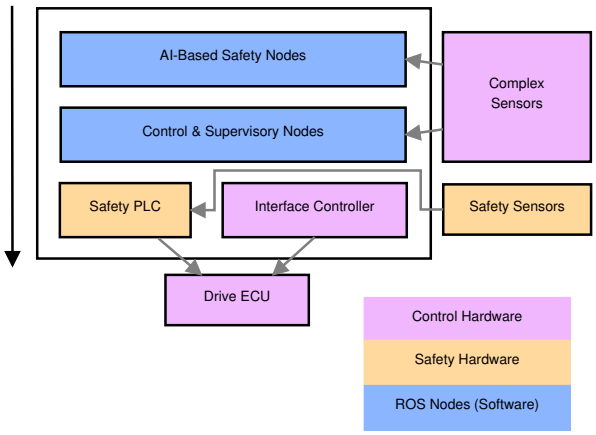


FIGURE 6: nUWAy Shuttle Multi-level Safety Architecture.

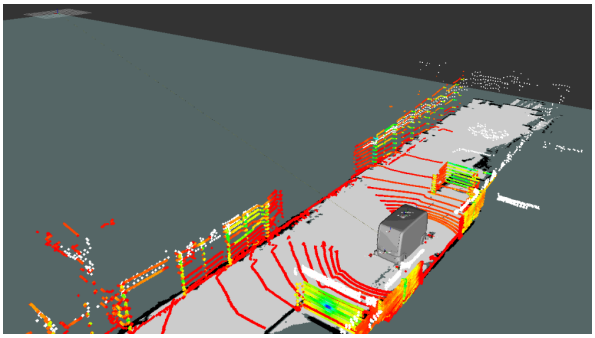


FIGURE 7: LIDAR point cloud of nUWAy shuttle.

the vehicle. (Fig. 7). However, this leaves the vehicle blind to obstacles lower than 30 cm.

2) **Emergency stops (e-stops)**: Pushing one of the four in-vehicle stop buttons will immediately stop the vehicle and disarm the doors.
3) **Redundant braking systems**: The vehicle features multiple braking methods, including a fail-safe emergency brake.

These functions are implemented using an industrial safety PLC, which operates independently of the navigation system and interconnects with the drive-control ECUs for speed detection etc. This approach provides robustness in this application and is well suited to the specialised vehicle, however additional safeguards must be implemented at a higher level by means of software within the autonomous driving control system.

VII. Application of Software Framework

The Robot Operating System (ROS) is a widely adopted platform for robotics and autonomous vehicle research projects. It is utilised across various computing platforms, from embedded controllers to multicore CPUs and GPUs, and in numerous autonomous vehicles such as robots, driverless cars, drones, autonomous boats, and robot manipulators. However, its adoption within the industry is limited. Major companies such as Argo AI, Audi, Volkswagen, and Ford do not utilise ROS [29]. NASA/JPL employs ROS for development projects and proof-of-concept builds but re-implements flight systems without ROS [30]. One exception is the startup Apex.AI, which is developing an OEM-independent automotive operating system based on ROS [28].

The central concern regarding ROS is its reliability and robustness, often deemed insufficient for safety-critical systems [31]. This project's experience revealed that individual ROS software nodes, such as LIDAR sensor nodes, are prone to crashing. Consequently, additional efforts are necessary to ensure the overall vehicle system remains in a safe state, including automated detection and recovery operations to prevent collisions due to sensor information loss. Alternatives to ROS include the DDS middleware system [32], VxWorks [33], and other real-time operating systems (RTOS). Despite the superior time and safety-critical performance of these alternatives, particularly RTOS, the modularity requirements of the framework led to the adoption of ROS, as it provides inherent modularity through its publish-subscribe architecture. It should be noted that ROS 1 lacks real-time capability, whereas ROS 2 incorporates some real-time features and more recent integration with commercial RTOS [23].

