Defence Research and Development Canada
Recherche et développement pour la défense Canada

Control algorithms for a Shape-shifting Tracked Robotic Vehicle climbing obstacles

I. Vincent

Defence R&D Canada

Technical Report
DRDC Suffield TR 2008-123
December 2008

Control algorithms for a Shape-shifting Tracked Robotic Vehicle climbing obstacles

I. Vincent
Defence R&D Canada – Suffield

Defence R&D Canada – Suffield

Technical Report

DRDC Suffield TR 2008-123

December 2008

Principal Author

Original signed by I. Vincent

I. Vincent

Approved by

Original signed by S. Monckton

S. Monckton
Acting Head/AISS Section

Approved for release by

Original signed by P. D'Agostino

P. D'Agostino
Head/Document Review Panel

© Her Majesty the Queen in Right of Canada as represented by the Minister of National Defence, 2008

© Sa Majesté la Reine (en droit du Canada), telle que représentée par le ministre de la Défense nationale, 2008

Abstract

Research in mobile robot navigation has demonstrated some success in navigating flat world while avoiding obstacles. However, algorithms which analyze complex environments in order to climb three-dimensional obstacles have had very little success due to the complexity of the task. Unmanned ground vehicles currently exhibit simple autonomous behaviours compared to the human ability to move in the world.

This research work aims to design controllers for a shape-shifting tracked robotic vehicle, thus enabling it to autonomously climb obstacles by adapting its geometric configuration. Three control algorithms are proposed to solve the autonomous locomotion problem for climbing obstacles. A reactive controller evaluates the appropriate geometric configuration based on terrain and vehicle geometric considerations. As a scripted controller is difficult to design for every possible circumstance, learning algorithms are a plausible alternative. A neural network based controller works if a task resembles a learned case. However, it lacks adaptability. Learning in real-time by reinforcement and progress estimation facilitates robot control and navigation. This report presents the reinforcement learning algorithm developed to find alternative solutions when the reactive controller gets stuck while climbing an obstacle. The controllers are validated and compared with simulations.

Résumé

L’avancement scientifique en navigation des robots mobiles a démontré quelques succès en navigation 2D et en évitement d’obstacle. Cependant, le défi d’analyser des environnements complexes pour traverser des obstacles a eu peu de succès vu la complexité de la tâche. Les véhicules terrestres non-pilotés exhibent présentement de simples comportements autonomes en comparaison aux habiletés des humains à se mouvoir dans leur environnement.

Ce travail de recherche vise à concevoir des contrôleurs pour qu’un véhicule robotisé qui peut adapter sa configuration géométrique puisse gravir des obstacles de manière autonome. Trois algorithmes de contrôle sont proposés afin de résoudre le problème de mobilité autonome pour traverser des obstacles. Un contrôleur réactif détermine la configuration géométrique appropriée du robot en considérant le relief du terrain et la géométrie actuelle du véhicule. Comme un contrôleur scripté est difficile à concevoir pour toutes les circonstances possibles, des algorithmes d’apprentissage machine sont une alternative possible. Un réseau de neurones artificiels fonctionnent si une tâche ressemble à un cas appris. Toutefois, il manque de faculté d’adaptation. Apprendre en temps réel par renforcement et estimation de la progression facilite le contrôle du robot et la navigation. Ce rapport présente un algorithme de contrôle par renforcement machine développé pour trouver des alternatives quand le contrôleur réactif reste pris durant la traversée d’un obstacle. Les contrôleurs sont validés et comparés en simulation.

This page intentionally left blank.

Executive summary

Control algorithms for a Shape-shifting Tracked Robotic Vehicle climbing obstacles

I. Vincent; DRDC Suffield TR 2008-123; Defence R&D Canada – Suffield; December 2008.

Background: Unmanned Ground Vehicles (UGV) currently exhibit simple autonomous behaviours compared to the human ability to move in the world. The challenge of analyzing complex environments to traverse obstacles has had very little success due to the complexity of the task. As military conflicts shift from traditionnal battlefields to urban settings, UGV locomotion must improve to better address the needs of the Canadian Forces. This research investigates mobility behaviours that exploit the shape-shifting capabilities of a hybrid UGV that combines tracked and legged locomotion for climbing obstacles.

Principal results: Control algorithms were designed to get the mobile robot to vary its geometry for climbing obstacles. The controllers were validated and compared in simulations. The first scientific contribution is the creation of a world representation suitable to intelligent mobility control algorithms. The second scientific contribution is the development of a reactive controller. It is efficient most of the time and generates a smooth progression of the vehicle. However, it can get stuck. As a scripted controller is difficult to design for every circumstance, learning algorithms are a plausible alternative. The third scientific contribution is the development of a neurocontroller trained with supervised runs. This demonstrates that the robot can successfully learn to climb obstacles by copying a supervisor. However, the artificial neural network does not adapt once the training is over and the motion is not as smooth and predictive as for the reactive controller. The last scientific contribution is the development of a methodology combining the reactive controller with reinforcement learning. This controller provides online adaptation of the reactive behaviour when the vehicle is stuck. By penalizing or reinforcing some actions, the system can optimize the locomotion and overcome bad situations in real-time. This combines adaptation with reactivity creating a more robust controller.

Significance of results: The controllers presented show the promise of using reinforcement learning with reactivity on mobile robots for complex terrain navigation. It incorporates adaptation abilities into the system, and produces improved UGV locomotion.

Future work: Real robot testing will be conducted to verify the controllers reliability and robustness to real environments, sensors and actuators.

Sommaire

Control algorithms for a Shape-shifting Tracked Robotic Vehicle climbing obstacles

I. Vincent ; DRDC Suffield TR 2008-123 ; R & D pour la défense Canada – Suffield ; décembre 2008.

Contexte : Les véhicules terrestres non-pilotés exhibent présentement de simples comportements autonomes en comparaison aux habiletés des humains à se mouvoir dans leur environnement. Le défi d'analyser des environnements complexes pour traverser des obstacles a eu très peu de succès vu la complexité de la tâche. Comme les conflits militaires se déplacent des champs de bataille traditionnels aux milieux urbains, la mobilité des robots doit être améliorée pour mieux répondre aux besoins des Forces Canadiennes. Ce projet de recherche étudie les comportements locomoteurs qui exploitent la capacité d'un véhicule terrestre non-piloté d'adapter sa géométrie pour produire des configurations appropriées pour l'environnement qu'il traverse. Le véhicule hybride combine la locomotion à pattes et celle à chenilles pour gravir des obstacles.

Principaux résultats : Des algorithmes de contrôle ont été développés afin d'amener le robot à varier sa geométrie pour traverser les obstacles. Les contrôleurs ont été validés et comparés en simulations. La première contribution scientifique est la création d'une représentation du monde appropriée pour des algorithmes de contrôle en mobilité intelligente. La seconde contribution scientifique est le développement d'un contrôleur réactif. Il est efficace la plupart du temps et génère une douce progression du véhicule. Cependant, il arrive que le robot demeure pris durant la traversée d'un obstacle. Comme il est difficile de scripter pour toutes les circonstances possibles, des algorithmes d'apprentissage machine sont une alternative plausible. La troisième contribution scientifique est le développement d'un neurocontrôleur entraîné avec un superviseur. Cela démontre que le robot peut apprendre avec succès à gravir des obstacles en copiant un superviseur. Cependant, le réseau de neurones artificiel ne s'adapte plus une fois l'entraînement complété et la progression du véhicule n'est pas aussi douce et prédictive que celle du contrôleur réactif. La dernière contribution scientifique est le développement d'une méthodologie combinant un contrôleur réactif à un contrôleur d'apprentissage par renforcement. Le contrôleur d'apprentissage par renforcement adapte le comportement du contrôleur réactif lorsque celui-ci est pris. En pénalisant ou en renforçant certaines actions, le système peut optimiser la locomotion et se déprendre de mauvaises situations, et ce, en temps réels. Le contrôleur combine la faculté d'adaptation à la réactivité et crée un contrôleur plus robuste.

Signification des résultats : Les contrôleurs présentés démontrent l'intérêt pour l'utilisation de l'apprentissage machine combinée avec la réactivité sur les robots mobiles pour naviguer des terrains complexes. Cela incorpore des abilités d'adaptation au système, et produit une locomotion améliorée du véhicule.

Travaux futurs : Des tests seront conduits sur le vrai robot afin de vérifier la fiabilité et la robustesse des contrôleurs avec de vrais environnements, capteurs et actuateurs.

This page intentionally left blank.

Table of contents

List of figures

Nomenclature

(x_L, y_L, z_L) Range data point Cartesian coordinates in the laser reference frame.

α Learning rate.

β Angle between the track centerline and the bottom of the track.

$\Delta\phi_F$ Front track orientation error.

Δx Forward distance traveled.

γ Discount factor.

$\hat{\phi}_B^d$ Desired back axle angular position estimation.

$\hat{\phi}_F^d$ Desired front axle angular position estimation.

κ Pan angle.

ω_x, ω_y, ω_z Rotational rates.

ϕ_B^d Desired back axle angular position.

ϕ_B^d Desired back axle angular position.

ϕ_F^d Desired front axle angular position.

ϕ_F Current front axle angular position.

π Control policy.

ψ Yaw angle.

σ Roll angle.

τ Staircase inclination.

θ Pitch angle.

ξ Angle between the center and the current laser beam positions.

a Action.

a_x Forward acceleration.

B_H,B_V Orthogonal distances between the robot body centre and the front track origin.

D Range data point.

d Evaluation distance for the reactive controller.

f State transition function.

$Grid_{map}$ Terrain map grid.

H_L Homogeneous transformation matrix from the laser reference frame to the robot reference frame.

I Agent.

i Terrain map x-axis cell index.

I_r Immediate reward.

j Terrain map pan angle index.

K Distance between the front track wheels.

L Distance from the robot body centre to the extremity of the front track in the extended configuration.

LWA Locally weighted averaging.

LWR Locally weighted regression.

P_0 Front track origin.

P_e Progress estimator.

P_k Terrain elevation corresponding to δ_{max}.

$P_{0,ego}$ Vector representing the front track origin in the robot reference frame.

P_{ego} Vector representing a range data point in the robot reference frame.

Q Maximum discounted cumulative reward.

R Big wheel radius.

r Reward.

s State.

S_B Back axle search direction.

S_F Front axle search direction.

SF Safety factor.

t Time.

T_H, T_V Orthogonal distances between the robot body centre and the laser beam.

u Small wheel radius.

V^{π} Discounted cumulative reward achieved by a control policy.

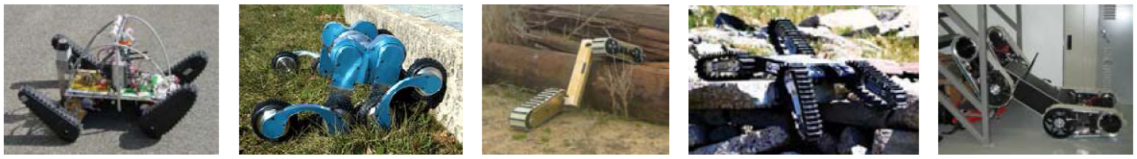
v_x, v_y, v_z	Linear velocities.
IMU	Inertial Measurement Unit.
MDP	Markov decision process.
MSE	Mean squared error.
STRV	Shape-shifting Tracked Robotic Vehicle.
UGV	Unmanned Ground Vehicle

1 Introduction

1.1 Background

Robotics is gaining popularity in military operations to facilitate soldier tasks and decrease their exposure to dangerous situations. Researchers around the world are working on autonomous and semi-autonomous robots to provide soldiers with more intelligent prototypes. The Family of Future Combat Vehicles study [2] from the Canadian Directorate of Land Concepts and Doctrine describes the envisioned unmanned ground vehicles in the Army of Tomorrow. Also, the American Office of the Secretary of Defense published the Unmanned Systems Roadmap 2007-2032 memorandum [3] which emphasizes the importance of unmanned systems as a new class of military tools. The main challenges include partial comprehension of the world due to imprecise sensors, navigation in uncertain environments, control of imperfect actuators to provide useful locomotion in complex terrains, and motion planning according to robot abilities. The intelligence required for autonomous vehicles demands advances in many fields of robotics. Mobility requirements for unmanned ground vehicles (UGV) are expected to increase significantly as military conflicts shift from operations in the open terrain to urban settings.

UGV currently exhibit simple autonomous behaviours compared to the human ability to move in the world. The purpose of this study is to develop intelligent mobility algorithms to create autonomous locomotion in a complex terrain. In this research project, exploration of locomotion is addressed by the variable geometry mobility class of autonomous robotic vehicle behaviours. Shape-shifting robotics is currently an active research area. At the University of Freiburg (Germany), researchers implement autonomous navigation strategies for the *Lurker* robot [4, 5] (Figure 1(a)) to climb ramps and stairs by adapting its tracks' orientation. They are developing a planner which maps terrain classes to specific robot behavioural skills. The Swiss Federal Institute of Technology is developing the shape-shifting robotic platform *Octopus* [6] (Figure 1(b)). This 15 degree-of-freedom hybrid vehicle is designed to climb obstacles and rough terrain, by combining the adaptability of legs with the efficiency of wheels. No controller seems to have been published for that platform. The Defense Joint Robotics Program is developing the *Novel Unmanned Ground Vehicle* [7, 8] (Figure 1(c)), a 6 degree-of-freedom tracked robot, to learn mobility by adaptive control of action patterns (scripted subroutines) based on conditionings. It is currently teleoperated or used with local reactive control. Autonomous Solutions has developed *Chaos* [9, 10] (Figure 1(d)), a small UGV designed for search, reconnaissance and surveillance in unstructured environments. It can walk, climb stairs, clamber over obstacles and traverse steep



(a) Lurker (b) Octopus (c) NUGV (d) Chaos (e) STRV

Figure 1: *Shape-shifting robotic vehicles in different research labs.*

slopes using different modular running gears attached to the vehicle. Intelligent mobility behaviours will be developed to reduce the complexity of the vehicle operation. Finally, the Autonomous Intelligent Systems Section at Defence R & D Canada proceeds to a variable geometry mobile vehicle research project using a small UGV called the *Shape-shifting Tracked Robotic Vehicle (STRV)* (Figure 1(e)), to climb obstacles autonomously.

This technical report describes intelligent mobility algorithms developed for the STRV. This hybrid UGV combines tracked and legged locomotion to produce various configurations suitable to the environment it is dealing with. Tracked locomotion enables fast motion on open terrain. On the other hand, legged locomotion is suitable for complex terrain and climbing over obstacles. A hybrid mechanism combining both locomotions helps in finding suitable solutions to a variety of terrains.

