Autonomous Mobile Robot Navigation in Dynamic Real-World Environments Without Maps With Zero-Shot Deep Reinforcement Learning

Shathushan Sivashangaran

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Mechanical Engineering

May 06, 2024

Blacksburg, Virginia

Keywords: Autonomous Mobile Robot, Cognitive Navigation, Deep Reinforcement Learning, Dynamic Obstacle Avoidance, Unstructured Terrain

Autonomous Mobile Robot Navigation in Dynamic Real-World Environments Without Maps With Zero-Shot Deep Reinforcement Learning

Shathushan Sivashangaran

(ABSTRACT)

Operation of Autonomous Mobile Robots (AMRs) of all forms that include wheeled ground vehicles, quadrupeds and humanoids in dynamically changing GPS denied environments without a-priori maps, exclusively using onboard sensors, is an unsolved problem that has potential to transform the economy, and vastly improve humanity's capabilities with improvements to agriculture, manufacturing, disaster response, military and space exploration. Conventional AMR automation approaches are modularized into perception, motion planning and control which is computationally inefficient, and requires explicit feature extraction and engineering, that inhibits generalization, and deployment at scale. Few works have focused on real-world end-to-end approaches that directly map sensor inputs to control outputs due to the large amount of well curated training data required for supervised Deep Learning (DL) which is time consuming and labor intensive to collect and label, and sample inefficiency and challenges to bridging the simulation to reality gap using Deep Reinforcement Learning (DRL). This dissertation presents a novel method to efficiently train DRL with significantly fewer samples in a constrained racetrack environment at physical limits in simulation, transferred zero-shot to the real-world for robust end-to-end AMR navigation. The representation learned in a compact parameter space with 2 fully connected layers with 64 nodes each is demonstrated to exhibit emergent behavior for Out-of-Distribution (OOD) generalization to navigation in new environments that include unstructured terrain without maps, dynamic obstacle avoidance, and navigation to objects of interest with vision input that encompass low light scenarios with the addition of a night vision camera. The learned policy outperforms conventional navigation algorithms while consuming a fraction of the computation resources, enabling execution on a range of AMR forms with varying embedded computer payloads.

Autonomous Mobile Robot Navigation in Dynamic Real-World Environments Without Maps With Zero-Shot Deep Reinforcement Learning

Shathushan Sivashangaran

(GENERAL AUDIENCE ABSTRACT)

Robots with wheels or legs to move around environments improve humanity's capabilities in many applications such as agriculture, manufacturing, and space exploration. Reliable, robust mobile robots have the potential to significantly improve the economy. A key component of mobility is navigation to either explore the surrounding environment, or travel to a goal position or object of interest by avoiding stationary, and dynamic obstacles. This is a complex problem that has no reliable solution, which is one of the main reasons robots are not present everywhere, assisting people in various tasks. Past and current approaches involve first mapping an environment, then planning a collision-free path, and finally executing motor signals to traverse along the path. This has several limitations due to the lack of detailed pre-made maps, and inability to operate in previously unseen, dynamic environments. Furthermore, these modular methods require high computation resources due to the large number of calculations required for each step that prevents high real-time speed, and functionality in small robots with limited weight capacity for onboard computers, that are beneficial for reconnaissance, and exploration tasks. This dissertation presents a novel Artificial Intelligence (AI) method for robot navigation that is more computationally efficient than current approaches, with better performance. The AI model is trained to race in simulation at multiple times real-time speed for cost-effective, accelerated training, and transferred to a physical mobile robot where it retains its training experience, and generalizes

to navigation in new environments without maps, with exploratory behavior, and dynamic obstacle avoidance capabilities.

Dedication

To my parents, Kala, and Sivashangaran.

Acknowledgments

First and foremost, I thank my advisor Dr. Azim Eskandarian for his outstanding guidance during my PhD. His experience, passion for science, work ethic and confidence in me, were instrumental in developing my skills, enabling me to do my best work.

I appreciate the members of my committee, Dr. Alexander Leonessa, Dr. Dylan Losey and Dr. Thinh Doan for their constructive feedback. I also thank the administrative staff in the Mechanical Engineering department for the excellent assistance, and swift purchase of equipment.

I am grateful to all my lab mates, past and present, at the Autonomous Systems and Intelligent Machines Lab, for the insightful discussions and contributions, and my friends at Virginia Tech, for their companionship which made the PhD journey enjoyable.

Last but not least, I thank my parents and brother for their unwavering support.

Contents

List of Figures

List of Tables

List of Abbreviations

α	Entropy loss weight
β	Slip angle
δ	Steering angle
η	Observation FoV
γ	Discount factor
κ	Longitudinal slip ratio
$\mathcal{O}$	Observation space
ω	Angular velocity
ϕ	Critic network parameters
ϕ_s	Servo motor position
$\pi(s \mid \theta)$	Actor network
ψ	Heading angle
θ	Actor network parameters
A	Action space
a	Action
ax_t	Axle track

D	Discounted accumulated reward
F_{fx}	Longitudinal tire force at the front tires
F_{fy}	Lateral tire force at the front tires
F_{rx}	Longitudinal tire force at the rear tires
F_{ry}	Lateral tire force at the rear tires
G_{π}	Gradient of the actor output with respect to its parameters
G_{a}	Gradient of the critic output with respect to the action selected by the actor network
H	Target entropy
I_z	Polar moment of inertia
I_h	Camera image height
I_w	Camera image width
K_m	AMR velocity to motor RPM gain
K_s	Steering angle to servo position gain
L	Loss function
l_f	Distance from the center of mass to the front axle
l_r	Distance from the center of mass to the rear axle
l_{wb}	Wheelbase
M	Mini-batch of sampled data
m	Mass

m_o	AMR velocity to motor RPM offset
$Q(s, a \mid \phi)$	Critic network
R	Scalar reward
r	Range measurement
s	Observed state
s_o	Steering angle to servo position offset
T	Throttle
T_r	Turning radius
v	Linear velocity
V_m	DC brushless motor RPM
x, y, z	Position coordinates
AGV	Autonomous Ground Vehicle
AI	Artificial Intelligence
AMR	Autonomous Mobile Robot
APF	Artificial Potential Field
API	Application Programming Interface
AutoVRL	AUTOnomous ground Vehicle deep Reinforcement Learning simulator
AV	Autonomous Vehicle
CAD	Computer Aided Design

CARMA	Cooperative Automation Research Mobility Applications
CAV	Connected Autonomous Vehicle
CDA	Cooperative Driving Autonomy
CNN	Convolutional Neural Network
COCO	Common Objects in Context dataset
CPU	Central Processing Unit
DDPG	Deep Deterministic Policy Gradient
DL	Deep Learning
DNN	Deep Neural Network
DPG	Deterministic Policy Gradient
DRL	Deep Reinforcement Learning
DSRC	Dedicated Short-Range Communication
EES	Exploration Efficiency Score
EQS	Exploration Quality Score
ESC	Electronic Speed Controller
F1	Formula 1
FHWA	Federal Highway Administration
FoV	Field of View
GA	Genetic Algorithm

GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPU	Graphics Processing Unit
GTE	Grand Touring Endurance
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
LMP1	Le Mans Prototype
LMPC	Learning Model Predictive Control
LSTM	Long Short-Term Memory
ML	Machine Learning
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
OOD	Out-of-Distribution
PCM	Point Cloud Map
PGM	Probability Graph Method
PPO	Proximal Policy Optimization
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RNN	Recurrent Neural Network

ROS	Robot Operating System
RPM	Revolutions per Minute
RRT	Rapid exploration Random Tree
SAC	Soft Actor-Critic
SAR	Search and Rescue
SLAM	Simultaneous Localization and Mapping
SOM	System on Module
TCP/IP	Transmission Control Protocol/Internet Protocol
TD3	Twin-Delayed Deep Deterministic Policy Gradient
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol
URDF	Unified Robot Description Format
USDOT	United States Department of Transportation
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
VIO	Visual Inertial Odometry
ViT	Vision Transformer
X-CAR	eXperimental vehicle platform for Connected Autonomy Research

XML	eXtensible Markup Language
XTENTH-CAR	eXperimental one-TENTH scaled vehicle platform for Connected autonomy and All-terrain Research
YOLO	You Only Look Once object detection system

Chapter 1

Introduction

1.1 Background and Motivation

Autonomous Mobile Robots (AMRs) of all forms such as wheeled ground vehicles [1, 2, 3, 4, 5, 6], quadrupeds [7, 8, 9, 10, 11, 12] and humanoids [13, 14, 15, 16, 17, 18] are beneficial tools for a wide variety of tasks in agriculture [19, 20, 21, 22, 23, 24, 25], manufacturing [26, 27, 28, 29, 30, 31], disaster response [32, 33, 34, 35, 36, 37], Search and Rescue (SAR) [38, 39, 40, 41, 42, 43], military [44, 45, 46, 47, 48, 49] and extraterrestrial planetary exploration [50, 51, 52, 53, 54]. Once robust and reliable, these robots have the potential to transform the economy, and vastly improve humanity's capabilities. Operation in new, dynamically changing environments without prior maps that may be GPS denied such as caves and lave tubes on Mars, unknown buildings, contested military regions and areas effected by natural disasters such as fire or earthquake remains an unsolved problem [55, 56, 57, 58, 59, 60].

Automating a mobile robot is conventionally modularized into perception [61, 62, 63], path and motion planning [64, 65, 66, 67, 68, 69, 70], and control [71, 72, 73]. Current state-of-the-art approaches utilize Simultaneous Localization and Mapping (SLAM) [74, 75, 76, 77, 78, 79, 80] to simultaneously estimate robot state using onboard sensors, and construct a model of the environment the sensors perceive to map the surrounding unknown environment in real-time, followed by trajectory planning and control for collision free, task specific navigation in the perceived environment. A dependence on sequential motion estimation that is subject to mapping and pose estimate errors is a major drawback to SLAM. Moreover, conventional motion planning algorithms are dependent on specific environmental configurations [81] which limits their effectiveness in adapting to information poor, dynamically changing environments such as areas where emergency hazards occur, and unexplored subterranean environments.

Advancements in Artificial Intelligence (AI) and computer technology enable intelligent mobile robots capable of high autonomy via intricate world representations. Modular navigation models that integrate SLAM with Machine Learning (ML) approaches such as Deep Reinforcement Learning (DRL) effectively explore unknown environments with static obstacles [82, 83]. However, modular methods require multiple Deep Neural Networks (DNNs) for each subtask which is computationally expensive, and infeasible for real-time execution at high sampling rates for high speed applications, or on small, resource constrained AMRs such as the Multi-Modal Mobility Morphobot [84] tested by NASA for Mars exploration.

End-to-end AMR navigation approaches that directly map sensor inputs to control outputs are comparatively much more computationally efficient, and eliminate the need for explicit feature extraction and engineering [85, 86, 87]. Most works in end-to-end mobile robot navigation and autonomous driving in industry utilize vision based Deep Learning (DL) that incorporate Convolutional Neural Networks (CNNs) [88, 89, 90] and Vision Transformers (ViTs) [91, 92, 93], due to the challenges that arise when training end-to-end DRL models such as sample inefficiency, and difficulty in converging to useful policies. However, supervised training of DNNs requires vast amounts of curated data, often in the order of millions of miles of driving data for Autonomous Vehicles (AVs) which is costly, time consuming and labor intensive to collect and label. Unlike DL, DRL does not require well curated training data, and is effective in simulation with just two neural networks with two layers each during training, and can execute complex learned behavior using a single two-layer actor network during inference [94, 95]. An added benefit to DRL over DL is the potential to learn policies unknown to humans outside a training dataset to achieve superhuman performance, as seen with every AI agent that has surpassed humans in board games, video games and drone racing to date [96, 97].

A compact ubiquitous end-to-end DRL AMR navigation model that adapts to new environments without prior reprogramming or dependence on maps, with the capability to function across a plethora of physical AMR forms ranging from small reconnaissance Autonomous Ground Vehicles (AGVs) to bipedal humanoids, with varying payloads for embedded compute, has potential to significantly improve mobile robot utility, and expedite deployment in dynamic real-world environments.

1.2 Problem Statement

Motion planning models that incorporate DNNs and actor-critic Reinforcement Learning (RL) enable robotic systems to learn optimal, end-to-end policies in continuous and high-dimensional action spaces directly from characteristics of high-dimensional sensory input data to intelligently select goal driven actions in obstacle filled unstructured terrain in the absence of prior knowledge and detailed maps [56, 98, 99, 100]. Despite the advantages compared to contemporary methods, that have limitations that prevent robust deployment in real-world dynamic scenarios, few works have focused on real-world end-to-end navigation using DRL [101, 102] due to the difficulty in converging to reliable policies, and sample inefficiency which necessitates tens of thousands of collisions to reinforce positive behavior [103] that can result in potential physical damage to hardware making the training process expensive and time consuming.

Training with safety controllers to prevent collisions [104, 105] for feasible direct training in the real-world denies full experimentation with a complete range of observations and actions, and offline DRL trained on collected data limits the agent to policies within the known dataset, which must be sufficiently large. Moreover, training AMR navigation using onboard sensor observations requires hardware to be untethered to a workstation unlike for manipulation, locomotion or navigation with external sensors [106], hence online training must be performed on the embedded computer or transmitted to an external workstation, which slows down the process.

Training in simulation to transfer the learned policy to the real-world provides a number of benefits that include cost effective training, accelerated learning at rates faster than real-time speed and quick iterative tuning of hyperparameters, and reward components. Bridging the simulation to reality gap, however, can be arduous due to the differences between simulator and real-world physics [107, 108]. Contemporary approaches to bridge the reality gap utilize domain randomization [109] and modeled sensor and actuator noise to learn more robust policies that work over a wider range of domains that encompass the real-world, however these may result in suboptimal policies as a consequence of the agent being overconditioned to prevent overfitting, and in the case of injecting sensor and actuator noise into the simulator, has been shown to result in worse performance when transferred to reality [110]. Progress in simulation to real-world model transfer has been mainly limited by the lack of high-fidelity simulation tools and assets. Hence, to investigate techniques for zero-shot simulation to real-world model transfer, accurate digital robot and environment models must be developed, and packaged with a physics engine.

Incorrectly defined rewards lead to undesirable behaviors and suboptimal policies, and designing an appropriate reward function that guides the learning process effectively can be non-trivial. Therefore, in order to fully realize the benefits of DRL, formulations ranging from sparse to shaped rewards in combination with different types of DRL algorithms must be evaluated. Furthermore, generalization of a computationally efficient policy for navigation in real-world dynamic environments without prior maps necessitates dynamic obstacle avoidance, robustness to increased sensor noise in unstructured terrain, and intrinsic exploratory behavior, consequently the behavioral similarity between observations across these various tasks must be modeled in a compact DNN.

1.3 Contributions

Motorsport has a history of driving automobile innovation, spearheaded by competitive teams challenged with developing novel vehicle design, aerodynamics, control systems, and power delivery solutions to improve on-track performance. Racing pushes automobiles to the limit to extract maximum performance, and in doing so, pushes the envelope of contemporary technology. These technological advances trickle down to consumer automobiles resulting in improved road safety, and efficiency [111, 112, 113, 114]. The rise of autonomous vehicles this decade, both in roads and on the racetrack, offers a similar development avenue for AMR technology.

This dissertation presents a novel DRL method for end-to-end AMR navigation trained with significantly fewer samples in simulation in a constrained racetrack environment at physical limits for cognitive Out-of-Distribution (OOD) generalization to new, unknown environments without maps or state estimation, exclusively using onboard sensors transferred zero-shot with no additional training in the real-world.

The developed method does not require a high-level planner for explicit exploration goal-positions, and utilizes a fraction of the computation resources required for conventional navigation algorithms. This enables high autonomy in previously unseen environments, and provides a computationally efficient, robust foundation for generalized AMR System 1 behavior without a-priori maps that is compatible with a range of AMR forms with varying embedded computer payloads, and application specific System II algorithms. The contributions of this dissertation are:

- Formulate a DRL framework for end-to-end AMR exploration.
- Formulate metrics to quantitatively evaluate quality and efficiency of exploration trajectories.
- Evaluate different combinations of DRL reward formulations and algorithms.
- Develop AMR hardware and simulation platforms for zero-shot simulation to real-world model transfer.
- Significantly increase DRL sample efficiency with more vigorous training in a constrained environment at physical limits for OOD generalization.
- Characterize intrinsic exploratory behavior, dynamic obstacle avoidance and robustness in unstructured terrain in a single, computationally efficient, compact 2 layer DNN.
- Formulate a safety supervisor to prioritize collision-free motion during potential edge cases.
- Develop an add-on controller to the DRL model to navigate to objects of interest in all lighting conditions, including no-light scenarios with a night vision camera.

1.4 Dissertation Framework

The rest of the dissertation is organized as follows. After introducing AMRs, challenges to navigation in real-world environments, and contributions of this dissertation in Chapter 1, Chapter 2 reviews conventional and current state-of-the-art AMR navigation algorithms, and DRL in general in robotics. Chapter 3 presents a DRL method for task-independent AMR exploration in new environments without maps. Chapter 4 presents a Nonlinear Model Predictive Control (NMPC) racing algorithm, and discusses autonomous racing, and drawbacks with conventional methods. The hardware and simulation platforms developed to conduct this work are detailed in Chapter 5. Chapter 6 describes the methodology for OOD zero-shot simulation to real-world DRL for AMR navigation. The results are presented, analyzed and discussed in Chapter 7, and Chapter 8 discusses future work directions, and concludes the dissertation.

Chapter 2

Review of Literature

AMRs of various forms have existed for decades, but are not currently reliable for full autonomy in real-world applications, independent of human operators. Extensive research has converged to data-driven AI approaches that make use of recent advancements in ML, sensors, and computer technology to model systems, that outperform conventional methods that formulate dynamics and kinematics. This is due to the difficulty in accurately modeling physics, which is an approximation at best for even a four-wheeled vehicle, that is much less complex than a bipedal humanoid. Simplified vehicle models such as the bicycle model are not viable beyond a threshold velocity, and more complex models that take into account nonlinearities significantly increase computation cost. Comparatively, data-driven approaches outperform physics derived models, and are more computationally efficient to execute. This chapter reviews conventional, and contemporary state-of-the-art AI driven AMR navigation algorithms, including approaches specific to unstructured terrain with increased sensor and actuator noise, and dynamic obstacle avoidance. Moreover, autonomous racing algorithms, and DRL in robotics applications in general such as manipulation, and locomotion are reviewed.

2.1 Conventional Navigation Algorithms

Navigation in environments without prior knowledge is conventionally solved with SLAM [115, 116, 117]. AMR trajectories require optimization for shortest path, minimum energy consumption and training time [118, 119]. Conventional navigation algorithms utilized for path and motion planning in the occupancy map generated by SLAM have several limitations, due to the dependence on specific environmental configurations which limits effectiveness in adapting to information poor, dynamically changing environments such as areas where emergency hazards occur, and unexplored subterranean environments.

These comprise graph search algorithms such as Dijkstra, A* and D* [120, 121, 122, 123] that are well-defined and simple to use but are inefficient in complex, dynamic environments and have poor robustness to noise interference and errors in the environment model, random sampling algorithms such as Probability Graph Method (PGM) and Rapid exploration Random Tree (RRT) [124, 125, 126] that select random scatter points in the entire environment space to search for the optimal path between the starting and end points making them susceptible to poor real-time performance, sub-optimal solutions and high computation cost, Artificial Potential Field (APF) [127, 128, 129, 130] that is efficient but prone to local minima traps, Bug algorithms [131, 132, 133], such as Bug1 and Bug2, that combine motion towards the goal with contour-following behavior to navigate around obstacles, but are inefficient in environments with a multitude of obstacles, and nature inspired algorithms such as fuzzy logic [134, 135, 136] that is robust, but requires prior knowledge in the form of user defined knowledge based logic and rules, and Genetic Algorithm (GA) [137, 138, 139, 140] which is ideal for the global optimal solution and suitable for complex problems, but has poor local search ability and slow convergence rate.

The methods lack the ability to adapt autonomously to unknown environments since the algorithms rely on pre-made maps, and lack the flexibility to navigate in dynamic scenarios such as areas with high pedestrian traffic flow, like crosswalks, where collision avoidance and time constraints are critical. Furthermore, the reliance on an environment map increases inaccuracies due to resolution limits and sensor errors, that raise safety concerns.

2.2 Navigation With Machine Learning

Recent works combine AI with SLAM for a modular approach to navigation that incorporate multiple DNNs for mapping, planning and control [141, 142, 143, 144, 145, 146]. In [110], a neural SLAM module was utilized for navigation in static object filled indoor environments to predict the map of the surrounding environment and agent pose, using which a global policy trained to maximize environment coverage using DRL, was used to compute a long-term goal using a CNN. Subsequently, a local policy utilizing a Recurrent Neural Network (RNN) computed the navigation action utilizing RGB observations and a short-term goal, derived from the long-term goal. The method was compared to an end-to-end approach that used high-dimensional image input, and was shown to outperform it by 67% due to the large image domain gap between the simulator used for training, and the real-world. A pre-trained ViT was utilized in [147] in combination with a diffusion policy for undirected exploration and goal-conditioned action computation in previously unseen environments, evaluated in smooth surfaces at a low speed.

Numerous works have focused on utilizing DRL for task-driven navigation [56, 148, 149, 150, 151, 152] aided by state estimation observations to specific goal positions generated by a high-level planner. In [103], the authors trained and evaluated an indoor robot equipped with camera, GPS and compass sensors in simulation, to navigate to a target location from a random initial position in an unseen map, for over 180 days of GPU-time parallelly with 64

GPUs, in 3 days using Decentralized Distributed Proximal Policy Optimization (DD-PPO). However, task-independent exploration of a new environment to facilitate various downstream applications has received significantly less attention [153]. Effective exploration in GPS-denied environments such as indoor and subterranean environments encountered in SAR, military and extraterrestrial planetary exploration require even greater training times, and different DRL approaches.

2.3 Traversal in Unstructured Terrain

Navigation in unstructured terrain such as forests and other off-road environments includes additional challenges due to the increase in sensor and actuator noise that arise from changes in elevation [154, 155]. Prior works involved extraction of traversability features using geometric information, and statistical processing techniques [156, 157, 158] for informed motion planning. Learning based methods [159, 160] were used to classify terrain to determine traversability [161, 162, 163] which can be applied to real-world environments without the need for explicitly designed rules that is required with classical approaches, however these methods require manually labeled data in large quantities, and result in potentially low accuracy due to discretization of the environment for simplification of traversability classification.

A DRL model was trained in simulation and deployed in a real-world laboratory rough terrain environment [82] to output navigation actions in the form of forward or backward motion for a specific distance, or rotation angle for left or right turns using AMR pose and a 2D elevation map that represented the average height in discretized grid cells, as observations. The elevation map was derived from a 3D mesh that represented the surface geometry of the terrain, reconstructed from a 3D point cloud map generated using SLAM from RGB-

D input via Poisson surface reconstruction [164]. Such modular methods utilize simplified control inputs to compensate for the additional computation cost which prohibits real-time, continuous actions, and require goal-positions from a separate high-level planner.

2.4 Dynamic Obstacle Avoidance

Dynamic obstacle avoidance is a critical capability for AMRs that operate in complex and uncertain environments [165, 166, 167, 168, 169]. Modular methods operate at low sampling rates due to the larger number of operations required per action compared to end-to-end approaches, hence have limited adaptability to perceive and avoid dynamic obstacles in real-time. To account for this, a prediction module is added to estimate trajectories of dynamic obstacles for informed planning [170], that needs to solve trajectories for different object classes such as humans, animals, and robots, or add a buffer to maintain safety, which can lead to compromised performance.

DRL was utilized to improve the conventional A-star algorithm for path planning with an improved dynamic window approach [171] to train weight coefficients to generate safer trajectories that can be tracked by a motion planner. Furthermore, a Long Short-Term Memory (LSTM) network encoder was used in combination with DRL [172] to adapt to stochastic scenarios by encoding a variable number of dynamic obstacles into a fixed-length representation. To expedite research in dynamic obstacle avoidance, a benchmarking suite [173] for evaluating obstacle avoidance approaches in highly dynamic environments was developed for training, testing, and deployment of navigation planners, including both model-based and learning-based methods, on various robotic platforms, in which learning-based approaches performed better than model-based planners, when evaluated for goal-position navigation in known environments.

2.5 Autonomous Racing Algorithms

Autonomous racing requires quick execution of control signals [174, 175, 176, 177, 178, 179], that has potential to transfer to other AMR applications. Competitions with one-tenth scaled and full-size AVs facilitate development, and testing of racing algorithms. The most common approaches [180, 181, 182, 183] utilize optimal control such as Model Predictive Control (MPC) and sampling methods such as RRT with separate hierarchical path and motion planners in mapped environments.

A pre-trained representation of visual features was utilized to bootstrap online DRL training in [184] to automate real-world training using goal checkpoints provided by a high-level planner tracked using Visual Inertial Odometry (VIO) in indoor settings, and GPS in outdoor environments where VIO was inaccurate for state estimation. Few works utilized end-to-end DRL transferred zero-shot from simulation to reality due to the lack of high-fidelity simulation tools. Safe collision-free real-world training utilizing a safety supervisor [104, 185] was developed to bypass the reality gap that trained conservative policies evaluated in simulation, and rectangular and oval real-world racetracks. Environments of similar complexity were used for real-world DRL evaluation [186, 187], but did not generalize to new layouts, or perform in more complex multidirectional racetracks.

2.6 Deep Reinforcement Learning in Robotics

DRL was utilized to solve a number of robotics problems such as manipulation [188, 189, 190, 191], and locomotion [192, 193, 194, 195], including bipedal locomotion [196] where the trained model was transferred zero-shot from simulation to the real-world using a virtual spring model to account for the simulator's inability to accurately model joint dynamics, and for dexterous reorientation of objects facing downwards [197], with gravity adding complexity, where the policy was trained in simulation using synthetic point cloud without rendering, and fine-tuned with rendered point cloud to minimize computation complexity in addition to utilizing depth measurements over RGB images for real-world transfer. Moreover, DRL was utilized to learn agile soccer skills for bipedal robots [198] in simulation using position observations for zero-shot real-world transfer utilizing an external motion capture system.

2.7 Summary

AMR navigation in unknown environments is an unsolved research problem. A multitude of works have focused on developing goal-position navigation controllers, which are reliant on a high-level planner, and GPS for state estimation, however task-independent exploration of previously unseen environments, especially in GPS-denied or limited coverage environments such as forests with thick foliage, inside buildings, subterranean environments, and extraterrestrial planets has not been extensively developed.

Modular approaches are more common than end-to-end methods due to the high training sample requirement for the latter that gives rise to difficulties related to data collection, and simulation to reality transfer. Separating navigation into mapping, localization, path, and motion planning facilitates functional implementation, but is computationally expensive due to the multiple algorithms required for each sub-task that can be infeasible for real-time, high-speed applications, and on small AMRs with limited payload for onboard compute.

The addition of dynamic obstacles, and unstructured terrain further increases complexity, and requires additional modules to predict dynamic obstacle trajectories, and traversability features. An end-to-end DRL navigation model, that characterizes intrinsic exploratory behavior, dynamic obstacle avoidance, and robustness to increased sensor and actuator noise in unstructured terrain, in a compact, computationally efficient DNN, has potential to significantly improve feasibility of real-world AMR deployment.

Chapter 3

Deep Reinforcement Learning for Autonomous Mobile Robot Exploration

AMRs are essential tools for a wide range of applications stemming from the ability to operate in hazardous environments with minimal human operator input [118, 199]. The inclusion of LIDAR and visual SLAM in the perception pipeline is a key enabler for contemporary AMR navigation in environments that are GPS-denied with no access to a-priori maps [200]. SLAM enables AMRs to simultaneously estimate vehicle state utilizing on-board sensors and construct a model of the environment the sensors perceive, following which effective motion planning is paramount for successful operation. Conventional motion planning algorithms are dependent on prior knowledge of environment characteristics and offer limited utility in information poor, dynamically altering environments. Moreover, SLAM is subject to mapping and pose estimate errors as a consequence of estimating sequential motion, and a modular navigation model is computationally demanding as it comprises multiple DNNs with AI approaches, for each subtask.

End-to-end navigation utilizing a single DRL network achieved with reward shaping without a dependence on mapping and long-term goal generation is of benefit to early stage task-independent exploration of unknown environments in adverse environmental conditions for data collection to build more accurate maps, and facilitate down-stream applications. Towards realizing the potential of DRL for AMR navigation in information poor environments, this chapter presents and evaluates a DRL framework with shaped rewards for cognitive AMR navigation solely utilizing distance measurement observations obtained using LiDAR or camera RGB-D point cloud, and compares state-of-the-art off-policy DRL algorithms' ability to safely navigate and explore obstacle filled terrain without prior knowledge of environment characteristics. Range measurements were chosen as observations over RGB images as these are more efficient, providing a complete understanding of the surrounding environment at a fraction of the data size. Animals such as bats use echolocation for navigation [201] solely utilizing an understanding of distances to objects, hence it is reasonable to expect a DRL agent to learn a reliable, robust navigation policy exclusively utilizing range measurements. The decrease in the size of the observation space enables faster learning of policies in more complex environments that require a large number of training samples.

The DRL architecture comprises feedforward neural networks for the critic and actor representations in which the actor network strategizes linear and angular velocity control actions given current state inputs, that are evaluated by the critic network which learns and estimates Q-values to maximize an accumulated reward. On-policy Actor-Critic DRL algorithms such as Proximal Policy Optimization (PPO) [202], that are most commonly used in prior work, are robust to hyperparameter tuning and straightforward to implement, but are sample inefficient as these require new training samples for every policy update, which makes learning an effective policy for complex tasks computationally exorbitant. Off-policy Actor-Critic DRL algorithms such as Deep Deterministic Policy Gradient (DDPG) [203], Twin Delayed Deep Deterministic Policy Gradient (TD3) [204] and Soft Actor-Critic (SAC) [205] reuse past experience stored in a replay buffer for learning, thus have higher sample efficiency. Hence, these three off-policy DRL algorithms were trained and compared in two environments of varying complexity, and further evaluated in a third with no prior training or knowledge of map characteristics. The agent is shown to learn optimal policies at the end of each training period to chart quick, collision-free exploration trajectories, and is extensible, capable of adapting to an unknown environment without changes to network architecture or hyperparameters. The best algorithm was further evaluated in a realistic 3D environment.

The rest of the chapter is organized as follows. Section 3.1 provides a background on DRL. The proposed DRL framework is detailed in section 3.2. The training and evaluation methodologies are described in section 3.3. The results are presented and analyzed in section 3.4, and section 3.5 provides closing remarks and summarizes the chapter.

3.1 Background on Deep Reinforcement Learning

RL is a ML framework inspired by trial-and-error animal learning to train agents that interact with the surrounding environment by promoting or discouraging actions utilizing reward feedback signals designed to gauge effectiveness of executed actions. DL, a key ML component, utilizes DNNs to form an abstract, distinguishable high-level representation from low-level input features. DRL algorithms combine DL and RL to extract unknown environment features from high-dimensional input data utilizing DNNs, and decide control actions using RL. Figure 3.1 portrays the DRL framework.

A RL agent observes its environment s_i at each time step t, and selects an action a_i from action space A, conforming to a learned policy $\pi(a_i \mid s_i)$ that maps states to actions. The expectation of a discounted, accumulated reward $D_i = \Sigma_{k=0}^{\infty} \gamma^k R_{i+k}$ at each state is maximized during learning, where $\gamma \in (0,1]$ is the discount factor, and R_i is the scalar reward signal for selecting action a_i [206].

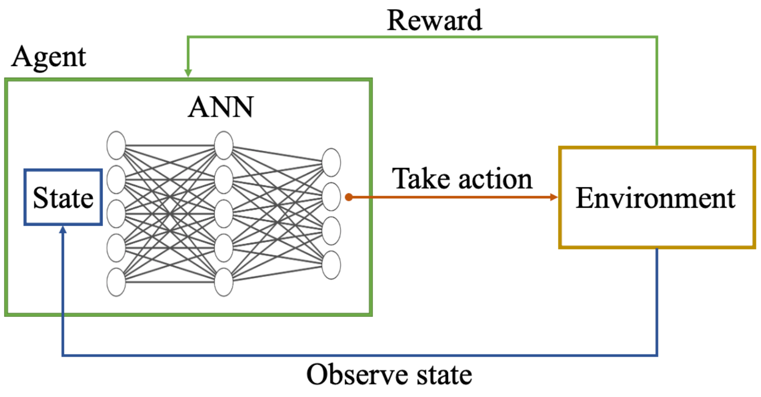


Figure 3.1: Schematic of deep reinforcement learning framework.

3.1.1 Actor-Critic Framework

An actor-critic framework utilizing deep function approximators that combines both value-based and policy-based RL is the preferred method to learn policies in continuous and high-dimensional action spaces, required for robotics applications. This method leverages the joint computing and decision-making abilities of the actor and critic neural networks to yield low variance and fast speeds when updating gradients. Figure 3.2 illustrates the actor-critic framework.

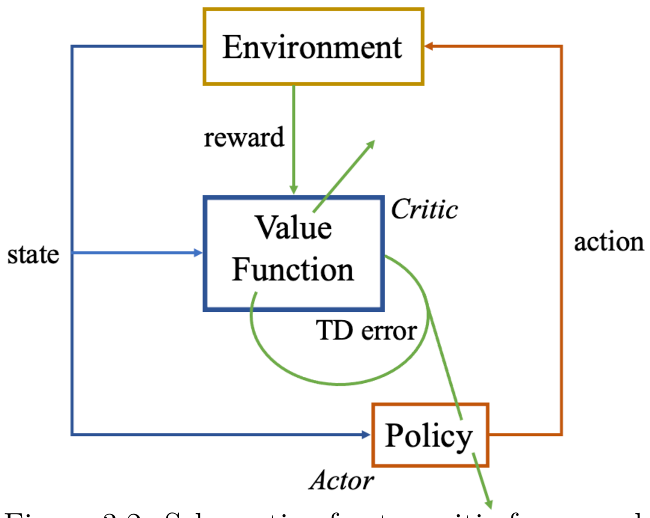


Figure 3.2: Schematic of actor-critic framework.

The actor network strategizes an action output selected from a continuous action space using policy gradient, utilizing the current state as the input. The critic evaluates the chosen actions and outputs the associated approximate Q-value for the current state and selected action using an approximated value function to counter the large variance in the policy gradients. In off-policy algorithms, sample data accumulated in a replay buffer is utilized to update and approximate the value function yielding higher sample efficiency than on-policy algorithms. The two networks compute the action prediction for the current state at each time step to generate a temporal-difference error signal.

3.1.2 Deep Deterministic Policy Gradient

DDPG [203] is a model-free, off-policy actor-critic RL algorithm that combines DNNs with the actor-critic representation of standard Deterministic Policy Gradient (DPG) [207] to successfully implement control sequences in a continuous action space. The actor, $\pi(s \mid \theta)$ and critic, $Q(s, a \mid \phi)$ each comprise fully-linked, two-layer feedforward DNNs with Rectified Linear Unit (ReLU) activation functions.

The loss L is minimized across all sampled experiences to update the critic parameters, ϕ,

$$L = \frac{1}{M} \sum_{i=1}^{M} (Y_i - Q(s_i, a_i \mid \phi))^2 \tag{3.1}$$

Here M is a random mini-batch of experiences, and Y is the target value function computed as follows,

$$Y_i = R_i + \gamma Q_t(s_{i+1}, \pi_t(s_{i+1} \mid \theta_t) \mid \phi_t) \tag{3.2}$$

θ_t and ϕ_t are parameters of the target actor π_t and target critic Q_t respectively, that have the same structure and parameterization as π and Q. The agent periodically updates θ_t and ϕ_t using the latest θ and ϕ values to improve the stability of the optimization.

The actor parameters, θ are updated using a sampled policy gradient $\nabla_\theta J$ to maximize the expected discounted reward,

$$\nabla_\theta J \approx \frac{1}{M}\sum_{i=1}^{M} G_{ai} G_{\pi i} \tag{3.3}$$

Here G_a is the gradient of the critic output with respect to the action selected by the actor network computed as follows,

$$G_{ai} = \nabla_a Q(s_i, \pi(s_i \mid \theta) \mid \phi) \tag{3.4}$$

G_π is the gradient of the actor output with respect to its parameters,

$$G_{\pi i} = \nabla_\theta \pi(s_i \mid \theta) \tag{3.5}$$

3.1.3 Twin-Delayed Deep Deterministic Policy Gradient

TD3 is designed to improve learned policies by preventing overestimation of the value function [204]. Two Q-value functions are learned simultaneously, and the minimum is used for policy updates. Moreover, the policy is updated less frequently than the Q-value function to further improve learned policies.

The parameters of the critic, $Q_k(s, a \mid \phi_k)$, where $k = 2$ is the number of critics, are updated by minimizing the loss L_k as follows,

$$L_k = \frac{1}{M}\sum_{i=1}^{M}(Y_i - Q_k(s_i, a_i \mid \phi_k))^2 \tag{3.6}$$

The target value function Y is computed as follows,

$$Y_i = R_i + \gamma \min_k(Q_{tk}(s_{i+1}, clip(\pi_t(s_{i+1} \mid \theta_t) + \varepsilon) \mid \phi_{tk})) \tag{3.7}$$

Here θ_t and ϕ_{tk} are parameters of the target actor π_t and target critics Q_{tk}, and ε is noise added to the computed action to promote exploration. The action is clipped based on the noise limits.