The IEEE Standard 1633-2016 defines software reliability (SR) as the probability that software will not cause a system failure for a specified time under specified conditions [34]. Unlike hardware, which is subject to wear and tear, software failures are typically due to design faults caused by human errors or oversight [35]. As such, software reliability is not affected by external conditions, nor can it be improved by running multiple instances of the same program for redundancy. Instead, design diversity, or the use of different codes performing the same task, can offer redundancy. Within our framework, the SOTAI regime, applicable to complex ROS nodes for autonomous navigation, is realised through the implementation of three mechanisms which are detailed in [1]:

1) Model Validation through Simulated Testing
2) Model Supervision through Diverse Architectures
3) Model Supervision through Hard-coded Constraints

VIII. Autonomous Driving Software Stack

The autonomous system was initially designed and deployed using ROS with a mixture of in-house and standard open-source packages. However, during initial testing, it was determined that more suitable packages were available in ROS 2, offering additional reliability features and the system was converted to the long-term support release of ROS 2. The following outlines the primary packages employed in the nUWAy shuttle bus:

1) **Nav2 stack** [36]: ROS 2 offers a plugin library that provides an array of resources for hot-swapping various solutions. Moreover, it establishes a system be-

haviour tree with associated recovery actions, typically used to ensure sufficient progress towards the goal. If adequate progress is not made, a fallback action, such as alternate global path planning, may be performed.

2) **Localisation and mapping**: A crucial aspect for the shuttle bus is the capacity to build and localise on a given map. For map building, there are a limited number of packages available, with the most prominent being the SLAM toolbox [37], which utilises scan matching to generate a smooth map. The Cartographer package [38] was briefly considered but was found to be only partially implemented in ROS 2. For localisation SLAM toolbox provides a service for loading and localising a generated "pose-graph" map, however AMCL was selected as it better handles larger scale maps. AMCL requires a separate map server, which is available in Nav2.
3) **Global planner**: Nav2 includes several "grid-based" global planners as plugins, which use either Djikstra's or the A* algorithm to compute a path. NavFn and Smac [36] were compared to evaluate potential benefits from feasibility-based planners.
4) **Local planner**: The local planner selection proved critical; Nav2 provides several local planner controllers that can be "hot-swapped". The nUWAy shuttle bus was tested with two primary packages: TEB (Time Elastic Band) [39] and Regulated Pure Pursuit [36], each with distinct advantages and disadvantages to reliability.

While selecting, tuning, and improving suitable open-source packages is crucial for creating a reliable driving system, the key objective of this project is to maximise reliability. This is achieved by establishing a monitor node that assesses the overall system's health and takes actions accordingly to ensure reliability. The monitor node's first aim is to initiate each node sequentially in the correct order, based on dependencies and resource requirements, to prevent node failure and overuse of system resources. This approach also eliminates race conditions in a distributed system, such as the one on the nUWAy shuttle bus, by enabling a single PC to monitor and control all nodes. On system startup, the first node is activated, and the monitor listens on the corresponding topic until node data is received, confirming successful node initiation and allowing the subsequent node to start.

Alternative system monitoring methods, such as heartbeats and watchdogs are implemented in the hardware systems and may be integrated between the monitoring node and other services in future, however, critical systems running under Nav2 are managed by a life cycle manager that handles many of these capabilities. The monitor's primary focus is on flexibility to ensure the project's scalability for future research. An external incident recorder has been developed, which activates when unexpected changes to the global path occur and records current and planned trajectories, event time, and camera data. This information feeds continuous improvement of the system.

Lastly, the monitor node establishes several monitoring sessions that captured critical information from the other nodes which is used to enhance fault tolerance. Node failures are determined by examining the current node list and listening for data on corresponding topics; allowing detection of faults in nodes that tend to fail silently. Once a failed node is detected, information about its location, failure type, and restoration methods is logged.