The use of robots for autonomous operations in complex terrains poses difficulty in developing perception, control and learning algorithms that are widely applicable, fast to compute and adaptive to changing ground conditions. The development of intelligence will provide the STRV with the capability of choosing appropriate geometric configurations to interact efficiently with the surrounding environment. Thus, it will be able to handle unforeseen scenarios and to increase the mobility performance.

Research in learning algorithms for mobile robots is limited. It has been applied to path-planning and obstacle avoidance to orient a vehicle in simple structured environments for small navigation tasks [11–20]. There has been limited application of learning algorithms to shape-shifting platforms for choice of geometry based on terrain structure.

1.2 Motivation and problem definition

Algorithms which analyze complex environments in order to climb three-dimensional obstacles have had very little success due to the complexity of the task. The objective of this study is to develop control algorithms which allow the mobile robot to autonomously vary its geometry in order to climb obstacles. This requires a good comprehension of the terrain, and adaptive behaviours handling uncertainties.

In this report, the challenge is limited to:

- Linear shapes such as steps, boxes, ramps and staircases, which are common shapes in urban settings.
- Obstacles fixed and solid.
- Obstacles wider than the vehicle.
- Track nominal propulsion speed constant at 2 km/h.

1.3 Potential contributions

The first scientific contribution is the creation of a world representation suitable to intelligent mobility control algorithms.

The second scientific contribution is the development of a reactive controller providing practical control of actuators and pitch stability. It evaluates the appropriate geometric configuration based on terrain and vehicle geometric considerations. As a scripted controller is difficult to design for every possible circumstances, learning algorithms are a plausible alternative.

The third scientific contribution is the development of a neurocontroller trained offline with supervised runs. This demonstrates that the robot can successfully learn to climb obstacles by copying a supervisor. It removes the necessity to script for every possible obstacle. However, the artificial neural network controller lacks adaptability once the training is over. Some adaptation mechanism needs to be integrated to find a suitable solution when the vehicle is stuck.

The last scientific contribution is the development of a methodology combining a reactive controller with a reinforcement learning controller. The reinforcement learning controller provides online adaptation of the reactive behaviour when the UGV is stuck. By penalizing or reinforcing some actions based on progress estimation, the system can optimize the locomotion and overcome bad situations in real-time. This combines adaptation to reactivity creating a more robust controller.

1.4 Report organization

This study consists of:

Section 1: *Introduction.* The introductory section describes the importance of developing intelligent mobility controllers for navigating complex terrain. It also presents the research problem of how to exploit the variable geometry capabilities of the STRV to autonomously produce various configurations suitable to the obstacle it is climbing.

Section 2: *Literature Review.* This section reviews current research in the field of learning for mobile robot navigation. It first explains why learning becomes essential in controlling UGV behaviour in a complex environment. Then it discusses the importance of reinforcement learning for a real-time mobile robot autonomously exploring its world.

Section 3: *Robotic platform.* This section describes the robotic vehicle, and the sensors used for the project. First, it provides a detailed description of the STRV, its sensors and the hardware. Then, it illustrates the interactions between the robot and its environment.

Section 4: *Perception module.* This section presents a perceptual module developed to fuse sensor data into a terrain map.

Section 5: *Control algorithms.* This section details three controllers designed to autonomously control the vehicle while climbing obstacles. It describes a reactive controller based on geometric considerations, an artificial neural network controller learning mobility offline with supervision, and a reinforcement learning controller improving the robot behaviour based on online reinforcements.

Section 7: *Simulation.* This section reports experiments with the STRV in a simulated environment. Two tests demonstrate the capabilities and limitations of the proposed controllers. The first test evaluates the robustness of the controllers to the variation of height and depth of box-shaped obstacles. The second test determines the steepest staircase each controller climbs and descends for different tread dimensions.

Section 8: *Conclusion.* This section summarizes the results of the experiments and the scientific contributions of the project. It also presents future works and possible improvements to the controllers.

2 Literature Review

Learning for mobile robot navigation is reviewed throughout. The robot environment is first defined. The reason for using learning and especially reinforcement learning has been analysed. Finally, existing control techniques for mobile robot navigation are studied.

2.1 Robot environment

An autonomous robot relies on sensor data to make decisions and navigate within the world. Its mobility is influenced by the terrain geometry and physical properties, and the vehicle's intrinsic capabilities. The terrain types vary from structured to complex. As many types of environment are possible, it is unrealistic to attempt to model all details. The robot has to adapt to the environment and make decisions in real-time. The control and motion planning of robotic platforms in complex terrain with varying ground conditions, sensor measurement uncertainty, planned motion not accurately followed and limited computation resources is a very challenging problem [21].

A typical mobile robot relies on sensors such as a laser range finder for 3D terrain mapping, an inertial measurement unit for orientation, rotational rate and acceleration measurements, a global positioning system for localization (in outdoor applications), etc. The collection of sensory inputs can be used to describe the state of the system. The robot may also have effectors to influence that state, which are usually motors and actuators. The UGV continuously maps the sensory inputs to the effector outputs. An action is an effector's output applied to the environment by the robot. Since the sensory inputs are characterized by uncertainty, error and noise, the robot's comprehension of the state of the world is only partial. This means that taking the same action in the same perceived state may result in a different outcome, therefore this is a nondeterministic system. In contrast, a deterministic system generates the same result given the same action and perceived state.

2.2 Why learning?

A shape-shifting robot must control its effectors to generate useful locomotion patterns. By sequencing different shapes it can progressively conform to the terrain geometry and climb obstacles. In this way, the variable geometry robot has the ability to traverse obstacles that are untraversable for a "single-configuration" robot. Modeling the robot and the environment in an attempt to proceed directly to programming of shape shifting for adaptation to the terrain is unrealistic since there are a very large number of possible states, as well as a large number of possible actions. For the same reason and because the system is nondeterministic, preparation of a table with all state-action pairs is also unrealistic. This problem requires a learning process that can deal with continuous state and action spaces, and optimizes the robot mobility in the case of unplanned outcomes.

The necessity of learning is explained by Kaelbling [22], as follows:

> *"The problem of programming an agent to behave correctly in a world is to*

> *choose some behaviour, given that the rest of the parameters of the agent and world are fixed. If the programmer does not know everything about the world, or if he wishes the agent he is designing to be able to operate in a variety of different worlds, he must program an agent that will learn to behave correctly."*

For these reasons, researchers have focused their efforts on learning algorithms to control robot behaviours. Research in machine learning has seen an increase of interest in the past 25 years. This is partly due to computational resource improvements. Mitchell [1] introduces different types of learning. Kaelbling [22] explores machine learning algorithms in the context of designing embedded systems that adapt their behaviour to a changing environment.

2.3 Why reinforcement learning?

Since mobile robots navigate in complex and nondeterministic environments, learning the function mapping the states to the actions is quite complex. Furthermore, the robot needs to learn and adapt since its sensors provide partial and imperfect representations of the environment. Finally, common machine learning algorithms expect training examples to be a set of state-action pairs. However, for a real-time mobile robot autonomously exploring the world, training examples are usually not available under the form of state-action pairs. For these reasons, researchers have been focusing their efforts in a particular branch of machine learning called reinforcement learning. This methodology learns to map the actions that maximizes reinforcement with training examples of the form $r(s,a)$, where the reward r is a function of the current state s and action a.

A robot perceives the world, with noise and uncertainty associated with its sensors. Through trial-and-error interactions with the environment, the learning system maps the inputs to some rewards or penalties, and learns the desirability of being in various states. Then, the controller chooses the best action to perform based on the information learned in order to achieve its goals. Sutton and Barto [23] give an introduction to reinforcement learning methods.

2.4 Q-learning

Q-learning [24] is an algorithm that can learn the optimal control strategy from the rewards. The best control strategy is the policy that selects the actions that maximize the robot's cumulative reward. The control policy outputs the optimal action, to reach specific goals, given the current state. A reward function assigns reinforcements expressing the desirability of the actions. This function is designed to reinforce or penalize the behaviours based on different observations and goals. According to the actions it chooses to execute, the robot can favour the exploration of unexplored states and actions to acquire new information. It can also favour the exploitation of high reward states and actions already learned to maximize the cumulative reward.

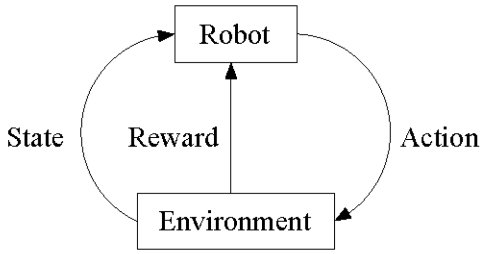


Figure 2: *Robot-environment interactions.*

2.4.1 Markov decision process

A robot interacts with its environment as represented in Figure 2. From Mitchell [1], at each time step t, the robot is in a state s_t and chooses to perform an action a_t. The environment provides the system with a reward $r_t = r(s_t, a_t)$ reflecting the desirability of the chosen action in the current state, and produces the next state $s_{t+1} = f(s_t, a_t)$ where f is the state transition function expressing the transition probability to the next state. Functions r and f are not necessarily known to the system. In a Markov decision process (MDP), the two functions depend entirely on the current state and action, and are independent of all prior history.

Several learning algorithms apply to MDPs. However, real world mobile robots usually deal with continuous state and action spaces rather than a finite set of discrete states. Furthermore, the environment is nondeterministic therefore the transition from one state to another does not depend only on the current state and action. The output can't be predicted with 100% certainty, as with MDPs, because there are multiple possible outcomes for each input. In a complex world, task achievement with the MDP assumption is usually unrealistic. Mataric [25] describes why robots learning in nondeterministic environments do not fit this assumption. Reinforcement learning in continuous state and action space, and nondeterministic system is a current research challenge.

2.4.2 Learning the optimal control strategy

To interact with the environment and reach its goals, the robot must learn a policy that maps states into actions. From [1], if the robot follows the policy π from an initial state s_t, the discounted cumulative reward $V^{\pi}(s_t)$ that the robot will gain if it executes the policy from that state is expressed by Equation 1. The discount factor γ reflects the relative importance of immediate versus delayed rewards.

$$V^{\pi}(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... \tag{1}$$

The robot should learn the policy that maximizes $V^{\pi}(s_t)$ for all states. The optimal policy π^* is as follows:

$$\pi^* = \arg\max_{\pi} V^{\pi}(s), \forall s. \tag{2}$$

This formula is difficult to apply directly, since training data is usually not available in this form. The Q function addresses this problem.

From [1], the maximum discounted cumulative reward that can be achieved is $Q(s, a)$. It is defined as the reward obtained a is performed in s, plus the discounted value of following the optimal policy thereafter.

$$Q(s,a) \equiv r(s,a) + \gamma V^{\pi^*}(f(s,a)) \tag{3}$$

The optimal policy is then the maximum Q value over the entire set of possible actions for that state.

$$\pi^*(s) \equiv \arg\max_a Q(s,a) \tag{4}$$

As

$$V^{\pi^*}(s) = \arg\max_b Q(s,b), \tag{5}$$

where b is a possible action which could be performed in the next state, the Q value can be rewritten as follows:

$$Q(s,a) = r(s,a) + \gamma \max_b Q(f(s,a),b). \tag{6}$$

This formula provides the robot with the possibility of learning the Q function instead of the V^{π^*} function and thus determining the optimal action without prior knowledge of the r and f functions. It only needs to choose the action maximizing $Q(s, a)$.

```
for each state-action pair do
    Initialize the table entry $\hat{Q}(s,a)$ to zero.
end
Sense the current state $s_t$.
repeat
    Choose an action $a_t$ and perform it.
    Receive a reward $r_t$.
    Observe the next state $s_{t+1}$.
    Update $\hat{Q}(s_t, a_t)$ using the following training rule:
        $\hat{Q}(s_t, a_t) \leftarrow r_t + \gamma \max_b \hat{Q}(s_{t+1}, b)$    (7)
        $s_t \leftarrow s_{t+1}$    (8)
until end of run
```

Figure 3: *Algorithm for deterministic MDP from [1].*

Figure 3 presents the Q-learning algorithm for a deterministic MDP. It iteratively approximates the Q values and stores them in a table. There is an entry for each state-action pair. $\hat{Q}$ converges to Q if the system is a deterministic MDP and it converges asymptotically with state-action pair visits.

A nondeterministic environment may generate a different reward each time the same state-action pair is visited. Thus, $\hat{Q}$ is continuously altered and does not converge to Q. A new training rule is required to achieve convergence. From [23], $\hat{Q}$ is given by:

$$\hat{Q}(s_t, a_t) \leftarrow (1-\alpha)\hat{Q}(s_t, a_t) + \alpha(r_t + \gamma \max_b \hat{Q}(s_{t+1}, b)), \tag{9}$$

The learning rate α is commonly decayed to give more importance to the first stages of learning.

As mentioned in [1], Q-learning represents each state-action pair has an entry in a lookup table. It does not generalize Q values for unseen pairs. Since convergence is proven only if every state-action pair is visited infinitely often, it is unrealistic for large and continuous spaces. A solution is to combine function approximation methods with Q-learning. For instance, the lookup table may be replaced by a backpropagation artificial neural network using each $\hat{Q}(s, a)$ update as a training example. The artificial neural network is trained to output $\hat{Q}$ as shown below,

$$input\ (s, a)\ \rightarrow\ artificial\ neural\ network\ \rightarrow\ output\ \hat{Q}(s, a). \tag{10}$$

2.5 Existing control algorithms

The literature contains a number of diversified algorithms developed to navigate mobile robots. Those with the potential of making a contribution to a control algorithm for the STRV are presented here.

2.5.1 Reactive systems

A reactive system senses the environment and reacts to changes. A control engineer programs the action to select based on the sensed environment. It is difficult to script a perfect controller, however reactive systems can be efficient in many circumstances. Some reactive systems for tracked mobile robots climbing stairs can be found in the literature.

Dornhege [4] uses a state machine. From an elevation map, a behaviour map is built by classifying the terrain structures (flat ground, ramp and wall). Then, the starting location and orientation of the transition edge between each structure are evaluated, and a cost is assigned. The A* algorithm is used to plan a path. For each transition, the corresponding skill subroutine, a state machine, is performed. It includes drive a ramp, climb up a stair, lift up and drive down from a pallet.

Fair [26] uses sequences of actions. An automated stairclimbing wheelchair, which is manually operated by the user, is modified to climb stairs autonomously. A laser scanner detects risers and infrared sensors detect dropoffs. The stair negotiating algorithm utilizes a sequence of actions with different triggers for each action transition.