The actor parameters are updated similar to DDPG using Equation 3.3 where G_a is computed as follows and G_π is computed as in Equation 3.5.

$$G_{ai} = \nabla_a \min_k(Q_k(s_i, \pi(s_i \mid \theta) \mid \phi)) \tag{3.8}$$

3.1.4 Soft Actor-Critic

SAC [205], similar to DDPG and TD3, is a model-free, off-policy actor-critic RL algorithm. In addition to maximizing the long-term expected reward, SAC maximizes the entropy of the policy, which is a measure of the policy uncertainty at a given state. A higher policy entropy promotes exploration, hence the learned policy balances exploitation and exploration of the environment.

The agent utilizes a stochastic actor that outputs mean and standard deviation, using which an unbounded action is randomly selected from a Gaussian distribution. The entropy of the policy is computed during training for the given observation using this unbounded probability distribution. Bounded actions that comply with the action space are generated by applying *tanh* and scaling operations to the unbounded action.

The critic parameters are updated at specific time step periods by minimizing the loss function in Equation 3.6, similar to TD3 for k critics.

The target value function Y is computed as the sum of R_i, the minimum discounted future reward from the critic networks, and the weighted entropy as follows,

$$\begin{gathered} Y_i = R_i + \gamma \min_k (Q_{tk}(s_{i+1}, \pi(s_{i+1} \mid \theta) \mid \phi_{tk})) \\ -\alpha ln\pi(s_{i+1} \mid \theta) \end{gathered} \tag{3.9}$$

Here α is the entropy loss weight. The entropy weight is updated by minimizing the loss function, L_α where H is the target entropy as follows,

$$L_\alpha = \frac{1}{M} \sum_{i=1}^{M} (-\alpha ln\pi(s_i \mid \theta) - \alpha H) \tag{3.10}$$

The stochastic actor parameters are updated by minimizing the objective function J_π,

$$J_\pi = \frac{1}{M} \sum_{i=1}^{M} (-\min_k (Q_{tk}(s_i, \pi(s_i \mid \theta) \mid \phi_{tk})) + \alpha ln\pi(s_i \mid \theta)) \tag{3.11}$$

3.2 Deep Reinforcement Learning Framework

This section presents the DRL framework, and reward design for quick, efficient and collision-free AMR exploration in new, unknown environments.

3.2.1 Network Architecture

In order to maximize the long-term reward, designed to encourage quick, efficient, and collision-free exploration of the environment, the DRL agent makes strategic linear and angular velocity action decisions for the current time step, v_t and ω_t. These decisions are based on LiDAR or camera RGB-D point cloud range measurements from the current and previous time steps r_t and r_{t-1}, the previous time step's action $a_{t-1} = (v_{t-1}, \omega_{t-1})$, and the corresponding reward value, R. The proposed DRL architecture for AMR exploration is shown in Figure 3.3.

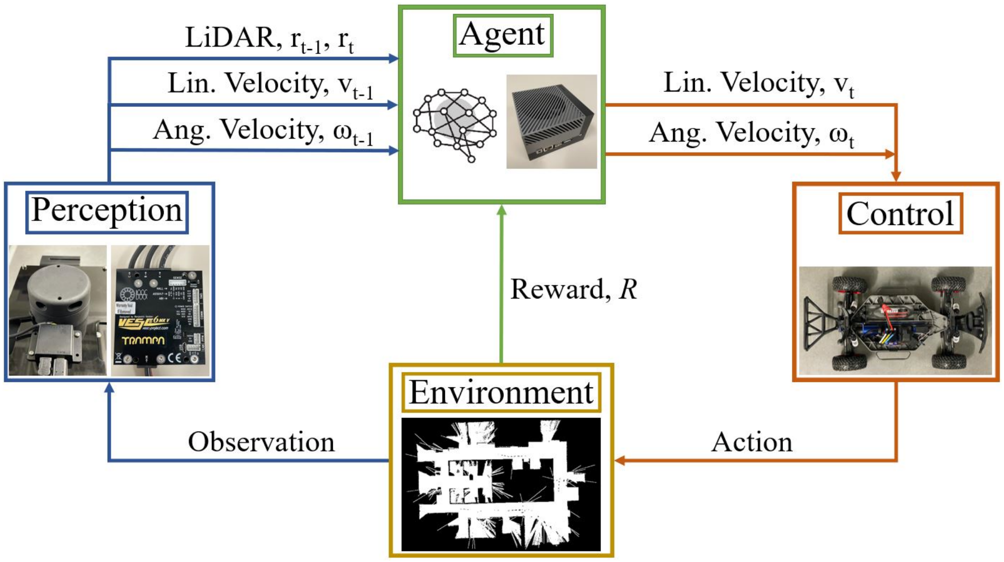


Figure 3.3: Deep reinforcement learning framework for autonomous mobile robot exploration without maps.

The agent updates the policy at each time step during training using the selected action, the prior and current observations after executing the action, and scalar reward feedback, with the objective of maximizing the long-term reward to promote collision-free exploration.

The addition of global position information obtained via GPS to the observation space will improve learning sample efficiency, and can encourage more efficient exploration, however it is not viable in environments that are underground, or inside buildings. To mitigate this, the reward is specifically shaped to encourage the agent to navigate previously unexplored regions solely utilizing range and odometry measurements.

3.2.2 Reward Shaping

The reward function is application specific, and designed to encourage the agent to explore its environment efficiently, quickly and safely without collisions. To compare the exploration capabilities of DDPG, TD3 and SAC, a minimalist reward function was shaped as presented in [99],

$$R = 0.0075r^2 + 1.5v^2 - 0.6\omega^2 \tag{3.12}$$

A positive reward was applied to the square of the minimum range measurement, r to incentivize obstacle avoidance. This reward is highest when the agent is at a greater distance from obstacles, encouraging the generation of paths devoid of obstacles. The agent was additionally rewarded for swift navigation through positive reinforcement of linear velocity, v. To encourage efficient exploration, a negative reward was applied to angular velocity, ω to discourage repeated circular motion in the same vicinity. High coefficients for r^2 and v^2 led to a compromise between obstacle avoidance ability and exploratory behavior, hence a balance was determined through trial-and-error experimentation to prioritize both exploration, and collision avoidance.

The presented DRL framework can be trained with more specialized application specific reward functions, such as for SAR operations in forests with thick foliage or urban regions covered in rubble in the aftermath of a natural disaster that require AMRs to specifically explore shaded regions underneath trees, boulders or buildings inaccessible to aerial surveillance using Unmanned Aerial Vehicles (UAVs). The reward for this application can be shaped as,

$$R_{SAR} = \begin{cases} 0.0075r^2 + 1.5v^2 - 0.6\omega^2 & \\ 2.0 & \text{if } 2.0 < r < 2.5 \\ -50 & \text{if } r < 1.0 \end{cases} \tag{3.13}$$

In addition to the reward in Equation 3.12, the agent is assigned a reward of 2.0 when it is between 2.0 and 2.5 m of the nearest object to promote exploration in shaded regions close to obstacles unobservable to aerial surveillance. In order to improve the safety of exploration trajectories, the agent is assigned a penalty of 50.0 when it is within 1 m of the nearest obstacle.

The focus of this chapter is on evaluating the proposed DRL framework for end-to-end cognitive AMR exploration without maps, and comparing the performance of sample efficient off-policy algorithms for this task, hence the generalized reward function in Equation 3.12 was used for comparison analysis. The best algorithm was further evaluated using the application specific shaped reward in Equation 3.13 to demonstrate the effectiveness of the proposed method.

3.3 Training and Evaluation

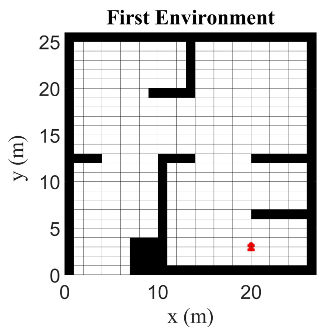
The MATLAB Robotics System [208] and Reinforcement Learning [209] Toolboxes, and Simulink were utilized to model the AMR, and train the DRL agent to compare off-policy algorithms DDPG, TD3 and SAC using the reward in Equation 3.12. The best algorithm was further evaluated in AutoVRL [210], an in-house high-fidelity simulator developed using open-source tools for simulation to real-world DRL using the shaped reward in Equation 3.13.

3.3.1 Environments

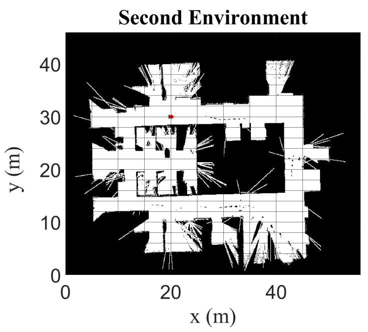
The DRL agents were trained in two distinct environments for comparison. The first environment, depicted in Figure 3.4a, is a simple 25 m x 25 m space with walls that the agent must steer clear of. The second environment, illustrated in Figure 3.4b, is a more complex 40 m x 40 m space with walls and various obstacles, dotted in black, that the agent must additionally avoid. The trained agents from each environment were evaluated in a third environment, illustrated in Figure 3.4c without prior training or knowledge of map characteristics to evaluate the robustness, and performance of the learned policies in a new, unknown environment with the same network architecture and hyperparameters.

The AMR, identified with a red symbol on the training maps, is set to a random starting position at the start of each training episode to enhance policy learning. This reset ensures that the agent is not biased towards any particular initial location.

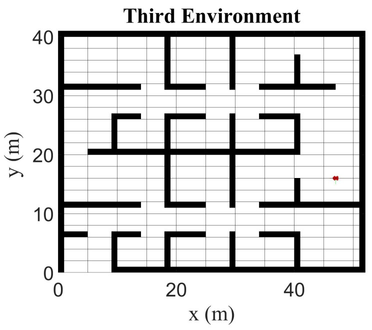
To further evaluate the effectiveness of the DRL architecture for intelligent application specific exploration, a realistic 3D 20 m x 20 m outdoor environment with tree and boulder objects, illustrated in Figure 3.5, was utilized to train and evaluate the best algorithm de-



(a)



(b)



(c)

Figure 3.4: Training and evaluation environments with DRL agent marked in red at a randomized initial location. (a) First training environment. (b) Second training environment. (c) Evaluation environment.

termined from analyses of prior results.

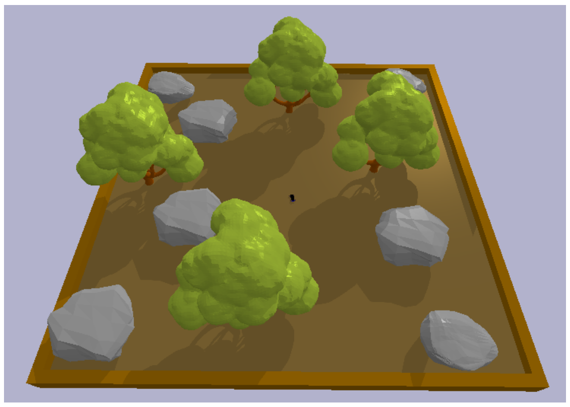


Figure 3.5: Outdoor environment with tree and boulder objects.

This environment includes shaded regions beneath trees and boulders unobservable to aerial surveillance, hence the shaped reward in Equation 3.13 was utilized to train the DRL agent to specifically explore shaded regions inaccessible to UAVs.

3.3.2 Exploration Quality and Efficiency Quantification

The post-training trajectories learned by the agent were evaluated for exploration quality and efficiency by assigning an Exploration Quality Score (EQS) and Exploration Efficiency Score (EES) to quantitatively compare the performance of each tested algorithm. The training and evaluation environments are segmented, as shown in Figure 3.6 for the first training environment. Each environment was divided into 2 m^2 segments, and further segmented by circles of increasing radii with radius $10n$, where $n = 1, 2, 3, ...$

The EQS for the generated trajectory was scored as follows,

$$EQS = \Sigma_{n=1}^{n_{max}} (n \Sigma_{j=1}^{j_{max}} j) \tag{3.14}$$

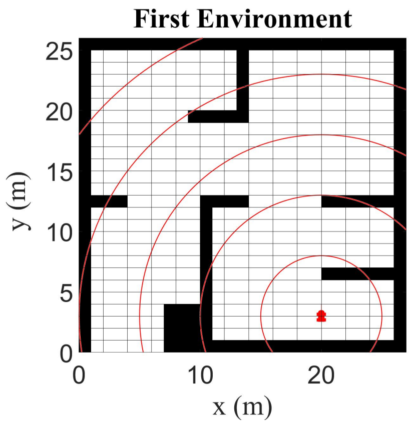


Figure 3.6: Occupancy grid map of first environment segmented in circles of increasing radii to score exploration quality and efficiency.

Here j is the number of 2 m^2 segments covered by the trajectory. The EQS is greater when the trajectory covers a higher number of segments, and traverses further away from the AMR's initial position to regions encompassed by circles of larger radii. 2 m^2 segments were chosen since it is reasonable for the AMR's onboard sensors to collect viable data for down-stream applications within this region.

The EES was scored as follows where d is the trajectory distance,

$$EES = \frac{EQS}{d} \tag{3.15}$$

The EES is higher when the EQS is large, and the distance travelled is small. A higher EES indicates better energy efficiency, and exploration performance.

3.3.3 AMR Model

XTENTH-CAR [211], a proportionally scaled experimental vehicle platform, designed with similar hardware and software architectures as the full-size X-CAR [212] connected autonomous vehicle, was modeled and trained in simulation. The XTENTH-CAR AMR has a wheelbase of 0.32 m and utilizes the Ackermann steering mechanism.

The AMR's kinematics were computed using a bicycle model, portrayed in Figure 3.7, where the front and rear wheels are represented by a single wheel located at the center of each axle. This model is accurate for use at low speeds and offers a good balance between model accuracy and computation cost [213] for evaluation of the DRL agent.

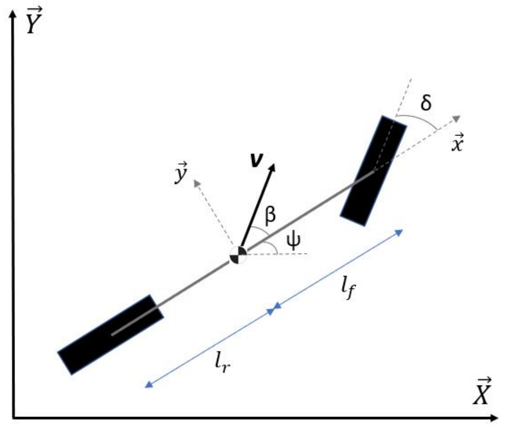


Figure 3.7: Schematic of kinematic bicycle model.

The bicycle model is represented by the following equations,

$$\dot{x} = v\,cos(\psi + \beta) \tag{3.16}$$

$$\dot{y} = v\,sin(\psi + \beta) \tag{3.17}$$

$$\dot{\psi} = \frac{v}{l_r}\,sin(\beta) \tag{3.18}$$

$$\beta = tan^{-1}\left(\frac{l_r}{l_f + l_r}\,tan(\delta)\right) \tag{3.19}$$

Here x and y are position coordinates of the AMR's center of mass, ψ is the angle of the AMR's heading with respect to the inertial reference frame, slip angle β is the angle between the velocity vector of the AMR's center of mass and its longitudinal axis, l_f and l_r are distances from the center of mass to the front and rear axles respectively, and velocity, v and steering angle, δ are control inputs.

3.3.4 Training Conditions

A training episode was concluded when the agent encountered an obstacle or completed the maximum number of steps permitted in a single episode. Subsequently, the agent was reset to a randomly determined starting location to initiate the next episode.

The DRL agent was trained to a total of 10,000 episodes, each with a maximum of 1000 steps in the first environment, and 20,000 episodes, each with a maximum of 2000 steps in the second, to facilitate rapid iterative learning. The best algorithm was trained for 5000 episodes, each with a maximum of 1000 steps in the 3D outdoor environment.

Hyperparameters that were modified with non-default values are listed in Table 3.1.

Table 3.1: Non-Default Hyperparameters

Hyperparameter	**Value**
Discount Factor (γ)	0.995
Actor Learn Rate	0.00005
Critic Learn Rate	0.0005
Target Smooth Factor	0.001
Mini Batch Size	128
Experience Buffer Length	1,000,000

3.4 Results and Discussion

This section presents DRL training results that include post-training exploration trajectories and corresponding average return and steps achieved by the agent each episode iteration during the training period, utilizing DDPG, TD3 and SAC algorithms.

3.4.1 Training Performance

An Intel i7 11700K CPU and GeForce RTX 3070 Ti GPU were used for training. Table 3.2 summarizes the training times for each DRL algorithm in the evaluated environments.

Table 3.2: Training Times

Agent	**Training Time (Hrs)**
First Environment: DDPG	66.0
First Environment: TD3	95.3
First Environment: SAC	156.7
Second Environment: DDPG	84.8
Second Environment: TD3	128.1
Second Environment: SAC	203.9

SAC required the longest training time, followed by TD3 and DDPG which required the least. On average, training in the second, more complex environment required 31% longer training time than in the first, over twice the number of training episodes. DDPG required 28.5%, TD3 34.4% and SAC 30.1% longer to train in the second environment.

In the first environment, TD3 required 44.4% longer to train than DDPG, and SAC 137.4% longer than DDPG and 64.4% longer than TD3. In the second environment, TD3 required 51.1% longer to train than DDPG, and SAC 140.4% longer than DDPG and 59.2% longer than TD3. On average, TD3 required 47.8% longer training time than DDPG, and SAC 138.9% longer than DDPG and 61.8% longer training time than TD3.

Training times ranged from 2.75 days in the first environment for DDPG to 8.5 days in the second environment for SAC. More optimal policies require longer training times to accommodate increased episode steps in the first environment, and more training episodes in the second.

3.4.2 First Environment

The order 50 moving average return and agent steps during training in the first environment are illustrated in Figures 3.8 and 3.9. The training results in the first environment are summarized in Table 3.3.

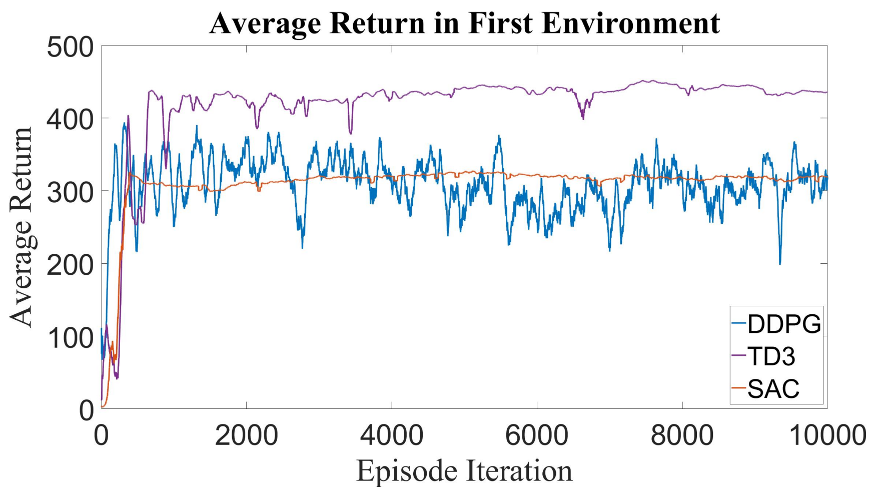


Figure 3.8: Order 50 moving average return during training in the first environment.

Table 3.3: First Environment Training Results

Algorithm	**Convergence Episode**	**Average Return**	**Average Steps**	**EQS**	**EES**
DDPG	170	318	865	81	1.82
TD3	960	435	1000	**113**	2.40
SAC	390	320	1000	91	**2.63**

DDPG converged first at 170 episodes with an average return of 318 and 865 average steps.

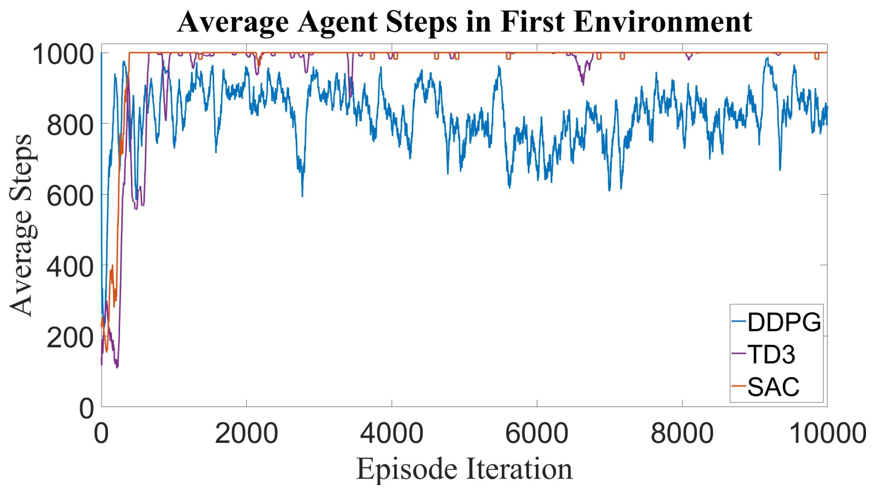


Figure 3.9: Order 50 moving average agent steps during training in the first environment.

TD3 converged last at 960 episodes with an average return of 435 and 1000 average steps, and SAC converged at 390 episodes with an average return of 320 and the maximum 1000 average steps. TD3 achieved the highest EQS, and SAC the highest EES.

DDPG learned the least optimal policy with the lowest average return, agent steps, EQS and EES. TD3 achieved the highest return, and the maximum 1000 steps, however SAC achieved 1000 exploration steps more consistently post training convergence. Unlike TD3 which solely maximized the long-term expected reward, SAC additionally maximized the entropy of the policy to promote exploration. Consequently, TD3 learned a policy with a higher return, but SAC learned the better policy for agent exploration.

The trajectories in the first environment for each algorithm post training completion are illustrated in Figure 3.10.

Each algorithm achieved 1000 episode steps without collision. SAC covered the most ground, and exhibited the most efficient exploratory behavior which will result in the greatest energy savings. TD3 is next best, followed by DDPG which was the most inefficient, covering the same region multiple times.

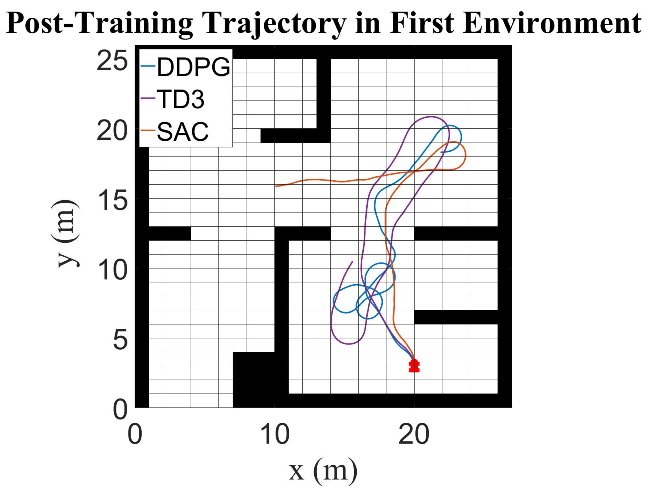


Figure 3.10: Trajectories in the first environment post training completion.

3.4.3 Second Environment

The order 50 moving average return and agent steps during training in the second environment are illustrated in Figures 3.11 and 3.12. The training results in the second environment are summarized in Table 3.4.

Training for 20,000 episodes was insufficient for the DRL algorithms to learn an optimal policy in the second environment. At the end of the training period, DDPG achieved an average return of 125 and 530 average steps, TD3 obtained an average return of 230 and 715 average steps, and SAC converged to a local maximum at 10,620 episodes with an average return of 210 and 1050 average steps. Training was limited to 20,000 episodes to gauge performance in a reasonable time frame, however, continued training over 75,000 to 100,000 episodes will enable the agents to learn an optimal policy to traverse the more complex terrain over an indefinite number of exploration steps.

Training DDPG, TD3 and SAC algorithms in the second environment for 20,000 episodes required a total of 416.8 hours, as such, it is infeasible to evaluate the algorithms for 75,000+ episodes with the existing setup. More powerful computer hardware is required. Similar to

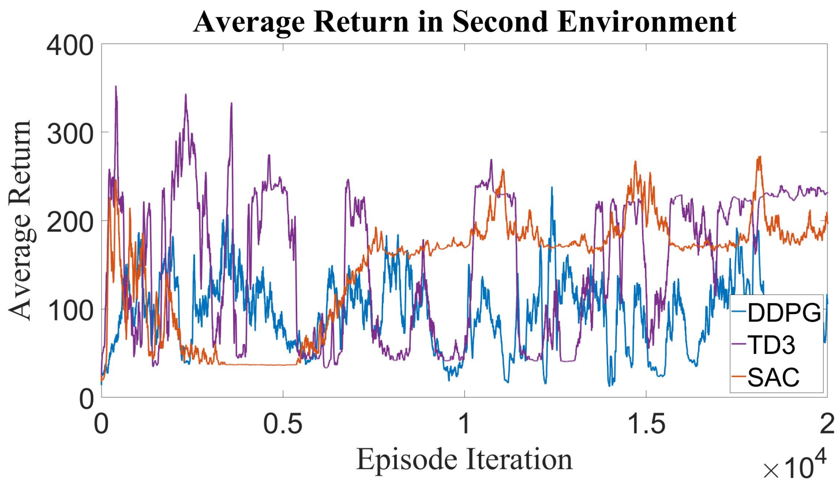


Figure 3.11: Order 50 moving average return during training in the second environment.

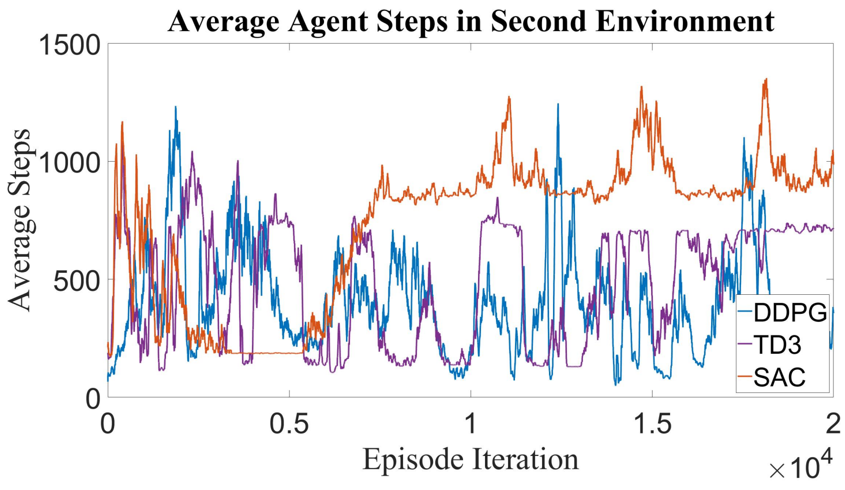


Figure 3.12: Order 50 moving average agent steps during training in the second environment.

the training results in the first environment, DDPG learned the least optimal policy achieving the lowest return, agent steps, EQS and EES. TD3 achieved the highest return, however SAC learned a more optimal policy achieving the highest EQS and EES.

The trajectories in the second environment for each algorithm post training completion are illustrated in Figure 3.13.

SAC achieved the best performance, learning a trajectory that covered the most distance.

Table 3.4: Second Environment Training Results

Algorithm	**Average Return**	**Average Steps**	**EQS**	**EES**
DDPG	125	530	60	1.92
TD3	230	715	67	1.97
SAC	210	1050	**110**	**2.24**

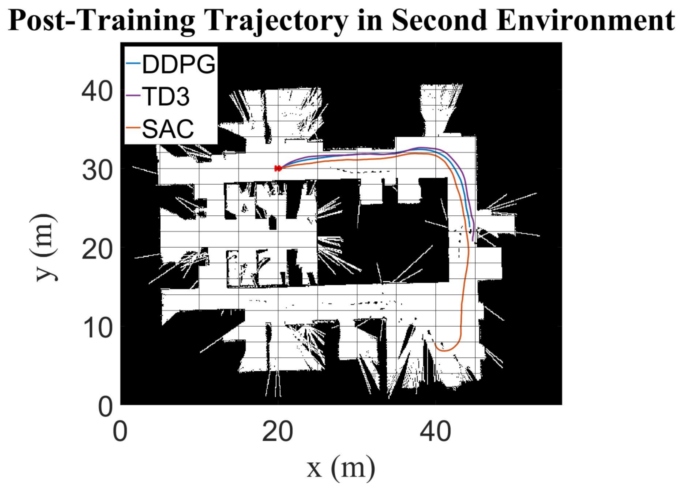


Figure 3.13: Trajectories in the second environment post training completion.

TD3 and DDPG yielded similar performance, with TD3 being a marginal improvement.

3.4.4 Trained Policy Evaluation

The six agents, DDPG, TD3 and SAC, trained in two different environments were evaluated in a third unknown environment with no prior training or knowledge of environment characteristics, to evaluate the extensibility of the ubiquitous DRL architecture for AMR exploration in information poor environments. Figure 3.14 portrays the trajectories for each agent in the third environment. The evaluation results are summarized in Table 3.5.

The SAC agent trained in the second, more complex environment demonstrated the best performance, covering the most ground, efficiently with the highest EQS and EES. The

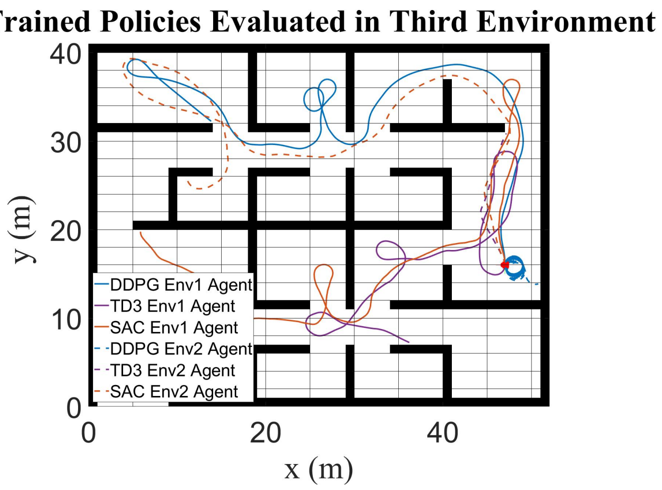


Figure 3.14: Trained DRL agents evaluated in the third environment.

Table 3.5: Third Environment Evaluation Results

Agent	Steps	EQS	EES
DDPG Env1	2151	252	2.36
TD3 Env1	1533	102	1.33
SAC Env1	2971	197	1.95
DDPG Env2	2893	3	0.02
TD3 Env2	321	25	1.56
SAC Env2	2839	**303**	**3.15**

DDPG agent trained in the first environment performed second best, covering more ground than either TD3 agent. DDPG trained in the second environment covered the second highest distance, but yielded the worst exploratory behavior with the least EQS and EES, repeatedly traversing a circular trajectory in the same vicinity. TD3 agents covered less ground, and exhibited less efficient exploratory behavior than either SAC agent. DDPG trained in the first environment performed better than that trained in the second, as the characteristics of the evaluated environment are more similar to the first than the second. The SAC agents were most robust to differences in environment characteristics, and performed better when trained in the more complex environment. Both SAC agents performed well, with the agent trained in the simple first environment achieving the third highest EQS and EES.

The reward function weights and network hyperparameters can be further engineered for this application, and the agent trained over a longer period with more episode steps each episode iteration to learn an improved policy that explores the surrounding environment indefinitely.

3.4.5 Shaped Reward Evaluation

SAC was utilized to learn a policy in a realistic 3D environment using the shaped reward in Equation 3.13 designed for AMR exploration in shaded regions unobservable to UAVs. The order 50 moving average return during training is illustrated in Figure 3.15. A post-training trajectory from a randomized initial position is illustrated in Figure 3.16.

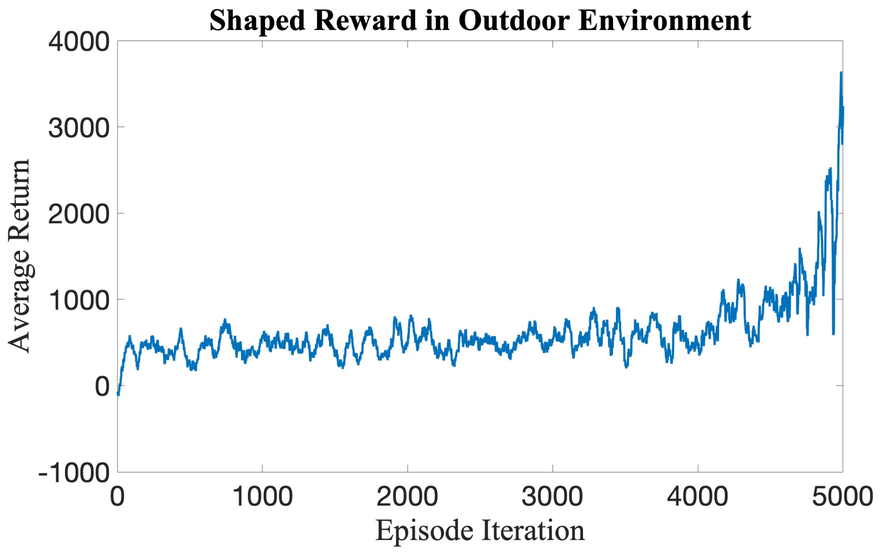


Figure 3.15: Order 50 moving average return during training in the outdoor environment.

The average return increased to 3642 at the end of the training period, yielding a policy that successfully promoted exploration in shaded regions beneath trees.

Bridging the simulation to real-world gap to transfer policies learned in simulation to real-world robotic systems is a current area of active research. The large number of episodes required to sufficiently train the agent renders simulation training an essential component

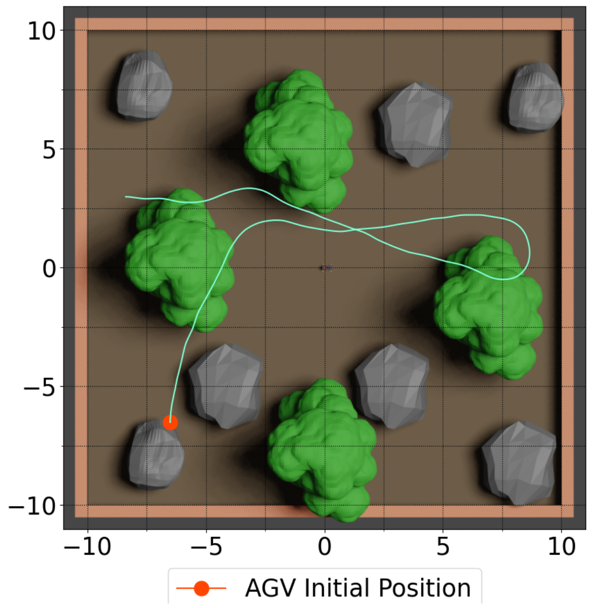


Figure 3.16: Trajectory in the outdoor environment post training completion.

for DRL in robotics applications to minimize cost and possible physical damage caused by collisions during training. Substantial computation cost is required for training, however, post-training implementation of DRL agents is significantly less expensive, which makes DRL a powerful tool for real-time AMR motion planning and control in environments without a-priori maps.

3.5 Summary

This chapter presented an ubiquitous DRL architecture for intelligent AMR exploration without a-priori maps. Three actor-critic off-policy DRL algorithms, DDPG, TD3 and SAC, were trained in two environments of varying complexity, and further evaluated in a third with no prior knowledge of map characteristics. Simulation results demonstrated the effectiveness of the proposed DRL architecture, reward function and training conditions for quick, efficient and collision-free AMR navigation. SAC achieved the best performance, yielding trajectories that covered the greatest distance, and demonstrated the most efficient exploratory behavior. Learning requires substantial computation cost, requiring up to 8.5 days for SAC in the second, complex environment using an Intel i7 11700K CPU and GeForce RTX 3070 Ti GPU. Improved policies with higher post-training episode steps require greater training times. Despite the high training cost, post-training implementation of DRL agents is significantly less expensive, which makes DRL a powerful tool for real-time AMR exploration in information poor, dynamically altering environments. Reward shaping renders the proposed DRL framework a versatile tool for a multitude of AMR applications. SAC was trained using a reward shaped for AMR exploration in regions inaccessible to aerial surveillance in a realistic outdoor environment, yielding a trajectory that specifically explored shaded regions underneath trees, demonstrating the adaptability of the proposed DRL framework.

Chapter 4

Autonomous Racing

Motorsport has historically driven automobile innovation by challenging the world's best car manufacturers to design, and develop vehicles that push limits of contemporary technology, and compete at physical vehicle limits. Autonomous driving is a rapidly evolving field that garners interest in industry, government, and research due to substantial improvements in road safety, and traffic flow. The three fundamental components to realize full self-driving are, perception for environment mapping, localizing, and object detection, motion planning for trajectory design to traverse, and complete tasks in the environment, and control to execute actuator commands to track the desired trajectory [81, 212, 214]. Autonomous racing is a byproduct of advancements in autonomous driving, and guarantees innovation in the field through the design of state-of-the-art perception, motion planning, and control algorithms developed to perform in fast-paced, multi-object environments at high speeds, operating at a vehicle's acceleration, and tire limits.