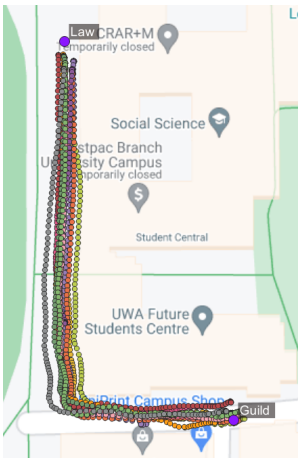


FIGURE 8: Multiple recorded shuttle drives between two campus stops.

IX. Evaluation of Autonomous Driving

The performance of the autonomous driving system is evaluated across their respective subsections.

A. Optimising nUWAy Shuttle Path Accuracy

Nine key stops across the breadth of the university campus are defined with routes between them experiencing high pedestrian traffic, especially during peak times. Groups of students further limit driving space. Fig.8 shows drives between the Law School and Student Guild stops. GPS accuracy and signal availability for Real-Time Kinematic (RTK) corrections cause slight deviations in the bus's starting position. The current system demonstrates consistent navigation using a basic path planning system.

During initial testing, unsuitable software components were replaced. Observations of student and pedestrian behaviour informed local planner selection. Initially, Navigation 2 (Nav2) implementation of Timed Elastic Band (TEB) was used for path planning due to its popularity. A performance review [40] determined that TEB and NavFn were superior in combination. However, TEB occasionally generated incoherent paths (Fig. 9a), requiring manual intervention.

Pedestrians seemed disoriented around the shuttle bus during TEB's operation due to abrupt direction changes. This

led to selecting an alternative local planning algorithm. The 'regulated pure pursuit' algorithm offers a simpler alternative to TEB, functioning like a carrot on a stick. It lacks dynamic obstacle avoidance but incorporates velocity scaling based on obstacle proximity. It improved the shuttle's route following ability, and pedestrians could anticipate its path.

Initial trials of the "regulated pure pursuit" algorithm showed promising results on straight paths with minimal interference. However, it struggled on corners (Fig. 9b). Two primary solutions include using a different global planner to address feasibility concerns or tuning inflation zones for safer cornering.

Previously, the NavFn global planner was used in simple outdoor environments, while TEB handled unexpected issues. However, NavFn does not consider path feasibility, which may result in poor performance. Consequently, the Smac planner provided by Nav2 was tested but proved unsuitable due to its continuous global path adjustments.

The focus shifted to adjusting inflation zones for reliable navigation along straight paths and safe cornering. Increasing the inflation zone larger than the shuttle bus size resolved the issue. Fig. 9c demonstrates the expanded inflation zone. The light blue regions represent impassable lethal objects and a decay value is set to gradually decrease the object's severity.

B. Disengagement Events Analysis

This section evaluates the autonomous driving system performance before implementing the monitor node. We used a black-box testing approach due to the codebase's complexity, which includes open-source packages and student contributions. We characterised the system's efficacy by the mean time between failures (MTBF) during routine operations, such as path planning and navigation. Failures were assessed by their frequency, severity, and whether a system restart was required. We identified four primary areas for optimisation to improve system reliability and performance:

1) Low-level motor driver communication
2) System initialisation
3) Vehicle localisation
4) Miscellaneous driving challenges

Over three weeks, we documented 686 system failures from 57 hours of data, resulting in an average MTBF of 4.98 minutes. Table 1 shows the failure categories before and after improvements.