2.5.2 Neural network based systems

ALVINN (Autonomous Land Vehicle In a Neural Network) [17] is a learning system installed on the Navlab at Carnegie Mellon University. It drives in a variety of roads and environments, at speeds up to 55 miles per hour. The network architecture is a single hidden layer feedforward neural network. The input is a 30x32 camera image. It has 4 hidden nodes and 30 output nodes. The output layer represents the appropriate steering angle to maintain the vehicle on the road and prevent collision. The network is trained using the backpropagation learning algorithm. In supervised mode, the network imitates a human driver.

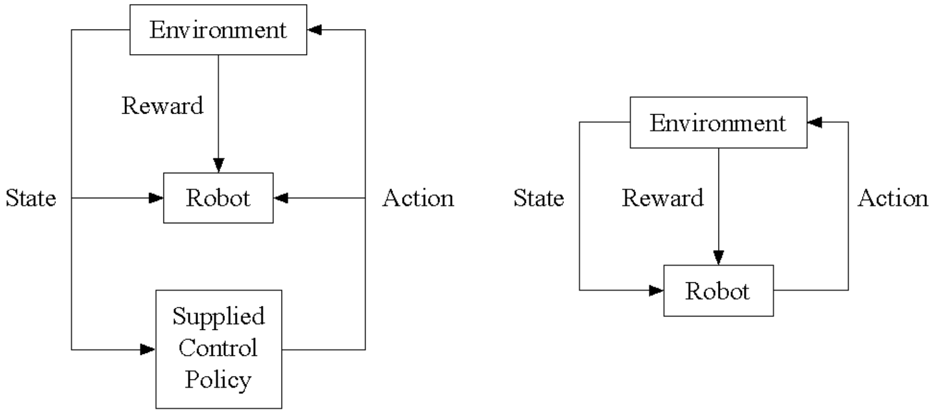
2.5.3 Reinforcement learning systems

Reinforcement learning has been applied to mobile robot navigation tasks such as corridor following, obstacle avoidance, and A-to-B mobility. Those with the potential of making a contribution to a learning algorithm for the STRV are presented here.

2.5.3.1 Acceleration of Q-learning for continuous space control tasks

Smart and Kaelbling [11, 27] introduce a Q-learning algorithm named the HEDGER prediction algorithm. In continuous state control tasks, it replaces Q values lookup tables by an approximation of the target function. Assuming that each training example can be represented by a point in an n-dimensional Euclidean space, HEDGER uses locally weighted regression (LWR) for interpolation within the training data, and locally weighted averaging (LWA) if the query point is outside the training data, to predict an approximation of the target function. LWR uses the training examples near the query point to construct a local approximation of the target function in the neighbourhood of that point. The contribution of each training example is weighted according to a function of its distance from the query point. In contrast, LWA predicts the target function value using all training examples, not just the surrounding points, also weighted by their distance to the query point. The approximation value, or Q value, obtained from LWR or LWA stays the same if no reward occurs. When reinforcement is observed, a standard Q-learning algorithm is used instead to iteratively improve the Q value. The HEDGER algorithm effectiveness has been demonstrated on a real robot executing an obstacle avoidance task and a corridor-following task.

Furthermore, the authors split learning into two phases represented in Figure 4. The first phase is a passive learning process where the robot is controlled by a supplied control policy. For instance, a human can drive the robot using a joystick. The learner passively watches the states, actions and rewards. It bootstraps information into its target function approximation. In the second phase of learning, the learned policy is in control of the robot and learning progresses using a standard Q-learning algorithm. The knowledge acquired in the first stage allows the robot to learn more effectively and reduce the time spent acting randomly in the second phase.



(a) Phase I

(b) Phase II

Figure 4: *Learning phases for the Hedger algorithm. In the first phase, the robot observes passively and learns the supplied control policy. In the second phase, the learned policy is in control of the robot and learning progresses using Q-learning.*

2.5.3.2 Careful design of the reinforcement functions

Mobile robots usually navigate in nondeterministic environments. Therefore, taking the same action in the same state may lead to a different next state and reinforcement. Also, rewards can be inconsistent, immediate or delayed. Mataric [28] discusses the reasons why traditional reinforcement learning methods, applied to deterministic MDP, perform poorly in such environments. The author highlights the importance of carefully designing the reinforcement function if the robot is to be able to learn successfully. She proposes a methodology that embeds human knowledge into reinforcement using heterogeneous reinforcement functions.

An immediate reward occurs when a goal is reached. A progress estimator evaluates the progress done accomplishing a specific goal. If m progress estimators P_e are designed, and n immediate rewards I_r incur, the total reinforcement r gained by the robot at time t is given by:

$$r_t = \sum_{i=1}^{m} w_i P_{e_i} + \sum_{i=1}^{n} w_{m+i} I_{r_i}. \tag{11}$$

The weights w_i correspond to the contribution of each component to the overall reinforcement. The approach was tested in a mobile robot group learning a foraging task. It has been demonstrated that a dense reward function, with multiple reinforcers and progress estimators, significantly accelerate learning.

2.5.3.3 Learning composite tasks with subtasks acting in parallel

Lin [29] proposes a Hierarchical Q-learning architecture (HQ-L) consisting of a group of Q-learning agents that learn the sub-problems of a main problem. When the robot observes a state s, each agent I suggests an action a_I. A switch agent chooses a winner k and executes the corresponding action a_k. The switch agent learns the Q-values $Q(s,k)$ of the best agent to select in each state. $Q(s,k)$ is easier to learn than $Q(s,a)$. The agents learn their Q values simultaneously. The update of the Q values is as follows for the winning agent:

$$Q_k(s_t,a_k) \leftarrow (1-\alpha)Q_k(s_t,a_k) + \alpha(r_k + \gamma \max_a Q_k(s_{t+1},a)) \tag{12}$$

and for the others:

$$Q_I(s_t,a_k) \leftarrow (1-\alpha)Q_I(s_t,a_k) + \alpha(r_I + \gamma \max_a Q_I(s_{t+1},a)). \tag{13}$$

Humphrys [30] compares a standard Q-learning agent to a HQ-L system in a simulation of a “house robot” application. The HQ-L architecture requires less memory since each agent only senses the subspace that is relevant to its reward function, and builds up Q values quicker than the standard Q-learning algorithm.

2.5.3.4 Reinforcement learning and action-selection with heterogeneous goals

Humphrys [30] presents an action-selection method for applications with heterogeneous goals. The author introduces a reinforcement learning method called W-learning. The action-selection process is learned while the W-learning agents compete to control the system. Different agents modify their behaviour based on whether or not they are succeeding in getting the robot to execute their action.

When a state s_t is observed, each agent I suggests an action with its corresponding strength $W_I(s_t)$ or W value. The W value indicates how important a specific action is for that agent. A switch agent selects the highest W value proposed by the agents (Equation 14) and executes its corresponding action a_k.

$$W_k(s_t) = \max_I W_I(s_t) \tag{14}$$

W-learning observes how bad it is when the requested action is not taken in this state by observing the reward r_I and the state s_{t+1} it led to. The agent I estimates a substitute W value, resulting from having executed action a_k in state s_t instead of a_I. When agent k is the winner, all other agents update with:

$$W_I(s_t) \leftarrow (1-\alpha)W_I(s_t) + \alpha(Q_I(s_t,a_I) - (r_I + \gamma \max_a Q_I(s_{t+1},a))). \tag{15}$$

This is the difference between the reward expected from the execution of the agent’s suggested action, and the reward obtained performing the winner’s action.

The agent learns by experiencing what the others want to do. When the leader changes, the agent is not required to learn a new $Q_I(s,a)$, it just changes the W value.

The W-learning algorithm has been tested in an ant world simulation. It represents the conflict between foraging food while avoiding predators. The algorithm has also been tested on a larger state space in a simulated “house robot” context.

Behaviour X	Active	Inactive
Positive reinforcement	a	b
No positive reinforcement	c	d
Negative reinforcement	j	k
No negative reinforcement	l	m

Table 1: *Performance table for an arbitrary behaviour X. It counts the number of times positive or negative reinforcement does or does not occur when the behaviour is active or inactive.*

2.5.3.5 Behaviour activation by reinforcement and precondition fulfilment

Maes and Brooks [15] present a 6-legged robot that learns to coordinate its legs to walk forward. The behaviour-based algorithm learns by reinforcement learning to activate different behaviours. Every behaviour learns when it should be active. This is possible by finding under which conditions the behaviour maximizes positive reinforcement and minimizes negative reinforcement, and how relevant it is to the global goal achievement.

Each behaviour keeps track of its performance in tables. Those tables contain the number of times positive and negative reinforcements happened when the behaviour was active and not active. Table 1 illustrates the performance table for an arbitrary behaviour X.

The correlation between positive reinforcement Pos and the activation status A of the behaviour is defined as:

$$Corr(Pos, A) = \frac{ad - cb}{\sqrt{(c+d)(b+d)(a+b)(a+c)}}. \tag{16}$$

It measures the degree to which the behaviour is correlated with the presence of positive reinforcement.

The relevance of a particular behaviour X is defined as:

$$Relevance(X) = Corr(Pos, A) - Corr(Neg, A), \tag{17}$$

where Neg means negative reinforcement. Relevance evaluates the probability that the behaviour becomes active. The reliability of a behaviour is defined as:

$$Reliability(X) = \min(\max(\frac{a}{a+c}, \frac{c}{a+c}),\ \max(\frac{j}{j+l}, \frac{l}{j+l})). \tag{18}$$

The closer the reliability is to one, the more consistent the behaviour. The algorithm decides whether the behaviour should improve itself or not based on its reliability.

Other statistics are required to select the appropriate behaviour. Some specific conditions are monitored. Table 2 illustrates a table for an arbitrary condition monitored. For instance, e is the number of times positive reinforcement happened when the behaviour X was active

Active Behaviour X	Condition ON	Condition OFF
Positive reinforcement	e	f
No positive reinforcement	g	h
Negative reinforcement	n	o
No negative reinforcement	p	q

Table 2: *Condition table for an arbitrary behaviour X. It counts the number of times positive and negative reinforcement does or does not occur when the behaviour is active and the condition is ON or OFF.*

and the condition was set to ON. Equation 19 evaluates the correlation between a positive reinforcement and a condition set to ON.

$$Corr(Pos, A, ON) = \frac{eh - fg}{\sqrt{(g+h)(f+h)(e+f)(e+g)}} \tag{19}$$

When the system notices a strong correlation between the condition being monitored and a certain reinforcement, it considers this condition as being a precondition for this particular behaviour. When a new condition is learned, the behaviour becomes active only when this condition is obtained.

The algorithm was tested on Genghis, a 6-legged robot. A trailing wheel provides the algorithm with positive reinforcement if the robot moves forward. Front and back touch sensors provide negative reinforcement if the body touches the ground. Each leg has a swing-leg-forward and a swing-leg-backward behaviour. An additional behaviour ensures horizontal balance of the platform. All behaviours learn the conditions under which they should become active.

2.5.3.6 Q-learning combined with neural networks

Junfei Qiao et al. [31] propose a learning action-selection controller for a mobile robot in a goal directed obstacle avoidance task. The authors use Q-learning to navigate autonomously, and a neural network to store the large state-action space. The neural network has a good ability of generalization and is used to approximate the Q function. The multilayer neural network has one input layer for the sensory information (7 sonar distance measurements and the angle between the current direction and the target), one hidden layer of 18 nodes using the sigmoid function, and one output layer generating the Q value for 7 possible steering angles. The neural network is trained with the backpropagation algorithm. The system selects the best action for each state according to the Q value. The reinforcement is based on the distance to surrounding obstacles and the proximity to the target location. The method is tested in a simulated environment on a robot having two steering wheels and one caster.

Similarly, Janusz and Riedmiller [16] apply Q-learning combined with neural networks to a mobile robot obstacle avoidance scenario. Experiments are conducted with a miniature 2-wheel mobile robot with eight infrared sensors located around the robot. The controller learns to select the velocity of the wheels in order to avoid collision. The controller observes the state. The neural network estimates the Q value for each possible action (forward, slow right/left turn, fast right/left turn). The action providing the best Q value is executed. Then, the system gets a reward and updates the weights in the neural network. The reinforcement is positive if the robot does not collide with an obstacle, negative otherwise.

Researchers have used reactive, artificial neural network and reinforcement learning controllers in mobile robot navigation tasks. Some interesting algorithms have been presented in this literature review. In complex terrains, robot mobility could be improved by learning the appropriate behaviours for the situations encountered. Reinforcement learning is an attractive concept since it can learn in real-time based on rewards and penalties. A lot of work still has to be done to provide robots with the ability to efficiently learn to behave in complex environments.

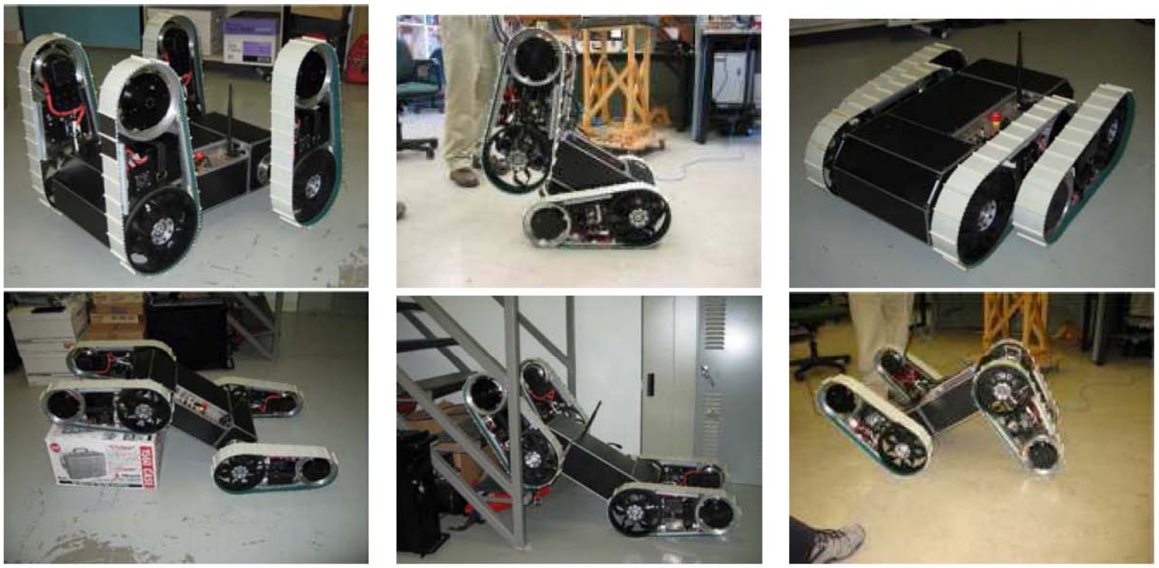


Figure 5: *STRV in different configurations.*

3 Robotic platform

The Autonomous Intelligent Systems Section at Defence R & D Canada proceeds to a variable geometry mobile vehicle research project on a small UGV called the STRV. Previous publications have presented the STRV, the variable geometry paradigm, the perception challenges to navigate with this platform, its characteristics and architecture [32–35].