Motion planning, and control for autonomous racing require maneuvering a vehicle to lap a racetrack as fast as possible, routinely operating in highly nonlinear domains adjacent to the vehicle's stability limits. Perception pipelines, incorporating LiDAR and camera data, extract key racetrack information utilizing lane detection for track boundaries, and object detection and tracking for observation of competing race cars for use in path planning algorithms [215, 216, 217].

Extensive research in this domain has resulted in a number of innovative motion planning and control strategies for autonomous race cars aimed at minimizing lap time. These include hierarchical Nonlinear Model Predictive Control (NMPC) strategies designed to handle extreme race car performance requirements [180, 215], and Learning Model Predictive Control (LMPC) developed to utilize data from prior laps to improve performance using either Iterative Learning Control (ILC) or a DNN [218, 219, 220]. Furthermore, state-of-the-art ML algorithms such as DRL for Unmanned Ground Vehicles (UGV) [99] can be applied to autonomous racing to train an optimal racing policy using a tailored reward function.

Racetrack limits confine autonomous racing to limited environment states, and as such, an exclusive optimal trajectory is present, to minimize lap time for a given vehicle's dynamics. Professional racing drivers operate race cars close to stability limits, conforming to an optimal racing line tracked through experience. Numerical optimization techniques are capable of designing an optimal race trajectory as good, or better than the best racing drivers. NMPC [221] is a control technique which optimizes a vehicle's predicted motion over a limited time horizon subject to constraints, enabling the design of a cost function to minimize lap time, and effectuate racetrack constraints, and vehicle input limits.

Complex high-fidelity system models allow for planning of real-world accurate racing trajectories, but are computationally expensive. The existence of a unique optimal racing trajectory enables offline computation of the time-optimal reference trajectory to be implemented in the actual race car, and tracked via a less computationally expensive low-level NMPC integrated with adaptive control to account for uncertainties. In this chapter, a novel high-level NMPC strategy to formulate the time-optimal reference trajectory in the global coordinate system is presented, with a cost function specifically designed to minimize lap time, subject to racetrack constraints, and vehicle limits. The NMPC trajectory planner is evaluated in three real-world racetracks, Circuit of the Americas, Circuit de Spa-Francorchamps, and Autodromo Nazionale Monza for three race car classes, Formula 1 (F1), Le Mans Prototype (LMP1), and Grand Touring Endurance (GTE), and shown to generate optimized trajectories for the tested scenarios. The NMPC strategy is shown to generate time-optimal trajectories for each vehicle class in the evaluated tracks, conforming to optimal racing lines demonstrated by professional racing drivers.

The rest of the chapter is organized as follows. Section 4.1 describes the nonlinear race car model used for NMPC computation, section 4.2 details the NMPC formulation, section 4.3 portrays results, and section 4.4 provides closing remarks, and discusses the drawbacks of conventional racing algorithms for real-world deployment.

4.1 Race Car Model

Typical race cars are four-wheeled rear-wheel driven vehicles, with steering input to the front tires. The dynamics of the race car are estimated using a nonlinear bicycle model, depicted in Figure 4.1, to evaluate the vehicle's future states [222]. This simplified low-fidelity system model offers reasonable computation time and model accuracy trade-off for validation of the proposed NMPC framework.

The position of the vehicle is described by x and y coordinates, and orientation ψ, defined at its center of mass. Velocities in x and y directions, v_x and v_y and yaw rate ω describe its motion. The six states are governed by the following state equations,

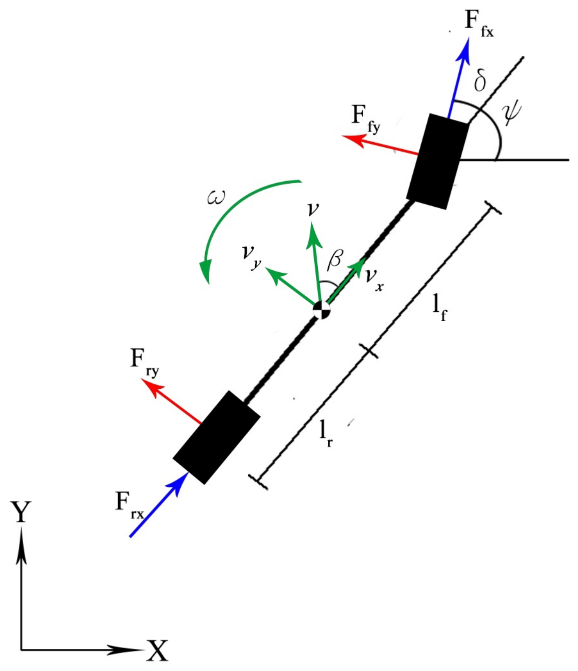


Figure 4.1: Schematic of nonlinear bicycle model.

$$\dot{x} = v_x cos\psi - v_y sin\psi \tag{4.1}$$

$$\dot{y} = v_x sin\psi + v_y cos\psi \tag{4.2}$$

$$\dot{\psi} = \omega \tag{4.3}$$

$$\dot{v}_x = \frac{1}{m}(F_{rx} + F_{fx} cos\delta - F_{fy} sin\delta + m v_y \omega) \tag{4.4}$$

$$\dot{v}_y = \frac{1}{m}(F_{ry} + F_{fx} sin\delta + F_{fy} cos\delta + m v_x \omega) \tag{4.5}$$

$$\dot{\omega} = \frac{1}{I_z}(l_f(F_{fx} sin\delta + F_{fy} cos\delta) - l_r F_{ry}) \tag{4.6}$$

Here m and I_z represent mass and polar moment of inertia respectively, δ is the steering angle, F_{fx}, F_{fy}, F_{rx} and F_{ry} are the longitudinal and lateral tire forces at the front and rear tires, and l_f and l_r add up to the wheelbase, and are the distances of the front and rear axles from the center of mass of the car respectively.

v_x and v_y are computed from the vehicle velocity v as follows,

$$v_x = vcos\beta \tag{4.7}$$

$$v_y = vsin\beta \tag{4.8}$$

$$\beta = tan^{-1}\left(\frac{l_r}{l_f + l_r} tan(\delta)\right) \tag{4.9}$$

Here β is the angle between the velocity vector of the center of mass of the car and its longitudinal axis.

4.1.1 Pacejka Tire Model

The tire forces were obtained using the simplified Pacejka Model [223]. The defined forces take into consideration lateral and longitudinal slip, and hence are a good approximation of the forces in the nonlinear bicycle model.

$$F_{iy} = D_i sin(C_i arctan(B_i \alpha_i)) \tag{4.10}$$

$$F_{ix} = D_i sin(C_i arctan(B_i \phi_i)) \tag{4.11}$$

Here $i = f, r$ indicate the front and rear tires respectively, and α_i and ϕ_i are parameters related to the lateral and longitudinal slips defined as follows,

$$\alpha_f = -arctan\left(\frac{\omega l_f + v_y}{v_x}\right) + \delta \tag{4.12}$$

$$\alpha_r = arctan\left(\frac{\omega l_r + v_y}{v_x}\right) \tag{4.13}$$

$$\phi_i = (1 - E)\kappa + \frac{E}{B}(arctan(B\kappa)) \tag{4.14}$$

The parameters B, C, D, and E are coefficients obtained using semi-empirical methods. κ is the longitudinal slip ratio which varies between 0 and 1.

4.2 Nonlinear Model Predictive Control Formulation

The NMPC framework for high level trajectory planning for an autonomous race car in known racetracks utilizes the vehicle model described in the previous section.

Longitudinal and lateral acceleration components, a_x and a_y were added to the states described above to encapsulate the vehicle's acceleration limits, and computed as follows,

$$a_x = \dot{v}_x \tag{4.15}$$

$$a_y = \dot{v}_y \tag{4.16}$$

Moreover the steering angle was added as an additional state obtained by integrating the steering rate input $\dot{\delta}$. The state vector is,

$$X = [x, y, \psi, v_x, v_y, \omega, a_x, a_y, \delta]^T \tag{4.17}$$

The inputs to the system are the steering rate and vehicle velocity. The input vector is,

$$U = [\dot{\delta}, v]^T \tag{4.18}$$

The output of the system, Y consists of all the states hence,

$$Y = X \tag{4.19}$$

The discrete form of the state equations for NMPC computation were obtained using Euler's method.

$$X_{k+1} = f(X_k, U_k) \qquad\qquad k = 0, ...P-1 \tag{4.20}$$

Here P is the prediction horizon for the controller. The NMPC algorithm optimizes a cost function $J(X, U)$ over the prediction horizon while satisfying the state and input constraints.

4.2.1 Cost Function

Racing requires completing laps around the track in a minimum time duration. Considering steering rate and vehicle speed inputs to the system, the task of racing requires the vehicle to achieve maximum possible speeds, so the total length of the track is covered in the least possible time. Hence the objective function for this problem was designed to minimize the total distance covered at each step divided by the absolute velocity at that instance to maximize the vehicle speeds at all instances. The cost function is,

$$\min_{X^k, U^k} J(X, U) = \sum_{k=1}^{P} \frac{S_k}{U_{2,k-1}} - \sum_{k=1}^{P} U_{2,k-1} \tag{4.21}$$

Here S_k is the euclidean distance between the current and the previous state.

$$S_k = \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2} \tag{4.22}$$

4.2.2 Design of Constraints

The states adhere to the vehicle model, hence the first constraint on the states is an equality constraint to follow the discrete form obtained in 4.20,

$$X_{k+1} - f_k(X_k, U_k) = 0 \tag{4.23}$$

Using the track boundary coordinates, constraints on the position of the vehicle for the prediction horizon were imposed to stay within boundaries as follows,

$$\begin{aligned} &\min(x^i_{lb}) < x_k < \max(x^i_{ub}) \\ &\min(y^i_{lb}) < y_k < \max(y^i_{ub}) \\ &\sqrt{(x_k - x_c)^2 + (y_k - y_c)^2} < w_{tr} \end{aligned} \tag{4.24}$$

Here the subscript k denotes the vehicle's current position, lb and ub correspond to lower and upper track bounds, c is the track center point and w_{tr} is half the track width.

Utilizing the distance from the centerline and the boundary coordinates for the next N steps, a binding area for the possible future positions from the current state was determined as illustrated in Figure 4.2.

The acceleration, steering angle, steering rate, and speed limits depend on vehicle characteristics. These states and inputs were constrained as follows,

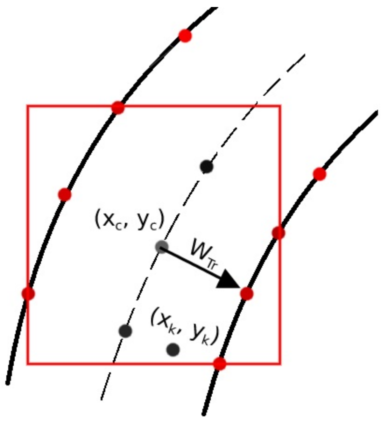


Figure 4.2: Track constraint binding area for two steps.

$$\begin{aligned}
a_{x,y,lb} &\leq a_{x,y,k} \leq a_{x,y,ub} \\
\delta_{lb} &\leq \delta_k \leq \delta_{ub} \\
\dot{\delta}_{lb} &\leq \dot{\delta}_k \leq \dot{\delta}_{ub} \\
0 &\leq v_k \leq v_{max}
\end{aligned} \tag{4.25}$$

4.2.3 Evaluation

The NMPC motion planner was evaluated in three real-world racetracks, Circuit of the Americas (COTA), Circuit de Spa-Francorchamps (Spa), and Autodromo Nazionale Monza (Monza) illustrated in Figures 4.3-4.5 respectively, for F1, LMP1, and LMGTE vehicles in simulation, using the MATLAB Model Predictive Control Toolbox [224]. Track data was obtained from [225].

These racetracks regularly host race events each year, and provide a realistic track layout to evaluate, and validate the NMPC motion planning algorithm for autonomous racing. The model was designed to be agnostic to vehicle type, and can be modified for each race car by

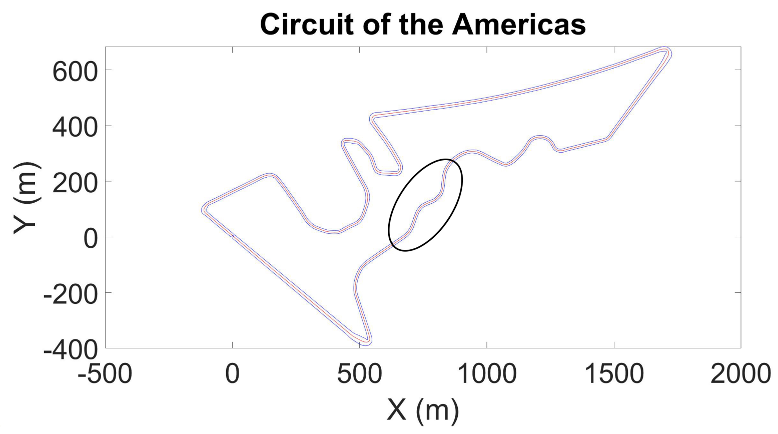


Figure 4.3: Circuit of the Americas.

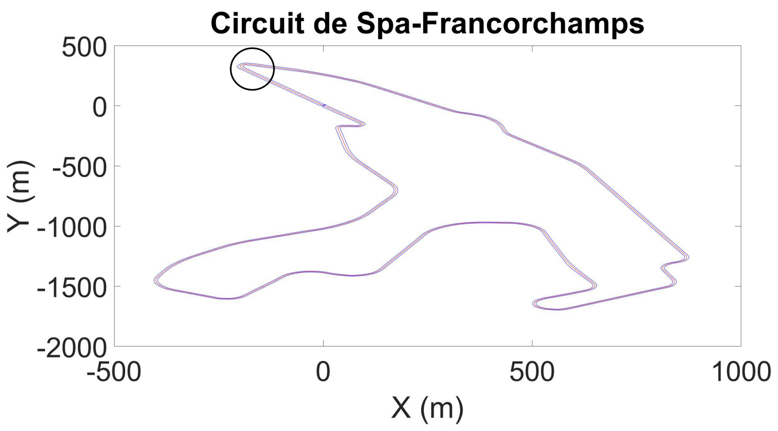


Figure 4.4: Circuit de Spa-Francorchamps.

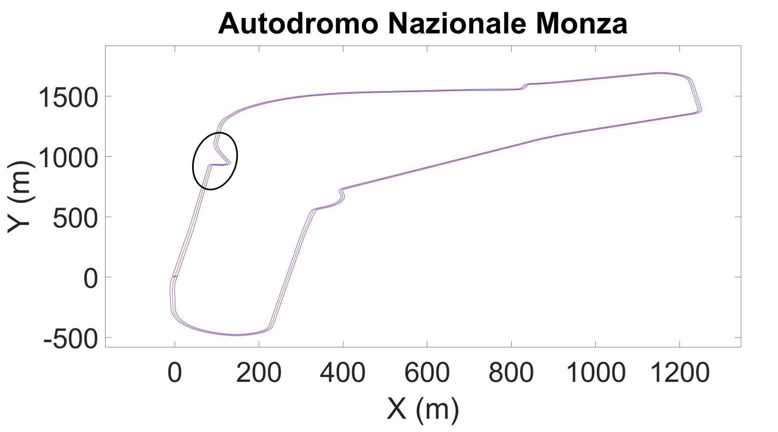


Figure 4.5: Autodromo Nazionale Monza.

Table 4.1: Race Car Parameters.

	F1	LMP1	LMGTE
$\boldsymbol{m(kg)}$	775	850	1250
$\boldsymbol{I_z(m^4)}$	707	945	2216
$\boldsymbol{l_f(m)}$	1.61	1.00	1.00
$\boldsymbol{l_r(m)}$	1.89	1.50	1.50
$\boldsymbol{\kappa}$	0.30	0.35	0.35
$\boldsymbol{v_{max}(m/s)}$	87.0	100.0	55.0
$\boldsymbol{a_{x,max}(m/s^2)}$	12.0	13.0	8.0
$\boldsymbol{-a_{x,max}(m/s^2)}$	20.0	23.0	18.0
$\boldsymbol{a_{y,max}(m/s^2)}$	59.0	45.0	24.0

altering input constraints, and model parameters. The race car parameters used in simulation are listed in Table 4.1. These parameters are estimates obtained from a combination of news articles, interviews and forum discussions since exact vehicle information is proprietary. Large prediction horizons yield improved results, but require high computation cost. A prediction horizon of 15, which corresponds to a track distance of $75m$, was used to balance computation cost, and provide sufficient information to solve the time-optimal trajectory.

4.3 Results and Discussion

The resulting time-optimal trajectories generated by the NMPC motion planning algorithm for each race car in the three racetracks, and corresponding velocity inputs, and steering angles derived from steering rate inputs are illustrated in Figures 4.6-4.14. Challenging corners in each track, circled in Figures 4.3-4.5, are used to demonstrate the effectiveness of the proposed NMPC strategy. Figures 4.6-4.8 depict trajectory, and input results for Turns 3-6 in COTA, Figures 4.9-4.11 for Turn 1 in Spa, and Figures 4.12-4.14 for Turns 1 and 2 in Monza.

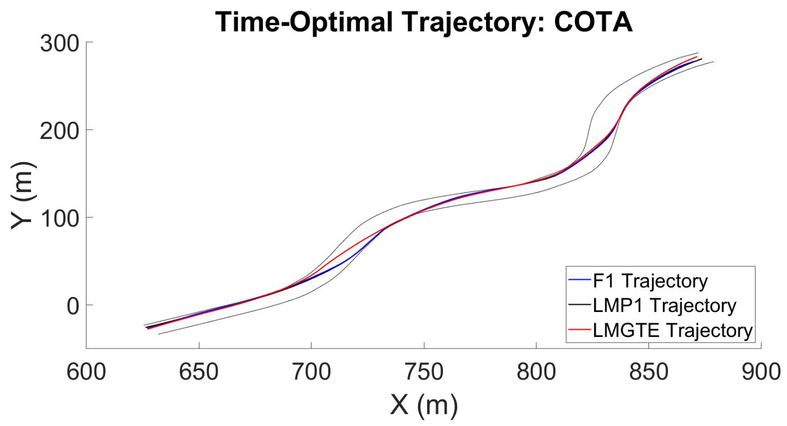


Figure 4.6: Time-optimal trajectories at COTA.

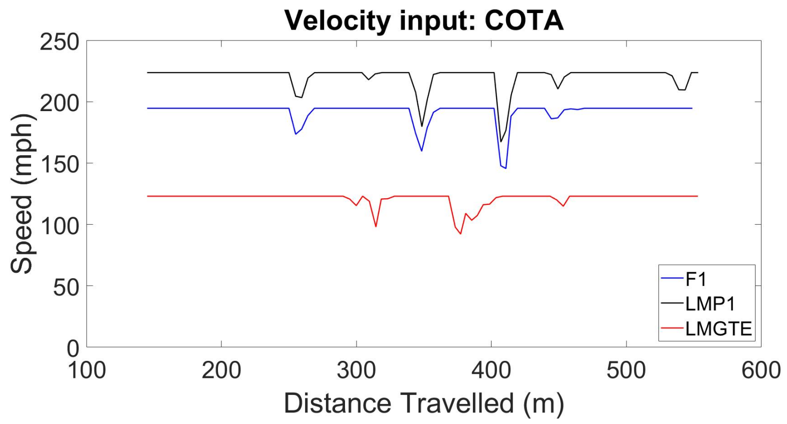


Figure 4.7: Velocity inputs at COTA.

The NMPC algorithm successfully planned an optimal trajectory for each vehicle designed to minimize lap time, subject to vehicle dynamics, and input limits. The generated trajectories exhibit ideal racing behavior. The race cars switch lanes to the far side when taking a corner, and swoop to the inside, hitting the apex before exiting on the far side past the corner, and accelerating. The corresponding velocity, and steering angle plots for each track segment convey ideal vehicle inputs solved by the algorithm to achieve these optimal trajectories. These inputs are dependent on the vehicle system model. High-fidelity models will yield

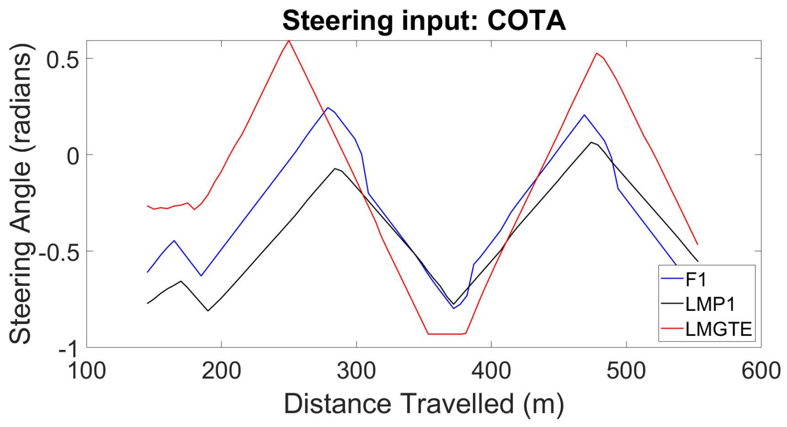


Figure 4.8: Steering inputs at COTA.

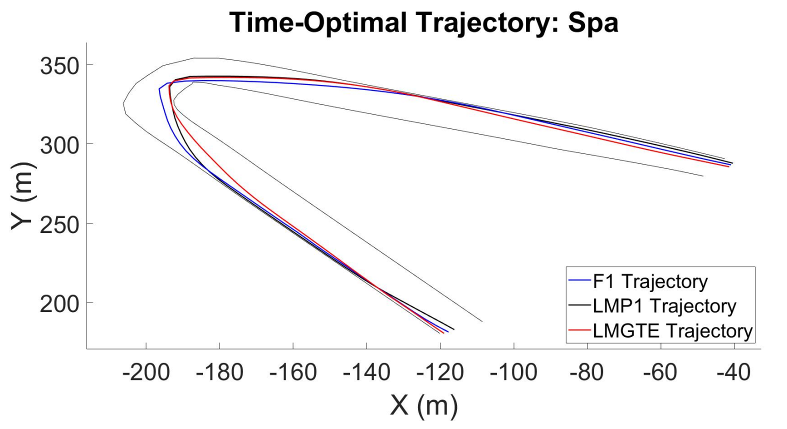


Figure 4.9: Time-optimal trajectories at Spa.

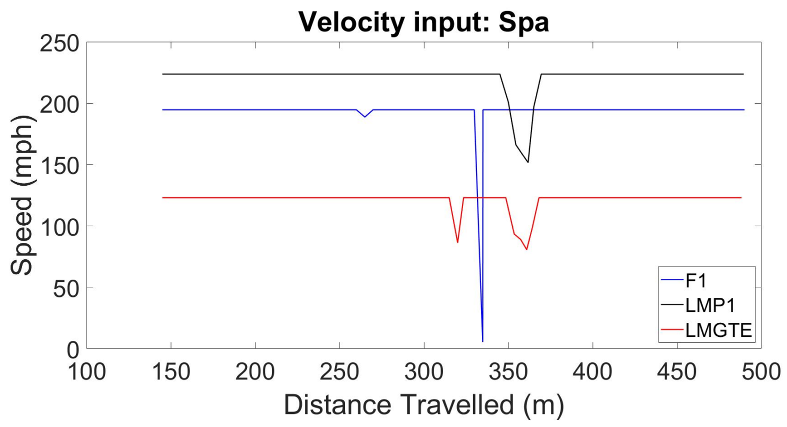


Figure 4.10: Velocity inputs at Spa.

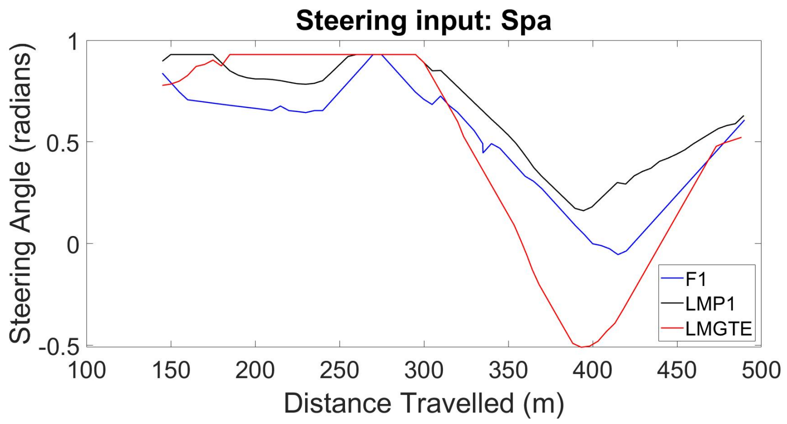


Figure 4.11: Steering inputs at Spa.

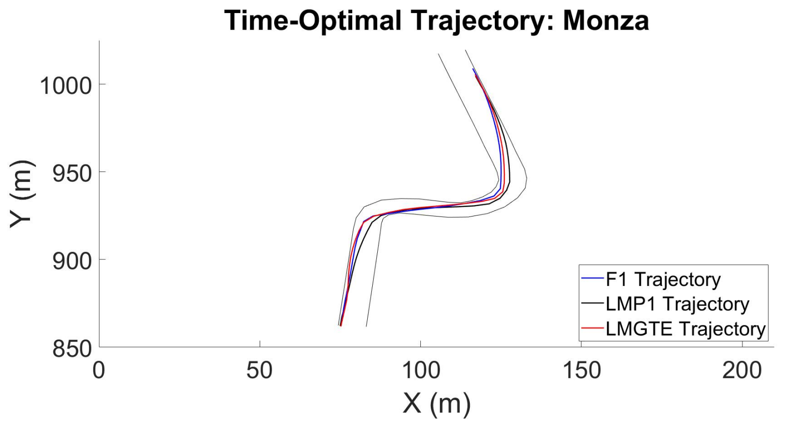


Figure 4.12: Time-optimal trajectories at Monza.

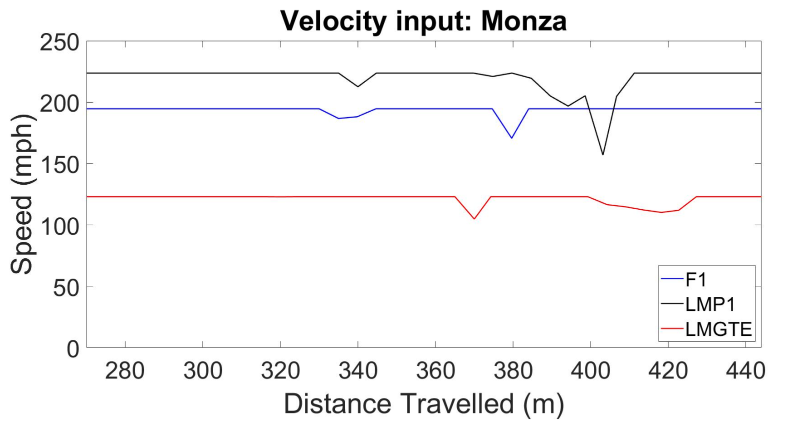


Figure 4.13: Velocity inputs at Monza.

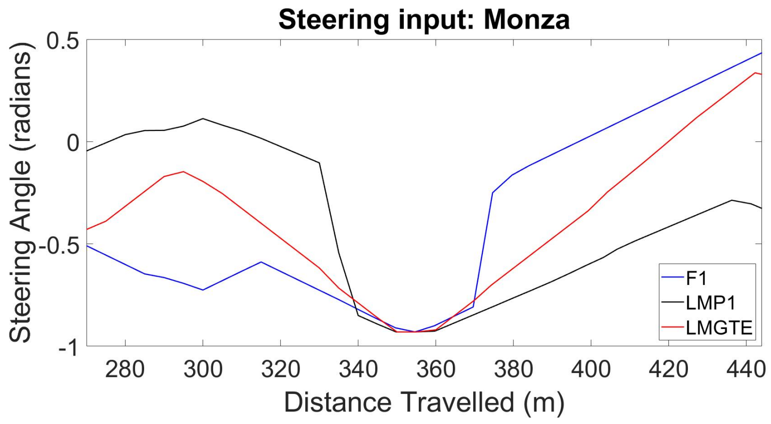


Figure 4.14: Steering inputs at Monza.

improved results that are more real-world accurate.

F1 cars have highest lateral acceleration and slip ratios, enabling high cornering speeds, conveyed through low number of fluctuations, and sharp changes to velocity inputs. LMGTE cars, most similar to road vehicles, have the least downforce, and top speed, and consequently require high steering input. LMP1 cars have highest top speed, but lower downforce than F1. These variations result in differences in the time-optimal trajectories for each race car.

4.4 Summary

The NMPC algorithm, with cost function, and constraints designed for time-optimal autonomous race car motion planning yielded optimal results. The generated trajectories were as expected, and conformed to racing lines demonstrated by professional racing drivers. The proposed NMPC strategy was evaluated in three real-world race tracks for three race car classes to compute an optimized trajectory to minimize lap time, and can be utilized for any combination of vehicle class and racetrack. To further improve the model, which is necessary for real-world deployment, velocity input can be replaced with longitudinal acceleration, and mapped to throttle position in combination with high-fidelity vehicle dynamics, which take into consideration center of mass height, aerodynamics, and resistive forces to obtain more real-world accurate control inputs. Implementing such models will further increase computation cost, that require offline computation of optimal racing trajectories, and do not guarantee optimal performance due to modeling approximations. Furthermore, the algorithm is dependent on racetrack boundary position information, that limits online adaptability to new racetrack layouts, and requires more compute for real-time SLAM, that may compromise high speed performance.

Chapter 5

Hardware and Simulation Platforms

DRL is an effective technique for AMR navigation, enabling intelligent action decisions in a continuous and high-dimensional action space that comply with a learned policy, solely utilizing high-dimensional sensory input data. Cognitive navigation that utilizes raw sensor data with no dependency on a-priori maps or GPS is a requisite for AMR applications in information poor, dynamically altering environments such as rescue operations in regions where natural disasters occur, mapping unexplored subterranian environments, and extraterrestrial planetary exploration. However, an optimal DRL policy is often application specific, and can require days or weeks of continuous training [95, 99].

Bridging the simulation to real-world gap is imperative for effective deployment of DRL for AMR, and other robotics applications [107, 108, 226, 227]. The substantial training time, and possible physical damage caused by collisions during training to reinforce application specific positive behavior renders real-world training markedly expensive.

A diverse variety of proprietary and open-source simulation tools are available for rigid-body dynamics [228, 229, 230]. For sim-to-real applications, high physical accuracy is required of the physics engine. Common physics engines and simulators utilized by researchers include Bullet, Gazebo, MuJoCo and Unreal Engine [82, 231, 232]. In addition to a physics engine, a simulation platform must include high-fidelity 3D environment and AMR models to construct a digital twin.

Hardware and simulation platforms developed in-house using open-source tools were used for training and evaluation. 3D Computer Aided Design (CAD) models of the physical robot sensors, actuators and chassis components were designed to create a high-fidelity digital twin for DRL training in simulation, and zero-shot real-world policy transfer.

eXperimental one-TENTH scaled vehicle platform for Connected autonomy and All-terrain Research (XTENTH-CAR) [211] developed for all-terrain DRL research using the NVIDIA Jetson Orin AGX, the most powerful embedded computer currently available, was used for real-world experiments. The robot, illustrated in Figure 5.1, is modeled by similar physics to a road vehicle, and is controlled using high-level linear velocity and steering angle actions which are converted to low-level propulsion and servo motor RPMs using an open source electronic speed controller. The maximum velocity and steering angle in either direction are $5m/s$ and 0.36^c. The velocity is capped for safety, and can be configured to be higher.

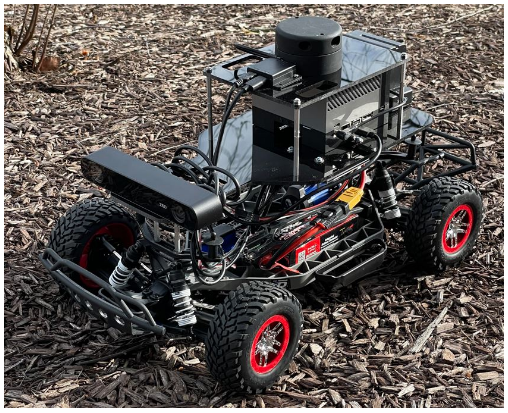


Figure 5.1: XTENTH-CAR wheeled mobile robot.

The platform is equipped with a 2D planar LiDAR, stereo camera, Inertial Measurement Unit (IMU), barometer, magnetometer and two batteries that each power the actuators and computer for increased run time for up to 2 hours on a single charge. The trained DRL policy utilizes under 25% of the available CPU and memory resources enabling smooth operation at high speeds and sampling rates.

AUTOnomous ground Vehicle deep Reinforcement Learning simulator (AutoVRL) [210] packages high-fidelity environment models and a digital twin of XTENTH-CAR, illustrated in Figure 5.2, generated using Unified Robot Description Format (URDF) files with the open source Bullet physics engine [233]. DRL algorithms are implemented using Stable Baselines3 [234] which utilizes the PyTorch Machine Learning (ML) framework [235] interfaced with the physics engine using the Gym Application Programming Interface (API) [236].

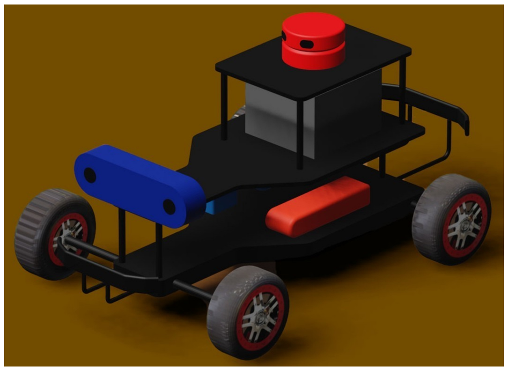


Figure 5.2: XTENTH-CAR digital twin.

AutoVRL includes implementations of GPS, IMU, LiDAR and camera sensors among which the 2D planar LiDAR was used for this work. Light transport and detection were modeled using ray tracing [237] to match real-world physics.

The source code for both platforms can be accessed in Appendix A. The rest of the chapter is organized as follows. Section 5.1 provides background, and describes actuation, computation and data processing, sensing, and evaluation results for the XTENTH-CAR wheeled mobile robot. The details of the AutoVRL simulator are presented in section 5.2.

5.1 Wheeled Mobile Robot

Connected Autonomous Vehicles (CAVs) are integral to the future of transportation, and have great potential to improve traffic flow, safety and fuel efficiency. CAVs are Autonomous Vehicles (AVs) with vehicle connectivity enabled for vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I) and vehicle-to-everything (V2X) communications [214, 238]. CAVs improve upon AV benefits to transportation via acquisition of information beyond the ego vehicle's Field of View (FoV) to achieve superior traffic throughput and energy economy.

All-terrain AMRs are beneficial tools for a wide range of applications, such as disaster response, automated mining, agriculture, military operations, search and rescue missions, and planetary exploration [118]. Intelligent navigation algorithms that incorporate DRL enable AMRs to cognitively chart collision-free motion trajectories in environments without a-priori maps [95, 99], which is a necessity in information poor, dynamically altering environments.

AMRs and CAVs are a continuously evolving research domain with substantial developments over the past decade. The complex nature of real-world traffic environments, and safety considerations pose challenges to experimental research and expeditious advancement of CAV technology. Full-scale experimental vehicles integrated with the CARMA^{SM} (Cooperative Automation Research Mobility Applications) platform developed by the U.S. Department of Transportation (USDOT) Federal Highway Administration (FHWA) [239] that utilize affordable, high quality hardware such as X-CAR [212] facilitate Cooperative Driving Autonomy (CDA) research and development. However, research that necessitates multiple vehicles requires a large, safe experimental environment, and becomes time-consuming and expensive using full-sized vehicles. Moreover, contemporary AMR DRL research is primarily driven by simulation experiments due to the high computation cost required for training and implementation of agents in the real-world.

To address these challenges, XTENTH-CAR, an open-source, cost-effective proportionally one-tenth scaled experimental vehicle platform was developed with best-in-class embedded computing, governed by the same physics as a full-size on-road vehicle.

Proportionally scaled AV testbeds are pertinent to AV and CAV research pipelines, enabling rapid experimental validation of algorithms prior to implementation and testing in full-size vehicles. XTENTH-CAR shares similar hardware and software architectures to the full-size X-CAR platform, enabling cross-platform development and real-world evaluation of novel cooperative perception, localization and motion planning algorithms. Furthermore, XTENTH-CAR is equipped with a state-of-the-art embedded CPU and GPU unit to facilitate experimental AMR Artificial Intelligence (AI) research. The NVIDIA Jetson AGX Orin System on Module (SOM) on XTENTH-CAR is NVIDIA's most capable, newly released SOM, featuring up to eight times greater performance than the previous generation [240]. It contains 2048-core NVIDIA Ampere architecture GPU with 64 Tensor Cores, 12-core Arm 64-bit CPU and 32GB LPDDR5 RAM in a small footprint with high-speed interfaces to support DRL research in a real-world lab environment.

The sensor suite comprises YDLIDAR G2, a 2D planar LiDAR, ZED 2 stereo camera with built-in Inertial Measurement Unit (IMU), barometer and magnetometer, and Vedder Electronic Speed Controller (VESC), an open-source Electronic Speed Controller (ESC) with drivers written for both versions of the Robot Operating System (ROS 1 & ROS 2) to facilitate static and dynamic object detection, mapping, localization and odometry, enabling full CAV and AMR capabilities. XTENTH-CAR's hardware architecture, and data flow between components, are illustrated in Figure 5.3.