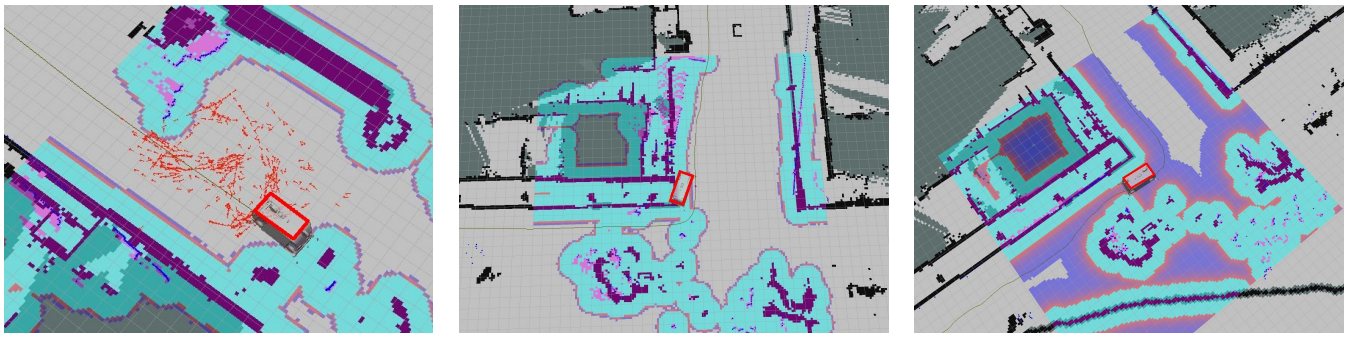
TABLE 1: Sources of failure before and after improvements.

Source	Before	After
Initialisation	15.6%	1.61%
Driving	22.45%	59.68%
Localisation	60.2%	35.48%
Low-level	1.75%	3.23%

Low-level failures can result from unsuccessful message exchanges between the interface board and PLC software. Initialisation failures occur when software nodes start, while driving failures relate to local planner software issues. Localisation failures arise when the system cannot load or accurately localise within a map. These categories were further classified by severity level: low, medium, or high. Low-severity failures can be resolved by non-technical users, while medium-severity failures need technical users to restart specific nodes. High-severity failures require a complete system restart or full power cycle of the shuttle bus.

Our focus was on minimising medium and high-severity failures in the nUWAy system, designed for operation by non-technical personnel. We addressed the most common issues in localisation and initialisation phases. Low-level communication failures were due to lost or delayed data packets. Enhancing load distribution and incorporating optimised data timeouts and fault recovery logic improved reliability, offering safety advantages.

A map of the shuttle's environment was created using the SLAM toolbox package, which provided a 2D map using localisation and LIDAR sensors. Despite the shuttle bus being equipped with high-quality GPS with RTK correction, a SLAM-based solution was found to be more suitable due



(a) Incoherent TEB paths (b) Corner cut by regulated pure pursuit (c) Improved by increased inflation zone

FIGURE 9: Issues encountered during local planner selection and tuning.

This article has been accepted for publication in IEEE Open Journal of Intelligent Transportation Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJITS.2024.3425165.

TABLE 2: Operations performed by SLAM Toolbox vs. AMCL.

	SLAM Toolbox		AMCL	
Operation	Failure	Success	Failure	Success
Map load (per hour)	5.727	2.009	0.098	3.610
Pose estimations (per hour)	1.004	2.132	0.293	6.244
MTBF (minutes)	8.24		55.91	

to drift caused by nearby buildings. ROS 2 offered two packages for this purpose: SLAM Toolbox and AMCL.

Initially, SLAM Toolbox was chosen but proved inadequate for our map sizes, causing service crashes during map loading. Table 2 compares map loading success rates using SLAM Toolbox and AMCL. SLAM Toolbox had a 25% success rate, often needing multiple software stack restarts. In contrast, AMCL was more reliable, with a 97% success rate and no performance issues due to smaller map sizes.

Table 2 shows SLAM Toolbox's poor localisation maintenance within a map, with an MTBF of 8.24 minutes. AMCL performed better, achieving an MTBF of nearly an hour. SLAM Toolbox failed to maintain localisation even with a smaller map.

C. Monitoring Node Implementation Improvements

Integrating the monitoring node into the ROS 2 framework significantly enhanced the nUWAy autonomous shuttle bus system performance. The startup issue with safety nodes launching before LIDAR drivers was resolved. A regulated startup sequence now ensures device drivers start and operate before activating dependent safety nodes.

TABLE 3: Comparison of failure severity levels pre- and post-monitor node implementation.