3.1 STRV

Figure 5 presents the STRV in different geometric configurations. This section describes the platform and the sensors chosen for this research project.

3.1.1 Platform

The platform consists of four independently driven tracks with two solid axles articulating the front and rear track pairs. Novel control methods will take advantage of the small size of the UGV, its robustness, its few degrees of freedom and its inherent ability to change geometry. The small size of the vehicle permits driving through typical door frames and staircases, making it a good platform for indoor applications. In addition, its ability to shift configuration provides an excellent solution to adapt to obstacle shapes.

The STRV's reference frame coordinate system is egocentric, defined at the robot body center, with the axes defined as: x-axis parallel to the forward motion, z-axis up along the gravity axis, and y-axis subsequently defined using the right hand rule. Roll σ occurs about the x-axis, pitch θ about the y-axis and yaw ψ about the z-axis. Figure 6 sketches the egocentric reference frame. Note that the forward velocity v_x, used in this document,

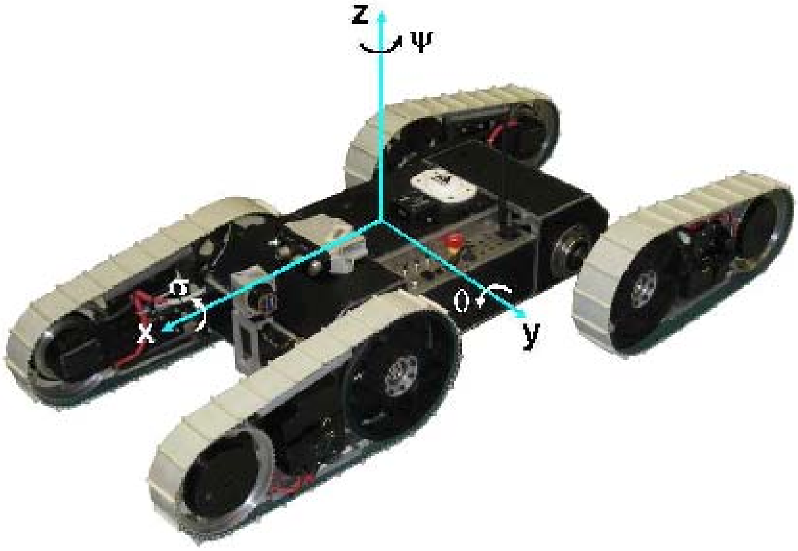
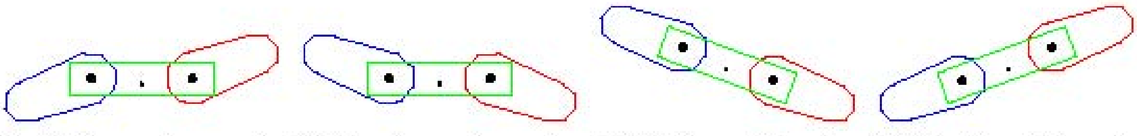


Figure 6: *Robot reference frame.*



(a) Positive axle angular position (b) Negative axle angular position (c) Positive pitch angle (d) Negative pitch angle

Figure 7: *Sign conventions of the axle angular position and the vehicle pitch. The front track is red and the back track is blue.*

is along the x-axis from the robot reference frame, which stays horizontal even if the robot pitches. Similarly, the upward velocity v_z is along the gravity axis. Figure 7 illustrates the sign conventions used in this document. The sign of the vehicle pitch is defined by the right hand rule using the robot reference frame, and the sign of the tracks' orientation is defined by the model in the simulator.

3.1.2 Sensors

As a small scale vehicle, the STRV has limited load carrying capacity with limited available power. As such, it poses a challenge when selecting sensors. They need to be small and light, and have a low power consumption. The robot has been outfitted with the following perception sensors:

- To orient the vehicle in the world and evaluate its progress traversing the terrain, the robot needs inertial measurement information. A good compromise between size and accuracy is the Microstrain 3DM-GX1. It gives triaxial acceleration, rotational rate and orientation with respect to the vehicle body center and magnetic north.

- To traverse complex terrain, a robot requires a good knowledge of the shape and location of obstacles. The Hukoyo URG-04LX scanning laser range finder shows the distance and direction to obstacles. It is mounted at the front of the vehicle to perceive an obstacle's shape and determine its location. The laser executes a 180° vertical scanline. The range data shows the vertical contour of the obstacles such as staircase, step, trench, wall and ramp. A pan mechanism rotates the URG laser over 90° degrees which, when combined with the vertical scanlines, produces a 3D scan of the environment, as shown in Figure 8.

  It was decided that collecting vertical scan lines and panning the camera provides a better precision to determine the vertical shape of the obstacles than a horizontal configuration. The URG laser makes 512 range measurements per scan line, and the pan mechanism makes up to 19 pan angles. For this reason, there is a better resolution

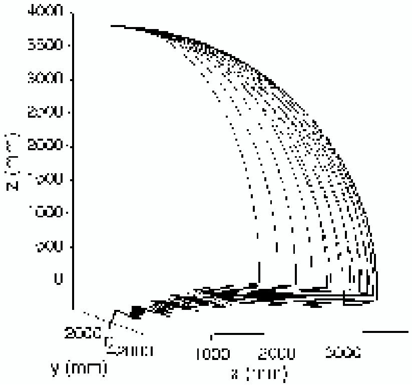


Figure 8: *3D scan of the environment using the Hukoyo URG-04LX scanning laser range finder panned by a servo controller mechanism.*

vertically (along the scan line) than horizontally. Since good vertical shape detection is desired to cross the obstacles, with accurate height and depth measurements, the vertical scan line is more appropriate than the horizontal scan line. In fact, the horizontal scan provides information to evaluate the obstacle width and the robot tolerates inaccuracy of the obstacle width measurement.

- The platform has six optical encoding systems: one on each track and one on each axle. Thus, the internal sensors provide the track velocities and the axle angular positions.
- For trajectory planning, a camera is mounted on the body. An operator can investigate the area and remotely control the trajectory of the robot by looking at the camera image.

3.2 Interactions

The robot perceives the environment and makes decisions on how to behave. The consequence of its motion is a change in the state. Figure 9 shows the sensory inputs and the controlled outputs to navigate the robot in the world. The inertial measurement unit (IMU) provides orientation, rotational rate and acceleration, the encoders provide track velocity and axle angular position, and the laser range finder provides distance measurements. The robot can then actuate the track velocity and the axle angular position to interact with its surroundings.

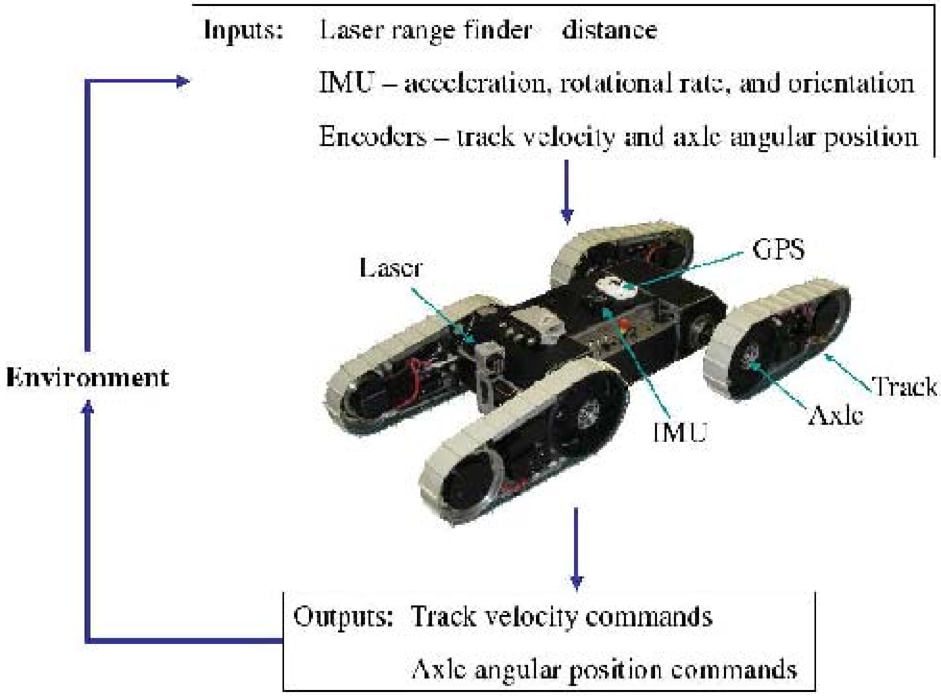
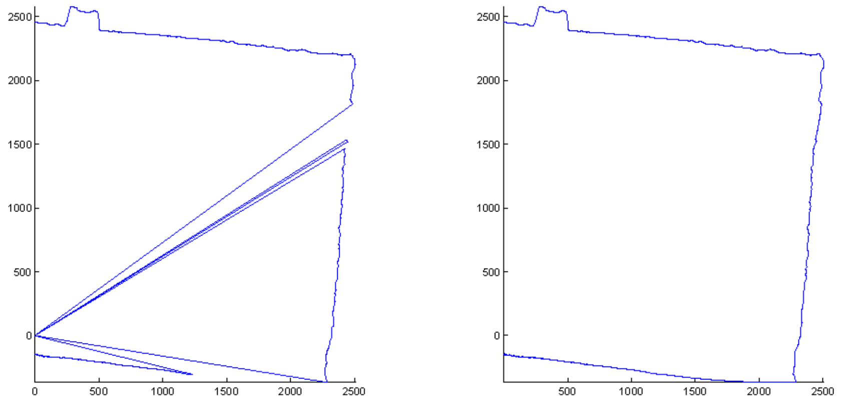

Inputs: Laser range finder distance
IMU – acceleration, rotational rate, and orientation
Encoders – track velocity and axle angular position
Laser
GPS
Environment
IMU
Track
Axle
Outputs: Track velocity commands
Axle angular position commands

Figure 9: *Robot interactions with the world through its sensors and actuators.*



(a) Range data with invalid detections.

(b) Range data after invalid detections removed by interpolating with adjacent valid datapoints.

Figure 10: *Illustration of real range finder raw data with invalid detections and the corresponding corrected image.*

4 Perception algorithm

In order for the STRV to interact with its surrounding, it must recognize obstacles that may affect its behaviour. The research project focuses on detecting physical characteristics such as obstacle location, height and depth that are useful for making control decisions. This section presents a first attempt to build a perception algorithm for intelligent mobility navigation.

The digital elevation map, or terrain map, is the most widely used world representation in mobile robot navigation [36–40]. This research project exploits elevation mapping to control the STRV negotiating obstacles.

For a vehicle operating under continuous motion, a persistent terrain map must fuse range data where each terrain scan is acquired at a different vehicle pose. The laser range finder generates a 180° vertical scan line of 512 range datapoints. The angle between the center and the current laser beam positions is ξ. A pan mechanism rotates the URG laser sensor κ degrees every scan line to cover a 90° field of view in front of the STRV.

The laser is influenced by lighting conditions. Therefore, the raw range data includes noisy data. A first thing to do before analyzing the data is to remove those invalid detections. The method used is the interpolation with valid adjacent datapoints. Figure 10 shows raw range data with invalid detections and the resulting range data after interpolation.

A range data point D is first converted from a spherical (D, ξ, κ) to a Cartesian (x_L, y_L, z_L)

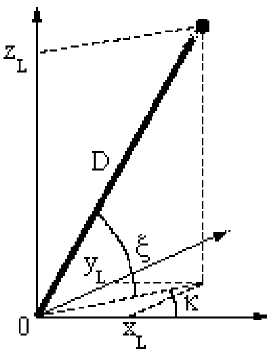


Figure 11: *Spherical to Cartesian coordinate system conversion.*

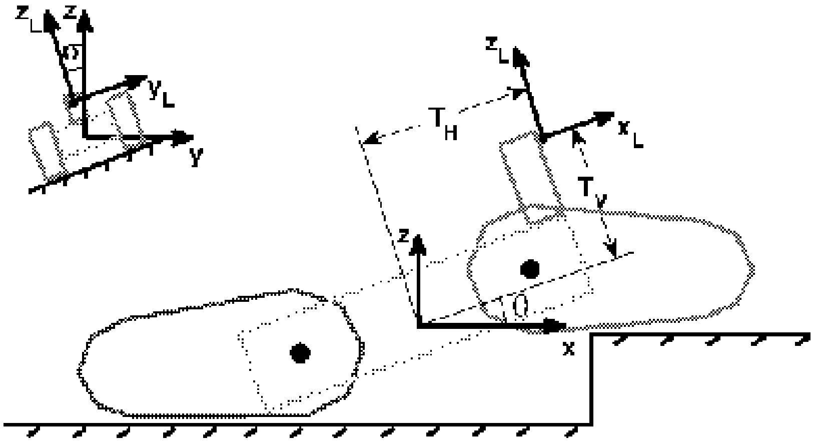


Figure 12: *Laser reference frame (x_L, y_L, z_L) and robot reference frame (x, y, z).*

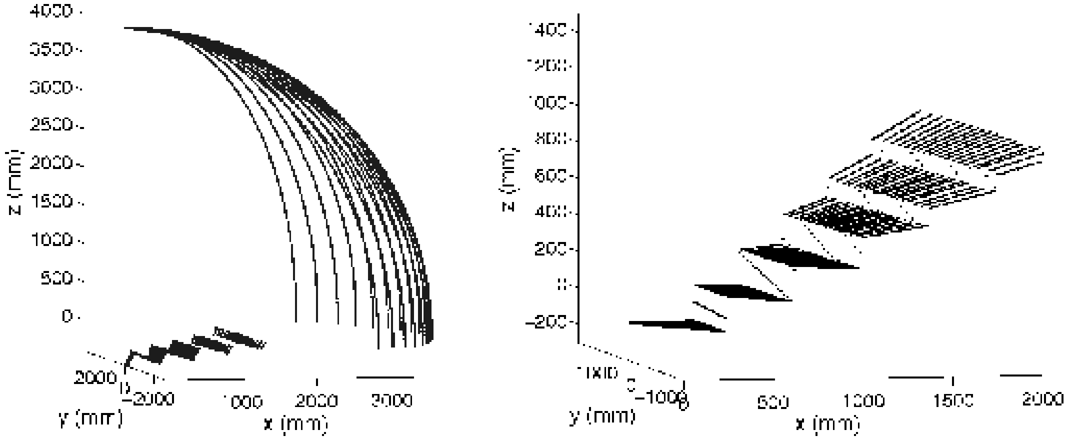
coordinate system in the laser reference frame as follows:

$$\begin{aligned} x_L &= Dcos(\xi)cos(\kappa) \\ y_L &= Dcos(\xi)sin(\kappa) \\ z_L &= Dsin(\xi). \end{aligned} \tag{20}$$

Figure 11 illustrates the conversion.