Simultaneous Localization and Mapping (SLAM) is implemented using Hector SLAM [241], a technique that uses the Extended Kalman Filter (EKF) for positioning and Gauss-Newton method for mapping. Hector SLAM was chosen for its optional odometry requirement, and

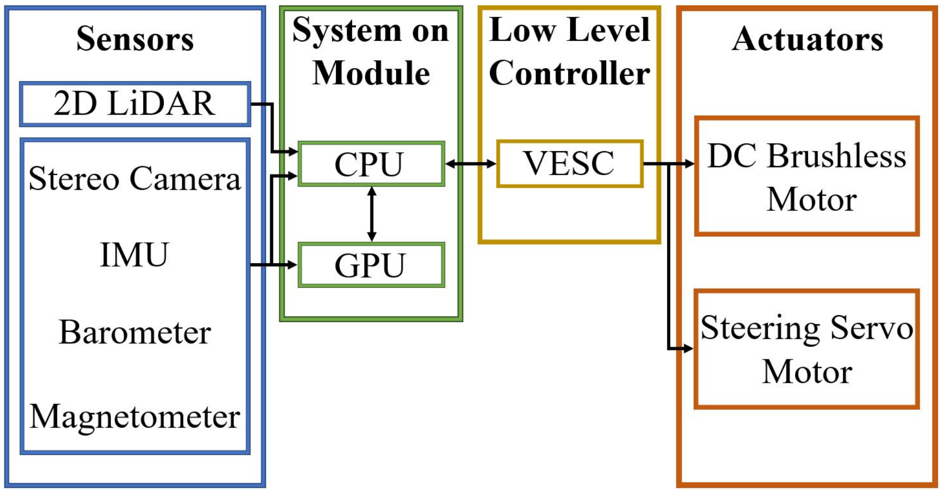


Figure 5.3: XTENTH-CAR hardware architecture and data flow.

low error rates and computation requirements [242]. Mask R-CNN [243], and the Common Objects in Context (COCO) dataset are utilized to validate Computer Vision (CV) and Machine Learning (ML) based object detection, tracking and identification. The CPU and memory usage for perception, control, and motion planning techniques that include collision avoidance using Model Predictive Control (MPC) integrated with Artificial Potential Function (APF) [244], and cognitive exploration utilizing DRL [95] are evaluated, and shown to execute efficiently in real-time.

The rest of the section is organized as follows. Subsection 5.1.1 provides background on past proportionally scaled experimental AVs and CAVs, and the ROS middleware utilized in XTENTH-CAR for communication between sensors and actuators. Subsection 5.1.2 outlines the actuators and chassis used to achieve accurate scaled on-road vehicle dynamics. XTENTH-CAR's computation and data processing SOM is discussed in subsection 5.1.3. The sensor suite utilized to enable a full range of on-road CAV and off-road AMR capabilities is detailed in subsection 5.1.4. Subsection 5.1.5 presents real-world environmental perception, actuator control and computation performance results, and subsection 5.1.6 concludes the section.

5.1.1 Background

Related Work

Early proportionally scaled experimental vehicle platforms utilized off-board computation, where sensor data is transmitted to a separate system for processing, following which control inputs are transmitted back to the vehicle. ETH Zurich built a 1/43 scale Ackermann steered AV platform using this mechanism for experimental evaluation of optimization based autonomous racing [180].

Advancements in computer technology enabled real-time onboard computation in larger 1/10 proportionally scaled AVs. The Berkeley Autonomous Race Car (BARC) developed by University of California, Berkeley utilized an ODROID single-board computer for data processing, and custom wheel encoders to estimate pose and velocity, for autonomous drifting and Learning Model Predictive Control (LMPC) based autonomous racing [245, 246]. Limitations of this platform include low computing capability of the ODROID, and dependence on wheel encoders built using light sensors.

The launch of NVIDIA's Jetson series SOM with embedded CPU and GPU, specifically designed to facilitate robotics research, aided implementation and evaluation of Machine Learning (ML) AV algorithms in scaled platforms. The MIT RACECAR utilized the NVIDIA Jetson TX1 and VESC for accurate odometry coupled with a sensor suite that included 2D LiDAR, stereo camera and IMU for autonomous racing and off-road applications [247].

Our lab at Virginia Tech developed ASIMcar [248], a low-cost scaled CAV platform that utilized the NVIDIA Jetson TX2, VESC for actuator control, and cost effective sensor suite that included an IR proximity sensor, IMU and camera with fisheye lens. XTENTH-CAR is a dual purpose evolution of ASIMcar with a more durable chassis for off-road experiments, and cross-platform architecture compatibility with the full-scale X-CAR CAV platform.

MuSHR developed by University of Washington utilizes the NVIDIA Jetson Nano which is now outdated, and no longer supported [249]. The open-source F1TENTH platform developed by University of Pennsylvania utilizes the NVIDIA Jetson Xavier NX Developer Kit, and provides a testing-oriented simulator [250]. This platform is primarily focused on robotics education and racing. The Xavier NX is lightweight, but not ideal for computationally expensive applications such as AMR DRL research. XTENTH-CAR is specifically built with best-in-class computation to facilitate state-of-the-art experimental CAV and AMR research.

Robot Operating System

ROS is an open-source modular framework and set of tools for robot software development [251, 252] that provides OS functionality on a heterogeneous computer cluster. A process on the loosely coupled ROS middleware, known as a node, is responsible for a specific task. Nodes can transmit or receive data from other nodes using a publish/subscribe model utilizing messages passed through logical channels called topics. XTENTH-CAR utilizes ROS to communicate sensor information to the computation unit for processing, and transmit computed control signals to the actuators.

The original ROS, now referred to as ROS 1, launched in 2007 with its last distribution release being ROS Noetic Ninjemys, built for Ubuntu 20.04 (Focal Fossa) release. It will be supported until May 2025 [253]. The new version of ROS, ROS 2, is a substantial revision of the ROS Application-Program Interface (API) designed to take advantage of contemporary technologies and libraries, and provide embedded hardware and real-time code support [254]. XTENTH-CAR software is written for both ROS 1 Noetic Ninjemys and ROS 2 Foxy Fitzroy to take advantage of the mature ROS 1 ecosystem, while ensuring continued support and improvements to ROS via ROS 2.

5.1.2 Actuation

In order to realistically emulate full-scale CAVs, a scaled platform must accurately simulate on-road vehicle dynamics. XTENTH-CAR is built around a proportionally 1/10 scale Traxxas Slash 4X4 VXL all-terrain Remote Controlled (RC) Ackermann-driven chassis [255]. This detailed chassis, depicted in Figure 5.4, features all-terrain long-travel suspension with oil-filled shock absorbers, heavy-duty driveshafts, slipper clutch for traction control and tunable steel gear differentials.

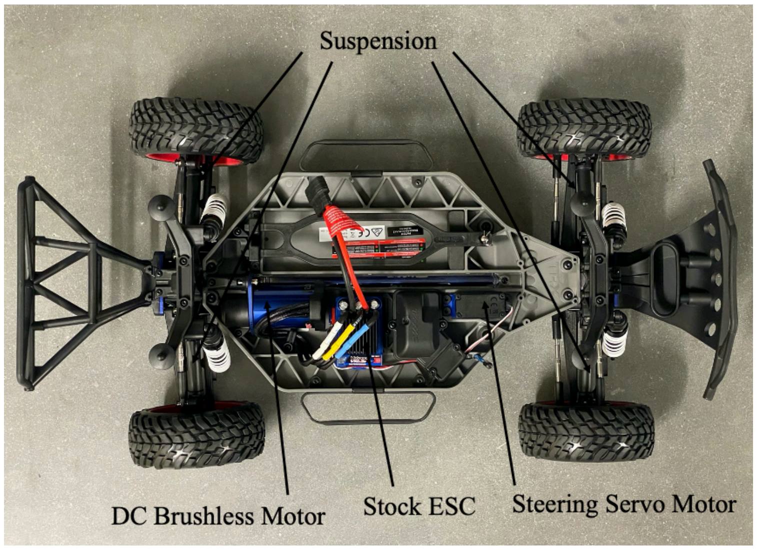


Figure 5.4: Proportionally scaled chassis used for XTENTH-CAR development.

The 10-turn, 3500 kV brushless motor used for propulsion enables a top speed of 60 mph, and a high-torque servo motor facilitates Ackermann steering [256]. The steering angles of the inside and outside wheels, δ_i and δ_o are derived as follows where l_{wb} is the wheelbase, ax_t is the axle track and T_r is the turning radius,

$$\delta_i = tan^{-1}\left(\frac{l_{wb}}{T_r - \frac{ax_t}{2}}\right) \tag{5.1}$$

$$\delta_o = tan^{-1}\left(\frac{l_{wb}}{T_r + \frac{ax_t}{2}}\right) \tag{5.2}$$

The two actuators are controlled using a VESC, an open-source ESC which replaces the stock ESC included with the chassis. The VESC, depicted in Figure 5.5, converts velocity and steering angle commands into motor Pulse Width Modulation (PWM) signals, and relays odometry for localization and motion planning via ROS. A 5000 mAh 11.1 V 3-Cell LiPo Battery is utilized to power the VESC and actuators.

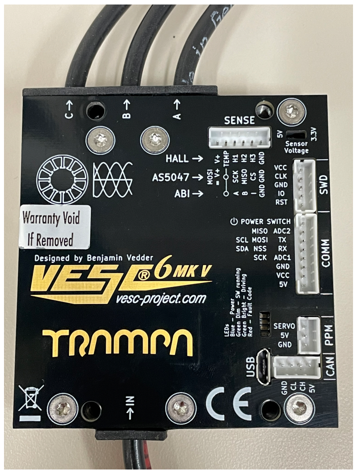


Figure 5.5: VESC open-source Electronic Speed Controller.

The Ackermann control inputs, v and δ are converted to DC brushless motor Revolutions per Minute (RPM), V_m and servo motor position, ϕ_s as follows,

$$V_m = K_m v + m_o \tag{5.3}$$

$$\phi_s = K_s \delta + s_o \tag{5.4}$$

Here the tunable parameters K_m and m_o are AMR velocity to motor RPM gain and offset, and K_s and s_o are steering angle to servo position gain and offset respectively.

The chassis shares similar structure to a full-size on-road vehicle, and is governed by the same physics. The chassis specifications are listed in Table 5.1.

Table 5.1: Chassis Specifications

Parameter	**Length (inches)**
Total Length	23.36
Axle Track	11.65
Wheelbase	12.75
Center Ground Clearance	2.83
Tire Diameter	4.31
Inner Wheel Diameter	3.0
Outer Wheel Diameter	2.2

Tires are interchangeable, and can be swapped for different applications. The stock tires included with the chassis are suitable for indoor CAV experiments. Traxxas canyon trail tires offer greater traction in outdoor environments, thus are a good addition for all-terrain AMR research.

5.1.3 Computation and Data Processing

The NVIDIA Jetson AGX Orin Developer Kit, illustrated in Figure 5.6, provides XTENTH-CAR's computation and data processing requirements. This SOM is NVIDIA's newest, and most powerful, capable of 275 TOPS AI performance and up to eight times greater performance than the previous generation Jetson AGX Xavier. The performance upgrade facilitates real-world DRL research, and high-speed experiments that require fast on-board processing.

Figure 5.6: NVIDIA Jetson AGX Orin Developer Kit.

The small form factor developer kit carries high speed interfaces that include two USB Type-C connectors, four USB Type-A connectors, USB Micro-B connector, DisplayPort, microSD slot and M.2 Key M slot with x4 PCIe Gen4 to facilitate sensor connectivity and expandable storage. The SOM components are listed in Table 5.2.

Table 5.2: System on Module Components

Item	**Component**
CPU	12-core Arm® Cortex®-A78AE v8.2 64-bit
GPU	2048-core NVIDIA Ampere 64 Tensor Cores
Memory	32GB 256-bit LPDDR5
Boot Drive	WD_BLACK 1TB SN770 M.2-2280 NVMe SSD
Internal Storage	64GB eMMC 5.1

The SOM and connected sensors are powered utilizing a 8800 mAh NP-F970 Battery mounted on a SmallRig NP-F Battery Adapter Plate, as illustrated in Figure 5.7. All electronics are mounted on custom acrylic laser cut platforms as shown in Figure 5.8. The use of separate batteries to power the SOC and actuators facilitates long lasting experiments.

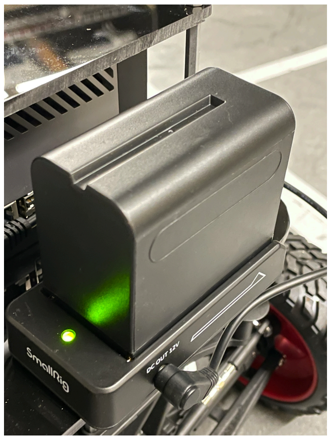


Figure 5.7: NP-F970 Battery and SmallRig NP-F Adapter.

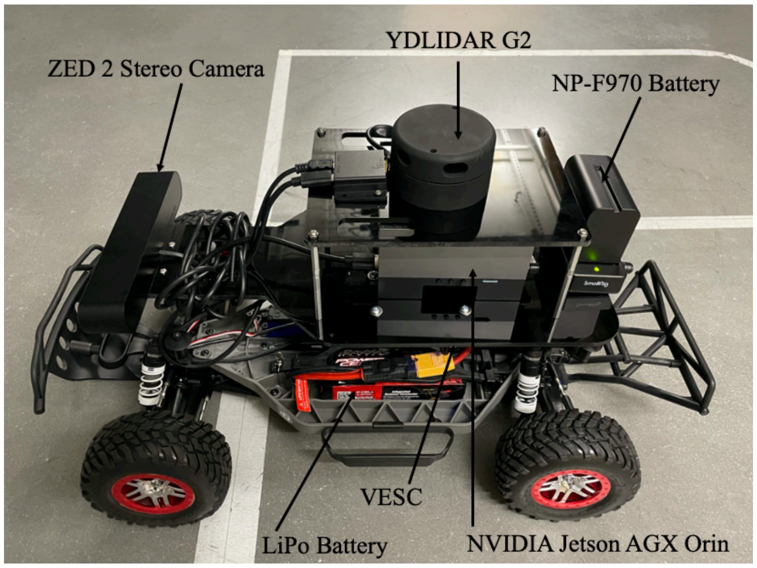


Figure 5.8: Fully assembled XTENTH-CAR.

5.1.4 Sensing

The XTENTH-CAR platform consists of a 120° FoV stereo camera with built-in IMU, barometer and magnetometer, and a 360° 2D LiDAR. Pose and velocity are accurately determined by the VESC, and Wi-Fi is used as a communication interface to interchange information.

Camera

Cameras capture light from the surrounding environment to create images, and convert them into electric signals for processing. Images can be used to detect road surfaces, lanes, traffic signs and lights, and objects. Sequential camera images enable dynamic object tracking, and visual odometry for pose estimation [257, 258]. A ZED 2 stereo camera from Stereolabs, illustrated in Figure 5.9, was used for CV in XTENTH-CAR. The similarly specced, rugged, dust, water and humidity resistant ZED 2i is suitable for outdoor experiments.

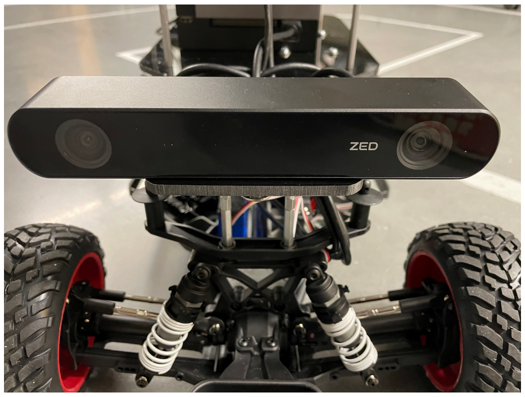


Figure 5.9: ZED 2 stereo camera mounted at the front of the chassis.

The camera system contains two camera sensors that each have an output resolution of 2208x1242 at 15 Frames per Second (fps), 1920x1080 at 30 fps, 1280x720 at 60 fps and 672x376 at 100 fps, with a combined FoV of 120° and depth range of 20 m to simulate human binocular vision, and capture 3D images. Moreover, ZED 2 includes an IMU, a barometer and a magnetometer for pose estimation, and atmospheric pressure and magnetic field measurements. The camera specifications are listed in Table 5.3.

Table 5.3: ZED 2 Stereo Camera Specifications

Characteristic	**Specification**
Max. Output Resolution	2208x1242 at 15 fps
Min. Output Resolution	672x376 at 100 fps
Field of View	120°
Depth Range	0.3-20 m
Depth Accuracy	<1% up to 3 m, <5% up to 15 m
Sensors	IMU, Barometer and Magnetometer

LiDAR

LiDAR (Light Detection and Ranging) is an active, remote sensing method that creates a precise point cloud map of the surrounding environment by emitting pulsed laser beams and measuring the time taken for reflected light to return to the system [259]. The sensor can be used to detect and determine distances of objects from the ego vehicle. Full-size AVs utilize one or more 3D LiDAR sensors for mapping and localization. A YDLIDAR G2, portrayed in Figure 5.10, a 2D 360° LiDAR with a maximum range of 12 m, was used for XTENTH-CAR. The LiDAR specifications are listed in Table 5.4.

Table 5.4: YDLIDAR G2 Specifications

Characteristic	**Specification**
Scan Angle	360°
Scan Frequency	5-12 Hz
Range Distance	0.12-12 m
Range Frequency	5000 Hz
Angle Resolution	0.36°-0.864°

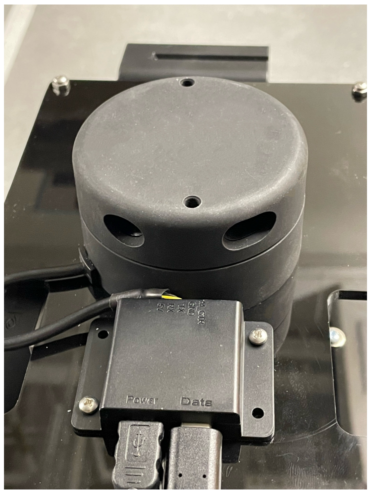


Figure 5.10: YDLIDAR G2 mounted on a level platform.

The LiDAR is mounted 11.5 inches from the surface of the ground, thus provides a 2D planar Point Cloud Map (PCM) of the surrounding environment at this height. This works especially well for CAV and AMR experiments on a small-scale platform, since experiments can accommodate the sensor's capabilities.

3D 360° LiDARs such as the Velodyne Puck draw more power, require substantially greater computation resources, and are heavier, which can lead to the small-scale chassis bottoming out. YDLIDAR G2 is lightweight, and has low computation and power consumption requirements which make it ideal for a scaled experimental vehicle platform.

Communication

CAVs such as X-CAR use Dedicated Short-Range Communication (DSRC) for direct inter-vehicle, and vehicle to infrastructure communication [260]. These one-way or two-way channels are specifically designed for automobiles in the Intelligent Transportation System (ITS) and a corresponding set of protocols and standards, eliminating reliance on cellular equipment and related infrastructure.

XTENTH-CAR utilizes either Transmission Control Protocol/Internet Protocol (TCP/IP) or User Datagram Protocol (UDP) for data transfer via Wi-Fi. The NVIDIA Jetson AGX Orin Developer Kit includes embedded Wi-Fi, and does not require a separate Wi-Fi card that is needed in previous versions.

5.1.5 Results and Discussion

Mapping and Localization

The 2D PCM generated by YDLIDAR G2 is depicted in Figure 5.11, with obstacles and surfaces marked in red. The vehicle's pose is conveyed by the coordinate frame.

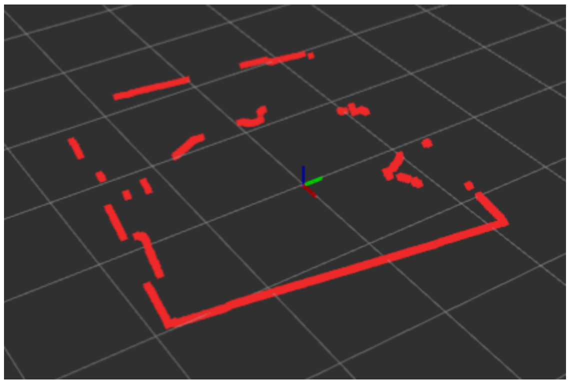


Figure 5.11: 2D point cloud map generated by YDLIDAR G2 with obstacles and surfaces marked in red.

2D LiDAR SLAM is computationally less expensive than visual SLAM [261], which allocates more resources for demanding real-time motion planning and control algorithms such as DRL and Nonlinear Model Predictive Control (NMPC) [262]. Figure 5.12 illustrates a map of an office space at Virginia Tech generated using Hector SLAM.

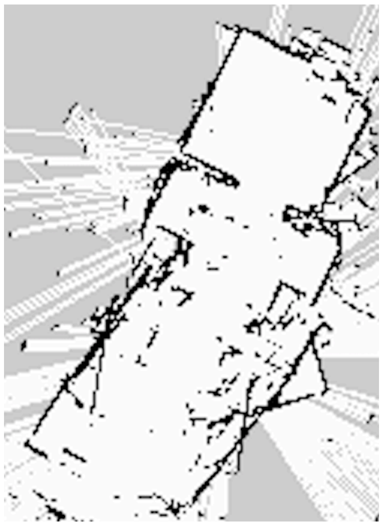


Figure 5.12: Map of an office space at Virginia Tech generated using Hector SLAM.

Object Detection, Tracking and Identification

Camera images and Mask R-CNN were utilized for real-time object detection, tracking and identification as shown in Figure 5.13.

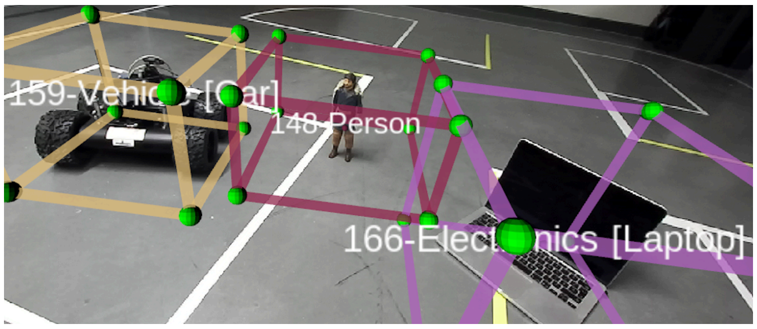


Figure 5.13: Object detection and tracking utilizing Mask R-CNN.

The CV ML algorithm accurately identifies a laptop, a humanoid action figure labelled as a person and a four-wheel differential drive mobile robot labelled as a vehicle. Similar RC humanoid action figures and wheeled mobile robots can emulate pedestrians and road traffic in 1/10th scaled CAV experiments.

Actuator Control

The two actuators, DC brushless motor and steering servo motor, are controlled by the VESC efficiently with minimal noise. Figure 5.14 illustrates the DC brushless motor cycle for a full range of velocity control inputs.

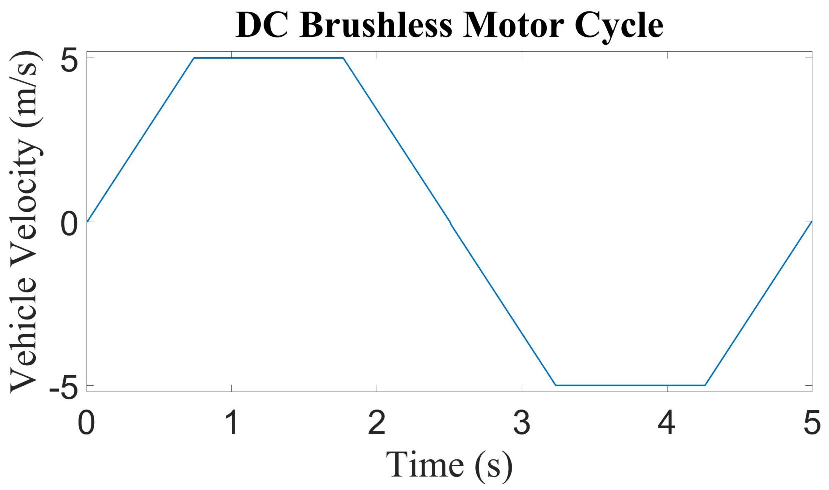


Figure 5.14: DC brushless motor cycle for full range of vehicle velocity control inputs.

Maximum and minimum velocities are set to 5 and -5 m/s by default for safety in indoor environments, but can be configured to be higher. The DC brushless motor accelerates XTENTH-CAR from 0 to 5 m/s in 0.75 s with an acceleration of 6.67 m/s^2. Deceleration performance is equivalent with the vehicle decelerating from 5 to -5 m/s in 1.5 s at -6.67 m/s^2.

The steering servo motor cycle for a full range of steering angle control inputs is illustrated in Figure 5.15.

The maximum and minimum steering angles are 0.36^c and -0.36^c respectively. The steering servo motor directs the front wheels to the maximum steering angle in either direction from 0^c in 0.069 s at a rate of 5.22 rad/s.

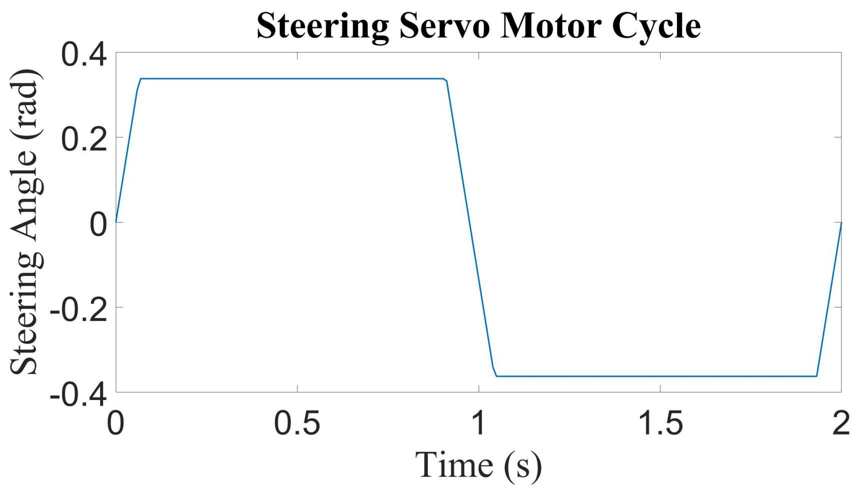


Figure 5.15: Steering servo motor cycle for full range of vehicle steering angle control inputs.

Computation Performance

A range of autonomous tasks were tested to evaluate real-time computation performance. The CPU and memory usage percentages for actuator control, LiDAR PCM, Hector SLAM, CV object detection and tracking, collision avoidance using MPC integrated with APF [244], and cognitive exploration utilizing DRL [95], trained for simulation to real-world policy transfer in AutoVRL [210], a high-fidelity AMR simulator developed using open-source tools for sim-to-real DRL research and development, are listed in Table 5.5.

CPU usage ranges for these tasks from 6.4% to 28.2 %, and memory usage from 8.8 % to 17.0 %. MPC is more CPU demanding than DRL, as it performs more computations at each time step when compared to a trained DRL policy. Conversely, the DRL agent requires more memory to load the trained policy. CV using Mask R-CNN was the most demanding task, nonetheless XTENTH-CAR handles the full perception load, motion planning with MPC or DRL, and actuator control efficiently, utilizing a fraction of the available CPU and memory resources enabling improved real-time performance and high sampling times for high speed experiments.

Table 5.5: CPU and Memory Usage for Various Autonomous Tasks

Task	**CPU Usage %**	**Memory Usage %**
Actuator Control	6.4	8.8
Actuator Control & PCM	7.0	9.0
Actuator Control, PCM & SLAM	14.0	11.7
Actuator Control, PCM & MPC	15.7	11.7
Actuator Control, PCM & DRL	7.6	12.6
Actuator Control, PCM & CV	23.3	15.7
Actuator Control, PCM, CV & MPC	28.2	16.1
Actuator Control, PCM, CV & DRL	24.2	17.0

5.1.6 Summary

This section introduced XTENTH-CAR, a proportionally scaled experimental vehicle platform for connected autonomy and all-terrain research with open-source software stack written for both ROS 1 & ROS 2. The platform utilizes the NVIDIA Jetson AGX Orin Developer Kit for best-in-class computation and data processing to facilitate experimental CAV and AMR research in state-of-the-art computationally expensive domains such as Deep Reinforcement Learning (DRL). After providing background, actuation to emulate a full-scale on-road CAV on a proportionally scaled platform was discussed. Moreover, the embedded computation unit, and sensing suite utilized to produce a complete understanding of the scaled vehicle's surrounding environment, required for experimental evaluation of CAV and AMR motion planning and control research were detailed. A range of autonomous tasks that include LiDAR PCM, SLAM and ML based CV for perception, MPC and DRL for motion planning, and actuator control were tested in real-time to evaluate computation performance, and shown to execute efficiently, utilizing a maximum of 28.2 % CPU and 17.0 % memory usage enabling high sampling times, and improved real-time performance.

5.2 Simulator

AUTOnomous ground Vehicle deep Reinforcement Learning simulator (AutoVRL), is an open-source high-fidelity simulator for sim-to-real AMR DRL research and development built upon the Bullet physics engine [233] utilizing the OpenAI Gym [236] python library to provide an Application Programming Interface (API) for communication between DRL algorithms and environments. State-of-the-art DRL algorithms are implemented using Stable Baselines3 (SB3) [234] which utilizes the PyTorch Machine Learning (ML) framework [235] based on the Torch library.

AutoVRL is equipped with sensor implementations of Global Positioning System (GPS), Inertial Measurement Unit (IMU), LiDAR and camera, high-fidelity model of the XTENTH-CAR AMR platform [211], and realistic environments for training and evaluation. The simulator is extensible, with new environments and AMR models straightforward to add. Dependencies are quick to install, and the provided python interface enables simple SB3 algorithm customization, implementation and evaluation of different DRL libraries or novel DRL algorithms, and execution of simulations.

The similar software frameworks in the simulator and real-world AMR, in Ubuntu with a python interface facilitate simulation to real-world DRL research. Policies learned in simulation are transferable to hardware using an identical software architecture, by swapping the PyBullet python bindings for the Bullet physics engine with a hardware interface, such as the commonly used Robot Operating System (ROS) framework [251, 254], for real-world action control inputs, and sensor observations.

The rest of the section is organized as follows. Subsection 5.2.1 details the simulator's architecture. Environment and AMR models are discussed in subsection 5.2.2. The sensor implementations of GPS, IMU, LiDAR and camera are presented in subsection 5.2.3. Subsection 5.2.4 describes actuation in AutoVRL. Training results in simulation and performance of simulator to hardware policy transfer are discussed in subsection 5.2.5, and subsection 5.2.6 concludes the section.

5.2.1 Architecture

AutoVRL comprises three components: PyBullet, OpenAI Gym and SB3. High-fidelity actions and observations are performed in the Bullet physics engine using PyBullet python bindings, and DRL is implemented via SB3, a set of customizable algorithms based on the PyTorch ML framework. The OpenAI Gym API communicates between the DRL algorithm and the Bullet physics engine. SB3 is easy to replace with alternate DRL libraries or custom DRL algorithms developed in PyTorch or TensorFlow for research evaluation. Figure 5.16 illustrates AutoVRL's architecture and data flow between modules.

Goal specific actions dependent on the reward function are strategized by the DRL algorithm each time step, and simulated in the PyBullet simulation environment via Gym. The updated observations are relayed back from the physics engine to Gym using which the episode termination condition is verified within Gym. This information is transmitted to the SB3 DRL algorithm for learning and policy updates.

The policy learned in simulation is applied to a real-world AMR using the same architecture, by replacing the PyBullet functions for action control inputs and sensor observations with the AMR's hardware interface, such as the ROS framework. In previous research [99], Simulink and the MATLAB Robotics System [208] and Reinforcement Learning [209] Toolboxes were utilized for DRL investigations, however, the use of MATLAB is computationally demanding and not optimal for real-time control of physical AMRs. AutoVRL operates on Ubuntu and provides a Python interface, commonly used in AMRs, enabling sim-to-real DRL research.

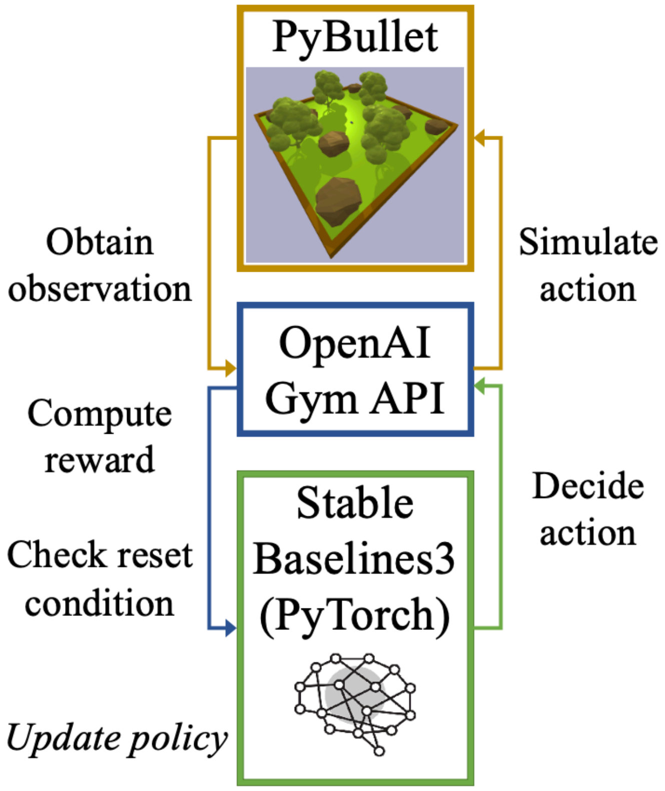


Figure 5.16: Schematic of AutoVRL architecture.

5.2.2 AMR and Environment Models

High-fidelity AMR and environment models are generated from 3D CAD models using Unified Robot Description Format (URDF) files, an eXtensible Markup Language (XML) file type that is also utilized in alternate simulation tools such as Gazebo. The URDF files for AMR and environment components include physical descriptions of each sub-component such as dimensions, joint type, color, and base position in the global coordinate system.

In order to facilitate simulation to real-world AMR research and development, AutoVRL includes a digital twin of XTENTH-CAR [211], a proportionally 1/10th scaled Ackermann steered vehicle for connected autonomy and all terrain research developed with best-in-class embedded processing to facilitate real-world AMR DRL research. XTENTH-CAR shares similar hardware and software architectures to the full-sized X-CAR [212] experimental vehicle, enabling DRL research for on-road Autonomous Vehicles (AVs) in addition to all-terrain AMRs. Figure 5.17 depicts the XTENTH-CAR AMR and digital twin. URDF files for digital twins of alternate, application specific AMRs are straightforward to generate, and load in AutoVRL using 3D CAD models.

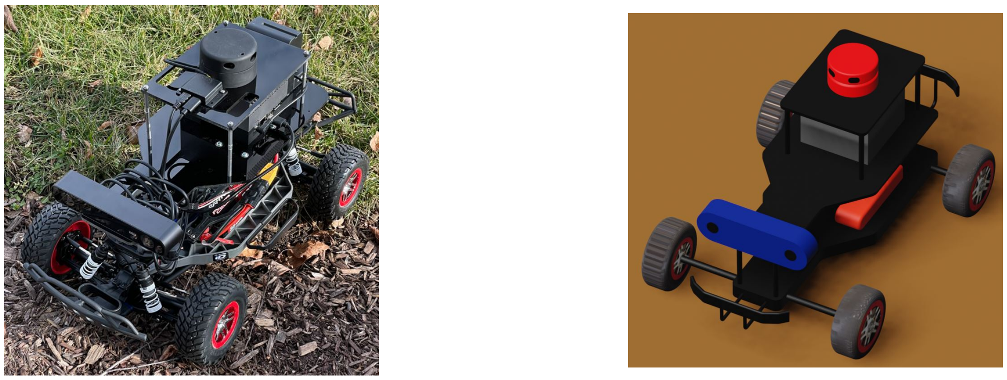
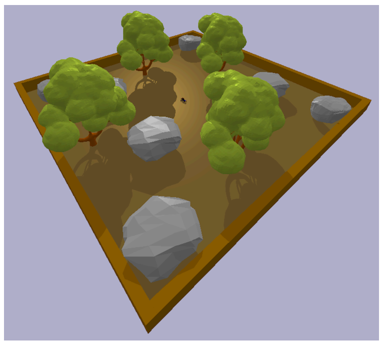

(a) (b)

Figure 5.17: (a) XTENTH-CAR Ackermann steered AMR. (b) XTENTH-CAR digital twin. LiDAR marked in red, and camera in blue.

Five environments are provided for training and evaluation. These include 20m x 20m and 50m x 50m outdoor and urban environments with realistic objects such as trees and boulders in the outdoor map, and buildings and passenger vehicles in the urban scenario, and an oval race track. The environments are illustrated in Figures 5.18 - 5.20.

The environments are simple to enlarge for larger AMRs, by modifying the URDF files, and adding additional objects to populate the map. Application specific scenarios, such as indoor household or office, and subterranean environments can be generated with open-source, or custom CAD models.