Severity	Pre-Implementation	Post-Implementation
Low	21.7%	80.7%
Medium	76.5%	16.1%
High	1.8%	3.2%

TABLE 4: MTBF comparison for severity levels and failure sources (in minutes).

Category	Level	Initial	Final
Severity	Low	23	25
	Medium	7	123
	High	284	615
Failure Source	Low-level	284	615
	Launch	32	1230
	Localisation	8	56
	Driving	22	33

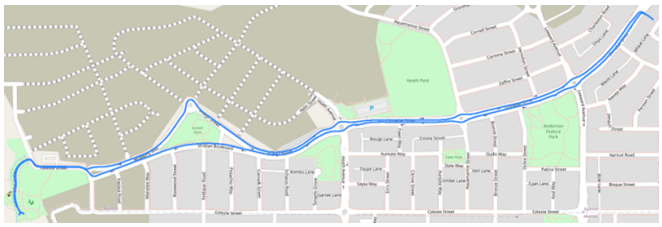


FIGURE 10: Eglinton shuttle bus route.

Table 3 compares failure severity levels before and after monitoring node implementation, showing a significant reduction in medium severity failures. Table 4 compares the MTBF for severity levels and failure categories, indicating improved system reliability and reduced frequency of high and medium severity consequences. Launch, localisation, and low-level failure probabilities decreased, leading to better overall system performance and stability primarily due to the monitoring node and better package selection. Hardware improvements are in progress to reduce low-level failures, including false positive prevention, watchdog implementation, and fault-tolerant recovery measures.

The monitoring node implementation also improved the overall systems ability to deal with failures and continue running after a failure has occurred. For example, during testing the GUI interface would often fail while driving; this was found to be an incompatibility between the package and long term release distribution of ROS 2. However, the modularised system allows the vehicle to continue to drive the provided path without triggering a disengagement during the failure. While the path is being driven the background watchdog will identify this failure and can fully recover the GUI functionality. Recent upgrades to the ROS software stack have reduced this issue however the occurrence highlights the importance of our implementation of additional supervisory functions within the ROS framework.

The next deployment of the nUWAy shuttle bus will take place on public roads in the suburb of Eglinton, driving route shown in Figure 10, in the north of Perth, Western Australia. This latest revision implements the updated ROS 2 distribution, Humble, and has been implemented using Docker containers to further modularise functions and segment failure points in both the building and running stages of the software. The watchdog architecture remains in use within the containers to ensure the system remains as safe and reliable as possible. Further work will extend our architecture to feature an on road system with two driving modes, which operate concurrently such that if one fails the other system can take control without disruption while the failed system is reset.

X. CONCLUSION

This study has presented a comprehensive analysis and implementation of an increasingly robust and dependable automation system for a passenger shuttle bus operating within

a shared space environment. Central to our investigation was the development of a flexible systems architecture, which allowed for the seamless integration of safety features with varying degrees of complexity, as well as the optimisation of open-source software in a modular manner to facilitate the advancement of both research and operational functionality. Hazard and risk assessment showed that further work is required to achieve fully autonomous operation without a human supervisor. We propose that this is achieved through the improvement of the reliability of the systems, in order to prevent dangerous conditions occurring and the provision of additional layers of protection within the system framework. There is significant scope to implement recent developments, particularly with regard to perception of hazards through advanced object detection and tracking systems e.g. [41] which are highly suitable in environments with multiple special purpose vehicles in operation. We ascertained that the existing ROS framework does not inherently provide the necessary mechanisms to guarantee the fulfilment of our safety and reliability criteria. Consequently, we have successfully devised and incorporated our own control measures to progressively work towards our ultimate objective of achieving a fully autonomous, unsupervised operation with enhanced intelligence embedded within the vehicle. Given the promising developments in ROS 2, we anticipate utilising the emerging features and mechanisms within this distribution to augment and refine the reliability of our systems. As we continue to advance our research, we recommend that future work should focus on evaluating the feasibility of transitioning to a RTOS and establishing standardised software engineering practices. Selection of algorithms should be guided by application and not by availability of functionality within a chosen framework to ensure the best performance. These measures will ensure that safety is maintained through consistent reliability as the project expands, further solidifying the effectiveness of our proposed automation system for passenger shuttle buses in shared spaces.