To convert the range data point from the laser reference frame (x_L, y_L, z_L) to the robot reference frame (x, y, z), the homogeneous transformation matrix H_L is applied as:

$$P_{ego} = H_L \begin{pmatrix} x_L \\ y_L \\ z_L \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}, \tag{21}$$



(a) Raw laser range data.

(b) Terrain map after ceiling extraction, homogeneous transformation and grid cell mapping.

Figure 13: *3D range detection of an ascending staircase.*

where H_L is as follows:

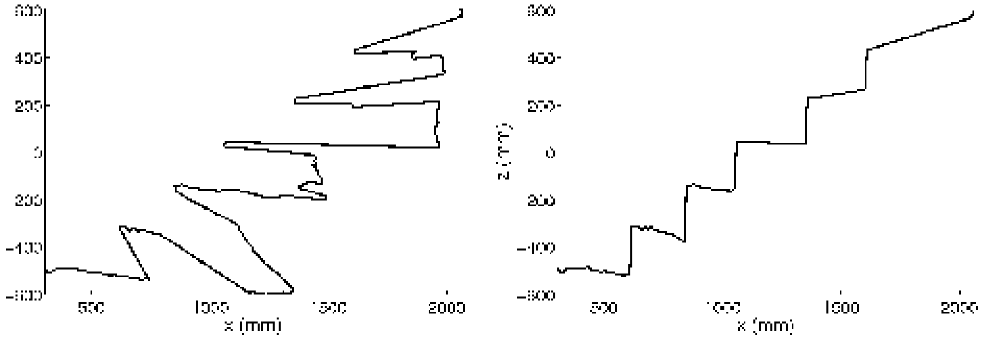
$$H_L = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) & T_H cos(\theta) - T_V sin(\theta) \\ -sin(\theta)sin(\sigma) & cos(\sigma) & -cos(\theta)sin(\sigma) & -sin(\sigma)(T_V cos(\theta) + T_H sin(\theta)) \\ sin(\theta)cos(\sigma) & sin(\sigma) & cos(\theta)cos(\sigma) & cos(\sigma)(T_V cos(\theta) + T_H sin(\theta)) \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (22)$$

and where σ and θ are the vehicle roll and pitch respectively. Figure 12 shows the variables used in the conversion. T_H and T_V are the orthogonal distances between the robot body centre and the laser beam. The resulting vector P_{ego} is the representation of the data point in the robot reference frame.

As each vertical scan line covers 180°, it is necessary to differentiate the ground datapoints from the ceiling datapoints, as shown on Figure 13(a). For this reason, the algorithm keeps datapoints from the start position of the scan line, below the robot, up to the furthest location along the x-axis, for each scan line. It rejects the following datapoints in the scan line. This assumption allows mapping of the ground and the obstacles rather than the ceiling. However, it does not represent an overhanging obstacle, which is not considered in this research project.

The terrain map captures the terrain elevation for each grid cell covering a 1-centimeter region along the x-axis. Linear interpolation provides elevation values between range data points. Among all data points associated with the same grid cell, the algorithm keeps the highest elevation value. The grid map is defined as 2 meters along the x-axis, and data are kept as an array of data point vectors P_{ego} per scan line or pan angle. Figure 13 shows the 3D range data and the resulting terrain map for a staircase.

Figure 14 demonstrates the terrain mapping method for a real laser scan line of a staircase without riser. This is a good example of how difficult it is to recognize obstacles with laser



(a) Raw laser range data

(b) Terrain map

Figure 14: *2D range detection of a real ascending staircase without* riser.

range data. Without developing an algorithm keeping only the highest elevation value per grid cell, it would be impossible to detect the staircase without risers.

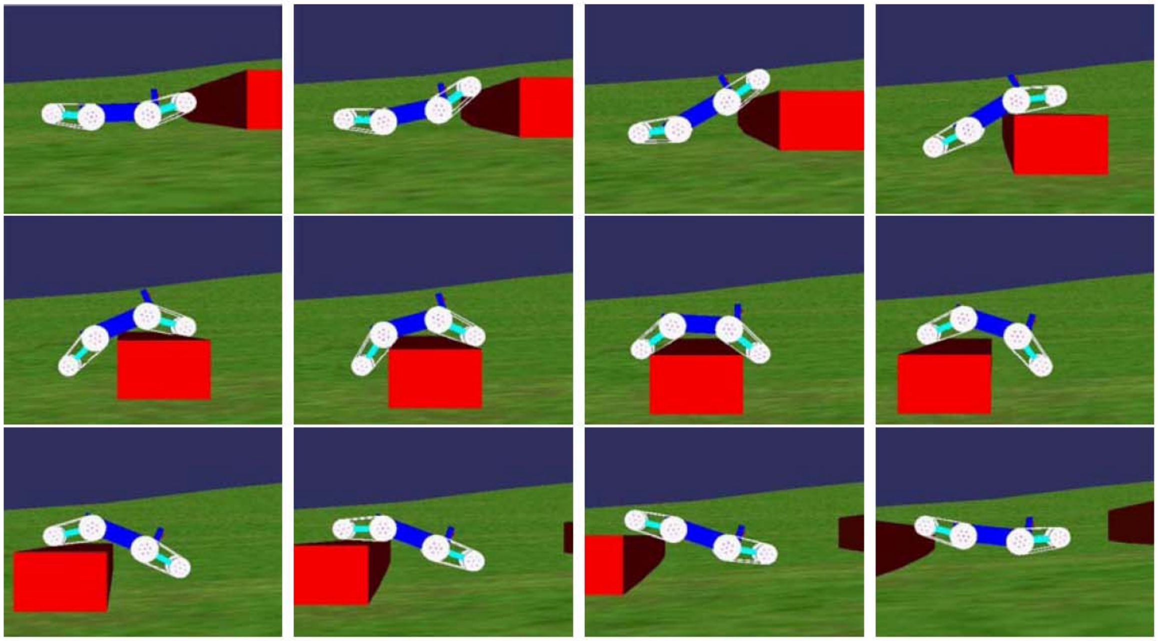


Figure 15: *Box climbing sequence illustrating the variation of the vehicle geometry to conform the terrain.*

5 Control algorithms

A box climbing sequence is depicted on Figure 15 and shows the capability of the STRV to vary its geometry to conform to the terrain. The intelligence to choose within its geometric configurations the one that best fits the situation, allows the robotic platform to interact with the world. This section explains the different control algorithms developed.

5.1 Reactive controller

An attractive concept in robotics is a robot learning in order to behave in a complex environment. However, most machine learning algorithms and neural networks are computationally expensive and require training data. An interesting alternative is to provide a robot with a reactive controller. A reactive system senses the environment and reacts to changes. The control engineer programs the action to select based on the sensed environment. It is difficult to program the perfect controller, however reactive systems can be efficient in many circumstances. This subsection presents a geometric-based reactive controller providing an estimation of a desirable actuation.

The proposed reactive controller orients the tracks. The main idea is to orient the bottom of the front track with the highest elevation in the next near-range distance in front of the vehicle as sketched in Figure 16. The back track follows the front track motion with a delay. This results in a snake-like behaviour.

Figure 17 illustrates the variables utilized in the algorithm,

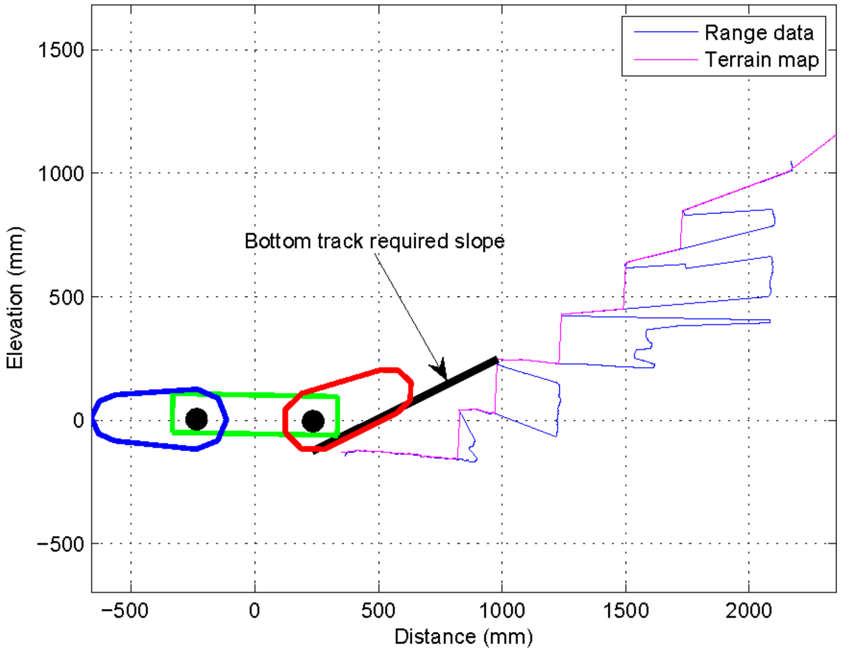


Figure 16: *The STRV climbing a staircase without risers. The laser range data is shown in blue, the terrain map in magenta, and the computed front track required orientation in black.*

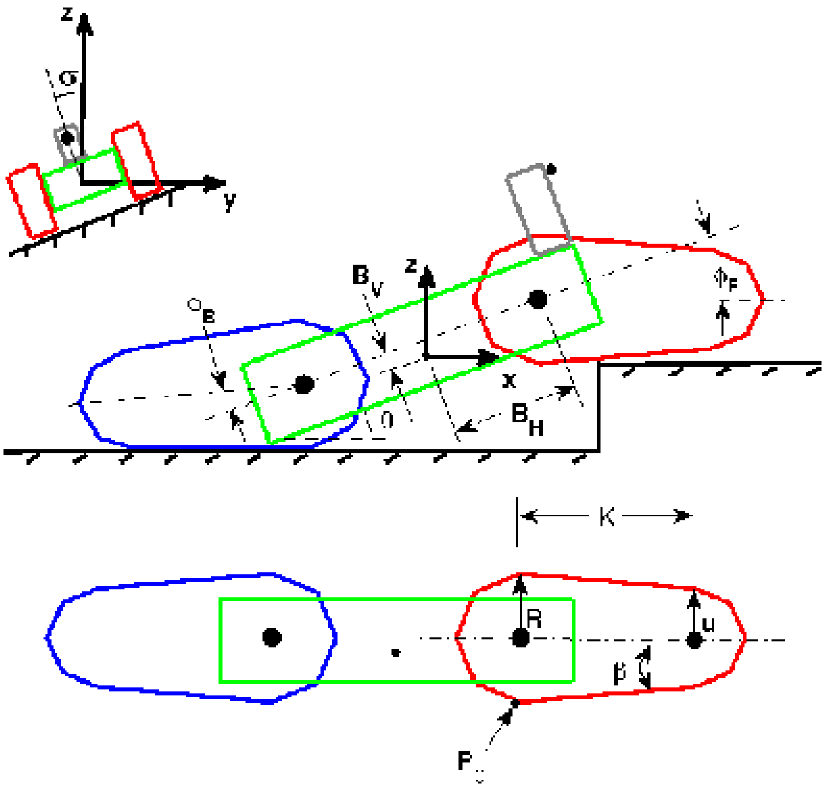


Figure 17: Variables used in the reactive controller.

where:

θ Pitch angle.

σ Roll angle.

ϕ_F Current front axle angular position.

ϕ_B Current back axle angular position.

B_H, B_V Orthogonal distances between the robot body center and the front track origin.

K Distance between the front track wheels.

R Big wheel radius.

u Small wheel radius.

P_0 Front track origin.

β Angle between the track centerline and the bottom of the track.

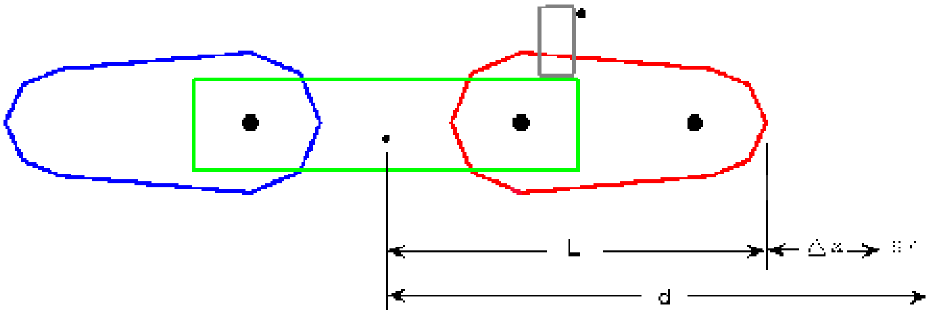


Figure 18: *Representation of the evaluation distance d used to compute a desired front axle angular position.*

The front track origin P_0 is set to the base of the front big wheel as presented in Figure 17. Given the vehicle orientation and axle angular positions, P_0 is first located in the robot reference frame. The homogeneous transformation matrix H_{P_0} is applied as:

$$P_{0,ego} = H_{P_0} \begin{pmatrix} 0 \\ 0 \\ -R \\ 1 \end{pmatrix}, \tag{23}$$

where H_{P_0} is as follows:

$$H_{P_0} = \begin{bmatrix} cos(\theta - \phi_F) & 0 & sin(\theta - \phi_F) & B_H cos(\theta) + B_V sin(\theta)) \\ sin(\sigma) sin(\theta - \phi_F) & cos(\sigma) & -sin(\sigma) cos(\phi_F - \theta) & sin(\sigma)(B_H sin(\theta) - B_V cos(\theta)) \\ cos(\sigma) sin(\phi_F - \theta) & sin(\sigma) & cos(\sigma) cos(\phi_F - \theta) & cos(\sigma)(B_V cos(\theta) - B_H sin(\theta)) \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{24}$$

and where ϕ_F is the current front axle angular position, R the big wheel radius, and B_H and B_V the origin orthogonal distances between the robot body centre and the front track origin. The resulting vector $P_{0,ego}$ is the representation of the front track origin in the robot reference frame.