(a)

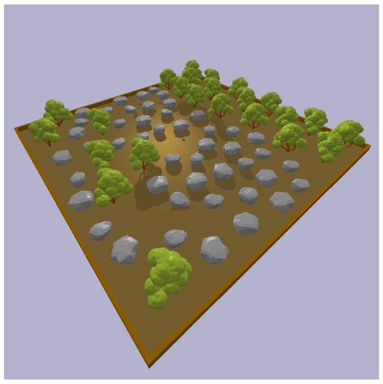
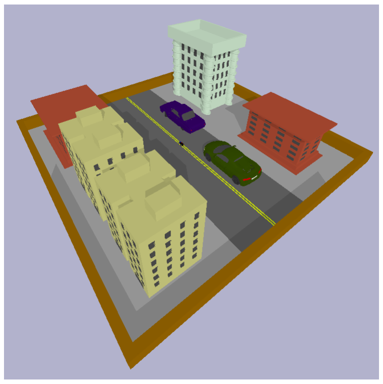

(b)

Figure 5.18: Outdoor environments with tree and boulder objects. (a) 20m x 20m. (b) 50m x 50m.


(a)

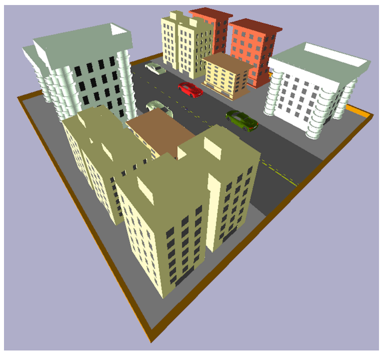

(b)

Figure 5.19: Urban environments with building and passenger vehicle objects. (a) 20m x 20m. (b) 50m x 50m.

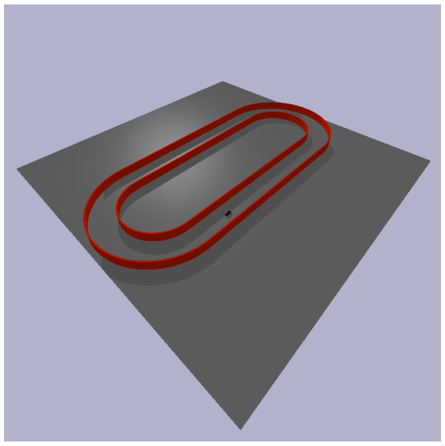


Figure 5.20: Oval race track environment.

5.2.3 Sensing

This section presents AutoVRL's sensor suite. The simulator is equipped with implementations of key AMR sensors that include GPS, IMU, LiDAR and camera.

GPS

GPS is a Global Navigation Satellite System (GNSS) that provides precise geo-spatial positioning. Global position and orientation of the simulated AMR are extracted using the PyBullet getBasePositionAndOrientation function. The position (x, y, z) is in Cartesian world coordinates, and the orientation is a quaternion of the form (x, y, z, w). The getEuler-FromQuaternion PyBullet function is used to convert the quaternion orientation to Euler angles, roll, pitch and yaw (ϕ, θ, ψ).

IMU

IMUs measure acceleration, angular rates and orientation. Linear and angular velocities in Cartesian global coordinates of the form $(\dot{x}, \dot{y}, \dot{z})$ and $(\dot{\phi}, \dot{\theta}, \dot{\psi})$ are acquired using the getBaseVelocity PyBullet function. Accelerations, both linear and angular, are computed using this information, and the sampling time t_s.

LiDAR

LiDAR (Light Detection and Ranging) detects and determines distances of objects from the ego vehicle by measuring the time taken for reflected pulsed laser beams to return to the system [259]. AutoVRL utilizes ray tracing, a technique to model light transport, and compute light rays hitting and reflecting off surfaces [263], to simulate LiDAR. The ray tracing implementation of LiDAR in AutoVRL is illustrated in Figure 5.21 where the red rays convey detected objects, and green rays miss obstacles.

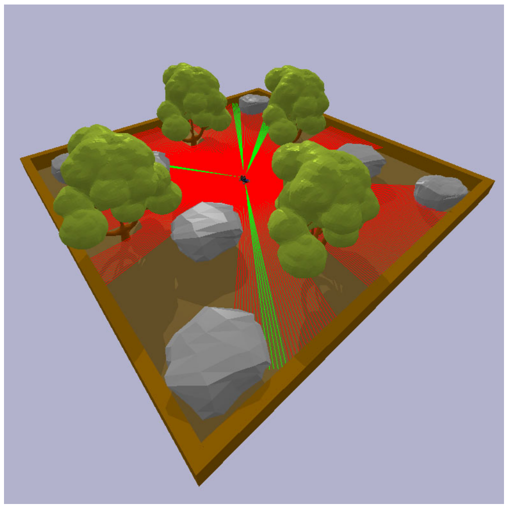


Figure 5.21: Ray tracing implementation of LiDAR. Red rays hit obstacles, and green rays detect none.

The origin coordinates of the rays (x_{RF}, y_{RF}, z_{RF}) are computed as follows,

$$x_{RF} = x_{L,A}cos(\psi_A) + x_A \tag{5.5}$$

$$y_{RF} = y_{L,A} sin(\psi_A) + y_A \tag{5.6}$$

$$z_{RF} = z_{L,A} + z_A \tag{5.7}$$

Here $x_{L,A}$, $y_{L,A}$ and $z_{L,A}$ are coordinates of the LiDAR with respect to the AMR's body frame, x_A, y_A and z_A represent the position of the AMR in the global coordinate frame, and ψ_A is the AMR's yaw.

The position of the end point of n rays (x_{RT}, y_{RT}, z_{RT}) with a maximum ray length r_{max} are generated as follows,

$$x_{RT} = r_{max} sin\left((\frac{\pi}{4} - \psi_A) + \eta(\frac{1:n}{n})\right) \tag{5.8}$$

$$y_{RT} = r_{max} cos\left((\frac{\pi}{4} - \psi_A) + \eta(\frac{1:n}{n})\right) \tag{5.9}$$

$$z_{RT} = z_{L,A} \tag{5.10}$$

Here $(\frac{\pi}{4} - \psi_A)$ represents the LiDAR's orientation, and $\eta(\frac{1:n}{n})$ segments a specified angle η, which in Figure 5.21 is 2π, to n equally spaced rays.

The hit position of each ray (x_{HP}, y_{HP}) is obtained from the PyBullet rayTestBatch function, using which the euclidean distance to detected obstacles r_o is determined,

$$r_o = \sqrt{(x_{HP} - x_{RF})^2 + (y_{HP} - y_{RF})^2} \tag{5.11}$$

Camera

Cameras convert light captured from the surrounding environment into electric signals for image processing. Images are used to detect and identify objects, track dynamic objects, and estimate pose via visual odometry.

AutoVRL renders images from an arbitrary AMR camera position using PyBullet's OpenGL GPU visualizer and CPU renderer. Two 4 by 4 matrices, the view matrix and the projection matrix are used to specify the synthetic camera using the computeViewMatrix and computeProjectionMatrixFOV PyBullet functions.

Images are generated from the view and projection matrices utilizing the getCameraImage PyBullet function, which returns a RGB image, a depth buffer, and a segmentation mask buffer with unique object IDs of detected objects at each pixel.

5.2.4 Actuation

The action outputs from the DRL algorithm are normalized in the range [-1 1]. These are converted to vehicle control inputs, which for an Ackermann steered AMR are throttle T in the range [0 1], and steering angle δ in the range $[\delta_{min}\delta_{max}]$ computed as follows,

$$T = min(max(a_T, 0), 1) \tag{5.12}$$

$$\delta = max(min(a_\delta, \delta_{max}), \delta_{min}) \tag{5.13}$$

Here a_T and a_δ are the normalized action outputs strategized by the DRL policy.

The vehicle control inputs are applied to the simulated AMR using the setJointMotorControlArray PyBullet function. The number of joints the control inputs are applied to is dependent on the AMR model. For an Ackermann steered AMR, the steering angle is applied to two joints, and joint speed to four. Figure 5.22 provides a generalized schematic of actuation in AutoVRL agnostic of AMR model.

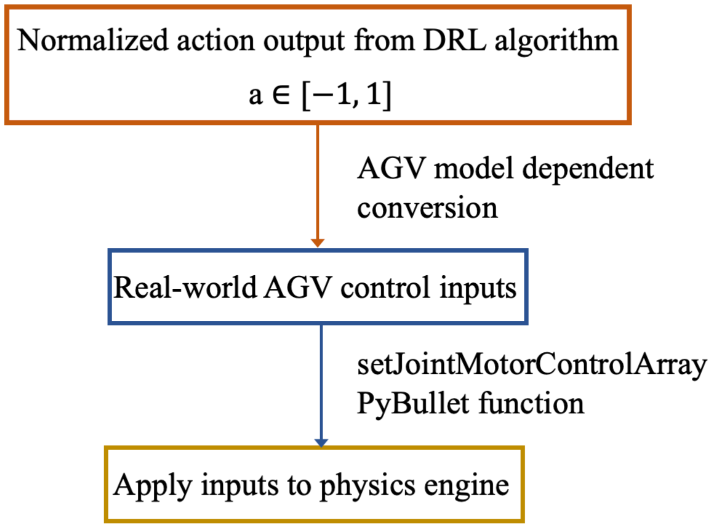

Normalized action output from DRL algorithm
a ∈ [−1, 1]
AGV model dependent conversion
Real-world AGV control inputs
setJointMotorControlArray PyBullet function
Apply inputs to physics engine

Figure 5.22: Actuation in AutoVRL agnostic of AMR model.

Resistive forces are accounted for to improve the fidelity of the simulated action. Friction f is computed for the current time step using drag constants C_d and C_r, and the joint speed from the previous time step $v_{j,i-1}$,

$$f = v_{j,i-1}(v_{j,i-1}C_d + C_r) \tag{5.14}$$

The acceleration of each joint is determined as follows, from the throttle input T, throttle constant C_T and resistive force f,

$$\dot{v}_{j,i} = C_T T - f \tag{5.15}$$

The speed of each joint for the current time step $v_{j,i}$ is computed as,

$$v_{j,i} = v_{j,i-1} + t_s \dot{v_{j,i}} \tag{5.16}$$

5.2.5 Results and Discussion

The training and evaluation results for two applications which each utilize a custom reward formulation are presented in this section. The first is a search application where the AMR is trained to explore shaded regions unobservable to aerial surveillance to detect subjects of interest. The exploration policy was learned using LiDAR range sensor observations, and Computer Vision (CV) was utilized to detect and identify subjects. The second policy was trained for autonomous racing. Moreover, the policies learned in simulation were implemented on a physical XTENTH-CAR to gauge real-time computation performance. An Intel Core i9 13900KF CPU and NVIDIA GeForce RTX 4090 GPU were used for training.

Exploration and Search

AMRs frequently work in tandem with Unmanned Aerial Vehicles (UAVs) in search and exploration applications such as military and Search and Rescue (SAR) operations. AMRs are especially beneficial in exploring regions with poor aerial visibility such as areas with thick foliage, and high obstacle density. The reward R_{search} was formulated to promote exploration in dense, obstacle filled terrain as follows, and trained using the Soft Actor-Critic (SAC) [205] algorithm which was found to be the best sample efficient off-policy DRL algorithm for AMR navigation in prior work [95].

$$R_{search} = \begin{cases} 0.005r_m^2 + 5.0T^2 - 2.0\delta^2 & \\ 2.0 & \text{if } 2.0 < r_m < 2.5 \\ -50 & \text{if } r_m < 1.0 \end{cases} \tag{5.17}$$

Here r_m is the minimum LiDAR range distance, which determines the AMR's displacement to the nearest obstacle. A positive reward of 0.005 was applied to the square of r_m to incentivize the agent to traverse trajectories devoid of obstacles. Positive and negative rewards of 5.0 and -2.0 were applied to throttle, T and steering angle, δ control inputs respectively to encourage exploration, and prevent learning of a sub-optimal trajectory with repeated circular motion in the same vicinity. A reward value of 2.0 was assigned when the agent was between 2.0 and 2.5 m of the nearest object to promote exploration near obstacles, that are often inaccessible to aerial surveillance. A penalty of 50.0 was applied when the agent was within 1 m of the nearest obstacle to improve the safety of exploration trajectories.

The default SB3 SAC hyperparameters were used to train the agent for 50,000,000 steps over 9.5 days in the 20m x 20m outdoor environment. The learning curve is shown in Figure 5.23.

The order 100,000 moving average return increased rapidly during the first 1,000,000 training steps, and fluctuated around a value of 3.9 over the remainder of the training period. The minimum average return increased as training progresses which indicates that an improved policy with better convergence can be learned over a longer training period. Figure 5.24 illustrates single vehicle post-training trajectories in the training environment evaluated at various combinations of AMR initial position and SAR subject location.

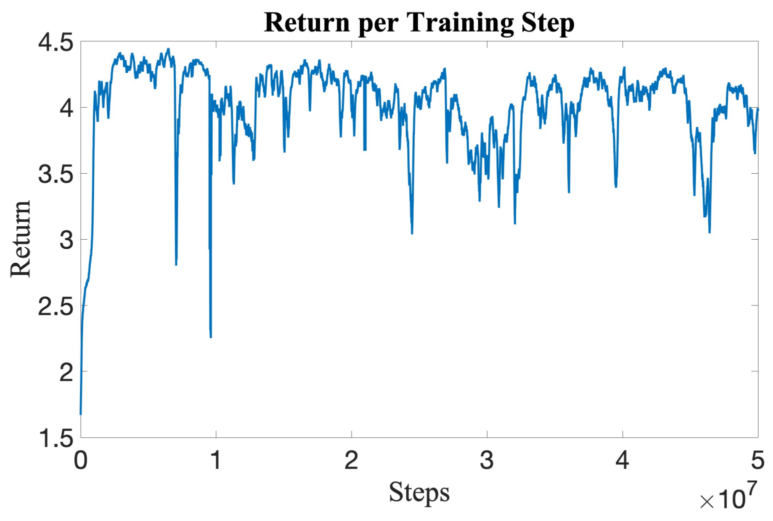


Figure 5.23: Order 100,000 moving average return during training in the $20m$ x $20m$ outdoor environment.

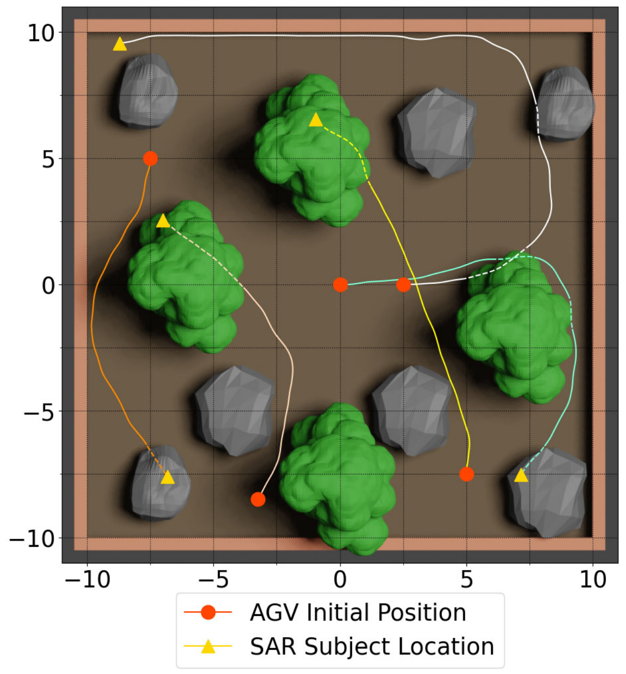


Figure 5.24: Post-training AMR trajectories in the outdoor environment from various initial positions.

The AMR located SAR subjects underneath trees and boulders that are unobservable to aerial surveillance, while maintaining a safe distance to obstacles solely utilizing raw sensor data with no prior knowledge of map characteristics. The trained agent was further evaluated in the $50m$ x $50m$ urban environment to test the extensibility, and robustness of the learned policy. Figure 5.25 illustrates single vehicle post-training trajectories in the evaluation environment.

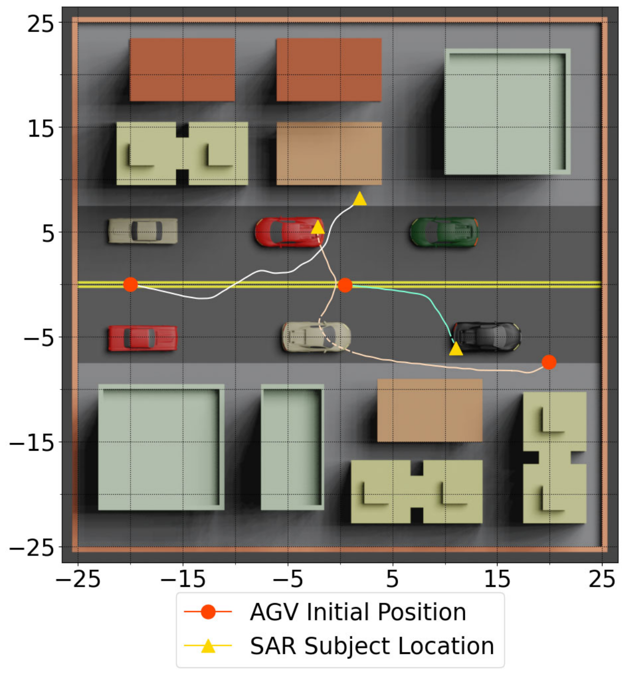


Figure 5.25: Post-training AMR trajectories in the urban environment from various initial positions.

The difference in map and object characteristics in the evaluation environment presented a new challenge, however, the agent was successfully able to strategize efficient, collision free trajectories to explore shaded regions beneath buildings, and underneath vehicles. A longer training period, tuned hyperparameters and refined reward function will enable more efficient learning, and yield an improved policy to chart more complex trajectories to explore the surrounding environment indefinitely.

Autonomous Racing

Motorsport drives automobile innovation, and autonomous racing has gained increasing popularity in recent years. A conservative reward function, R_{racing} was formulated as follows to promote safe, time-optimal trajectories to gauge the effectiveness of the AutoVRL simulator. The agent was trained for 10,000,000 steps over 1.4 days using Proximal Policy Optimization (PPO) [202], an on-policy DRL algorithm which is less sample efficient than SAC, but potentially more robust to hyperparameter tuning, which is of benefit to simulation to real-world applications. The learning curve and post-training trajectory are illustrated in Figures 5.26 and 5.27.

$$R_{racing} = \begin{cases} 5.0T^2 - 2.0\delta^2 & \\ -50 & \text{if } r_m < 0.8 \end{cases} \tag{5.18}$$

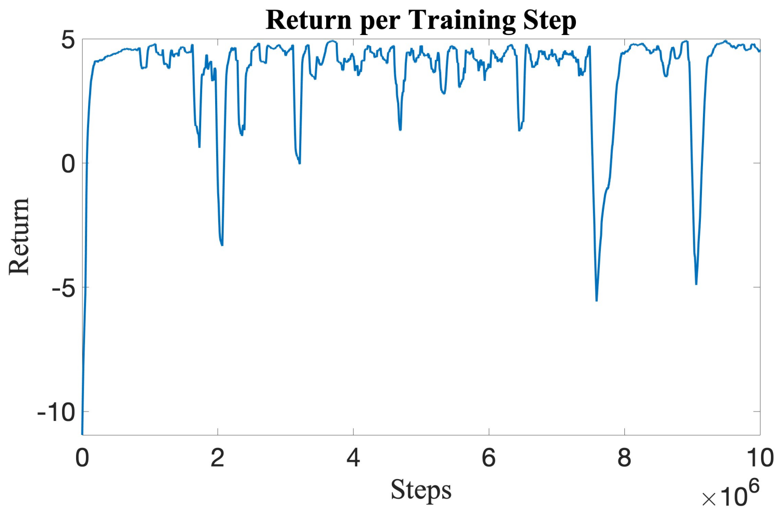


Figure 5.26: Order 100,000 moving average return during training in the oval racetrack.

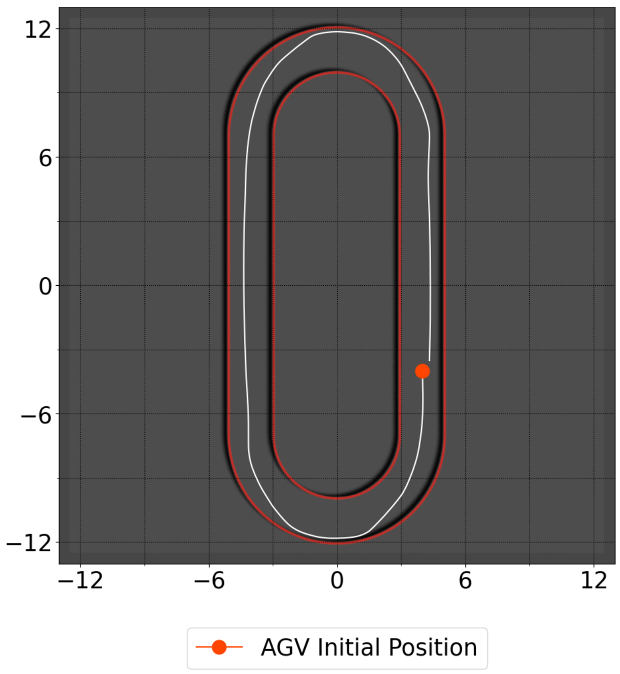


Figure 5.27: Post-training AMR trajectory in the oval racetrack.

The order 100,000 moving average return rapidly increased during the first 600,000 training steps to a value of 4.6. The trajectory at the end of training was collision-free, but not time-optimal as the agent traversed around the longer outside curve of the corner. A time-optimal trajectory in which the agent switches lanes to the outside, and swoops to the inside of the corner before exiting can be learned with a refined reward function. The results demonstrate the versatility of AutoVRL for a variety of AMR applications.

Real-World Performance

In order to test simulation to real-world performance, the policies learned in simulation were implemented on a physical XTENTH-CAR equipped with the NVIDIA Jetson AGX Orin System on Module (SOM). The AutoVRL ROS driver utilizes real-world sensor data to select actions using the policy learned in simulation, and applies the chosen control signals to the AMR's actuators in real-time. Table 5.6 lists the average CPU and memory usages for real-time actuator control, LiDAR Point Cloud Map (PCM), CV using Mask R-CNN and DRL for navigation.

Table 5.6: Real-World CPU and Memory Usage.

Task	**CPU Usage %**	**Memory Usage %**
Actuator Control, PCM & DRL	7.6	12.6
Actuator Control, PCM, CV & DRL	24.2	17.0

Despite the high training cost, real-time control using a trained DRL policy is significantly less computationally demanding. The Jetson AGX Orin utilized peak CPU and memory usage percentages of 24.2 and 17.0. The trained policy is under 10 MB, and works seamlessly in both the physics engine in which it can be trained for days or weeks, and on real-world hardware.

5.2.6 Summary

This section introduced AutoVRL, a high-fidelity AMR simulator for simulation to real-world DRL research and development that is beneficial for DRL applications that require days or weeks of continuous training. The simulator is built using open-source tools, and provides access to state-of-the-art DRL algorithms for a range of AMR applications. A digital twin of the XTENTH-CAR AMR and 3D environments that include racetracks, and outdoor and urban surroundings are included with AutoVRL. The simulator is extensible, with application specific AMR and environment models straightforward to add. DRL agents for search applications, and autonomous racing were trained in AutoVRL using off-policy and on-policy algorithms, and shown to learn favorable policies using custom reward formulations. Moreover, the trained policies were implemented on the real-world XTENTH-CAR AMR to assess real-time performance on an embedded computer, and demonstrated to utilize a fraction of the available CPU and memory resources.

Chapter 6

Methodology

The DRL model was trained in simulation in a constrained racetrack to navigate the environment time-optimally at physical limits, and transferred zero-shot to the real-world for OOD generalization to new racetrack layouts, navigation in unstructured terrain, and dynamic obstacle avoidance. Figure 6.1 illustrates the training and deployment schematic.

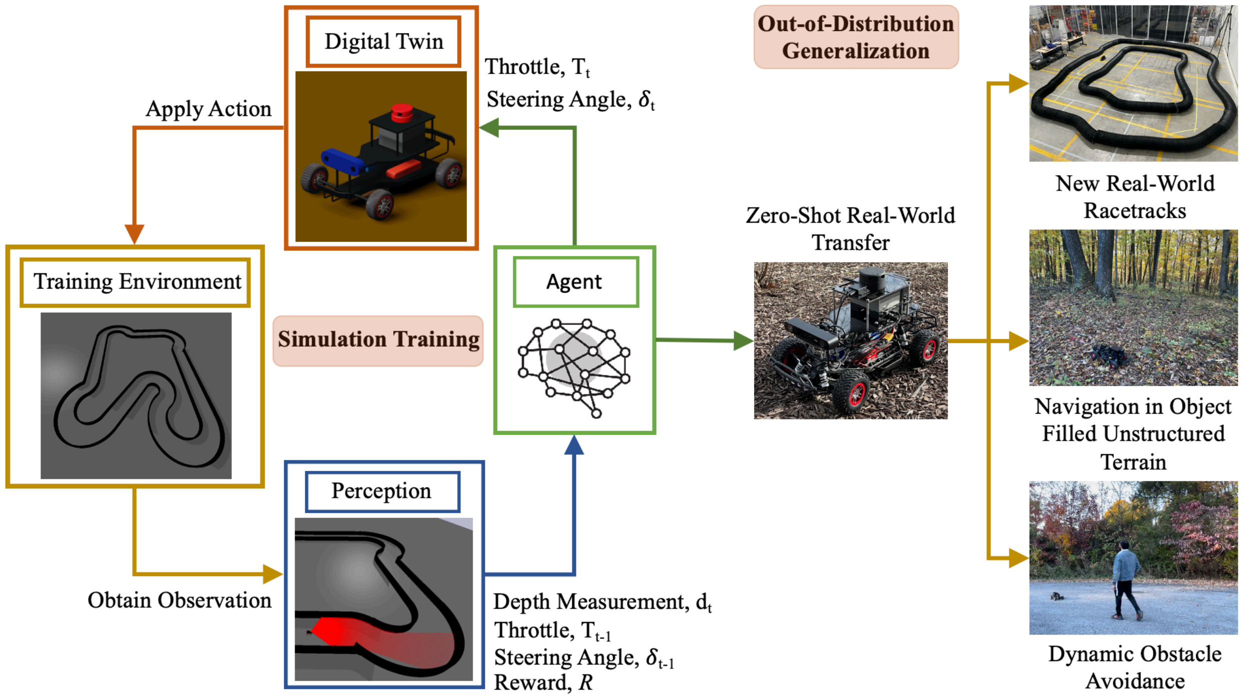

Digital Twin
Throttle, T_t
Steering Angle, δ_t
Apply Action
Out-of-Distribution Generalization
New Real-World Racetracks
Zero-Shot Real-World Transfer
Training Environment
Simulation Training
Agent
Navigation in Object Filled Unstructured Terrain
Perception
Obtain Observation
Depth Measurement, d_t
Throttle, T_{t-1}
Steering Angle, δ_{t-1}
Reward, R
Dynamic Obstacle Avoidance

Figure 6.1: Methodology for training and deployment. The DRL model was trained in simulation in a constrained racetrack at physical limits and transferred zero-shot to the real-world for OOD generalization to new racetrack layouts, exploration in unstructured terrain and dynamic obstacle avoidance.

The policy $\pi_\theta(a_t \mid \mathcal{O}_t)$ was trained using Proximal Policy Optimization (PPO) [202], an online on-policy actor-critic algorithm [206], to compute continuous actions in action space a for continuous observations in observation space $\mathcal{O}$ at each discrete time step t, to maximize expected future rewards utilizing a DNN parameterized with weights θ. PPO was compared to the best off-policy DRL algorithm for AMR exploration determined from prior work, SAC, with a combination of reward formulations, and found to perform better in the constrained racetrack environment for OOD generalization, despite SAC outperforming the on-policy algorithm significantly when trained in obstacle filled environments.

PPO approximately maximizes the objective function $L_t(\theta)$ each iteration utilizing stochastic gradient ascent, where $\hat{\mathbb{E}}_t$ represents the empirical average for a finite batch of samples, c_1 and c_2 are coefficients, and S is an entropy bonus to augment the objective to ensure exploration,

$$L_t(\theta) = \hat{\mathbb{E}}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](\mathcal{O}_t)] \tag{6.1}$$

The actor and critic networks each comprise 2 fully connected layers with 64 nodes each with *tanh* activation that share parameters for efficient computation. The policy objective function $L_t^{CLIP}(\theta)$ utilized by the actor network is computed as follows, where $r_t(\theta) = \frac{\pi_\theta(a_t|\mathcal{O}_t)}{\pi_{\theta old}(a_t|\mathcal{O}_t)}$, ϵ is a hyperparameter, and $\hat{A}_t$ is an estimator for the advantage function computed over T time steps using discount factor γ,

$$L_t^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{6.2}$$

The probability ratio $r_t(\theta)$ is clipped to avoid excessively large updates to the policy. The value function squared-error loss $L_t^{VF}(\theta)$ utilized by the critic network is computed as follows,

$$L_t^{VF}(\theta) = (V_\theta(\mathcal{O}_t) - V_t^{target})^2 \tag{6.3}$$

The actor selects actions to maximize the expected return of the policy, while the critic evaluates the quality of the chosen actions by minimizing the error between the estimated return V_t^{target} and the actual return $V_\theta(\mathcal{O}_t)$ during training. Post training completion, the actor network is solely used during inference to compute actions for each observation.

6.1 Observation Space

Depth observations that can be obtained using LiDAR or camera RGB-D point cloud were used as observations since these provide a complete spatial understanding of the surrounding environment at a fraction of the data size of RGB images enabling faster learning of policies, and efficient real-time execution, similar to animals that use echolocation as a result of natural evolution, such as bats [201]. Moreover, the domain gap between simulation and real-world depth measurements is substantially smaller than for images which facilitates more accurate end-to-end policy transfer without high-level feature extraction from raw sensor inputs. The observation space $\mathcal{O}$ is defined as follows where r_i is the depth measurement at each angle increment in the observation FoV,

$$\mathcal{O} = [r_i]^{1\times170} \tag{6.4}$$

The 1D observation space comprises 170 depth measurements spread across a frontal 120° FoV. The size of the observation space was found to have a significant effect on the learning ability of the agent. Higher resolution observations with 1000 measurements resulted in substantially slower learning and an inability to converge due to the less pronounced differences during object detection. Conversely, low resolution observations with 100 measurements were prone to missing small objects. The minimum resolution of 170, determined through empirical experimentation, provided sufficient detail for detection of small objects with a 5 *cm* radius with pronounced alteration in the space for detected object profiles that enabled quick learning of accurate behavior. A frontal 120° FoV was chosen over a 360° FoV to also enable faster learning of desirable behavior as a consequence of the agent being more focused on objects directly in front. Larger FoVs took longer to converge, and did not improve obstacle avoidance ability or exploration efficiency. This matches natural evolution where animals perceive the surrounding environment without compromise efficiently with a limited FoV.

6.2 Action Space

The DRL algorithm selects normalized continuous actions a_T and a_δ in the range [-1 1] for each observation. The two dimensional vector action space a is defined as follows,

$$a = (a_T, a_\delta); \quad a_T, a_\delta \in [-1, 1] \tag{6.5}$$

High-level control inputs, throttle T in the range [0 1], and steering angle δ in the range $[\delta_{min}\delta_{max}]$ were computed from the normalized actions as follows,

$$T = min(max(a_T, 0), 1) \tag{6.6}$$

$$\delta = max(min(a_\delta, \delta_{max}), \delta_{min}) \tag{6.7}$$

Throttle control commands were linearly mapped to the AMR's linear velocity. No further processing was performed on the steering control inputs.

6.3 Reward Formulation

Shaped and sparse rewards were tested with the former penalizing collisions and states close to racetrack boundaries in addition to rewarding the square of the throttle output, and the latter solely rewarding high throttle output. The sparse reward performed better, both for racing, and OOD generalization to navigation in unknown environments, yielding optimal trajectories in simulation. Shaped rewards that were identical to, and variations of, those presented in Chapters 3 and 5, resulted in excessive wall avoidance behavior that slowed lap time.

The sparse reward formulation T^2 maximized velocity at each state subject to the AMR's unknown dynamics, however resulted in excessive swerving when transferred to the real-world. To correct this, the following reward was formulated where $a_{\delta,prev}$ and $a_{\delta,curr}$ are unprocessed steering angles determined by the learning agent at the previous and current time steps,

$$R = \begin{cases} 5T^2 & \\ -2.0 & \text{if } a_{\delta,prev}a_{\delta,curr} = -1 \end{cases} \tag{6.8}$$

The product of the previous and current steering angle outputs was penalized if it was negative at the upper bound to mitigate imperfections in the physics engine that the agent exploited during accelerated training to learn rapidly oscillating steering angle outputs. The correction component was formulated after observing the initial policy transfer and error patterns to re-train the model with the difference between simulator and real-world physics accounted for, which resulted in no increase in training time or performance in simulation, but transferred zero-shot to the real-world. The weights of each reward component were chosen to prioritize maximizing the throttle output for time-optimal performance.

6.4 Training and Evaluation

Training was performed in a handcrafted multidirectional racetrack with varying cornering radii, as illustrated in Figure 6.2, using an Intel Core i9 13900KF CPU and NVIDIA GeForce RTX 4090 GPU for 20,000,000 steps that corresponded to 15,747 training episodes, accelerated at 30 times real-time speed, in 48 wall-clock hours. An episode concluded when the agent collided with the racetrack boundary, following which it was reset to the same starting position.

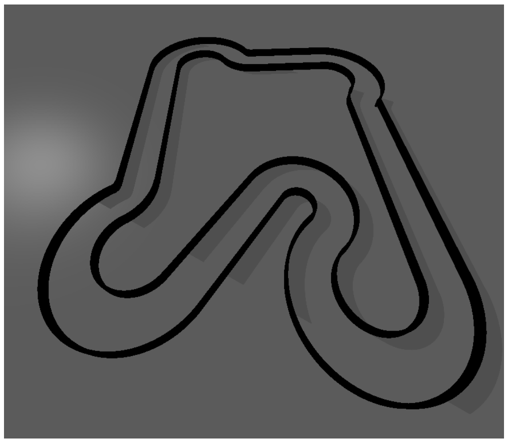


Figure 6.2: Multidirectional racetrack used for training.

The mixture of turning directions and lengths prevented the model from overfitting to a specific layout to enable OOD generalization by providing an array of observation sequences in a constrained environment for learning of obstacle avoidance behavior at the maximum attainable velocity within the laws of physics, which required planning sequences of actions using detected object profiles, that facilitated dynamic obstacle avoidance at a sufficient sampling rate. The hyperparameters used for training are summarized in Table 6.1.

Table 6.1: Training Hyperparameters

Hyperparameter	**Value**
Discount Factor (γ)	0.99
Learning Rate	0.0003
Rollout Buffer Size	2048
Batch Size	64
Number of Epochs	10

OOD generalization for navigation was tested in simulation in a densely cluttered 50m x 50m outdoor environment, illustrated in Figure 6.3, that included tree and boulder objects with explorable regions underneath that are unobservable to aerial surveillance, where ground AMRs are beneficial for exploration.

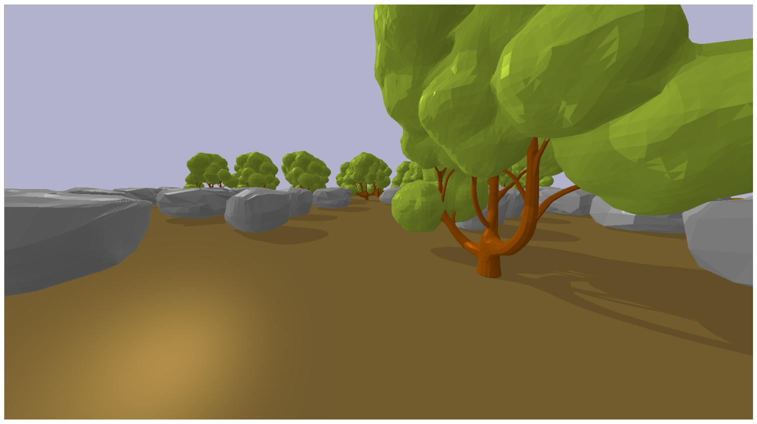


Figure 6.3: Simulated outdoor environment used for evaluation of OOD generalization.

Real-world policy transfer was tested in five scenarios that include racing in different racetrack layouts to the training environment, navigation in an object filled laboratory environment, evaluation of exploration behavior in a parking lot, navigation in unstructured forest terrain, and dynamic obstacle avoidance.

Racing performance was evaluated in three unknown racetracks with energy absorbent barriers, as depicted in Figure 6.4 to gauge zero-shot policy transfer, and generalization to new configurations outside the training distribution.