ACKNOWLEDGMENT

The REV team would like to acknowledge the support of all its sponsors, especially Stockland, Allkem, CD Dodd, and Dyflex.

REFERENCES

[1] T. Drage, K. L. Lim, J. E. H. Koh, D. Gregory, C. Brogle, and T. Braunl, "Integrated modular safety system design for intelligent autonomous vehicles," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, Jul. 2021.

[2] M. Murtaza, C.-T. Cheng, M. Fard, and J. Zeleznikow, "The importance of transparency in xxiinaming conventions, designs, and operations of safety features: from modern ADAS to fully autonomous driving functions," *AI & Society*, 2022.

[3] "Autonomous vehicle technology: A guide for policymakers," Society of Automotive Engineers, Warrendale, USA, Tech. Rep. SAE J3016_202104, 2021.

[4] J. M. Anderson *et al.*, "Autonomous vehicle technology: A guide for policymakers," RAND Corporation, Santa Monica, USA, Tech. Rep., 2014.

[5] J. Fleetwood, "Public health, ethics, and autonomous vehicles," *American Journal of Public Health*, vol. 107, no. 4, pp. 532–537, 2017.

[6] M. Geisslinger, R. Trauth, G. Kaljavesi, and M. Lienkamp, "Maximum acceptable risk as criterion for decision-making in autonomous vehicle trajectory planning," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 4, pp. 570–579, 2023.

[7] P. Feth, R. Adler, T. Fukuda, T. Ishigooka, S. Otsuka, D. Schneider, D. Uecker, and K. Yoshimura, "Multi-aspect safety engineering for highly automated driving," in *Developments in Language Theory*. Porto: Springer International Publishing, 2018, pp. 59–72.

[8] V. V. Dixit, S. Chand, and D. J. Nair, "Autonomous vehicles: Disengagements, accidents and reaction times," *PLOS ONE*, vol. 11, no. 12, p. e0168054, Dec. 2016.

[9] B. Schlager, T. Goelles, S. Muckenhuber, and D. Watzenig, "Contaminations on lidar sensor covers: Performance degradation including fault detection and modeling as potential applications," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 738–747, 2022.

[10] L. Hacker and J. Seewig, "Insufficiency-driven dnn error detection in the context of sotif on traffic sign recognition use case," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 4, pp. 58–70, 2023.

[11] S. S. Banerjee, S. Jha, J. Cyriac, Z. T. Kalbarczyk, and R. K. Iyer, "Hands off the wheel in autonomous vehicles?: A systems perspective on over a million miles of field data," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Jun. 2018.

[12] "Road vehicles — Functional safety," International Organization for Standardization, Geneva, CH, Standard, Aug. 2018.

[13] S. Salih and R. Olawoyin, "Fault injection in model-based system failure analysis of highly automated vehicles," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 2, pp. 417–428, 2021.

[14] "Functional safety of electrical/electronic/programmable electronic safety-related systems," IEC, Geneva, CH, Standard, Apr. 2018.

[15] "Road vehicles — Safety of the intended functionality," International Organization for Standardization, Geneva, CH, Standard, Jan. 2019.

[16] "Uniform provisions concerning the approval of vehicles with regard to automated lane keeping systems," United Nations Economic Commission for Europe, Geneva, CH, Standard UN Regulation No. 157, 2023.