To calculate a desired front axle angular position, the reactive controller does not consider the entire terrain map. Instead, it considers only a short distance d, called evaluation distance, and represented in Figure 18. d is computed as follows:

$$d = L + \Delta x + SF, \tag{25}$$

where L is the distance from the robot body center to the extremity of the front track in the extended configuration, Δx is the expected forward distance traveled for the next time step t, and SF is a 10-centimeter safety factor to consider further than the track extremity when the velocity is null. The faster the robot goes, the longer d must be to compute a desired front axle angular position since the time the robot has to react is shorter. The expected traveled distance is given as:

$$\Delta x = v_{x_{t-1}} t + \frac{1}{2} a_{x_t} t^2, \tag{26}$$

where v_x and a_x are the forward velocity and acceleration respectively along the robot reference frame x-axis.

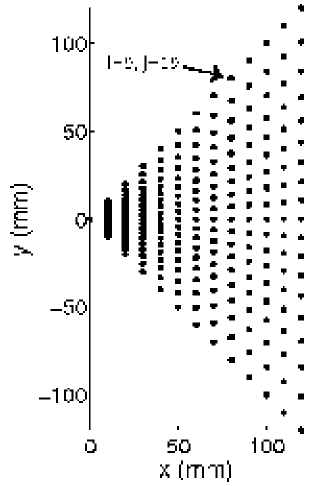


Figure 19: *Partial terrain map grid. Each dot is the (x,y) grid location of a datapoint.*

A terrain map $Grid_{map}$ is illustrated in Figure 19. Each dot is the (x,y) grid location of a datapoint. For row i and column j in $Grid_{map}$, a grid location is expressed by:

$$(x_i, y_{i,j}) = \left(0.01(i-1),\ 0.01(i-1)tan\left(\frac{\pi}{36}(j-10)\right)\right), \tag{27}$$

where $\frac{\pi}{36}$ comes from $\frac{\pi}{2}$ field of view divided in 19 pan angles or 18 increments. The coefficient 0.01 converts the results from centimeters to meters, as each grid cell i covers a one-centimetre region along the x-axis. This is the projection of each dot on the x and y-axis. For instance, the array position $i = 9$ and $j = 19$ gives $(x_9, y_{9,19}) = (0.080, 0.080)$ meters. The magnitude z of the elevation in each (x,y) location can be expressed as $Grid_{map}(i,j) = z$.

As illustrated in Figure 20, δ is the angle between the horizontal and a segment from the track origin P_0 to a terrain elevation. For row i and column j in $Grid_{map}$, δ_{ij} is computed as follows:

$$\delta_{ij} = tan^{-1}\left(\frac{Grid_{map}(i,j) - P_{0,ego}(3)}{\sqrt{(x_i - P_{0,ego}(1))^2 + (y_{i,j} - P_{0,ego}(2))^2}}\right) \tag{28}$$

for all terrain elevations over d. $P_{0,ego}$ is given by Equation 23. The maximum angle is δ_{max} and its corresponding terrain elevation is P_k. The line formed from the track origin P_0 to P_k represents the desired orientation of the bottom of the front track to cross the terrain over d.

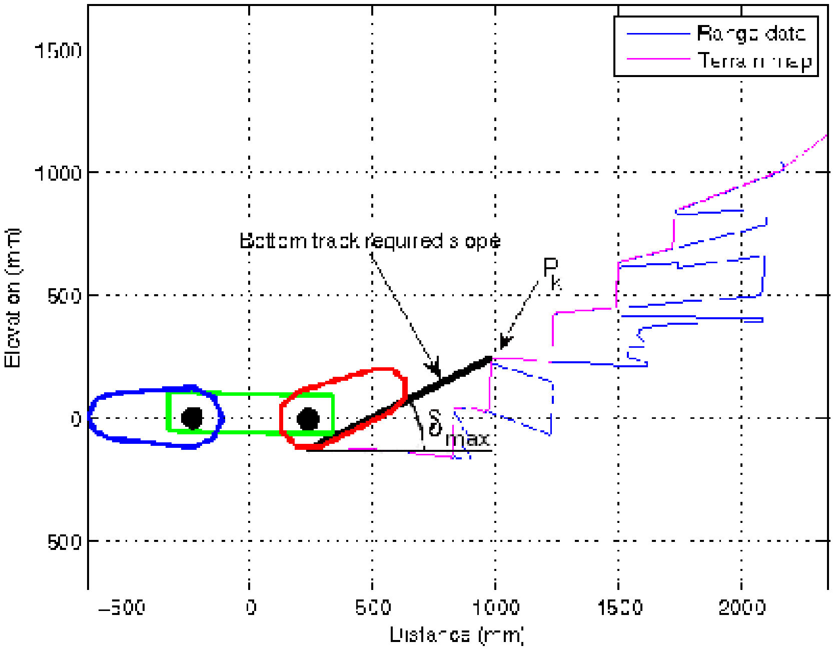


Figure 20: *Desired front track orientation to traverse the terrain.*

To compute the front axle angular position, the orientation variation must be added to the current position. However, δ_{max} is computed for the bottom of the track instead of the track centerline. β represents the angle between the track centerline and the bottom of the track. β is computed by:

$$\beta = tan^{-1}\left(\frac{R-u}{K}\right), \tag{29}$$

where K is the distance between the front track wheels, and u and R are the small and big wheel radius respectively.

Figure 21 illustrates the variables used in the following equations. The desired front track orientation is evaluated by subtracting β from δ_{max} as follows:

$$DesiredTrackOrientation = \delta_{max} - \beta. \tag{30}$$

The current track orientation is given by substracting the pitch angle from the current front axle angular position.

$$CurrentTrackOrientation = \phi_F - \theta \tag{31}$$

Then, the front track orientation error $\Delta\phi_F$ is obtained by substracting the current track orientation from the desired track orientation as follows:

$$\Delta\phi_F = DesiredTrackOrientation - CurrentTrackOrientation \tag{32}$$

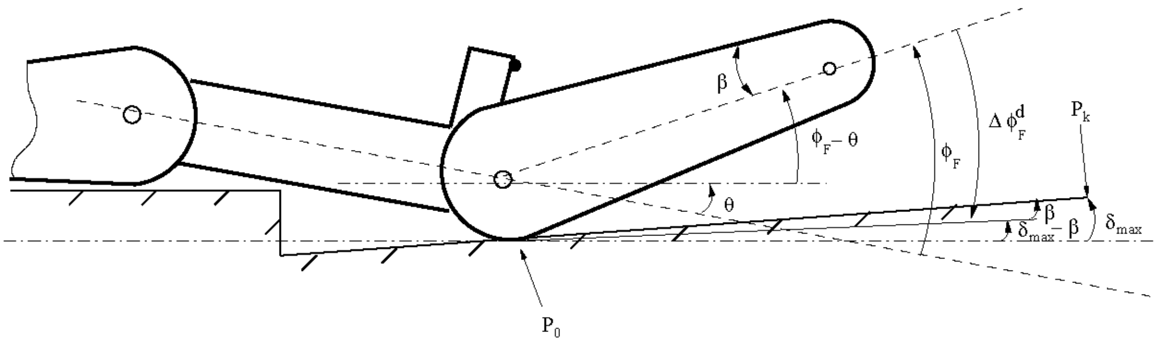


Figure 21: *Variables to evaluate the desired front axle angular position required to orient the track with the terrain to cross.*

Finally, the desired front axle angular position ϕ_F^d is given by the current front axle angular position plus the front track orientation error.

$$\phi_F^d = \phi_F + \Delta\phi_F \tag{33}$$

ϕ_F^d is limited to ±180°.

The back axle executes the front axle opposite commands, but delayed with respect to the vehicle velocity. A table manages sequencing of the commands. The delay is established based on the vehicle nominal speed. It equals the axle span divided by the vehicle nominal speed.

To maintain stability and avoid excessive pitching as much as possible, it is checked that the mass center of the track does not pass above or below the body mass centre. Equation 34 shows the stability equations where ϕ_F^d and ϕ_B^d are the desired front and back axle angular positions, and θ is the vehicle pitch angle.

$$\begin{array}{l}
if\ (\phi_F^d - \theta > 90^\circ)\ and\ (\theta < 0)\ then\ \phi_F^d = -90^\circ - \theta \\
if\ (\theta - \phi_F^d > 90^\circ)\ and\ (\theta > 0)\ then\ \phi_F^d = 90^\circ - \theta \\
if\ (\phi_B^d + \theta > 90^\circ)\ and\ (\theta > 0)\ then\ \phi_B^d = 90^\circ - \theta \\
if\ (\phi_B^d - \theta > 90^\circ)\ and\ (\theta < 0)\ then\ \phi_B^d = 90^\circ + \theta
\end{array} \tag{34}$$

A reactive controller is a good first step. However, human-scripted algorithms prove difficult and time consuming to understand, design, and tune for a UGV that possesses multiple modes of locomotion and navigates various terrains. The production of mobility behaviours needs learning control algorithms and flexibility to changing conditions.

5.2 Artificial neural network controller

Learning is required to improve the robot behaviour and its adaptability to new obstacles. Neurocontrollers have been studied by some researchers to solve mobile robot collision-free path planning problems [41–43]. As stated in [43], the interesting properties of neural networks are their nonlinear dynamics, their natural complexity, and their adaptability and learning ability. Artificial neural networks can be used to copy an existing controller in supervised learning mode, or to self-tune by reinforcement learning.

In the ALVINN project [17], Dean Pomerleau at Carnegie Mellon University used a multilayer feedforward backpropagation neural network to learn steering position based on road images. This concept has several similarities with controlling the angular position of the axles based on terrain elevation images. Following this idea, this subsection describes the control algorithm developed through an artificial neural network for the STRV.

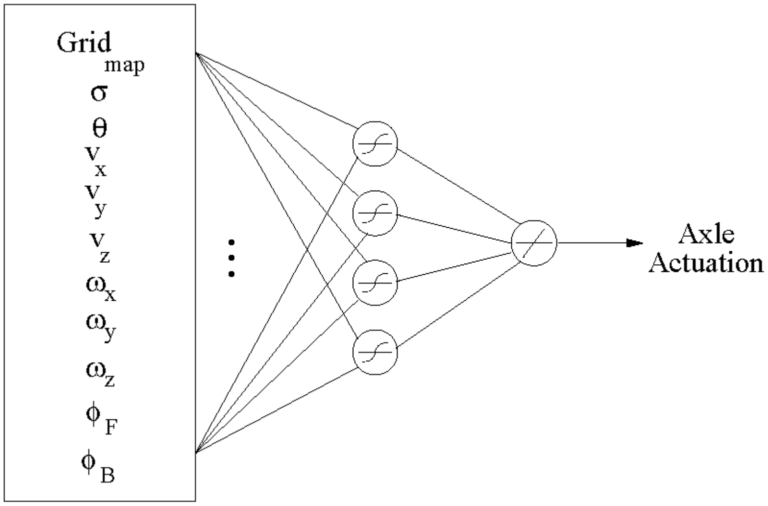


Figure 22: *Multilayer feedforward neural network architecture.*

The neural network architecture, presented in Figure 22, consists of four nodes using the hyperbolic tangent sigmoid transfer function in the hidden layer, followed by a single-node linear transfer function in the output layer. The input layer consists of the roll σ and pitch θ angles, triaxial linear velocity $(v_x,\ v_y,\ v_z)$ and triaxial rotational rates $(\omega_x,\ \omega_y,\ \omega_z)$ in the robot reference frame, and axle angular positions $(\phi_F,\ \phi_B)$. Moreover, the input layer includes the terrain map for 5 pan angles $(-20°,\ -10°,\ 0°,\ 10°)$ of 201 grid cells each as illustrated in Figure 23. The Matlab Neural Network toolkit refuses to process more pan angles using the Levenberg-Marquardt backpropagation algorithm, because it requires too much memory while training the network. However, the system could learn to navigate effectively with only one pan angle, thus using 5 pan angles was only to increase the system robustness and accelerate the learning process. Every input activation is normalized [-1,1]. Each of the 1015 input units is fully connected to the hidden layer units, which are in turn fully connected to the linear output layer unit. The input activations include more than

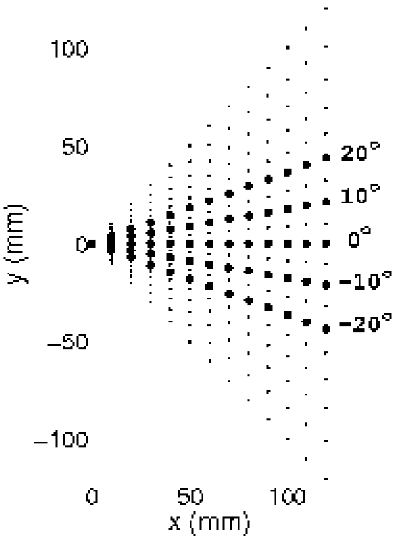


Figure 23: *$Grid_{map}$ cells whose corresponding terrain elevations are used as inputs to the neural network. The biggest grid cells in this partial view of the terrain map grid represent the five pan angles chosen as network inputs.*

just the terrain map to involve vehicle dynamics in the training process. This should help maintain robot stability and motion smoothness.

The neural network is trained to produce a suitable front axle angular position using the Levenberg-Marquardt backpropagation algorithm [44]. In backpropagation, the input activation is propagated forward through the network to determine the output. The result is then compared with the target value. The weights are slightly modified to rectify the output activation error.

The neural network is tuned using a supervised learning technique. The expectation is to drive the robot through some obstacles to tune the network, and then obtain an appropriate motion of the robotic platform traversing similar obstacles with the learned behaviour. Thus, the control engineer wouldn't have to script how to achieve the task, the robot would learn to behave by observing passively the supervisor output for every inputs. Since the robot is broken, it is currently impossible to collect real remotely controlled run data. Also, the simulator doesn't provide the user with the ability to control the simulated robot with a joystick. Therefore, the most readily available supervisor is the reactive controller. It trains the network with appropriate axle angular positions based on the sensory inputs and the robot geometric configuration.