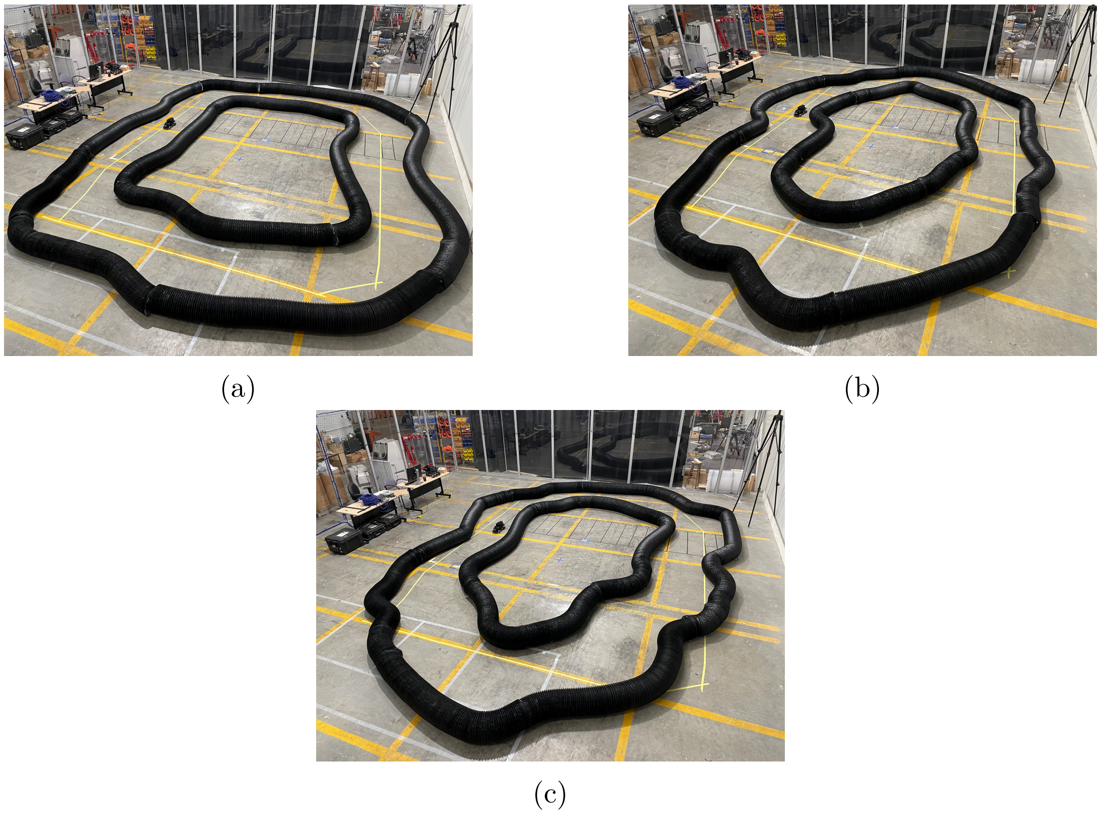
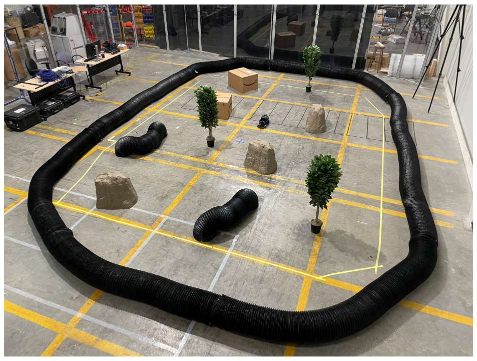
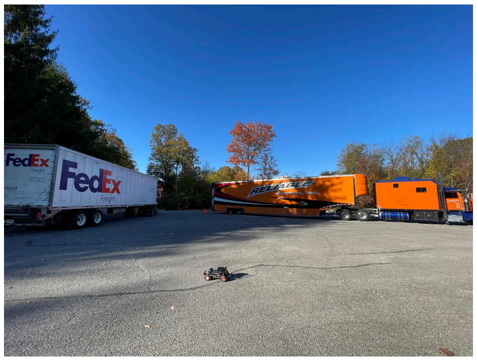

(a) (b) (c)

Figure 6.4: Real-world racetrack layouts used to evaluate zero-shot simulation to real-world policy transfer. (a) Straight and simple corners. (b) Long cornering length. (c) Multiple cornering directions in quick succession.

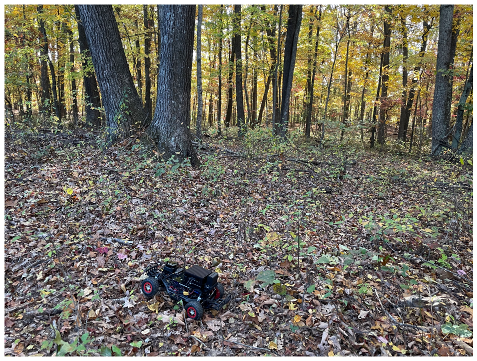
OOD navigation capabilities were tested in a laboratory obstacle filled experimental environment depicted in Figure 6.5a that was populated with a range of obstacle shapes and sizes. Exploration behavior and efficiency were tested in a parking lot, illustrated in Figure 6.5b, and robustness in unstructured terrain with increased sensor noise was evaluated in a forest, depicted in Figure 6.5c. Dynamic obstacle avoidance was evaluated for social navigation for pedestrian avoidance at an operation speed of $1m/s$.



(a)



(b)



(c)

Figure 6.5: Environments used to evaluate exploration without a-priori maps. (a) Laboratory obstacle filled experimental environment. (b) Parking lot. (c) Forest.

6.5 Safety Supervisor

In order to prioritize collision-free motion, a safety supervisor was devised using the range sensor's complete frontal 120° observation space $\mathcal{O}_{sensor}$, which is 10 times larger than the DRL agent's observation space. This facilitated safe maneuvers in tight regions such as 90° wall boundaries in indoor environments that are particularly challenging for an Ackermann steered AMR, by providing reverse throttle commands in a specific steering direction dependent on the region, right, r_r or left, r_l past the center of $\mathcal{O}_{sensor}$ that the minimum range measurement r_{min} was below the critical safety range measurement r_{crit}. The formulation for the safety supervisor is as follows where A_{ss} is the set of executed actions (T_{ss}, δ_{ss}) at each time step,

$$A_{ss} = (T_{ss}, \delta_{ss}) = \begin{cases} (T_{DRL}, \delta_{DRL}) & \text{if } r_{min} > r_{crit} \\ (-T_{DRL}, \delta_{min}) & \text{if } r_{min} < r_{crit}, r_{min} \in r_r \\ (-T_{DRL}, \delta_{max}) & \text{if } r_{min} < r_{crit}, r_{min} \in r_l \end{cases} \tag{6.9}$$

The actions computed by the DRL model (T_{DRL}, δ_{DRL}) were executed if the minimum range measurement was above the safety threshold, and otherwise bypassed by the safety supervisor to enable the AMR to safely reverse out of tight spaces before the DRL model took over.

6.6 Navigation to Objects

The positions of objects of interest in a RGB camera frame were determined using a pre-trained object detection model. The You Only Look Once (YOLO) detection system [264] was utilized to obtain bounding boxes of objects in the Common Objects in Context (COCO) dataset which comprises 80 object categories that include people, animals and vehicles. In order for real-time execution, the Compute Unified Device Architecture (CUDA) API was used on the Jetson Orin AGX embedded computer to utilize the GPU to speed up inference 20 times, from 0.4 seconds to 0.02. An ELP night vision camera was used for object detection in low-light and no-light conditions. This was mounted to a modified configuration of XTENTH-CAR, illustrated in Figure 6.6, with a lower positioned, higher resolution RPLIDAR S2 for improved DRL model, and safety supervisor performance. Navigation to a person, identified as an object of interest, was evaluated in well lit and no-light conditions in a multi-room hallway, depicted in Figure 6.7.

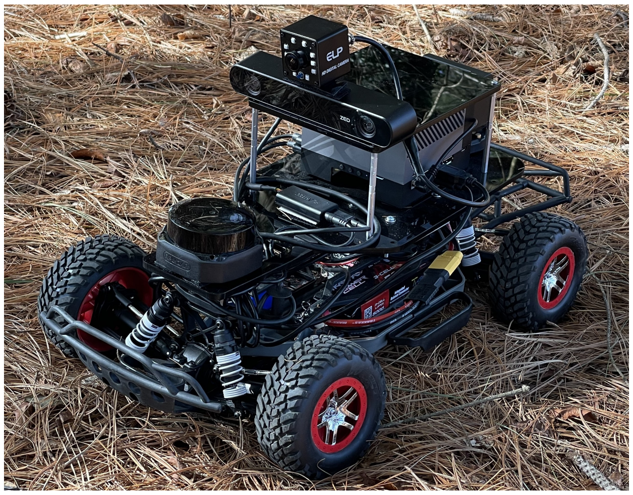


Figure 6.6: Modified configuration of XTENTH-CAR with ELP night vision camera and higher resolution RPLIDAR S2.

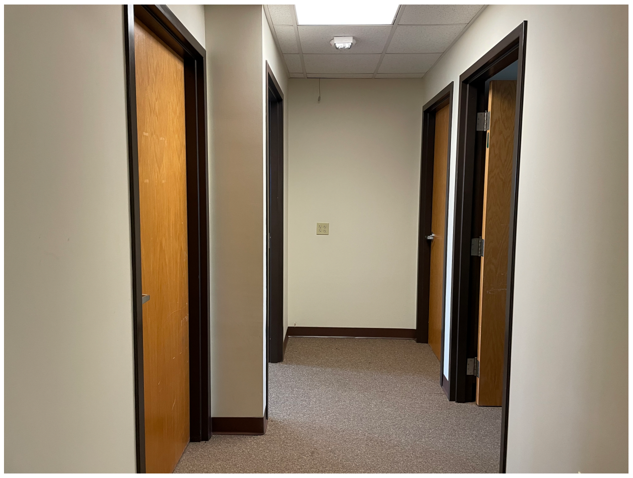


Figure 6.7: Multi-room hallway used to evaluate navigation to an object of interest in different lighting conditions.

The position of the bounding box of the detected object, with coordinates $(x_{ul}, y_{ul}, x_{br}, y_{br})$ where x_{ul} and y_{ul} are the upper left, and x_{br} and y_{br} are the bottom right coordinates, was used to compute the navigation action A_{nav} to navigate to a specific object, which comprises T_{nav} and δ_{nav}, as follows,

$$A_{nav} = (T_{nav}, \delta_{nav}) = \begin{cases} (0,0) & \text{if } (x_{br} - x_{ul}) > x_{crit} \\ (T_{DRL}, \delta_{min}) & \text{if } x_{ul} > I_w/2, y_{br} < y_{crit} \\ (T_{DRL}, \delta_{max}) & \text{if } x_{br} < w/2, y_{br} < y_{crit} \end{cases} \tag{6.10}$$

The AMR was halted if $(x_{br} - x_{ul})$ was above x_{crit} which was 80% of the image width I_w that corresponded to the object in close proximity, which signalled that navigation was complete. The throttle commands computed by the DRL model T_{DRL} and either the minimum or maximum steering angles δ_{min} and δ_{max} were applied if the lower y coordinate of the bounding box y_{br} was below y_{crit} which was 80% of the image height I_h, which occurs when the object is further away. The direction of steering was dependent on the position of the object in the left or right half of the image. Figure 6.8 illustrates the navigation framework that incorporates the DRL model, safety supervisor and navigation controller.

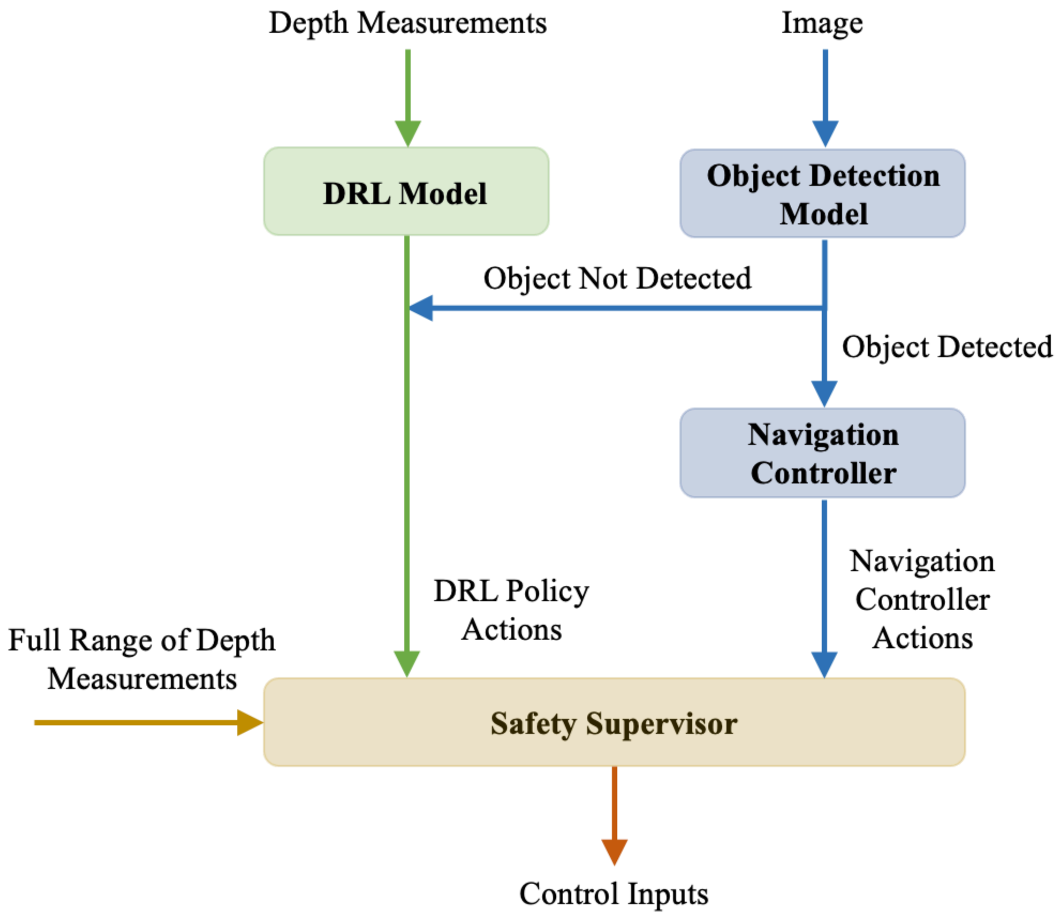


Figure 6.8: Schematic of navigation framework.

The DRL model computes control inputs for exploration end-to-end from depth measurements in observation space $\mathcal{O}$ that is a subset of the full range of depth measurements from the sensor $\mathcal{O}_{sensor}$ utilized by the safety supervisor. If an object of interest is detected, the navigation controller computes actions to traverse to it. Both sets of computed actions are verified by the safety supervisor prior to execution. The order of control commands in increasing order of priority is the navigation controller, DRL model and safety supervisor. The DRL policy actions bypass the navigation controller if δ_{DRL} is in the opposite direction of δ_{nav} to avoid obstacles detected by the DRL model, on the path between the AMR and identified object.

Chapter 7

Results and Discussion

This chapter presents and analyzes the simulation and real-world results for racing, and zero-shot OOD generalization to navigation in new environments without maps. A video of the results can be accessed in Appendix B.

7.1 Learning Curve

The learning curve during training is illustrated in Figure 7.1 where the order 1000 moving average reward is plotted against the training episode.

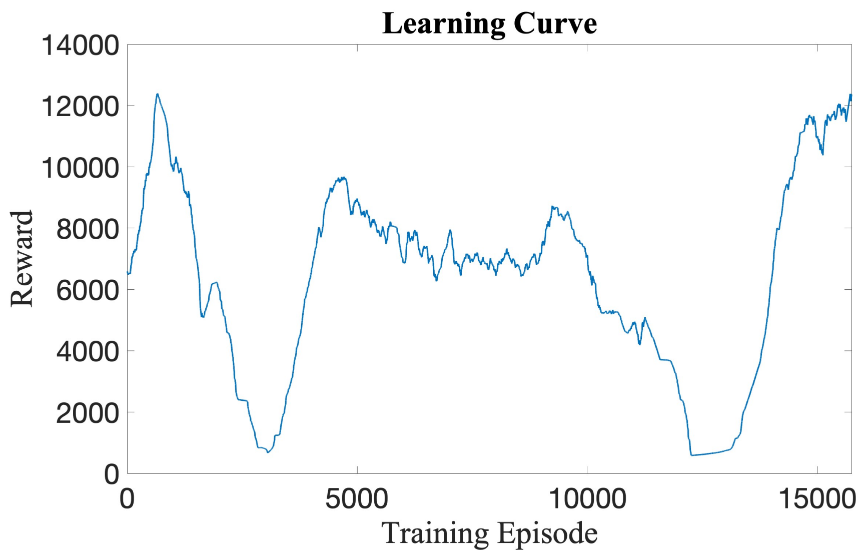


Figure 7.1: Order 1000 moving average reward each training episode.

An average reward of 12,383 was attained post training completion. The learning curve is highly nonlinear, similar to non-trivial environments in prior work [95, 99]. At episode 657, a high reward was obtained, however, the policy was overfitted at this point, since the agent was unable to replicate results consistently as can be observed when the average reward drops with further training, as different action sequences were experimented to collect training samples to improve the long term cumulative return.

At episode 4,675, a local maximum was attained at which point the trained policy computed actions to navigate the racetrack with time-optimal trajectories for 2 laps before collision. To improve performance, training was continued, during which the average reward was constant for 4,900 episodes at the local maximum, after which at episode 9575, further action sequences were experimented to learn an optimal policy at the end of the training period at episode 15,747. The policy learned at the end of the nonlinear learning curve mapped observations to actions to navigate the racetrack safely without collision for 20 tested laps, with the same time-optimal trajectory solved at the local maximum.

7.2 Performance in Training Environment

The time-optimal post training trajectory in the racetrack used for training is depicted in Figure 7.2.

Racing requires prediction and planning to execute control signals in sequence to maximize velocity at physical limits in highly nonlinear domains. The learned policy selected actions at these limits to navigate multiple turns of varying directions and radii conforming to an optimal racing trajectory that navigated to the apex of each corner to traverse the shorter inside region to minimize lap time, identical to professional racing drivers.

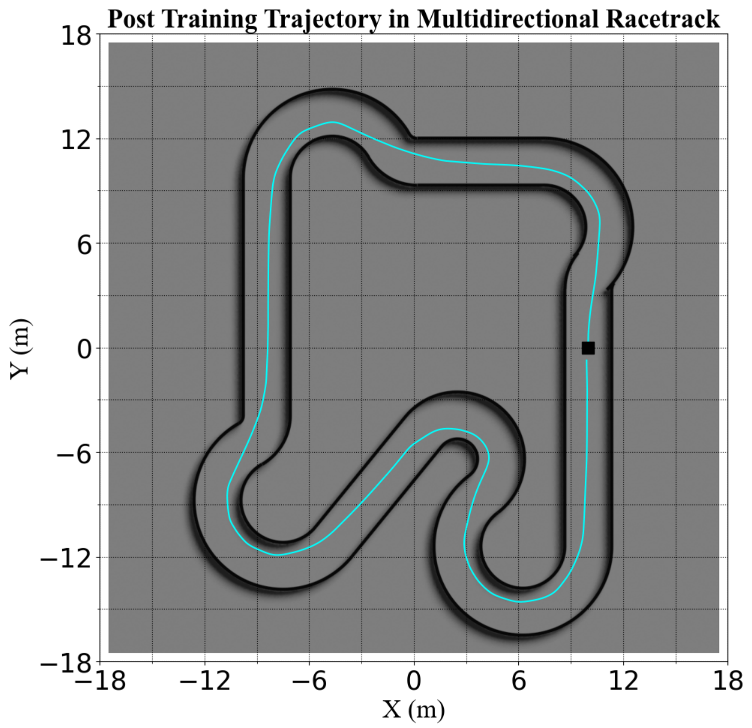


Figure 7.2: Time-optimal trajectory in the multidirectional racetrack used for training.

Unlike conventional racing algorithms that use optimization methods such as Model Predictive Control (MPC) and sampling methods such as Rapid Random Trees (RRT) [262, 265], the DRL model computed control inputs using significantly fewer operations, and did not require prior maps or track boundary information.

7.3 Out-of-Distribution Generalization for Navigation in Simulated Outdoor Environment

The racing agent exhibited emergent behavior that extended to navigation in obstacle filled environments. OOD generalization for navigation in the $50m$ x $50m$ simulated outdoor environment is illustrated in Figure 7.3.

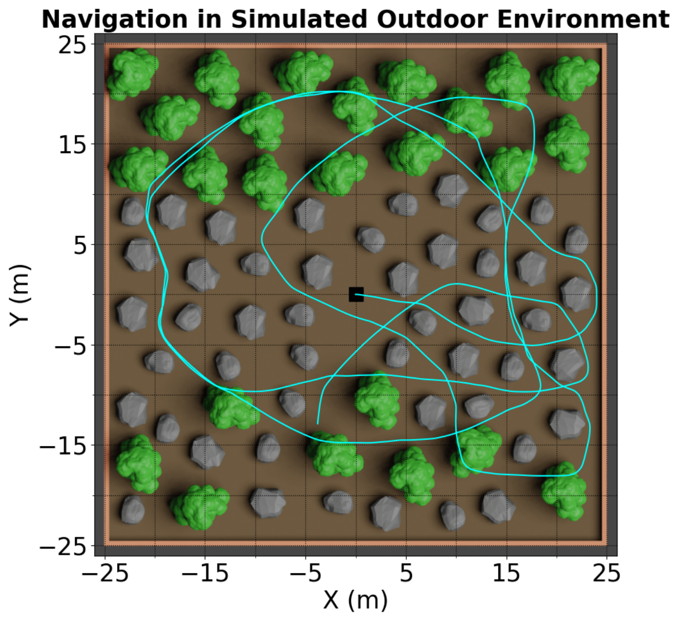


Figure 7.3: OOD generalization for navigation in the simulated outdoor environment.

The policy trained to race in 20,000,000 steps explored this environment more effectively than policies trained in 60,000,000 steps using PPO or SAC with different variations of the shaped reward in [95] specifically designed for exploration in obstacle filled environments without prior maps. SAC outperformed PPO when trained directly in the object filled environment, with the latter learning inefficient circular motion in the same vicinity despite the addition of a penalty to the reward for excessive steering inputs. In contrast, PPO learned a better policy in the racetrack environment where there were more pronounced alterations in the observation space, in which SAC took longer to converge, and did not learn the time-optimal trajectory due to more emphasis on exploratory behavior that led to conservative racetrack boundary avoidance.

The constrained racetrack environment with sparse rewards resulted in more rigorous training that enabled learning of desirable obstacle avoidance behavior with a higher success rate and three times fewer training samples than training in obstacle filled environments with shaped rewards. Moreover, the racing policy facilitated obstacle seeking behavior that is beneficial to ground AMRs for exploration and search applications in aerially occluded regions.

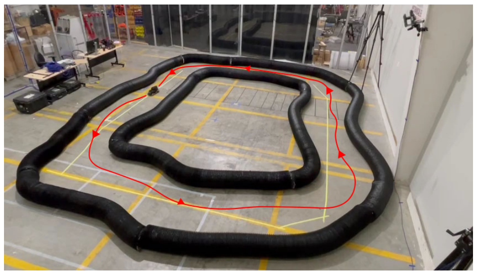
7.4 Zero-Shot Real-World Transfer

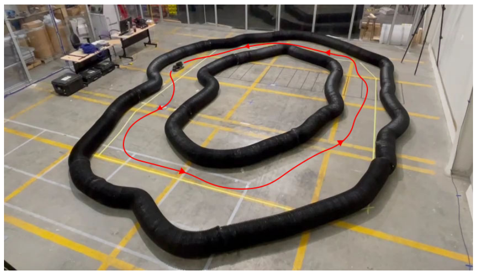
The policy learned in simulation was transferred zero-shot to the physical AMR, with no additional training in the real-world. Planar LiDAR range measurements were processed to match the simulated observation space. Sensor and actuator noise were not modeled and added to simulation training, nor was domain randomization utilized.

Time-optimal, collision free trajectories traversed in the three tested racetracks are shown in Figure 7.4, demonstrating zero-shot policy transfer trained in simulation without safety considerations for uncompromised performance, and generalization to new multidirectional racetracks.

The DRL model utilized less computation resources for better racing performance compared to a modified wall following PID controller and MPC combined with Artificial Potential Function (APF) [266], and lapped faster than non-expert humans with manual control. On average, the policy navigated the short racetracks 30% faster than the tested methods, while utilizing the same CPU usage as the PID controller, and half that of the optimization method.

The laboratory obstacle filled environment contained various objects of different profiles such as boulders, tubes and small pots. The trajectory for OOD generalization in the cluttered test environment is shown in Figure 7.5. The results demonstrate successful obstacle detection and avoidance, with high accuracy. Exploration was sufficient to collect data in all regions using onboard sensors, but can be further improved. The racing nature of the learned policy


(a)


(b)

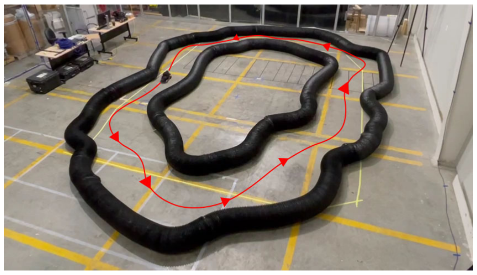

(c)

Figure 7.4: Trajectories in new real-world racetracks without prior maps. (a) Straight and simple corners. (b) Long cornering length. (c) Multiple cornering directions in quick succession.

led to high risk obstacle avoidance behavior where the AMR charted trajectories to avoid obstacles at the last possible moment, and favored trajectories between obstacles, that is beneficial for exploration and search applications.

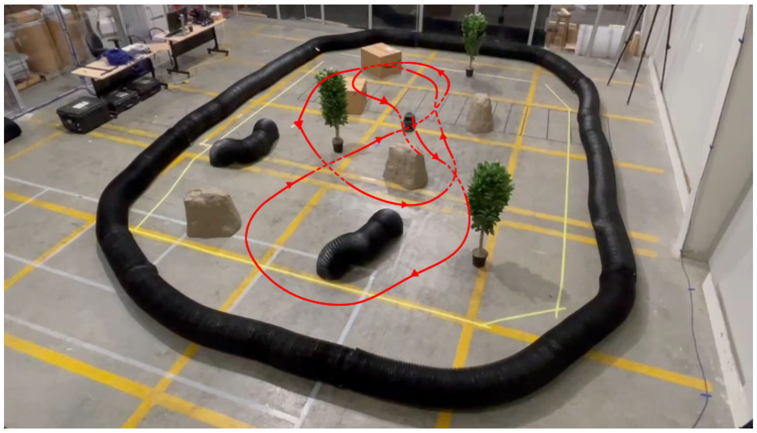


Figure 7.5: Exploration trajectory in the static object filled laboratory environment.

Exploration efficiency was further tested in a parking lot, shown in Figure 7.6. This environment had large regions of empty space with objects such as parked cars, trailers and paved risings that facilitated testing for deployment in urban settings for disaster relief or military operations. The AMR navigated the region without collision or repeated motion in the same vicinity, exhibiting efficient, and thorough exploration capability that demonstrates feasibility for autonomous deployment in environments without prior knowledge, or GPS coverage.

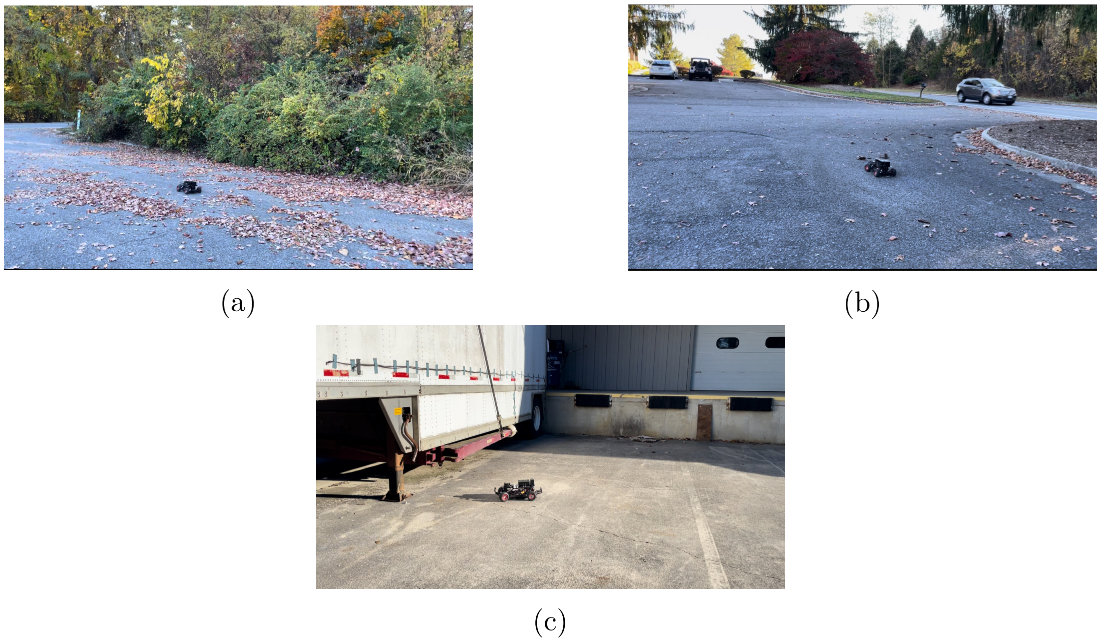


Figure 7.6: Exploration in a parking lot. (a) Intrinsic exploratory behavior. (b) Accurate obstacle avoidance. (c) Obstacle seeking behavior.

7.5 Navigation in Unstructured Terrain

Robustness to increased sensor noise was tested in unstructured forest terrain that included vegetation, soil and leaf litter, shown in Figure 7.7. The AMR did not collide with trees and branches during testing including narrow regions between sparsely detected branches, demonstrating strong resilience to added noise from changes in terrain elevation, and obstacle avoidance capability. The exploration trajectory was efficient, with new regions covered over the course of operation with no repeated coverage in the same vicinity, and the learned racing behavior was beneficial for improved search ability enabling navigation close to trees to seek and explore shaded regions that can be unobservable to aerial surveillance.

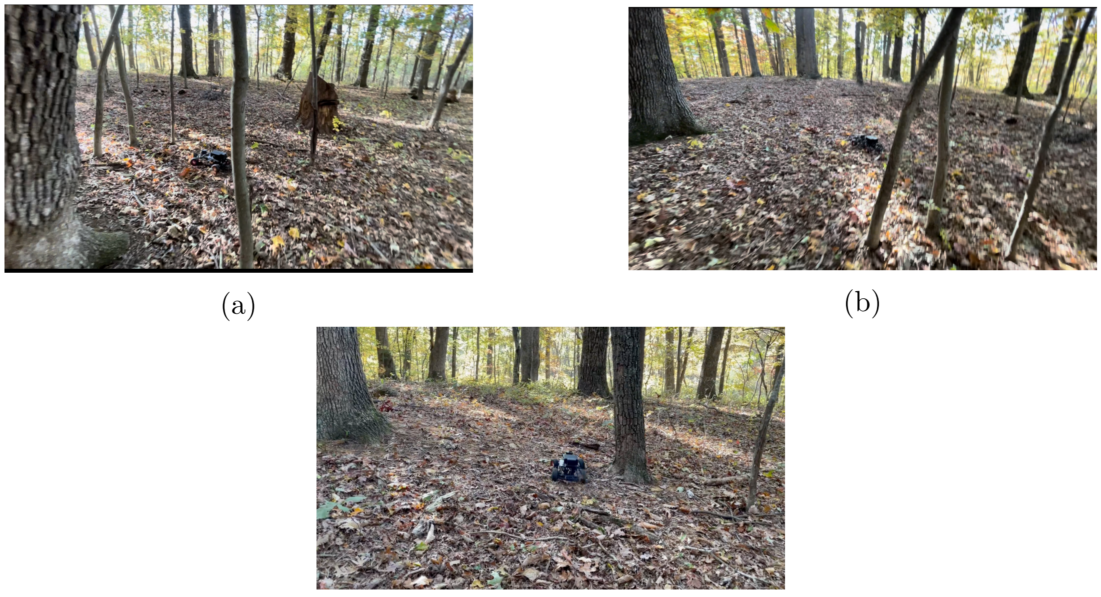

(a) (b) (c)

Figure 7.7: Exploration in unstructured forest terrain. (a) Navigation in narrow region between sparsely detected branches. (b) Avoidance of repeated motion in the same vicinity. (c) Obstacle seeking behavior that is beneficial for exploration of aerially occluded shaded regions.

The results demonstrate the effectiveness of the trained policy for SAR, military, and planetary exploration applications in hazardous regions inaccessible to humans. The method does not require a high-level planner to provide goal positions, as is necessary for conventional approaches which makes it more flexible and adaptable to various environments and applications, while maintaining high levels of safety and efficiency, enabling exploration and data collection in new, unknown environments for downstream tasks. An added benefit is the lack of computationally expensive SLAM algorithms that enables utility in high speed real-time applications, and execution on a range of AMRs with varying payloads for embedded computers.

7.6 Out-of-Distribution Generalization for Dynamic Obstacle Avoidance

The learned behavior extended to dynamic obstacle avoidance outside the training distribution, where directions of dynamic objects influenced action outputs for collision free navigation. Three test scenarios for moving pedestrian avoidance are shown in Figures 7.8, 7.9 and 7.10.

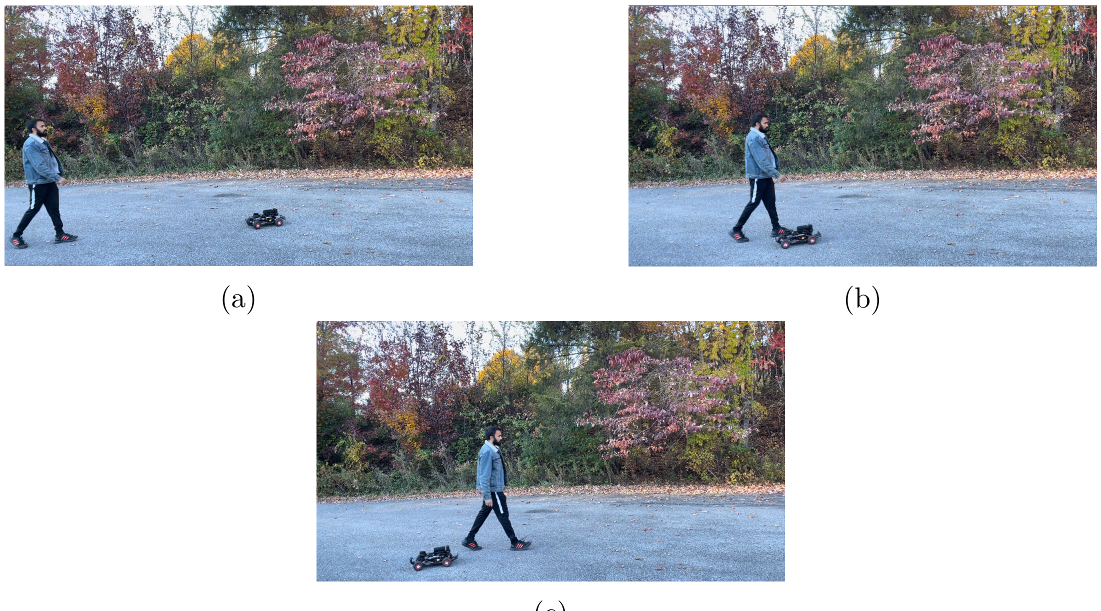

(a) (b) (c)

Figure 7.8: Dynamic pedestrian avoidance: Scenario 1. (a) Trajectories on collision course. (b) Evasive maneuver. (c) Dynamic obstacle avoided.

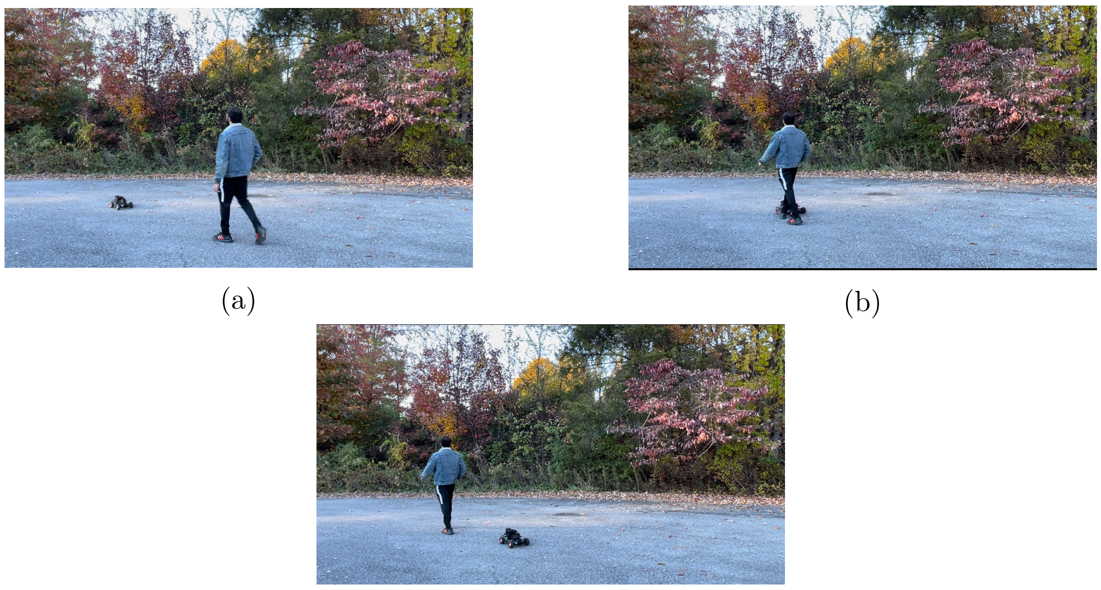

(a) (b) (c)

Figure 7.9: Dynamic pedestrian avoidance: Scenario 2. (a) Trajectories on collision course. (b) Evasive maneuver. (c) Dynamic obstacle avoided.