[17] M. Minielly, A. Berg and C. Decker, "Mercedes-Benz world's first automotive company to certify SAE Level 3 system for U.S. market," *Business Wire*, Jan. 2023. [Online]. Available: https://www.proquest.com/wire-feeds/mercedes-benz-world-s-first-automotive-company/ docview/2769612413/se-2

[18] Mercedes-Benz, "Introducing DRIVE PILOT: An Automated Driving System for the Highway," Mar. 2023. [Online]. Available: https://group.mercedes-benz.com/dokumente/innovation/sonstiges/2023-03-06-vssa-mercedes- benz-drive-pilot.pdf

[19] Ford Motor Company, "A matter of trust 2.0 – Ford's approach to developing self-driving vehicles," Jun. 2021. [Online]. Available: https://media.ford.com/content/dam/fordmedia/North America/US/2021/06/17/ford- safety-report.pdf

[20] K. L. Lim, T. Drage, C. Zhang, C. Brogle, W. W. L. Lai, T. Kelliher, M. Adina-Zada, and T. Braunl, "Evolution of a reliable and extensible high-level control system for an autonomous car," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 3, pp. 396–405, Sep. 2019.

[21] X. Larrucea, P. González-Nalda, I. Etxeberria-Agiriano, M. C. Otero, and I. Calvo, "Analyzing a ROS based architecture for its cross reuse in ISO26262 settings," in *Communications in Computer and Information Science*. Cham: Springer International Publishing, 2018, pp. 167–180.

[22] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, Kobe, Japan, 2009, p. 5.

[23] ROS Tutorial, "Real-time programming in ROS 2," May 2022. [Online]. Available: https://docs.ros.org/en/foxy/Tutorials/Real-Time-Programming.html

[24] K. L. Lim, T. Drage, R. Podolski, G. Meyer-Lee, S. Evans-Thompson, J. Y.-T. Lin, G. Channon, M. Poole, and T. Braunl, "A modular software framework for autonomous vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2018.

[25] T. H. Drage, "Development of a Navigation Control System for an Autonomous Formula SAE-Electric Race Car," Master's thesis,

The University of Western Australia, 2013. [Online]. Available: http://robotics.ee.uwa.edu.au/theses/2013-REV-Navigation-Drage.pdf
[26] A. Araujo, D. Portugal, M. S. Couceiro, and R. P. Rocha, "Integrating arduino-based educational mobile robots in ROS," in *2013 13th International Conference on Autonomous Robot Systems*, Apr. 2013.
[27] I. Malavolta, G. A. Lewis, B. Schmerl, P. Lago, and D. Garlan, "Mining guidelines for architecting robotics software," *Journal of Systems and Software*, vol. 178, p. 110969, Aug. 2021.
[28] J. Becker, "Das neue auto-os und die robotik," Interview, May 2022. [Online]. Available: https://intellicar.de/podcast/das-neue-auto-os-und-die-robotik/
[29] A. Boeing, Private Communication, Munich, May 2022.
[30] I. Nesnas, Private Communication, Pasadena, Jan. 2016.
[31] ROS Answers, "What are technical reasons for criticisms of ROS's Reliability/Robustness/Safety?" May 2022. [Online]. Available: https://answers.ros.org/question/317435/what-are-technical-reasons-for-criticisms-of-ross-reliabilityrobustnesssafety/
[32] DDS Foundation, "What is DDS?" May 2022. [Online]. Available: https://www.dds-foundation.org/what-is-dds-3/
[33] Wind River Systems, "VxWorks - The Leading RTOS for the Intelligent Edge," May 2022. [Online]. Available: https://www.windriver.com/products/vxworks
[34] "IEEE recommended practice on software reliability," Standard IEEE Std 1633-2016 (Revision of IEEE Std 1633-2008), 2017.
[35] M. R. Lyu, *Handbook of software reliability engineering*. Piscataway, NJ: IEEE Computer Society Press, 1996.
[36] S. Macenski, F. Martin, R. White, and J. G. Clavero, "The marathon 2: A navigation system," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020.
[37] S. Macenski and I. Jambrecic, "Slam toolbox: Slam for the dynamic world," *Journal of Open Source Software*, vol. 6, no. 61, p. 2783, 2021.
[38] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016.
[39] C. Roesmann, W. Feiten, T. Woesch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *ROBOTIK 2012; 7th German Conference on Robotics*, 2012, pp. 1–6.
[40] A. Filotheou, E. Tsardoulias, A. Dimitriou, A. Symeonidis, and L. Petrou, "Quantitative and qualitative evaluation of ROS-enabled local and global planners in 2d static environments," *Journal of Intelligent & Robotic Systems*, vol. 98, no. 3-4, pp. 567–601, Oct. 2019.
[41] Z. Meng, X. Xia, R. Xu, W. Liu, and J. Ma, "Hydro-3d: Hybrid object detection and tracking for cooperative perception using 3d lidar," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 8, pp. 4069–4080, 2023.