The reactive behaviour has limitations. Sometimes the robot gets stuck on its belly for instance. To train the network, the supervisor must generate successful runs and avoid getting stuck on an obstacle. To counter that problem, whenever the vehicle gets stuck, the robot executes a kick with both axles simultaneously. If it stays stuck after a first kick, it executes a bigger kick, and so on. This kicking behaviour is very simple to implement,

Node quantity	1	2	3	4	5
MSE	0.010498	0.007820	0.005935	0.005407	0.009012
Time (s)	60	213	465	926	1446

Table 3: *Network training results when varying the quantity of hidden nodes. Trials over 25 epochs using the Levenberg-Marquart backpropagation algorithm.*

however flipping over occurs often since the robot gets unstable when kicking while climbing obstacles. The only purpose of this behaviour is to provide the network being trained with a continuous motion of the vehicle crossing obstacles without getting stuck, when the reactive controller does not have an appropriate solution. Only successful runs are used in the training process.

The multilayer feedforward neural network is trained offline for different series of obstacles. The training set consists of upward and downward staircases. They vary from 0.05m to 0.50m tread depth, by 0.05m increments, and from 5° to 50° inclination, by 5° increments, with respect to the supervisor abilities. Moreover, the training set consists of boxes varying from 0.10m to 1.00m deep, by 0.10m increments, and from 0.05m high to the supervisor limits, by 0.01m increments.

Different neural network structures were tested to determine the configuration providing the best performance. First, the number of nodes in the hidden layer was varied. The accuracy of the network was evaluated through mean squared error (MSE). Table 3 shows that 4 nodes provided the optimal performance. Each table entry represents 7 different trials. Since this controller was trained offline, the training time was not considered, only the performance. Second, several training algorithms supported by the Matlab Neural Network toolkit were tested. Table 4 shows the performance obtained for several methods. The training proceeded on 100 epochs. Each table entry represents 2 different trials. The Levenberg-Marquart backpropagation [44], a second-order nonlinear optimization technique, provided the best mean squared error. For the best trial, the mean squared error performance was 0.4% over 25 epochs and Figure 24 shows the training curve obtained. Weights and biases computed during that trial are the ones used by the artificial neural network controller.

Training algorithm	MSE	Time (s)
Levenberg-Marquardt	0.0028	3645
Powell-Beale Restarts conjugate gradient descent	0.0158	39
Polak-Ribiere update conjugate gradient descent	0.0172	36
Scaled conjugate gradient descent	0.0194	31
One step secant method	0.0208	34
Fletcher-Reeves update conjugate gradient descent	0.0256	35
Resilient backpropagation	0.0264	15
Variable learning rate backpropagation with momentum	0.0446	21
Variable learning rate backpropagation	0.0628	18
Gradient descent with momentum	0.1252	18
Gradient descent	0.1297	18

Table 4: *Network training results for various training methods. Trials over 100 epochs.*

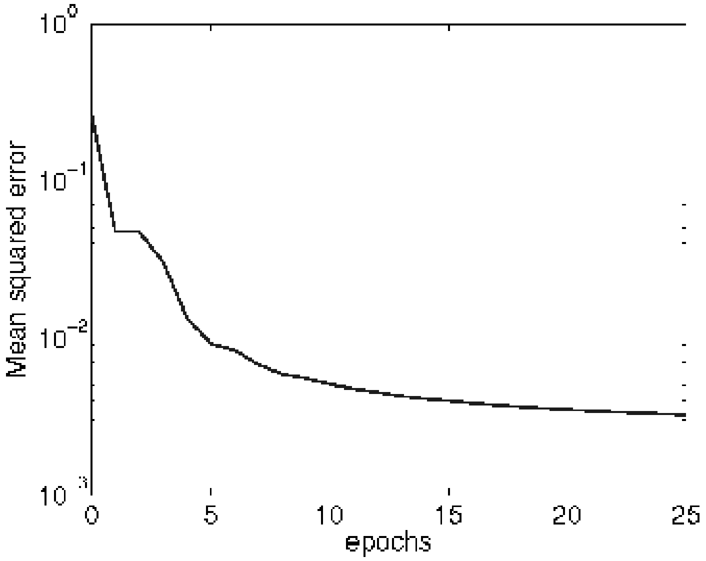


Figure 24: *Best Levenberg-Marquart backpropagation neural network training curve obtained over 25 epochs.*

5.3 Reinforcement learning controller

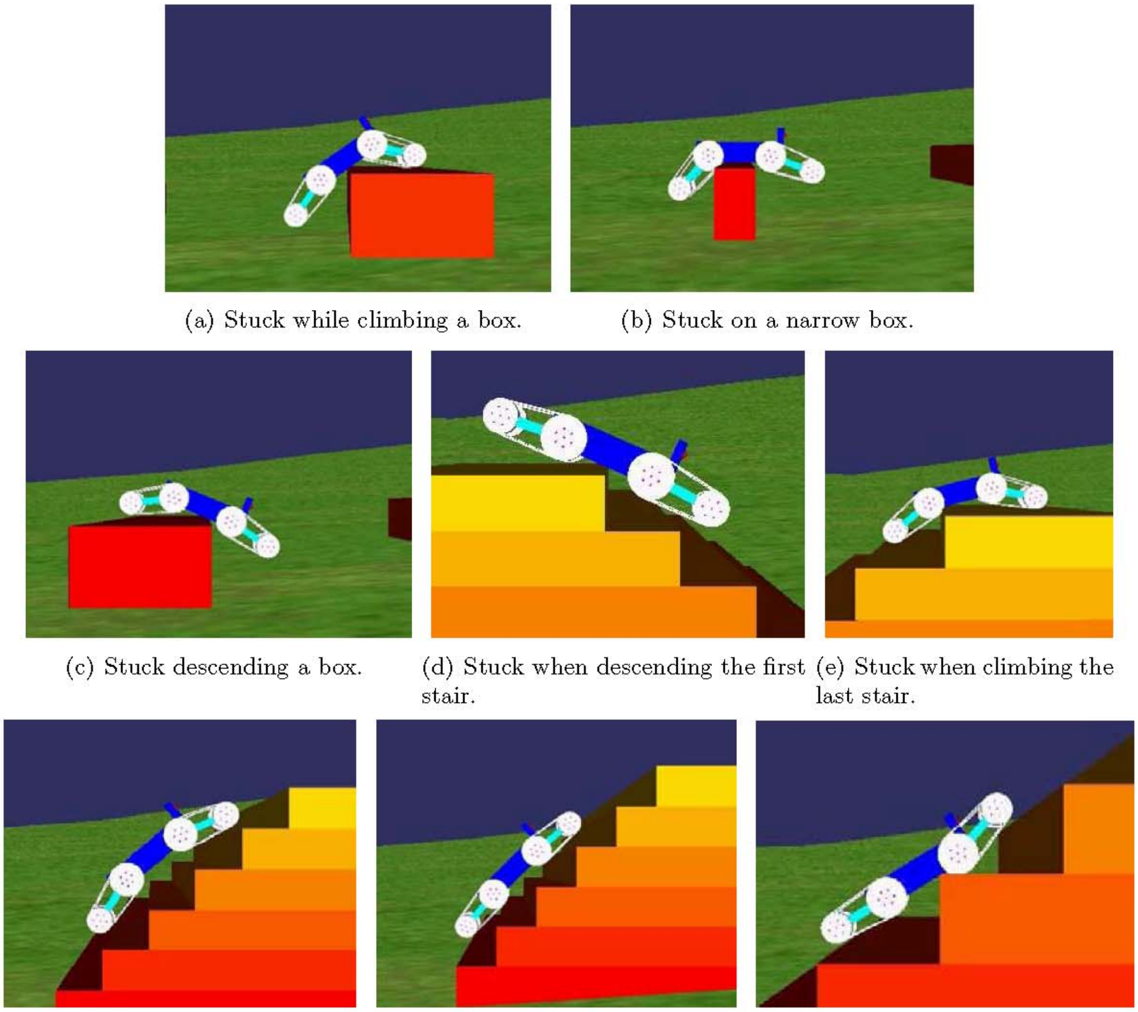
The reactive controller provides a good estimation of an appropriate solution for traversing the terrain. Although this controller is effective and produces a desirable predictive and smooth motion, it gets stuck in some situations. The system is assumed stuck when $\sqrt{v_x^2 + v_y^2 + v_z^2} < 0.01$ m/s or $v_x < 0$. Only forward movements are considered in this work, for this reason when $v_x < 0$ the robot is considered stuck. Also, if the nominal speed of the vehicle is zero, meaning the vehicle is stopped, the stuck situation does not apply. Recall that these velocities are in the robot reference frame and do not change orientation with the vehicle roll and pitch. The reactive controller gets stuck because it generates commands based on near-range views of the environment and the back axle only repeats the front axle commands with a delay. It does not take into account what is going on below the body, robot dynamics and uncertainties in perception. Reinforcement learning will be useful to search for alternative actuations and exit bad situations, when the reactive controller does not have an appropriate solution. It will also attempt to maintain the robot stability during the manoeuver.

Reinforcement learning adapts the robot behaviour to changing environments in real-time and learns actuation online. This report presents the reinforcement learning algorithm developed to select suitable STRV geometric configurations in situations where the reactive controller is stuck. It adjusts the two axle angular positions to make it more fit to progress under the terrain conditions.

Mobile robots rely on their sensors to make real-time decisions and navigate various terrains. They deal with inconsistent, complex and nondeterministic environments. Therefore, it is difficult to design a perfect controller. The solution is to make the robot learn to behave correctly. Reinforcement learning addresses the problem of how to learn the best action to perform based on rewards and penalties incurred from performing particular actions.

Several criteria are considered to build the reward function. During a successful navigation, the robot's forward velocity is important. It slows down during obstacle ascent and the linear velocity along the z-axis increases. A low or negative forward velocity usually signifies a difficult progression. Similarly, a null or small linear velocity along the z-axis while climbing an obstacle means the robot is stuck or in difficulty and should vary its geometry. Moreover, to maintain stability while climbing an obstacle, the more pitched forward the robot is, the less risk there is to back flip. This tends to bring the body closer to the obstacle. Another consideration is the mass center displacement. In most circumstances, when the vehicle is stuck on an obstacle, the robot must raise or move the mass center forward (in the robot reference frame). This increases the chance to exit the dead end situation. A mass center upper displacement, in the robot reference frame, usually brings the body nearer the obstacle, and often, this increases the traction surface area for better propulsion.

Figure 25 presents different situations where the robot is stuck. In the first case, the vehicle is stuck while climbing a box. The back legs need to push the hips upward and the front legs to get more horizontal, pitching the body forward. In case 25(b), the box is narrower than



(a) Stuck while climbing a box.

(b) Stuck on a narrow box.

(c) Stuck descending a box.

(d) Stuck when descending the first stair.

(e) Stuck when climbing the last stair.

(f) Stuck due to inaccurate elevation representation.

(g) Slip in a steep stair.

(h) Stuck on a tread deeper than the vehicle span.

Figure 25: *Situations where the robot is stuck. The reactive controller does not have an appropriate solution to keep progressing.*

the axle span. Therefore, the robot is stuck on its belly and has trouble pushing itself out of the bad situation. For low boxes, the robot may opt to lift itself up by pushing all tracks down, or for tall boxes, flip the back legs up then forward to propel the vehicle forward. In case 25(c), the robot is stuck on the belly while descending a box. The legs must generate a balancing motion to pitch the body forward. The back legs must push or move upward to displace the center of mass forward and help pitching the vehicle forward. The front legs must extend to move the center of mass forward. In the other cases, the robot is stuck in staircases. Cases 25(d) and 25(e) show the situation where the system is stuck on its belly and needs to push more with the back hips. Case 25(f) shows the situation where the robot is stuck in the stairs and the controller does not have an appropriate solution to progress, probably due to an inaccurate representation of the environment. In case 25(g), the robot slips because the staircase is steep. Finally, case 25(h) represents very challenging stairs where the treads are deeper than the axle span. The robot needs to generate a snake-like motion conforming to the shape of the stairs, increasing the traction surface.

Every situation requires a sequence of geometric configurations to exit the bad situation. How to select the configurations and the sequence order vary from one attempt to another. It depends on perception uncertainty, traction quality, obstacle dimensions, and vehicle dynamics. How much to rotate the axles, in which sequence and without flipping over, is a very challenging problem. Reinforcement learning may help to solve that problem.

Figure 26 presents the reinforcement learning controller architecture developed. When the system is stuck, this controller takes over control and adapts the behaviour to solve the situation. Then, the reactive controller takes over control again. Processed sensory inputs are fed into an artificial neural network which outputs the desired front and back axle angular position. On the next iteration, a reward function evaluates how good the vehicle progression is since the performed actuation. The actuation is updated based on the obtained reward. The artificial neural network is then trained to output this new actuation next time it receives the same inputs. If the progression is very bad, an exploration function explores new avenues. In this application, the exploration function switches the search direction in the update function. The following paragraphs detail each step.

The construction of the reward function is a difficult and very important task since it controls the learning process. The system won't learn the task if the reward function is not designed properly. Which elements should be considered for reinforcing an appropriate motion and solve a stuck situation? Through experimentation, it was determined that different stuck situations require different reinforcements to successfully accomplish the task. For this reason, the reward function is split in three components. The classification consists of 1) narrow boxes (Figure 25(b)) where P_k is lower than 0.15 meters below the body center and the vehicle pitch is greater than $-5°$, 2) upward stairs (Figures 25(f), 25(g) and 25(h)) where P_k is positive and θ is negative and 3) steps or any other situations (Figures 25(a), 25(c), 25(d) and 25(e)). After designing this reward function, it was clear that, to expand the robot abilities for climbing more diversified obstacle shapes, it would necessitate adding more classifications in this reward function.