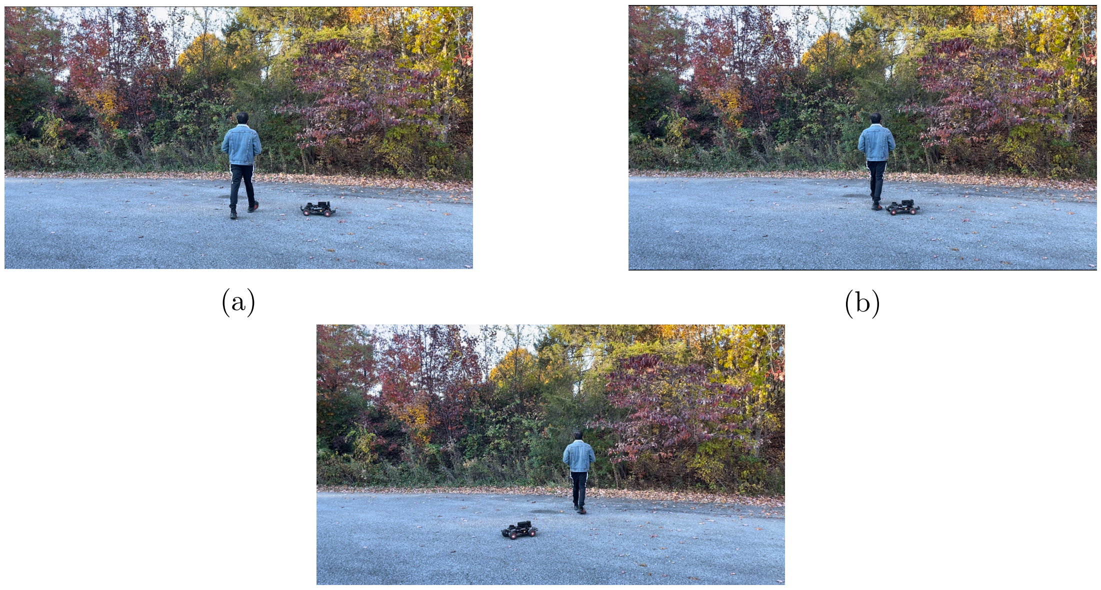

(a) (b) (c)

Figure 7.10: Dynamic pedestrian avoidance: Scenario 3. (a) Trajectories on collision course. (b) Evasive maneuver. (c) Dynamic obstacle avoided.

Prediction and planning required to execute control inputs in sequence to maximize velocity for racing at highly nonlinear physical limits transferred well to dynamic obstacle avoidance at a tested velocity of 1 m/s that represents a typical AMR operation speed around humans.

7.7 Navigation to Objects of Interest

Navigation to a person in a previously unseen environment was evaluated in a indoor hallway with multiple rooms, shown in Figure 7.11. Object detection, and navigation in a no-light setting is shown in Figure 7.12.

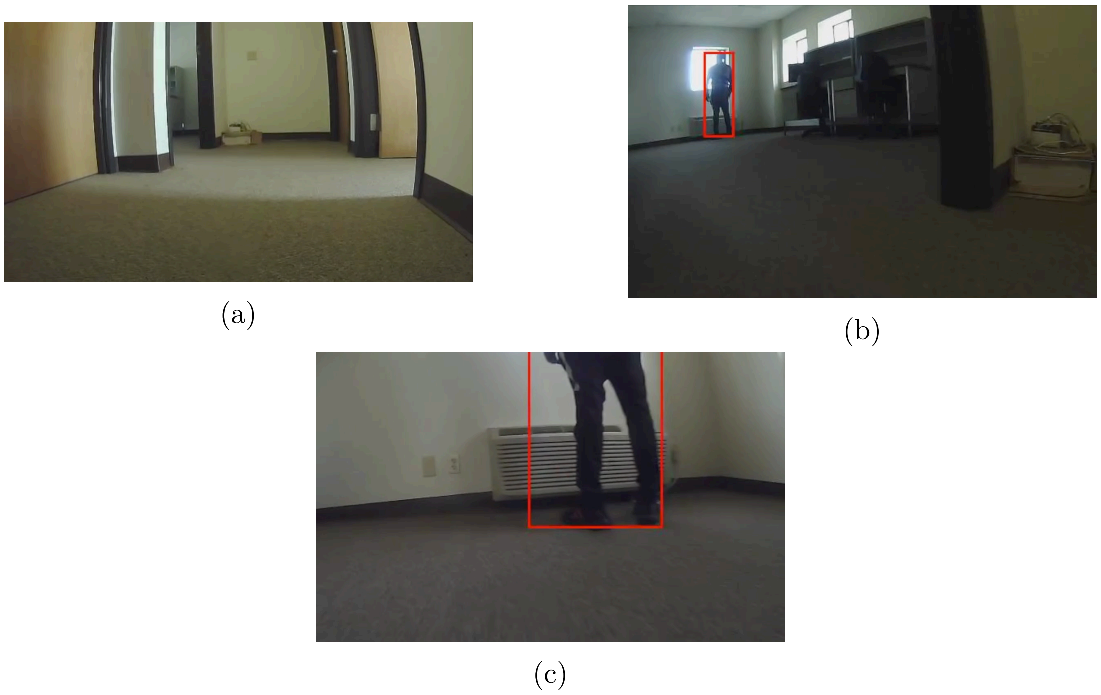

(a)
(b)
(c)

Figure 7.11: Navigation to person in indoor hallway. (a) Exploration in hallway. (b) Person detected in room. (c) Navigation to person.

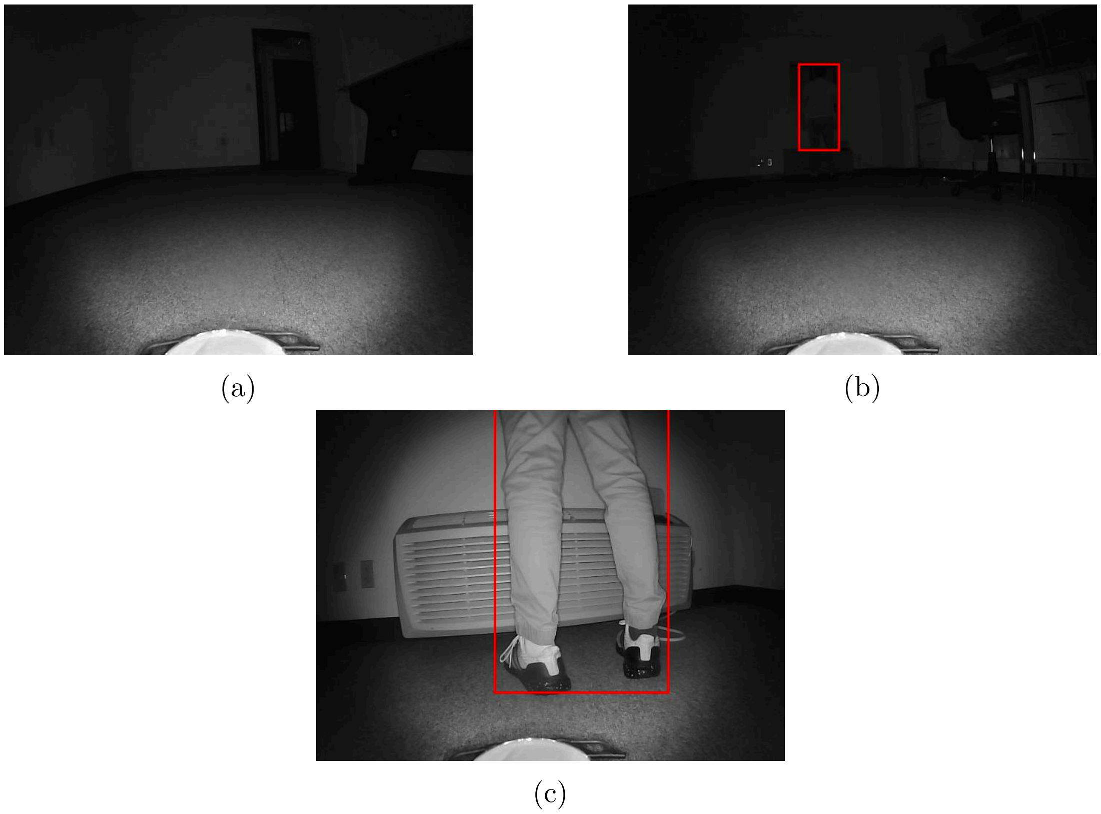

(a) (b) (c)

Figure 7.12: Navigation without light to person in indoor hallway. (a) Starting position in different room. (b) Person detected. (c) Navigation to person.

The AMR explored the unknown environment without collision before detecting the person, and navigating to the object of interest, demonstrating beneficial behavior for home assistance, military and SAR operations. The object detection model worked without alteration with images from the night vision camera, but had a lower detection range, between 1 and 2 m less, compared to the well lit scenario, which maintains functionality, and does not excessively compromise performance. The results demonstrate the compatibility of the DRL model with application specific System II algorithms, such as navigation to objects, or goal positions.

Chapter 8

Conclusions and Future Works

8.1 Conclusions

AMR utility in dynamic, real-world environments without detailed maps is an unsolved problem that has potential to transform the economy, and improve humanity's capabilities in agriculture, manufacturing, disaster response, SAR, military and extraterrestrial planetary exploration. Conventional navigation algorithms are dependent on specific environmental configurations obtained from a separate mapping and localization algorithm, and may utilize estimated physics derived models to compute control inputs, that have accuracy and computation efficiency trade-offs. Learning representations for end-to-end mobile robot navigation utilizing ML has a number of benefits that include computation efficiency and generalization across tasks for deployment at scale, however it is challenging due to the large training sample requirement, difficulty in converging to useful policies and accurately bridging the simulation to reality gap to leverage accelerated training in simulation. This dissertation presented a novel method to train DRL in simulation in a constrained racetrack environment at physical limits over 2 days on an Intel Core i9 13900KF CPU and NVIDIA GeForce RTX 4090 GPU transferred zero-shot to the real-world with no additional training. Emergent behavior in the trained agent was demonstrated for OOD generalization to navigation in new environments without prior maps, and dynamic obstacle avoidance. The learned policy utilized depth measurement observations for fast inference, operation over a wide range of environments that include those that are GPS denied and dimly lit such as caves, bunkers and forests at night, and accurate simulation to real-world policy transfer. The method is robust to sensor noise, and functions in unstructured forest terrain, utilizing a fraction of the computation resources required for conventional modular navigation methods enabling execution in a range of AMR forms with varying payloads for onboard compute.

Chapter 1 elucidated AMRs, and the importance of robust, real-world navigation for operation in a range of tasks. Furthermore, the limitations of contemporary approaches, and benefits and challenges of end-to-end DRL were discussed. Zero-shot simulation to real-world model transfer was highlighted as a feasible alternative to DL trained on a large well, curated dataset that is expensive, and time consuming to collect, and the contributions of the dissertation towards sample efficient zero-shot OOD DRL for AMR navigation were summarized. The shortcomings of current methods towards real-world deployment were inferred in Chapter 2, where conventional and state-of-the-art AMR navigation algorithms, including approaches to unstructured terrain, and dynamic obstacle avoidance, in addition to DRL approaches in general in robotics applications such as locomotion and manipulation were reviewed.

In Chapter 3, a DRL framework for task-independent exploration of an unknown environment without prior maps was formulated. The approach does not require GPS or state estimation, and utilizes range measurement observations for end-to-end control input computation, enabling execution in a range of environments that include those that are GPS and light denied. Three off-policy DRL algorithms, DDPG, TD3 and SAC, that are more sample efficient than on-policy algorithms due to the use of a replay buffer to store, and reuse previously sampled data during training, were compared in simulation using EQS and EES metrics, formulated to quantify the quality and efficiency of exploration trajectories. SAC performed best for the exploration task, and was further used to evaluate a shaped reward for application specific exploration of shaded regions underneath objects such as trees and boulders that are often inaccessible to UAVs in SAR or military operations, to demonstrate the versatility of the DRL framework with tailored, task-specific reward formulations.

Autonomous racing was discussed in Chapter 4, in which a NMPC strategy was formulated to compute time-optimal racing trajectories, evaluated in three real-world racetracks. The limitations of conventional physics derived motion planning, and trade-off between computation efficiency and model accuracy were portrayed. Furthermore, the benefits of developing racing algorithms for potential utility in general AMR applications due to operation in demanding conditions at physical limits in constrained environments were explored. Despite the NMPC motion planning algorithm computing time-optimal racing trajectories in simulation, the nonlinear bicycle model utilized to estimate race car physics was an approximation to minimize computation cost, which will be higher for more accurate, high-fidelity physics models that are challenging to deploy in real-time in the real-world with embedded compute. Moreover, exact physics models of four-wheeled vehicles, that are much less complex than AMRs such as bipedal humanoids are unknown, and an estimation at best, that suggests data-driven approaches have potential to more accurately model the system, with improved computation efficiency.

In order to develop and evaluate simulation to real-world DRL model transfer, a wheeled mobile robot, and simulator with high-fidelity robot and environment models were developed, detailed in Chapter 5. The four-wheeled AMR included range and vision sensors paired with the most powerful embedded computer currently available for high-speed, real-time data processing, necessary for real-world embodied AI. A digital twin of the developed AMR, and 3D simulated environments that include outdoor and urban surroundings, and racetracks were modeled, and packaged with a physics engine and state-of-the-art DRL algorithms for accelerated, cost-effective training. DRL models were trained for exploration in shaded regions unobservable to aerial surveillance, and autonomous racing, using separate shaped rewards to demonstrate the utility of the developed simulator, and accelerated DRL training. The trained policies, which are under 4 megabytes in size, were transferred to the physical AMR, for efficient execution in real-time utilizing a fraction of the resources required for conventional methods. Both platforms are made open-source to facilitate AMR DRL research and development.

Chapter 6 detailed the methodology to leverage autonomous racing for OOD generalization to navigation in previously unseen real-world, dynamic environments without maps. A multidirectional racetrack with varying cornering radii and directions was modeled to train a DRL model in simulation, in a compact 2-layer DNN, to compute control inputs end-to-end directly from range measurements, for time-optimal trajectories. The behavioral similarity between observations for this task, and intrinsic exploratory behavior in obstacle filled environments facilitated zero-shot simulation to real-world transfer for exploration in new environments. The modeled prediction and planning required for high-speed racetrack navigation transferred to dynamic obstacle avoidance at sufficient sampling rates attained with the computationally efficient policy. Moreover, a safety supervisor was devised to prioritize safety during potential edge cases, and a controller was developed for navigation to objects of interest utilizing bounding box outputs inferred from a pre-trained object detection DNN.

The simulation and real-world experimental results were presented, analyzed and discussed in Chapter 7. The DRL model was evaluated in simulation in the racetrack used for training, and an outdoor environment with tree and boulder objects, where different reward formulations and algorithms were compared. Real-world experiments included racing in alternate racetrack layouts, exploration evaluation in an obstacle filled indoor laboratory environment, and parking lot, robust exploration with increased sensor and actuator noise in unstructured forest terrain, dynamic obstacle avoidance, and navigation to objects of interest in an indoor multi-room hallway.

The developed DRL method leveraged accelerated simulation training, and zero-shot end-to-end OOD policy transfer to significantly improve training sample efficiency for AMR navigation in new, unknown environments with intrinsic exploratory behavior for task-independent exploration, modeled in a compact, computationally efficient DNN that facilitates improved utility across a range of AMR forms. Furthermore, application specific System II algorithms, such as the evaluated controller for navigation to objects of interest, or goal position navigation, are compatible with the robust System I DRL policy for operation in unstructured, dynamic environments in all lighting conditions.

8.2 Future Works

There are multiple avenues for future work to further improve exploration efficiency, eliminate edge cases, learn end-to-end DRL policies with image observations for simulation to real-world transfer, and increase DRL training sample efficiency. During testing in indoor and outdoor environments, an edge case was found at 90° wall boundaries in empty space where the policy computed actions to navigate between the two wall directions, which required intervention from the safety supervisor. Observation space resolutions can be further experimented to better detect sharp 90° angles, while maintaining computation efficiency.

Exploration is not optimized for energy efficiency, but is suitable for real-world deployment as is for SAR, military, and exploration for data collection applications. The intrinsic exploratory behavior seeks objects, and avoids excessive repeated motion in the same vicinity. To further improve exploration efficiency, encoded images from visited locations can be matched in latent space using feature matching models [267, 268, 269, 270, 271] to maintain an intrinsic memory of explored regions, and directions.

The number of depth measurements and FoV had a significant impact on the learning ability of the agent, and OOD generalization. Planar depth observations are useful for fast inference, and operation in dimly lit environments such as caves, underground bunkers and forests at night where conventional cameras offer limited utility. 3D LiDAR or RGB-D camera observations can be utilized to learn more in-depth semantic knowledge using larger DNNs. Larger models are more capable of processing and integrating the higher dimensional information provided by these sensors to learn more detailed, context-aware representations of the environment that require significantly more compute for training, and powerful embedded hardware for real-time execution. A well-positioned 2D LiDAR provides the same obstacle avoidance and navigation capabilities as that of 3D spatial observations, and can be further augmented to improve the pre-trained agent's performance by superpositioning 3D spatial information that encompass the AMR's volumetric footprint while maintaining the same observation size, for improved spatial awareness, without an increase in compute cost, that is beneficial for many use cases.

15,747 collisions and resets were necessary for training completion which would have required significantly more time and labor in the real-world. Development of end-to-end DRL approaches is primarily limited by the lack of good simulation environments for simulation to real-world policy transfer. Depth measurements enable accurate policy transfer due to the accurate modeling in physics engines using ray tracing. Advances in neural surface reconstruction [272, 273, 274, 275] have potential for photorealistic, detailed scene construction in simulators from RGB video recordings, to better bridge the reality gap for end-to-end policies that utilize image observations.

Adding perturbations to enhance robustness, and domain randomization are common methods used to bridge the reality gap, but both can result in suboptimal policies that require longer training times to converge. The trained policy was robust to sensor and actuator noise in different environment configurations, and terrain without modelled noise or randomized physics parameters during training. The use of a reward component to learn the difference between observed real-world and simulator physics optimized policy transfer for effective zero-shot adaptation without an increase in convergence time. The DRL model's generalization capability is attributed to the learned representation that effectively captures the behavioral similarity between observations across various tasks, enabling the agent to adapt, and perform in new environments. The policy captures the underlying structure of the perceived environment in a compact 2-layer DNN, that facilitates quick computation. The results demonstrate the potential of AI agents to generalize to diverse and unrelated scenarios, and suggest that training in a constrained environment at physical limits may extend to other applications, such as UAV navigation [276] and particle accelerator optimization [277, 278], for more efficient training in a small parameter space with improved generalization capabilities.

The nonlinear learning curve makes it difficult to predict duration of training. With sufficient training samples, and entropy to promote exploration during training, overfitting and local maxima are surmounted. The significant drop in reward per training episode before an improved policy is learned, first for the local maximum, then for the optimal solution, is a function of the policy and value objective functions, entropy bonus, stochastic gradient ascent optimization, and hyperparameters. The sample efficiency of the training process can potentially be further improved with alterations to the training algorithm for faster, more efficient convergence with a task specific advantage function, optimization algorithm, and engineered hyperparameters.

Appendices

Appendix A

Source Code

A.1 XTENTH-CAR

Source code, hardware and software installation guides for the XTENTH-CAR AMR.

https://github.com/Shathushan-Sivashangaran/XTENTH-CAR

A.2 AutoVRL

Source code and installation guide for the AutoVRL simulator.

https://github.com/Shathushan-Sivashangaran/AutoVRL

Appendix B

Video of Results

Simulation and real-world results for racing, OOD generalization to navigation in new environments without maps, and dynamic obstacle avoidance.

https://youtu.be/jUtPaQV3Bd8

Bibliography

[1] Jo Yung Wong. *Theory of ground vehicles.* John Wiley & Sons, 2022.

[2] Yan Chen and Junmin Wang. Design and evaluation on electric differentials for over-actuated electric ground vehicles with four independent in-wheel motors. *IEEE Transactions on Vehicular Technology*, 61(4):1534–1542, 2012.

[3] Agnieszka Dąbrowska, Stanisław Konopka, Mirosław Przybysz, and Arkadiusz Rubiec. Ability to negotiate terrain obstacles by lightweight six-wheeled unmanned ground vehicles. In *10th International Conference ITELMS*, 2015.

[4] Rongrong Wang, Yan Chen, Daiwei Feng, Xiaoyu Huang, and Junmin Wang. Development and performance characterization of an electric ground vehicle with independently actuated in-wheel motors. *Journal of Power Sources*, 196(8):3962–3971, 2011.

[5] Jun Ni, Jibin Hu, and Changle Xiang. A review for design and dynamics control of unmanned ground vehicle. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 235(4):1084–1100, 2021.

[6] Liam Lynch, Thomas Newe, John Clifford, Joseph Coleman, Joseph Walsh, and Daniel Toal. Automated ground vehicle (agv) and sensor technologies-a review. In *2018 12th International Conference on Sensing Technology (ICST)*, pages 347–352. IEEE, 2018.

[7] Priyaranjan Biswal and Prases K Mohanty. Development of quadruped walking robots: A review. *Ain Shams Engineering Journal*, 12(2):2017–2031, 2021.

[8] Marc Raibert, Kevin Blankespoor, Gabriel Nelson, and Rob Playter. Bigdog, the rough-terrain quadruped robot. *IFAC Proceedings Volumes*, 41(2):10822–10825, 2008.

[9] Yibin Li, Bin Li, Jiuhong Ruan, and Xuewen Rong. Research of mammal bionic quadruped robots: A review. In *2011 IEEE 5th International Conference on Robotics, Automation and Mechatronics (RAM)*, pages 166–171. IEEE, 2011.

[10] Masahiro Fujita and Hiroaki Kitano. Development of an autonomous quadruped robot for robot entertainment. *Autonomous Robots*, 5:7–18, 1998.

[11] Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 international conference on robotics and automation (ICRA)*, pages 6295–6301. IEEE, 2019.

[12] Mrinal Kalakrishnan, Jonas Buchli, Peter Pastor, Michael Mistry, and Stefan Schaal. Fast, robust quadruped locomotion over challenging terrain. In *2010 IEEE International Conference on Robotics and Automation*, pages 2665–2670. IEEE, 2010.

[13] Kazuo Hirai, Masato Hirose, Yuji Haikawa, and Toru Takenaka. The development of honda humanoid robot. In *Proceedings. 1998 IEEE international conference on robotics and automation (Cat. No. 98CH36146)*, volume 2, pages 1321–1326. IEEE, 1998.

[14] Shuuji Kajita, Hirohisa Hirukawa, Kensuke Harada, and Kazuhito Yokoi. *Introduction to humanoid robotics*, volume 101. Springer, 2014.

[15] Hiroshi Ishiguro, Tetsuo Ono, Michita Imai, Takeshi Maeda, Takayuki Kanda, and Ryohei Nakatsu. Robovie: an interactive humanoid robot. *Industrial robot: An international journal*, 28(6):498–504, 2001.

[16] Kenji Kaneko, Kensuke Harada, Fumio Kanehiro, Go Miyamori, and Kazuhiko Akachi.

Humanoid robot hrp-3. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2471–2478. IEEE, 2008.

[17] Takayuki Kanda, Hiroshi Ishiguro, Michita Imai, and Tetsuo Ono. Development and evaluation of interactive humanoid robots. *Proceedings of the IEEE*, 92(11):1839–1850, 2004.

[18] Yu Ogura, Hiroyuki Aikawa, Kazushi Shimomura, Hideki Kondo, Akitoshi Morishima, Hun-ok Lim, and Atsuo Takanishi. Development of a new humanoid robot wabian-2. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 76–81. IEEE, 2006.

[19] Juan P Vasconez, George A Kantor, and Fernando A Auat Cheein. Human–robot interaction in agriculture: A survey and current challenges. *Biosystems engineering*, 179:35–48, 2019.

[20] Tom Duckett, Simon Pearson, Simon Blackmore, Bruce Grieve, Wen-Hua Chen, Grzegorz Cielniak, Jason Cleaversmith, Jian Dai, Steve Davis, Charles Fox, et al. Agricultural robotics: the future of robotic agriculture. *arXiv preprint arXiv:1806.06762*, 2018.

[21] Neha S Naik, Virendra V Shete, and Shruti R Danve. Precision agriculture robot for seeding function. In *2016 international conference on inventive computation technologies (ICICT)*, volume 2, pages 1–3. IEEE, 2016.

[22] Arcot Kumar, Ramarangula Sai Deepak, DS Kusuma, and DV Sreekanth. Review on multipurpose agriculture robot. *International Journal for Research in Applied Science and Engineering Technology*, 8(V), 2020.

[23] Robert Sparrow and Mark Howard. Robots in agriculture: prospects, impacts, ethics, and policy. *precision agriculture*, 22:818–833, 2021.

[24] Andrew English, Patrick Ross, David Ball, and Peter Corke. Vision based guidance for robot navigation in agriculture. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1693–1698. IEEE, 2014.

[25] Yucheng Jin, Jizhan Liu, Zhujie Xu, Shouqi Yuan, Pingping Li, and Jizhang Wang. Development status and trend of agricultural robot technology. *International Journal of Agricultural and Biological Engineering*, 14(4):1–19, 2021.

[26] Mikkel Rath Pedersen, Lazaros Nalpantidis, Rasmus Skovgaard Andersen, Casper Schou, Simon Bøgh, Volker Krüger, and Ole Madsen. Robot skills for manufacturing: From concept to industrial deployment. *Robotics and Computer-Integrated Manufacturing*, 37:282–291, 2016.

[27] Zhihao Liu, Quan Liu, Wenjun Xu, Lihui Wang, and Zude Zhou. Robot learning towards smart robotic manufacturing: A review. *Robotics and Computer-Integrated Manufacturing*, 77:102360, 2022.

[28] Janis Arents and Modris Greitans. Smart industrial robot control trends, challenges and opportunities within manufacturing. *Applied Sciences*, 12(2):937, 2022.

[29] Tadeusz Mikolajczyk. Manufacturing using robot. *Advanced Materials Research*, 463: 1643–1646, 2012.

[30] Pinar Urhal, Andrew Weightman, Carl Diver, and Paulo Bartolo. Robot assisted additive manufacturing: A review. *Robotics and Computer-Integrated Manufacturing*, 59:335–345, 2019.

[31] Izabela Nielsen, Quang-Vinh Dang, Grzegorz Bocewicz, and Zbigniew Banaszak. A methodology for implementation of mobile robot in adaptive manufacturing environments. *Journal of intelligent manufacturing*, 28:1171–1188, 2017.

[32] Max Schwarz, Tobias Rodehutskors, David Droeschel, Marius Beul, Michael Schreiber, Nikita Araslanov, Ivan Ivanov, Christian Lenz, Jan Razlaw, Sebastian Schüller, et al. Nimbro rescue: solving disaster-response tasks with the mobile manipulation robot momaro. *Journal of Field Robotics*, 34(2):400–425, 2017.

[33] Geert-Jan M Kruijff, Ivana Kruijff-Korbayová, Shanker Keshavdas, Benoit Larochelle, Miroslav Janíček, Francis Colas, Ming Liu, Francois Pomerleau, Roland Siegwart, Mark A Neerincx, et al. Designing, developing, and deploying systems to support human–robot teams in disaster response. *Advanced Robotics*, 28(23):1547–1570, 2014.

[34] G Clark Haynes, David Stager, Anthony Stentz, J Michael Vande Weghe, Brian Zajac, Herman Herman, Alonzo Kelly, Eric Meyhofer, Dean Anderson, Dane Bennington, et al. Developing a robust disaster response robot: Chimp and the robotics challenge. *Journal of Field Robotics*, 34(2):281–304, 2017.

[35] Yohei Kakiuchi, Kunio Kojima, Eisoku Kuroiwa, Shintaro Noda, Masaki Murooka, Iori Kumagai, Ryohei Ueda, Fumihito Sugai, Shunichi Nozawa, Kei Okada, et al. Development of humanoid robot system for disaster response through team nedo-jsk's approach to darpa robotics challenge finals. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 805–810. IEEE, 2015.

[36] Manolis Chiou, Georgios-Theofanis Epsimos, Grigoris Nikolaou, Pantelis Pappas, Giannis Petousakis, Stefan Mühl, and Rustam Stolkin. Robot-assisted nuclear disaster response: Report and insights from a field exercise. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4545–4552. IEEE, 2022.

[37] Shinji Kawatsuma, Mineo Fukushima, and Takashi Okada. Emergency response by robots to fukushima-daiichi accident: summary and lessons learned. *Industrial Robot: An International Journal*, 39(5):428–435, 2012.

[38] Yugang Liu and Goldie Nejat. Robotic urban search and rescue: A survey from the control perspective. *Journal of Intelligent & Robotic Systems*, 72:147–165, 2013.

[39] Farzad Niroui, Kaicheng Zhang, Zendai Kashino, and Goldie Nejat. Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. *IEEE Robotics and Automation Letters*, 4(2):610–617, 2019.

[40] Jennifer Casper and Robin R. Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 33(3):367–385, 2003.

[41] Angela Davids. Urban search and rescue robots: from tragedy to technology. *IEEE Intelligent systems*, 17(2):81–83, 2002.

[42] Ahmet Denker and Mehmet Can İşeri. Design and implementation of a semi-autonomous mobile search and rescue robot: Salvor. In *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, pages 1–6. IEEE, 2017.

[43] Yi Liu, Yuhua Zhong, Xieyuanli Chen, Pan Wang, Huimin Lu, Junhao Xiao, and Hui Zhang. The design of a fully autonomous robot system for urban search and rescue. In *2016 IEEE International Conference on Information and Automation (ICIA)*, pages 1206–1211. IEEE, 2016.

[44] Deepak Patil, Munsaf Ansari, Dilisha Tendulkar, Ritesh Bhatlekar, Vijaykumar Naik Pawar, and Shailendra Aswale. A survey on autonomous military service robot. In

2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), pages 1–7. IEEE, 2020.

[45] Florian Jentsch. *Human-robot interactions in future military operations.* CRC Press, 2016.

[46] Mrs NS Usha, S Priyadharshini, K Rohinee Shree, P Sabari Devi, and G Sangeetha. Military reconnaissance robot. *International Journal of Advanced Engineering Research and Science*, 4(2):237036, 2017.

[47] Patrick Lin, George Bekey, and Keith Abney. Autonomous military robotics: Risk, ethics, and design. 2008.

[48] Ingo Schwartz. Primus: autonomous driving robot for military applications. In *Unmanned Ground Vehicle Technology II*, volume 4024, pages 313–323. SPIE, 2000.

[49] Muhammad Sanaullah, Md Akhtaruzzaman, and Md Altab Hossain. Land-robot technologies: The integration of cognitive systems in military and defense. *NDC E-JOURNAL*, 2(1):123–156, 2022.

[50] Brian H Wilcox. Robotic vehicles for planetary exploration. *Applied Intelligence*, 2: 181–193, 1992.

[51] Paul S Schenker, Terrance L Huntsberger, Paolo Pirjanian, Eric T Baumgartner, Hrand Aghazarian, Ashitey Trebi-Ollennu, Patrick C Leger, Yang Cheng, Paul G Backes, Edward Tunstel, et al. Robotic automation for space: planetary surface exploration, terrain-adaptive mobility, and multirobot cooperative tasks. In *Intelligent Robots and Computer Vision XX: Algorithms, Techniques, and Active Vision*, volume 4572, pages 12–28. SPIE, 2001.

[52] Paul S Schenker, Terry L Huntsberger, Paolo Pirjanian, Eric T Baumgartner, and Eddie Tunstel. Planetary rover developments supporting mars exploration, sample return and future human-robotic colonization. *Autonomous Robots*, 14:103–126, 2003.

[53] Alex Ellery. *Planetary rovers: robotic exploration of the solar system*. Springer, 2015.

[54] Aravind Seeni, Bernd Schäfer, and Gerd Hirzinger. Robot mobility systems for planetary surface exploration–state-of-the-art and future outlook: a literature survey. *Aerospace Technologies Advancements*, 492:189–208, 2010.

[55] Uwe Jahn, Daniel Heß, Merlin Stampa, Andreas Sutorma, Christof Röhrig, Peter Schulz, and Carsten Wolff. A taxonomy for mobile robots: Types, applications, capabilities, implementations, requirements, and challenges. *Robotics*, 9(4):109, 2020.

[56] Kai Zhu and Tao Zhang. Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Science and Technology*, 26(5):674–691, 2021.

[57] Ron Alterovitz, Sven Koenig, and Maxim Likhachev. Robot planning in the real world: Research challenges and opportunities. *Ai Magazine*, 37(2):76–84, 2016.

[58] Mary B Alatise and Gerhard P Hancke. A review on challenges of autonomous mobile robot and sensor fusion methods. *IEEE Access*, 8:39830–39846, 2020.

[59] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[60] Noor Abdul Khaleq Zghair and Ahmed S Al-Araji. A one decade survey of autonomous mobile robot systems. *International Journal of Electrical and Computer Engineering*, 11(6):4891, 2021.

[61] Lanshen Guo, Minglu Zhang, Yang Wang, and Gengqian Liu. Environmental perception of mobile robot. In *2006 IEEE International Conference on Information Acquisition*, pages 348–352. IEEE, 2006.

[62] Cristiano Premebida, Rares Ambrus, and Zoltan-Csaba Marton. Intelligent robotic perception systems. *Applications of mobile robots*, pages 111–127, 2018.

[63] Johann Borenstein, Hobart R Everett, Liqiang Feng, and David Wehe. Mobile robot positioning: Sensors and techniques. *Journal of robotic systems*, 14(4):231–249, 1997.

[64] Jose Ricardo Sanchez-Ibanez, Carlos J Perez-del Pulgar, and Alfonso García-Cerezo. Path planning for autonomous mobile robots: A review. *Sensors*, 21(23):7898, 2021.

[65] N Sariff and Norlida Buniyamin. An overview of autonomous mobile robot path planning algorithms. In *2006 4th student conference on research and development*, pages 183–188. IEEE, 2006.

[66] Han-ye Zhang, Wei-ming Lin, and Ai-xia Chen. Path planning for the mobile robot: A review. *Symmetry*, 10(10):450, 2018.

[67] Lu Dong, Zichen He, Chunwei Song, and Changyin Sun. A review of mobile robot motion planning methods: from classical motion planning workflows to reinforcement learning-based architectures. *Journal of Systems Engineering and Electronics*, 34(2): 439–459, 2023.

[68] Hai Lin. Mission accomplished: An introduction to formal methods in mobile robot motion planning and control. *Unmanned Systems*, 2(02):201–216, 2014.

[69] Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[70] Sheha Ame Mnubi. Motion planning and trajectory for wheeled mobile robot. *International Journal of Science and Research*, 5(1):1064–1068, 2016.

[71] Spyros G Tzafestas. *Introduction to mobile robot control.* Elsevier, 2013.

[72] Spyros G Tzafestas. Mobile robot control and navigation: A global overview. *Journal of Intelligent & Robotic Systems*, 91:35–58, 2018.

[73] Jonathan Simpson, Christian L Jacobsen, and Matthew C Jadud. Mobile robot control. *Communicating Process Architectures*, page 225, 2006.

[74] Iker Lluvia, Elena Lazkano, and Ander Ansuategi. Active mapping and robot exploration: A survey. *Sensors*, 21(7):2445, 2021.

[75] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[76] Hamid Taheri and Zhao Chun Xia. Slam; definition and evolution. *Engineering Applications of Artificial Intelligence*, 97:104032, 2021.

[77] Andréa Macario Barros, Maugan Michel, Yoann Moline, Gwenolé Corre, and Frédérick Carrel. A comprehensive survey of visual slam algorithms. *Robotics*, 11(1):24, 2022.

[78] Rainer Kümmerle, Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, and Alexander Kleiner. On measuring the accuracy of slam algorithms. *Autonomous Robots*, 27:387–407, 2009.

[79] Josep Aulinas, Yvan Petillot, Joaquim Salvi, and Xavier Lladó. The slam problem: a survey. *Artificial Intelligence Research and Development*, pages 363–371, 2008.

[80] Luigi Nardi, Bruno Bodin, M Zeeshan Zia, John Mawer, Andy Nisbet, Paul HJ Kelly, Andrew J Davison, Mikel Luján, Michael FP O'Boyle, Graham Riley, et al. Introducing slambench, a performance and accuracy benchmarking methodology for slam. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 5783–5790. IEEE, 2015.

[81] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE intelligent vehicles symposium (IV)*, pages 163–168. IEEE, 2011.

[82] Han Hu, Kaicheng Zhang, Aaron Hao Tan, Michael Ruan, Christopher Agia, and Goldie Nejat. A sim-to-real pipeline for deep reinforcement learning for autonomous robot navigation in cluttered rough terrain. *IEEE Robotics and Automation Letters*, 6 (4):6569–6576, 2021.

[83] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[84] Eric Sihite, Arash Kalantari, Reza Nemovi, Alireza Ramezani, and Morteza Gharib. Multi-modal mobility morphobot (m4) with appendage repurposing for locomotion plasticity enhancement. *Nature communications*, 14(1):3323, 2023.

[85] Pranav Singh Chib and Pravendra Singh. Recent advancements in end-to-end autonomous driving using deep learning: A survey. *IEEE Transactions on Intelligent Vehicles*, 2023.