THOMAS H. DRAGE received the B.Sc. degree in physics and the B.Eng. (Hons) degree in electrical and electronic engineering from The University of Western Australia in 2014 and the MIDS degree from the University of California, Berkeley in 2020. In 2023 he received the Ph.D. degree in computer engineering from The University of Western Australia.

From 2015 to 2019 he was engaged in automation and instrumentation engineering roles with Chevron Australia and from 2020 to 2021 led data science projects with the global Chevron Technical Center. He is currently Engineering Manager of Orexplore Technologies, based in Perth, Western Australia, who develop and market advanced x-ray instruments. His research interests include systems design, functional safety and advanced sensor systems.

KIERAN QUIRKE-BROWN has received a Master of Professional Engineering in Electrical & Electronics engineering in 2017 from The University of Western Australia. Between 2017 and 2021 Kieran worked as a BMS Engineer with Schneider Electric while earning his second master's degree in Information Technology from The University of Queensland. In 2021 he returned to UWA as a Ph.D. student in robotics and automation; his research focus' on dynamic obstacle avoidance and path planning for autonomous vehicles. He also holds a strong passion for teaching the next generation of students.

ZHIHUI LAI received the B.E. (Hons.) degree from The University of Western Australia, Perth, Australia, in 2021, where he is currently pursuing a Ph.D. degree with a focus on deep learning methods for autonomous driving. His research interests include end-to-end self-driving based on deep neural networks, computer vision, object detection and tracking.

KAI LI LIM received the B.Eng. (Hons) degree in electronic and computer engineering from the University of Nottingham, Nottingham, U.K. in 2012, and the M.Sc. degree in computer science from Lancaster University, Lancaster, U.K. in 2014. He received the Ph.D. degree from The University of Western Australia, Perth, Australia, in computer engineering in 2020, sponsored by the Australian Government under the Research Training Program.

He currently holds the St Baker Fellowship in E-Mobility at the University of Queensland, Australia, is an adjunct research fellow of The University of Western Australia and most recently is a visiting scholar at the EV Research Centre of the University of California, Davis. His research interests include visual navigation, navigational algorithms, and electric vehicles.

THOMAS BRÄUNL (Senior Member, IEEE) become a Member (M) of IEEE in 1997 and a Senior Member (SM) in 2008 and has received a Diploma degree in Informatics from Universität Kaiserslautern, Germany, the M.S. degree in Computer Science from University of Southern California, Los Angeles, CA, USA, and the Ph.D. and Habilitation degrees in parallel processing from Universität Stuttgart, Germany. He is currently Professor in Electrical and Computer Engineering at The University of Western Australia, Perth, Australia. He has worked in Germany for Mercedes-Benz, BMW, and BASF and has held guest professor appointments at TU München, Germany, and Santa Clara University, CA, USA. He is creator of the EyeBot mobile robot and embedded controller family, and directs the Renewable Energy Vehicle Project (REV) at UWA, where so far four electric vehicles and two autonomous vehicles have been constructed, and a city-wide charging network has been established. Embedded Robotics (Springer 2008) and Parallel Image Processing (Springer 2001) are two of his major publications in book form.