[86] Zhijian Liu, Alexander Amini, Sibo Zhu, Sertac Karaman, Song Han, and Daniela L Rus. Efficient and robust lidar-based end-to-end navigation. In *2021 IEEE Interna-*

tional Conference on Robotics and Automation (ICRA), pages 13247–13254. IEEE, 2021.

[87] Ardi Tampuu, Tambet Matiisen, Maksym Semikin, Dmytro Fishman, and Naveed Muhammad. A survey of end-to-end driving: Architectures and training methods. *IEEE Transactions on Neural Networks and Learning Systems*, 33(4):1364–1384, 2020.

[88] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019, 2021.

[89] Jianxin Wu. Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, 5(23):495, 2017.

[90] Moez Krichen. Convolutional neural networks: A survey. *Computers*, 12(8):151, 2023.

[91] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):87–110, 2022.

[92] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41, 2022.

[93] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12104–12113, 2022.

[94] Yunlong Song, Angel Romero, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Reaching the limit in autonomous racing: Optimal control versus reinforcement learning. *Science Robotics*, 8(82):eadg1462, 2023.

[95] Shathushan Sivashangaran and Azim Eskandarian. Deep reinforcement learning for autonomous ground vehicle exploration without a-priori maps. *Advances in Artificial Intelligence and Machine Learning*, 3(2):1198–1219, 2023.

[96] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

[97] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[98] Shathushan Sivashangaran. Application of deep reinforcement learning for intelligent autonomous navigation of car-like mobile robot. Master's thesis, State University of New York at Buffalo, 2021.

[99] Shathushan Sivashangaran and Minghui Zheng. Intelligent autonomous navigation of car-like unmanned ground vehicle via deep reinforcement learning. *IFAC-PapersOnLine*, 54(20):218–225, 2021.

[100] Thomas Nakken Larsen, Halvor Ødegård Teigen, Torkel Laache, Damiano Varagnolo, and Adil Rasheed. Comparing deep reinforcement learning algorithms' ability to safely navigate challenging waters. *Frontiers in Robotics and AI*, 8, 2021.

[101] Jonáš Kulhánek, Erik Derner, and Robert Babuška. Visual navigation in real-world indoor environments using end-to-end deep reinforcement learning. *IEEE Robotics and Automation Letters*, 6(3):4345–4352, 2021.

[102] Haobin Shi, Lin Shi, Meng Xu, and Kao-Shing Hwang. End-to-end navigation strategy

with deep reinforcement learning for mobile robots. *IEEE Transactions on Industrial Informatics*, 16(4):2393–2402, 2019.

[103] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[104] Benjamin D Evans, Hendrik W Jordaan, and Herman A Engelbrecht. Safe reinforcement learning for high-speed autonomous racing. *Cognitive Robotics*, 3:107–126, 2023.

[105] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[106] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on Robot Learning*, pages 2226–2240. PMLR, 2023.

[107] Erica Salvato, Gianfranco Fenu, Eric Medvet, and Felice Andrea Pellegrino. Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning. *IEEE Access*, 9:153171–153187, 2021.

[108] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, pages 737–744. IEEE, 2020.

[109] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[110] Theophile Gervet, Soumith Chintala, Dhruv Batra, Jitendra Malik, and Devendra Singh Chaplot. Navigating to objects in the real world. *Science Robotics*, 8(79): eadf6991, 2023.

[111] Alberto Boretti and Ishak Aris. Regenerative braking of a 2015 lmp1-h racing car. Technical report, SAE Technical Paper, 2015.

[112] Alberto Boretti. *Kinetic energy recovery systems for racing cars.* SAE International, 2013.

[113] Willem Toet. Aerodynamics and aerodynamic research in formula 1. *The Aeronautical Journal*, 117(1187):1–26, 2013.

[114] Peter Wright and Tony Matthews. *Formula 1 technology.* SAE International, 2001.

[115] Shuhuan Wen, Yanfang Zhao, Xiao Yuan, Zongtao Wang, Dan Zhang, and Luigi Manfredi. Path planning for active slam based on deep reinforcement learning under unknown environments. *Intelligent Service Robotics*, 13:263–272, 2020.

[116] Zehui Meng, Hao Sun, Hailong Qin, Ziyue Chen, Cihang Zhou, and Marcelo H Ang. Intelligent robotic system for autonomous exploration and active slam in unknown environments. In *2017 IEEE/SICE International Symposium on System Integration (SII)*, pages 651–656. IEEE, 2017.

[117] Kruno Lenac, Andrej Kitanov, Ivan Maurović, Marija Dakulović, and Ivan Petrović. Fast active slam for accurate and complete coverage mapping of unknown environments. In *Intelligent Autonomous Systems 13: Proceedings of the 13th International Conference IAS-13*, pages 415–428. Springer, 2016.

[118] P Victerpaul, D Saravanan, S Janakiraman, and J Pradeep. Path planning of au-

tonomous mobile robots: A survey and comparison. *Journal of Advanced Research in Dynamical and Control Systems*, 9(12):1535–1565, 2017.

[119] Apoorva Khairnar, Shathushan Sivashangaran, and Azim Eskandarian. A comparison of motion planning methods for autonomous ground vehicle exploration and search. In *ASME International Mechanical Engineering Congress and Exposition*, volume 87639, page V006T07A069. American Society of Mechanical Engineers, 2023.

[120] Huijuan Wang, Yuan Yu, and Quanbo Yuan. Application of dijkstra algorithm in robot path-planning. In *2011 second international conference on mechanic automation and control engineering*, pages 1067–1069. IEEE, 2011.

[121] Jianming Guo, Liang Liu, Qing Liu, and Yongyu Qu. An improvement of d* algorithm for mobile robot path planning in partial unknown environment. In *2009 Second International Conference on Intelligent Computation Technology and Automation*, volume 3, pages 394–397. IEEE, 2009.

[122] Shaher Alshammrei, Sahbi Boubaker, and Lioua Kolsi. Improved dijkstra algorithm for mobile robot path planning and obstacle avoidance. *Comput. Mater. Contin*, 72: 5939–5954, 2022.

[123] Ali Alyasin, Eyad I Abbas, and Sundus D Hasan. An efficient optimal path finding for mobile robot based on dijkstra method. In *2019 4th Scientific International Conference Najaf (SICN)*, pages 11–14. IEEE, 2019.

[124] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2997–3004. IEEE, 2014.

[125] Luigi Palmieri and Kai O Arras. A novel rrt extend function for efficient and smooth mobile robot motion planning. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 205–211. IEEE, 2014.

[126] Jiadong Li, Shirong Liu, Botao Zhang, and Xiaodan Zhao. Rrt-a* motion planning algorithm for non-holonomic mobile robot. In *2014 proceedings of the SICE annual conference (SICE)*, pages 1833–1838. IEEE, 2014.

[127] Prahlad Vadakkepat, Kay Chen Tan, and Wang Ming-Liang. Evolutionary artificial potential fields and their application in real time robot path planning. In *Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512)*, volume 1, pages 256–263. IEEE, 2000.

[128] Pu Shi and Yiwen Zhao. Global path planning for mobile robot based on improved artificial potential function. In *2009 IEEE International Conference on Automation and Logistics*, pages 1900–1904. IEEE, 2009.

[129] Cao Qixin, Huang Yanwen, and Zhou Jingliang. An evolutionary artificial potential field algorithm for dynamic path planning of mobile robot. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3331–3336. IEEE, 2006.

[130] Wenbai Chen, Xibao Wu, and Yang Lu. An improved path planning method based on artificial potential field for a mobile robot. *Cybernetics and information technologies*, 15(2):181–191, 2015.

[131] S Sivaranjani, Divya A Nandesh, K Gayathri, R Ramanathan, et al. An investigation of bug algorithms for mobile robot navigation and obstacle avoidance in two-dimensional unknown static environments. In *2021 International Conference on Communication information and Computing Technology (ICCICT)*, pages 1–6. IEEE, 2021.

[132] Kimberly N McGuire, Guido CHE de Croon, and Karl Tuyls. A comparative study of bug algorithms for robot navigation. *Robotics and Autonomous Systems*, 121:103261, 2019.

[133] Anwar Al-Haddad, Rubita Sudirman, Camallil Omar, Koo Yin Hui, and Muhammad Rashid Jimin. Wheelchair motion control guide using eye gaze and blinks based on pointbug algorithm. In *2012 Third International Conference on Intelligent Systems Modelling and Simulation*, pages 37–42. IEEE, 2012.

[134] Patrick Reignier. Fuzzy logic techniques for mobile robot obstacle avoidance. *Robotics and Autonomous Systems*, 12(3-4):143–153, 1994.

[135] Hajer Omrane, Mohamed Slim Masmoudi, and Mohamed Masmoudi. Fuzzy logic based control for autonomous mobile robot navigation. *Computational intelligence and neuroscience*, 2016, 2016.

[136] Gianluca Antonelli, Stefano Chiaverini, and Giuseppe Fusco. A fuzzy-logic-based approach for mobile robot path tracking. *IEEE transactions on fuzzy systems*, 15(2): 211–221, 2007.

[137] Azzeddine Bakdi, Abdelfetah Hentout, Hakim Boutami, Abderraouf Maoudj, Ouarda Hachour, and Brahim Bouzouia. Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control. *Robotics and Autonomous Systems*, 89:95–109, 2017.

[138] Chaymaa Lamini, Said Benhlima, and Ali Elbekri. Genetic algorithm based approach for autonomous mobile robot path planning. *Procedia Computer Science*, 127:180–189, 2018.

[139] AT Ismail, Alaa Sheta, and Mohammed Al-Weshah. A mobile robot path planning

using genetic algorithm in static environment. *Journal of Computer Science*, 4(4): 341–344, 2008.

[140] Jianping Tu and Simon X Yang. Genetic algorithm based path planning for a mobile robot. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 1, pages 1221–1226. IEEE, 2003.

[141] Han Yu, Qing Wang, Chao Yan, Youyang Feng, Yang Sun, and Lu Li. Dld-slam: Rgb-d visual simultaneous localisation and mapping in indoor dynamic environments based on deep learning. *Remote Sensing*, 16(2):246, 2024.

[142] Jessica Halim. Enhancing robustness and efficiency in visual slam through integration of deep learning-based semantic segmentation techniques. 2024.

[143] Jiarui Hu, Mao Mao, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Cp-slam: Collaborative neural point-based slam system. *Advances in Neural Information Processing Systems*, 36, 2024.

[144] Ans Hussain Qureshi, Muhammmad Latif Anjum, Wajahat Hussain, Usama Muddassar, and Sohail Abbasi. One step back, two steps forward: learning moves to recover from slam tracking failures. *Advanced Robotics*, pages 1–16, 2024.

[145] Yu Wu, Niansheng Chen, Guangyu Fan, Dingyu Yang, Lei Rao, Songlin Cheng, Xiaoyong Song, and Yiping Ma. Navs: A neural attention-based visual slam for autonomous navigation in unknown 3d environments. *Neural Processing Letters*, 56(2):1–21, 2024.

[146] Sufang Wang and Weicun Zhang. Slam algorithms for autonomous mobile robots. In *Modeling, Identification, and Control for Cyber-Physical Systems Towards Industry 4.0*, pages 115–134. Elsevier, 2024.

[147] Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. *arXiv preprint arXiv:2310.07896*, 2023.

[148] Min-Fan Ricky Lee and Sharfiden Hassen Yusuf. Mobile robot navigation using deep reinforcement learning. *Processes*, 10(12):2748, 2022.

[149] Reinis Cimurs, Il Hong Suh, and Jin Han Lee. Goal-driven autonomous exploration through deep reinforcement learning. *IEEE Robotics and Automation Letters*, 7(2): 730–737, 2021.

[150] Reinis Cimurs, Jin Han Lee, and Il Hong Suh. Goal-oriented obstacle avoidance with deep reinforcement learning in continuous action space. *Electronics*, 9(3):411, 2020.

[151] Roman Parak and Radomil Matousek. Comparison of multiple reinforcement learning and deep reinforcement learning methods for the task aimed at achieving the goal. *MENDEL Journal*, 27(1):1–8, 2021.

[152] Laëtitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. Reward function and initial values: Better choices for accelerated goal-directed reinforcement learning. In *International Conference on Artificial Neural Networks*, pages 840–849. Springer, 2006.

[153] Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning exploration policies for navigation. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[154] Brian P Gerkey and Kurt Konolige. Planning and control in unstructured terrain. In *ICRA workshop on path planning on costmaps*. Citeseer, 2008.

[155] Xuesu Xiao, Joydeep Biswas, and Peter Stone. Learning inverse kinodynamics for accurate high-speed off-road navigation on unstructured terrain. *IEEE Robotics and Automation Letters*, 6(3):6054–6060, 2021.

[156] Tixiao Shan, Jinkun Wang, Brendan Englot, and Kevin Doherty. Bayesian generalized kernel inference for terrain traversability mapping. In *Conference on Robot Learning*, pages 829–838. PMLR, 2018.

[157] Philipp Krüsi, Paul Furgale, Michael Bosse, and Roland Siegwart. Driving on point clouds: Motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments. *Journal of Field Robotics*, 34(5):940–984, 2017.

[158] Panagiotis Papadakis. Terrain traversability analysis methods for unmanned ground vehicles: A survey. *Engineering Applications of Artificial Intelligence*, 26(4):1373–1385, 2013.

[159] Adarsh Jagan Sathyamoorthy, Kasun Weerakoon, Tianrui Guan, Jing Liang, and Dinesh Manocha. Terrapn: Unstructured terrain navigation using online self-supervised learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7197–7204. IEEE, 2022.

[160] David Silver, J Andrew Bagnell, and Anthony Stentz. Learning from demonstration for autonomous navigation in complex unstructured terrain. *The International Journal of Robotics Research*, 29(12):1565–1592, 2010.

[161] R Omar Chavez-Garcia, Jérôme Guzzi, Luca M Gambardella, and Alessandro Giusti. Learning ground traversability from simulations. *IEEE Robotics and Automation letters*, 3(3):1695–1702, 2018.

[162] Fabian Schilling, Xi Chen, John Folkesson, and Patric Jensfelt. Geometric and visual

terrain classification for autonomous mobile navigation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2678–2684. IEEE, 2017.

[163] Shengyan Zhou, Junqiang Xi, Matthew W McDaniel, Takayuki Nishihata, Phil Salesses, and Karl Iagnemma. Self-supervised learning to visually detect terrain surfaces for autonomous robots operating in forested terrain. *Journal of Field Robotics*, 29(2): 277–297, 2012.

[164] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, page 0, 2006.

[165] Chiara Fulgenzi, Anne Spalanzani, and Christian Laugier. Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1610–1616. IEEE, 2007.

[166] Animesh Chakravarthy and Debasish Ghose. Obstacle avoidance in a dynamic environment: A collision cone approach. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 28(5):562–574, 1998.

[167] Clark R Karr, Michael A Craft, and Jaime E Cisneros. Dynamic obstacle avoidance. In *Distributed Interactive Simulation Systems for Simulation and Training in the Aerospace Environment: A Critical Review*, volume 10280, pages 192–216. SPIE, 1995.

[168] F Kamil, S Tang, W Khaksar, N Zulkifli, and S Ahmad. A review on motion planning and obstacle avoidance approaches in dynamic environments. *Advances in Robotics & Automation*, 4(2):134–142, 2015.

[169] Chung-Hsin Wu and Kuei-Yuan Chan. Developing a dynamic obstacle avoidance system for autonomous mobile robots using bayesian optimization and object tracking: Implementation and testing. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 238(9):4007–4022, 2024.

[170] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020.

[171] Wei Guan and Kuo Wang. Autonomous collision avoidance of unmanned surface vehicles based on improved a-star and dynamic window approach algorithms. *IEEE Intelligent Transportation Systems Magazine*, 2023.

[172] Xiaoshan Gao, Liang Yan, Zhijun Li, Gang Wang, and I-Ming Chen. Improved deep deterministic policy gradient for dynamic obstacle avoidance of mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2023.

[173] Linh Kästner, Teham Bhuiyan, Tuan Anh Le, Elias Treis, Johannes Cox, Boris Meinardus, Jacek Kmiecik, Reyk Carstens, Duc Pichel, Bassel Fatloun, et al. Arena-bench: A benchmarking suite for obstacle avoidance approaches in highly dynamic environments. *IEEE Robotics and Automation Letters*, 7(4):9477–9484, 2022.

[174] Tantan Zhang, Yueshuo Sun, Yazhou Wang, Bai Li, Yonglin Tian, and Fei-Yue Wang. A survey of vehicle dynamics modeling methods for autonomous racing: Theoretical models, physical/virtual platforms, and perspectives. *IEEE Transactions on Intelligent Vehicles*, 9(3):4312–4334, 2024. doi: 10.1109/TIV.2024.3351131.

[175] Sihao Wu, Zhengwei Yang, Xiaopo Xie, Yilong Wang, Xinliang Wang, Qi Wang, Bofan Wu, Hongjun Zhang, Hanning Zhang, Haochun Ma, Xuanliang Zhang, and Haiying

Lin. Design and simulation of an autonomous racecar: Perception, slam, planning and control. In *2021 IEEE International Conference on Autonomous Systems (ICAS)*, pages 1–5, 2021. doi: 10.1109/ICAS49788.2021.9551125.

[176] Leonhard Hermansdorfer, Johannes Betz, and Markus Lienkamp. Benchmarking of a software stack for autonomous racing against a professional human race driver. In *2020 Fifteenth International Conference on Ecological Vehicles and Renewable Energies (EVER)*, pages 1–8, 2020. doi: 10.1109/EVER48776.2020.9242926.

[177] Bai Li, Yile Fang, Tian'ao Xu, Siji Ma, Haorui Wang, Yazhou Wang, Xinyuan Li, Tantan Zhang, Xuepeng Bian, and Fei-Yue Wang. Toward fair and thrilling autonomous racing: Governance rules and performance metrics for the autonomous one. *IEEE Transactions on Intelligent Vehicles*, 8(8):3974–3982, 2023. doi: 10.1109/TIV.2023.3298914.

[178] James Herman, Jonathan Francis, Siddha Ganju, Bingqing Chen, Anirudh Koul, Abhinav Gupta, Alexey Skabelkin, Ivan Zhukov, Max Kumskoy, and Eric Nyberg. Learn-to-race: A multimodal control environment for autonomous racing. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9773–9782, 2021. doi: 10.1109/ICCV48922.2021.00965.

[179] Danio Caporale, Adriano Fagiolini, Lucia Pallottino, Alessandro Settimi, Andrea Biondo, Francesco Amerotti, Federico Massa, Stefano De Caro, Andrea Corti, and Luca Venturini. A planning and control system for self-driving racing vehicles. In *2018 IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI)*, pages 1–6, 2018. doi: 10.1109/RTSI.2018.8548444.

[180] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based

autonomous racing of 1: 43 scale RC cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.

[181] Johannes Betz, Tobias Betz, Felix Fent, Maximilian Geisslinger, Alexander Heilmeier, Leonhard Hermansdorfer, Thomas Herrmann, Sebastian Huch, Phillip Karle, Markus Lienkamp, et al. Tum autonomous motorsport: An autonomous racing software for the indy autonomous challenge. *Journal of Field Robotics*, 40(4):783–809, 2023.

[182] Alexander Liniger. *Path planning and control for autonomous racing.* ETH Zurich, 2018.

[183] Eugenio Tramacere, Sara Luciani, Stefano Feraco, Salvatore Circosta, Irfan Khan, Angelo Bonfitto, and Nicola Amati. Local trajectory planning for autonomous racing vehicles based on the rapidly-exploring random tree algorithm. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 85369, page V001T01A009. American Society of Mechanical Engineers, 2021.

[184] Kyle Stachowicz, Dhruv Shah, Arjun Bhorkar, Ilya Kostrikov, and Sergey Levine. Fastrlap: A system for learning high-speed driving via deep rl and autonomous practicing. *arXiv preprint arXiv:2304.09831*, 2023.

[185] Benjamin Evans, Johannes Betz, Hongrui Zheng, Herman A Engelbrecht, Rahul Mangharam, and Hendrik W Jordaan. Accelerating online reinforcement learning via supervisory safety systems. *arXiv preprint arXiv:2209.11082*, 2022.

[186] Nathaniel Hamilton, Patrick Musau, Diego Manzanas Lopez, and Taylor T Johnson. Zero-shot policy transfer in autonomous racing: Reinforcement learning vs imitation learning. In *2022 IEEE International Conference on Assured Autonomy (ICAA)*, pages 11–20. IEEE, 2022.

[187] Michael Bosello, Rita Tse, and Giovanni Pau. Train in Austria, race in Montecarlo: Generalized RL for cross-track F1 tenth lidar-based races. In *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, pages 290–298. IEEE, 2022.

[188] Muhammed Saeed, Mohammed Nagdi, Benjamin Rosman, and Hiba H. S. M. Ali. Deep reinforcement learning for robotic hand manipulation. In *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, pages 1–5, 2021. doi: 10.1109/ICCCEEE49695.2021.9429619.

[189] Michał Bednarek and Krzysztof Walas. Comparative assessment of reinforcement learning algorithms in the taskof robotic manipulation of deformable linear objects. In *2019 4th International Conference on Robotics and Automation Engineering (ICRAE)*, pages 173–177, 2019. doi: 10.1109/ICRAE48301.2019.9043790.

[190] Recep Ozalp, Aysegul Ucar, and Cuneyt Guzelis. Advancements in deep reinforcement learning and inverse reinforcement learning for robotic manipulation: Toward trustworthy, interpretable, and explainable artificial intelligence. *IEEE Access*, 12: 51840–51858, 2024. doi: 10.1109/ACCESS.2024.3385426.

[191] Hengyue Guan. Analysis on deep reinforcement learning in industrial robotic arm. In *2020 International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI)*, pages 426–430, 2020. doi: 10.1109/ICHCI51889.2020.00094.

[192] Apan Dastider, Sayyed Jaffar Ali Raza, and Mingjie Lin. Safe locomotion within confined workspace using deep reinforcement learning. In *2021 Fifth IEEE International Conference on Robotic Computing (IRC)*, pages 111–114, 2021. doi: 10.1109/IRC52146.2021.00025.

[193] Muleilan Pei, Dongping Wu, and Changhong Wang. Quadruped robot locomotion in unknown terrain using deep reinforcement learning. In *2020 3rd International Conference on Unmanned Systems (ICUS)*, pages 517–522, 2020. doi: 10.1109/ICUS50048.2020.9274920.

[194] Chen Yu and Andre Rosendo. Multi-modal legged locomotion framework with automated residual reinforcement learning. *IEEE Robotics and Automation Letters*, 7(4): 10312–10319, 2022. doi: 10.1109/LRA.2022.3191071.

[195] Jinghong Yue. Learning locomotion for legged robots based on reinforcement learning: A survey. In *2020 International Conference on Electrical Engineering and Control Technologies (CEECT)*, pages 1–7, 2020. doi: 10.1109/CEECT50755.2020.9298680.

[196] Ilija Radosavovic, Tete Xiao, Bike Zhang, Trevor Darrell, Jitendra Malik, and Koushil Sreenath. Real-world humanoid locomotion with reinforcement learning. *Science Robotics*, 9(89):eadi9579, 2024.

[197] Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84):eadc9244, 2023.

[198] Tuomas Haarnoja, Ben Moran, Guy Lever, Sandy H Huang, Dhruva Tirumala, Jan Humplik, Markus Wulfmeier, Saran Tunyasuvunakool, Noah Y Siegel, Roland Hafner, et al. Learning agile soccer skills for a bipedal robot with deep reinforcement learning. *Science Robotics*, 9(89):eadi8022, 2024.

[199] Huihui Sun, Weijie Zhang, Runxiang Yu, and Yujie Zhang. Motion planning for mobile robots—focusing on deep reinforcement learning: A systematic review. *IEEE Access*, 9:69061–69081, 2021.

[200] Kamak Ebadi, Lukas Bernreiter, Harel Biggie, Gavin Catt, Yun Chang, Arghya Chatterjee, Christopher E Denniston, Simon-Pierre Deschênes, Kyle Harlow, Shehryar Khattak, et al. Present and future of slam in extreme underground environments. *arXiv preprint arXiv:2208.01787*, 2022.

[201] Gareth Jones and Marc W Holderied. Bat echolocation calls: adaptation and convergent evolution. *Proceedings of the Royal Society B: Biological Sciences*, 274(1612): 905–912, 2007.

[202] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[203] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[204] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[205] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[206] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[207] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

[208] The Mathworks, Inc. Robotics System Toolbox, 2023. URL https://www.mathworks.com/products/robotics.html. MATLAB. Natick, Massachusetts, United States.

[209] The Mathworks, Inc. Reinforcement Learning Toolbox, 2023. URL https://www.mathworks.com/products/reinforcement-learning.html. MATLAB. Natick, Massachusetts, United States.

[210] Shathushan Sivashangaran, Apoorva Khairnar, and Azim Eskandarian. Autovrl: A high fidelity autonomous ground vehicle simulator for sim-to-real deep reinforcement learning. *IFAC-PapersOnLine*, 56(3):475–480, 2023. ISSN 2405-8963.

[211] Shathushan Sivashangaran and Azim Eskandarian. Xtenth-car: A proportionally scaled experimental vehicle platform for connected autonomy and all-terrain research. In *ASME International Mechanical Engineering Congress and Exposition*, volume 87639, page V006T07A068. American Society of Mechanical Engineers, 2023.

[212] Goodarz Mehr, Prasenjit Ghorai, Ce Zhang, Anshul Nayak, Darshit Patel, Shathushan Sivashangaran, and Azim Eskandarian. X-CAR: An experimental vehicle platform for connected autonomy research. *IEEE Intelligent Transportation Systems Magazine*, pages 2–19, 2022. doi: 10.1109/MITS.2022.3168801.

[213] Philip Polack, Florent Altché, Brigitte d'Andréa Novel, and Arnaud de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE intelligent vehicles symposium (IV)*, pages 812–818. IEEE, 2017.

[214] Azim Eskandarian, Chaoxian Wu, and Chuanyang Sun. Research advances and challenges of autonomous and connected ground vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 22(2):683–711, 2019.

[215] Benedikt Wohner, Francisco Sevilla, Benedikt Grueter, Johannes Diepolder, Rubens Afonso, and Florian Holzapfel. Hierarchical nonlinear model predictive control for an autonomous racecar. In *2021 20th International Conference on Advanced Robotics (ICAR)*, pages 113–120. IEEE, 2021.

[216] Johannes Betz, Alexander Wischnewski, Alexander Heilmeier, Felix Nobis, Tim Stahl, Leonhard Hermansdorfer, and Markus Lienkamp. A software architecture for an autonomous racecar. In *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pages 1–6. IEEE, 2019.

[217] Miguel I Valls, Hubertus FC Hendrikx, Victor JF Reijgwart, Fabio V Meier, Inkyu Sa, Renaud Dubé, Abel Gawel, Mathias Bürki, and Roland Siegwart. Design of an autonomous racecar: Perception, state estimation and system integration. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 2048–2055. IEEE, 2018.

[218] Alexandra Tătulea-Codrean, Tommaso Mariani, and Sebastian Engell. Design and simulation of a machine-learning and model predictive control approach to autonomous race driving for the f1/10 platform. *IFAC-PapersOnLine*, 53(2):6031–6036, 2020.

[219] Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, 2019.

[220] Ugo Rosolia, Ashwin Carvalho, and Francesco Borrelli. Autonomous racing using learning model predictive control. In *2017 American control conference (ACC)*, pages 5115–5120. IEEE, 2017.

[221] Frank Allgöwer and Alex Zheng. *Nonlinear model predictive control*, volume 26. Birkhäuser, 2012.

[222] Stefan Fuchshumer, Kurt Schlacher, and Thomas Rittenschober. Nonlinear vehicle dynamics control-a flatness based approach. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 6492–6497. IEEE, 2005.

[223] Egbert Bakker, Lars Nyborg, and Hans B Pacejka. Tyre modelling for use in vehicle dynamics studies. *SAE transactions*, pages 190–204, 1987.

[224] The Mathworks, Inc. Model Predictive Control Toolbox. URL `https://www.mathworks.com/products/model-predictive-control.html`. MATLAB. Natick, Massachusetts, United States.

[225] Alexander Heilmeier, Alexander Wischnewski, Leonhard Hermansdorfer, Johannes Betz, Markus Lienkamp, and Boris Lohmann. Minimum curvature trajectory planning and control for an autonomous race car. *Vehicle System Dynamics*, 2019.

[226] Fabio Muratore, Michael Gienger, and Jan Peters. Assessing transferability from simulation to reality for reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence*, 43(4):1172–1183, 2019.

[227] Hao Ju, Rongshun Juan, Randy Gomez, Keisuke Nakamura, and Guangliang Li. Transferring policy of deep reinforcement learning from simulation to reality for robotics. *Nature Machine Intelligence*, pages 1–11, 2022.

[228] Kishy Kumar and Parminder Singh Reel. Analysis of contemporary robotics simulators. In *2011 International Conference on Emerging Trends in Electrical and Computer Technology*, pages 661–665, 2011. doi: 10.1109/ICETECT.2011.5760200.

[229] Jack Collins, Shelvin Chand, Anthony Vanderkop, and David Howard. A review of physics simulators for robotic applications. *IEEE Access*, 9:51416–51431, 2021. doi: 10.1109/ACCESS.2021.3068769.

[230] Trupti Kantale, Pooja Pawar, and Amol P. Bhagat. An overview of artificial intelligence based autonomous vehicle robotics simulators. In *2019 International Conference on Emerging Trends in Science and Engineering (ICESE)*, volume 1, pages 1–6, 2019. doi: 10.1109/ICESE46178.2019.9194653.

[231] Diego Ferigo, Silvio Traversaro, Giorgio Metta, and Daniele Pucci. Gym-ignition: Reproducible robotic simulations for reinforcement learning. In *2020 IEEE/SICE International Symposium on System Integration (SII)*, pages 885–890. IEEE, 2020.

[232] Quanyi Li, Zhenghao Peng, Lan Feng, Qihang Zhang, Zhenghai Xue, and Bolei Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence*, 2022.

[233] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

[234] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[235] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[236] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[237] Andrew S Glassner. *An introduction to ray tracing.* Morgan Kaufmann, 1989.

[238] Fábio Duarte and Carlo Ratti. The impact of autonomous vehicles on cities: A review. *Journal of Urban Technology*, 25(4):3–18, 2018.

[239] U.S. Department of Transportation Federal Highway Administration. CARMA products, December 2021. URL https://highways.dot.gov/research/operations/CARMA-products. Accessed: 2022-09-14.

[240] NVIDIA. Nvidia Jetson Orin, 2022. URL https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/. Accessed: 2022-09-14.

[241] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.

[242] Riadh Dhaoui and Amine Rahmouni. Mobile robot navigation in indoor environments: Comparison of lidar-based 2d slam algorithms. In *Design Tools and Methods in Industrial Engineering II: Proceedings of the Second International Conference on Design Tools and Methods in Industrial Engineering, ADM 2021, September 9–10, 2021, Rome, Italy*, pages 569–580. Springer, 2022.

[243] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[244] Xu Shang and Azim Eskandarian. Emergency collision avoidance and mitigation using model predictive control and artificial potential function. *IEEE Transactions on Intelligent Vehicles*, pages 1–15, 2023. doi: 10.1109/TIV.2023.3244193.

[245] J Gonzales, F Zhang, K Li, and F Borrelli. Autonomous drifting with onboard sensors. In *Advanced Vehicle Control AVEC'16*, pages 133–138. CRC Press, 2016.

[246] Ugo Rosolia and Francesco Borrelli. Learning how to autonomously race a car: A predictive control approach. *IEEE Transactions on Control Systems Technology*, 28 (6):2713–2719, 2019.

[247] Sertac Karaman, Ariel Anders, Michael Boulet, Jane Connor, Kenneth Gregson, Winter Guerra, Owen Guldner, Mubarik Mohamoud, Brian Plancher, Robert Shin, and John Vivilecchia. Project-based, collaborative, algorithmic robotics for high school students: Programming self-driving race cars at MIT. In *2017 IEEE Integrated STEM Education Conference (ISEC)*, pages 195–203, 2017. doi: 10.1109/ISECon.2017.7910242.

[248] Xihui Wu and Azim Eskandarian. An improved small-scale connected autonomous vehicle platform. *2019 ASME Dynamic Systems and Control Conference*, 2019. doi: 10.1115/DSCC2019-9121.

[249] Siddhartha S Srinivasa, Patrick Lancaster, Johan Michalove, Matt Schmittle, Colin Summers, Matthew Rockett, Joshua R Smith, Sanjiban Choudhury, Christoforos Mavrogiannis, and Fereshteh Sadeghi. MuSHR: A low-cost, open-source robotic racecar for education and research. *arXiv preprint arXiv:1908.08031*, 2019.

[250] Matthew O'Kelly, Hongrui Zheng, Dhruv Karthik, and Rahul Mangharam. F1TENTH: An open-source evaluation environment for continuous control and reinforcement learning. *Proceedings of Machine Learning Research*, 123, 2019. URL https://par.nsf.gov/biblio/10221872.

[251] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. ROS: An open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[252] Morgan Quigley, Brian Gerkey, and William D Smart. *Programming Robots with ROS: A practical introduction to the Robot Operating System.* "O'Reilly Media, Inc.", 2015.

[253] Open Robotics. ROS Noetic Ninjemys. URL https://wiki.ros.org/noetic. Accessed: 2022-09-14.

[254] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.

[255] Traxxas. Slash 4x4 VXL: 1/10 scale 4WD Electric Short Course Truck with TQi™ Traxxas link™ enabled 2.4GHz Radio System & Traxxas Stability Management (TSM), December 2016. URL https://traxxas.com/products/models/electric/slash-4x4-tsm?t=overview. Accessed: 2022-09-14.

[256] Thomas D Gillespie. Fundamentals of vehicle dynamics. *Society of Automotive Engineers*, 1:278, 1992.

[257] Zahra Soleimanitaleb, Mohammad Ali Keyvanrad, and Ali Jafari. Object tracking methods: A review. In *2019 9th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 282–288. IEEE, 2019.

[258] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, 2004. doi: 10.1109/CVPR.2004.1315094.

[259] Thinal Raj, Fazida Hanim Hashim, Aqilah Baseri Huddin, Mohd Faisal Ibrahim, and Aini Hussain. A survey on lidar scanning mechanisms. *Electronics*, 9(5):741, 2020.

[260] John B Kenney. Dedicated short-range communications (DSRC) standards in the United States. *Proceedings of the IEEE*, 99(7):1162–1182, 2011.

[261] Maksim Filipenko and Ilya Afanasyev. Comparison of various slam systems for mo-

bile robot in an indoor environment. In *2018 International Conference on Intelligent Systems (IS)*, pages 400–407. IEEE, 2018.

[262] Shathushan Sivashangaran, Darshit Patel, and Azim Eskandarian. Nonlinear model predictive control for optimal motion planning in autonomous race cars. *IFAC-PapersOnLine*, 55(37):645–650, 2022.

[263] Ingo Wald, William R Mark, Johannes Günther, Solomon Boulos, Thiago Ize, Warren Hunt, Steven G Parker, and Peter Shirley. State of the art in ray tracing animated scenes. In *Computer graphics forum*, volume 28, pages 1691–1722. Wiley Online Library, 2009.

[264] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[265] Gabriel Hartmann, Zvi Shiller, and Amos Azaria. Competitive driving of autonomous vehicles. *IEEE Access*, 10:111772–111783, 2022.

[266] Xu Shang and Azim Eskandarian. Emergency collision avoidance and mitigation using model predictive control and artificial potential function. *IEEE Transactions on Intelligent Vehicles*, 2023.

[267] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. Lightglue: Local feature matching at light speed. *arXiv preprint arXiv:2306.13643*, 2023.

[268] A. Baumberg. Reliable feature matching across widely separated views. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, volume 1, pages 774–781 vol.1, 2000. doi: 10.1109/CVPR.2000.855899.

[269] David M Mount, Nathan S Netanyahu, and Jacqueline Le Moigne. Efficient algorithms for robust feature matching. *Pattern Recognition*, 32(1):17–38, 1999. ISSN 0031-3203.

[270] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8922–8931, June 2021.

[271] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[272] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

[273] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H. Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8456–8465, June 2023.

[274] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6290–6301, June 2022.

[275] Hongrui Cai, Wanquan Feng, Xuetao Feng, Yan Wang, and Juyong Zhang. Neural surface reconstruction of dynamic scenes with monocular rgb-d camera. In S. Koyejo,

S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 967–981. Curran Associates, Inc., 2022.

[276] Hui Lv, YaDong Chen, Shibo Li, and Baolong Zhu. Improve exploration in deep reinforcement learning for uav path planning using state and action entropy. *Measurement Science and Technology*, 2024.

[277] Verena Kain, Simon Hirlander, Brennan Goddard, Francesco Maria Velotti, Giovanni Zevi Della Porta, Niky Bruchon, and Gianluca Valentino. Sample-efficient reinforcement learning for cern accelerator control. *Physical Review Accelerators and Beams*, 23(12):124801, 2020.

[278] A. L. Edelen, S. G. Biedron, B. E. Chase, D. Edstrom, S. V. Milton, and P. Stabile. Neural networks for modeling and control of particle accelerators. *IEEE Transactions on Nuclear Science*, 63(2):878–897, 2016. doi: 10.1109/TNS.2016.2543203.