Equation 35 presents the reward function. As recommended in [28], the developed reward

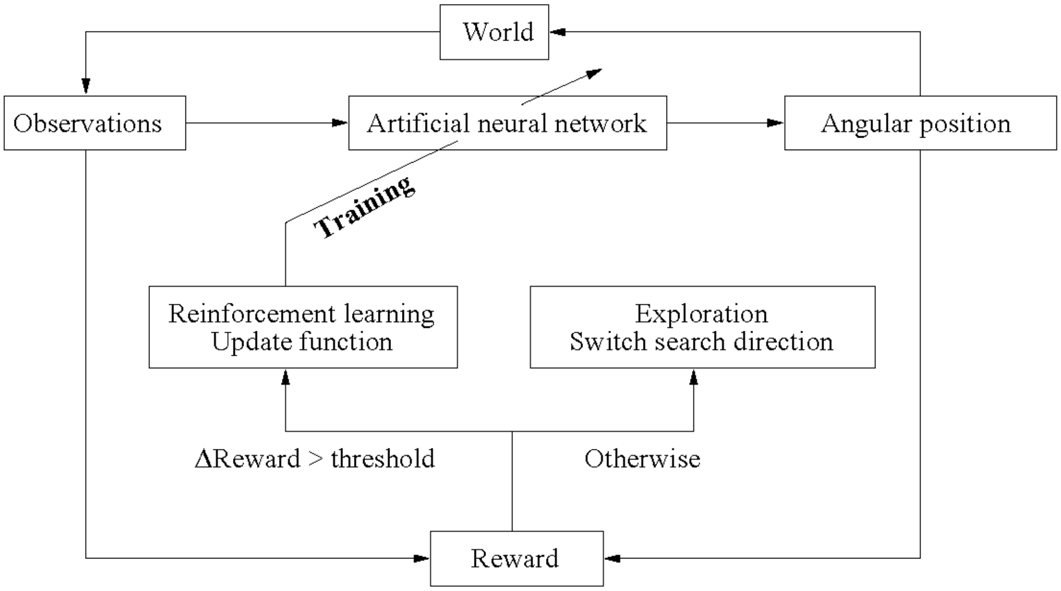


Figure 26: *Reinforcement learning controller architecture.*

function consists of several rewards and progress estimators to orient and accelerate learning. It consists of six elements. P_e is a progress estimator. It equals 1 when v_x is bigger than 0.1 m/s, otherwise P_e equals -1. It gives an important reward when the vehicle gets some forward velocity to reinforce the behaviour. Two other elements are the normalized forward and upward velocities, $\frac{v_x}{v_{max}}$ and $\frac{v_z}{v_{max}}$, reinforcing forward and upward motion. v_{max} is the maximum or nominal speed. Also, the normalized pitch angle $\frac{\theta}{\pi/2}$ reinforces forward pitch to keep the vehicle closer to the obstacle and avoid flipping back over. Finally, it is important to consider the mass center displacement. Appendix A presents the mass center location calculation. Reinforcing a forward mass center displacement (in the robot reference frame), $\frac{D_{x_t}-D_{x_{t-1}}}{|D_{x_{t-1}}|}$, usually reduces the risk of flipping back over. The mass center relative displacement is computed by dividing the displacement of consecutive iterations by the previous location. In the case of a division by zero, this reward component is set to zero to avoid an infinite reward. Similarly, reinforcing an upward mass center displacement (in the robot reference frame), $D_{z_t} - D_{z_{t-1}}$, lowers the body near the obstacle and often increases the traction surface. This should facilitate the vehicle propulsion. Only the sign of the mass center displacement is considered, $sgn\{D_{z_t} - D_{z_{t-1}}\}$. This reward component has been added mainly to solve situations where the robot is stuck on narrow boxes and needs to rotate the back tracks above the body to propel the vehicle forward.

$$r = \begin{cases} a_1 P_e + a_2 \frac{v_x}{v_{max}} + a_3 \frac{v_z}{v_{max}} + a_4 \frac{\theta}{\pi/2} + a_5 sgn\{D_{z_t} - D_{z_{t-1}}\} + a_6 (\frac{D_{x_t} - D_{x_{t-1}}}{|D_{x_{t-1}}|}), \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{if } P_k < -0.15 \ \& \ \theta > -5^\circ \\ b_1 P_e + b_2 \frac{v_x}{v_{max}} + b_3 \frac{v_z}{v_{max}} + b_4 sgn\{D_{z_t} - D_{z_{t-1}}\} + b_5 sgn\{\frac{D_{x_t} - D_{x_{t-1}}}{|D_{x_{t-1}}|}\}, \text{if } P_k > 0 \ \& \ \theta < 0 \\ c_1 P_e + c_2 \frac{v_x}{v_{max}} + c_3 \frac{\theta}{\pi/2} + c_4 (\frac{D_{x_t} - D_{x_{t-1}}}{|D_{x_{t-1}}|}), \ \text{otherwise.} \end{cases} \tag{35}$$

The numeric value for each reward coefficient is presented in Table 5 and expresses the strength of that component on the total reward. The coefficients were tuned by trial and error to reinforce some behaviours in specific situations. This is how the control engineer can incorporate his own knowledge about how to act in specific circumstances. The engineer decides what to reinforce and how much.

Coefficient	1	2	3	4	5	6
a	0.17	0.33	-0.5	0.5	0.17	0.03
b	0.17	0.33	0.17	0.08	0.08	
c	0.17	0.33	0.5	0.17		

Table 5: *Reward function coefficient numeric values. The row indicates the coefficient letter and the column the coefficient subscript.*

An artificial neural network stores the knowledge acquired by experience. The network structure was determined based on results obtained for the artificial neural network controller presented in Section 5.2. It is assumed that the network behaves similarly and the optimal structure for the neural network controller is assumed also optimal for the reinforcement learning controller. This assumption is necessary since it would be difficult to train online different network architectures to find the optimal one. Each run generates a different robot behaviour based on perception accuracy, vehicle dynamics and neural network decisions. Therefore, it would be pointless to compare the different architectures.

Figure 27 presents the artificial neural network architecture. It has 207 nodes in the input layer consisting of the terrain map at 0° pan angle, forward and upward vehicle velocity v_x and v_z, pitch angle θ, current front and back axles angular position ϕ_F and ϕ_B, and P_k from the reactive controller algorithm. Using the central pan angle provides good results. Adding more pan angles may increase the system robustness, but is more computationally expensive. The velocities and pitch angle give knowledge about the vehicle dynamics. The axles angular positions combined with P_k describes figuratively the kind of stuck situation. The hidden layer consists of 4 nodes using a hyperbolic tangent sigmoid transfer function. Finally, the output layer has 2 nodes combining linearly the hidden layer outputs. They represent the desired front and back axle angular positions ϕ_F^d and ϕ_B^d. This is different from the neural network controller which generates the front axle position only. Here, the back axle control assumption of repeating the front axle actuation with a delay does not apply.

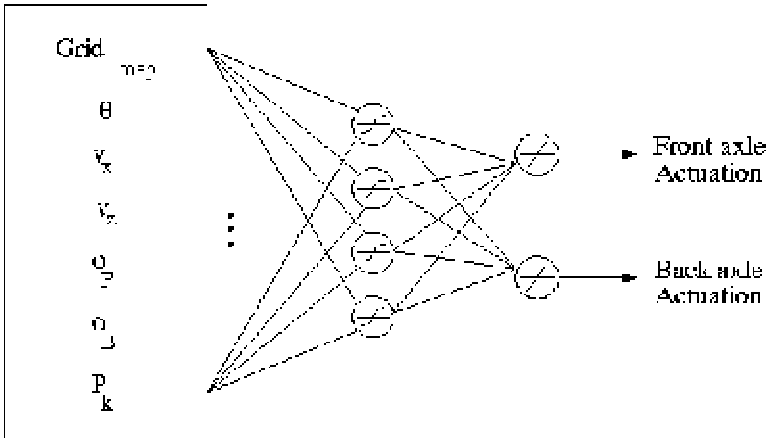


Figure 27: *Artificial neural network architecture for the reinforcement learning controller.*

The multilayer feedforward neural network is trained online using the Levenberg-Marquart backpropagation algorithm.

The system gets a reward each iteration. If both axle actuations were changed simultaneously, it would be impossible to know the impact of one axle actuation on the reward. Therefore, the controller changes one axle command per iteration.

Equation 36 presents the update function. ϕ_F and ϕ_B represent the current front and back axle angular positions. $\hat{\phi}^d_F$ and $\hat{\phi}^d_B$ are the desired axle angular position estimations fed into the artificial neural network. The learning rate α is set to 1. It represents how much of the reward component must influence the actuation. S_F and S_B are the search directions for the front and back axles. When the vehicle is stuck, the control switches from the reactive to the reinforcement learning controller. The artificial neural network outputs the learned actuations for the current situation. The search direction is the motion direction the axles must take to realize those actuations. Suppose the front legs are at 5° and the neural network simulates 10°, then the search direction is positive (10-5=5) and S_F is set to 1. If it was negative, S_F would be set to -1. The same evaluation is done for the back legs and provides S_B. The controller maintains the search direction until the exploration function switches it.

$$\begin{aligned} \hat{\phi}^d_F &\leftarrow \phi_F + \alpha r S_F \\ \hat{\phi}^d_B &\leftarrow \phi_B + \alpha r S_B \end{aligned} \tag{36}$$

This update function is used, rather than the Q-learning update function [24], because the controller was unsuccessful learning with the Q-learning algorithm. Therefore, the update function was simplified to successfully accomplish the navigation task. By analogy, if $\hat{\phi}^d$ is replaced by $\hat{Q}$ (without the subscript indicating front or back axle), the update function becomes:

$$\hat{Q}(s_t, a_t) \leftarrow \hat{Q}(s_t, a_t) + \alpha_t r_t S, \tag{37}$$

compared to the nondeterministic Q-learning update function

$$\hat{Q}(s_t, a_t) \leftarrow (1 - \alpha_t)\hat{Q}(s_t, a_t) + \alpha_t(r_t + \gamma \max_a \hat{Q}(s_{t+1}, a)). \tag{38}$$

It is important to keep in mind that the update function used here calculates an axle actuation instead of a maximum discounted cumulative reward, as for the Q-learning algorithm. All neurocontrollers presented in Subsection 2.5.3 have one neural network output node per possible action. The neural network outputs the Q values for each action and the robot performs the action with the best Q value. In our case, there is a large action space, therefore it is not possible to have one output node per action. For that reason, it was decided to output an actuation instead of a Q value.

The exploration is very important in reinforcement learning to acquire knowledge over the full action space and therefore make better action selection [23]. In this algorithm, the exploration function explores new avenues when the reinforcement learning function acts very poorly. That is, when the reward is worse than the previous reward for that actuator, minus a certain threshold, here 0.1. This threshold represents how long to wait before switching from exploitation of learned information to exploration of new avenues. A small threshold means that the robot explores new avenues more often than it exploits the learned behaviour. In contrast, a high threshold means that the robot is stuck longer before deciding to look for new avenues. The threshold was tuned by trial and error. In future work, a sensitivity analysis on that threshold should be done to figure out an optimal value. The exploration function switches the sign of the update function by inverting the sign of S_F or S_B in Equation 36 based on the axle the reward is associated with. This leads the system to search in the opposite direction. The exploration function is also called when an actuator reaches its limits to search in the other direction.

For safety concerns, the actuations are limited to a maximum of 30° increment per step. This safety is necessary since the network could generate an inadequate command and the vehicle could be damaged.

6 Path planning module

The STRV is not completely autonomous. An operator remotely controls the vehicle heading. The operator tells the robot where to go, but the control algorithm takes in charge mobility adaptation to the terrain, and how to traverse and climb obstacles. This technique reduces considerably the operator labour who just needs to plan the vehicle trajectory. This semi-autonomous characteristic simplifies the control algorithms and adds safety in the navigation process, since the operator tends to choose the paths that seem the most feasible.

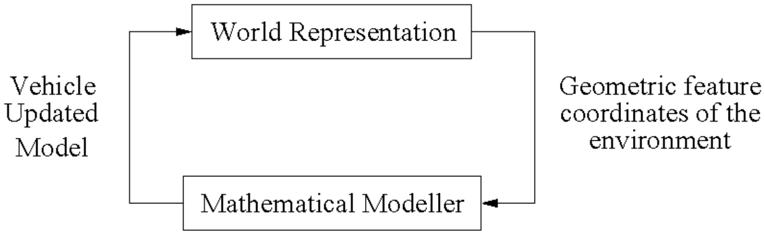


Figure 28: *Geometric feature coordinates of the environment are passed to the mathematical modeller which correctly positions the vehicle into the world.*

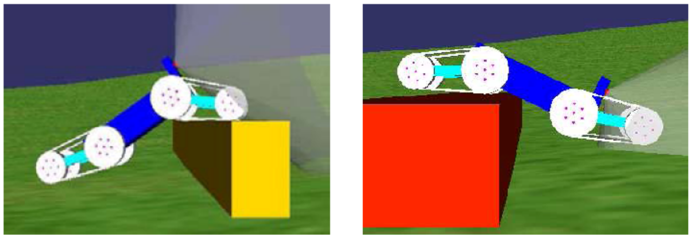


Figure 29: *Model of the STRV in the Vortex simulator.*

7 Simulation

This section presents tests using a simulator and compares the performance of the reactive, the artificial neural network and the reinforcement learning controllers.

7.1 Simulator

The simulator Vortex by CMLabs Simulations Inc. [45], a physics based engine for real-time simulation, is used as a modelling tool. As presented in Figure 28, to fill the gap between the real-world and the controller, relevant geometric features of the environment are extracted from a world representation, whose coordinates are passed on to the mathematical modeller. A model of the STRV that includes its dynamics, is then correctly positioned into the world representation, as illustrated in Figure 29. The Vortex simulator models accurately the physics of the ground vehicle, terrain and real-world objects. Designed for real-time simulation, the Vortex development platform is a great tool for testing and validating the performance and logic of control algorithms.

7.2 Testing and controllers comparison

Two tests were designed to evaluate the performance and limitations of the controllers. They consist of box and staircase crossing, two common obstacles in indoor environments.

7.2.1 Test 1: Box crossing

The first test is a series of boxes of different sizes. It evaluates the robustness of the controllers to the variation of height and depth of box-shaped obstacles. Performance criteria include adaptability to box dimensions, stability maintenance and ability to negotiate the obstacle.

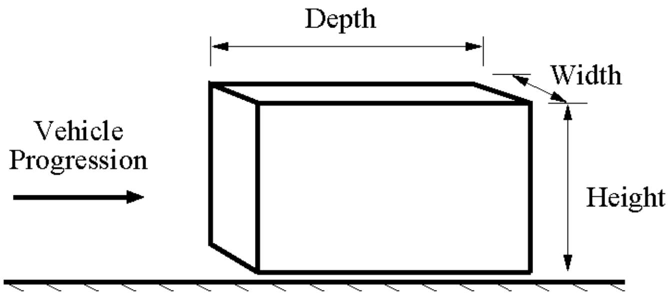


Figure 30: *Box parameters.*

Figure 30 illustrates the box parameters and Figure 31 presents the maximum box dimensions the vehicle can traverse employing a particular controller. For different box depths, the graph shows the maximum box height each controller navigates successfully. For each box depth corresponding to a datapoint in the graph, every height was tried starting at 5 cm by 1-cm increment until an unsuccessful trial occurred. Then many trials around that height were run to determine the maximum traversable box height.

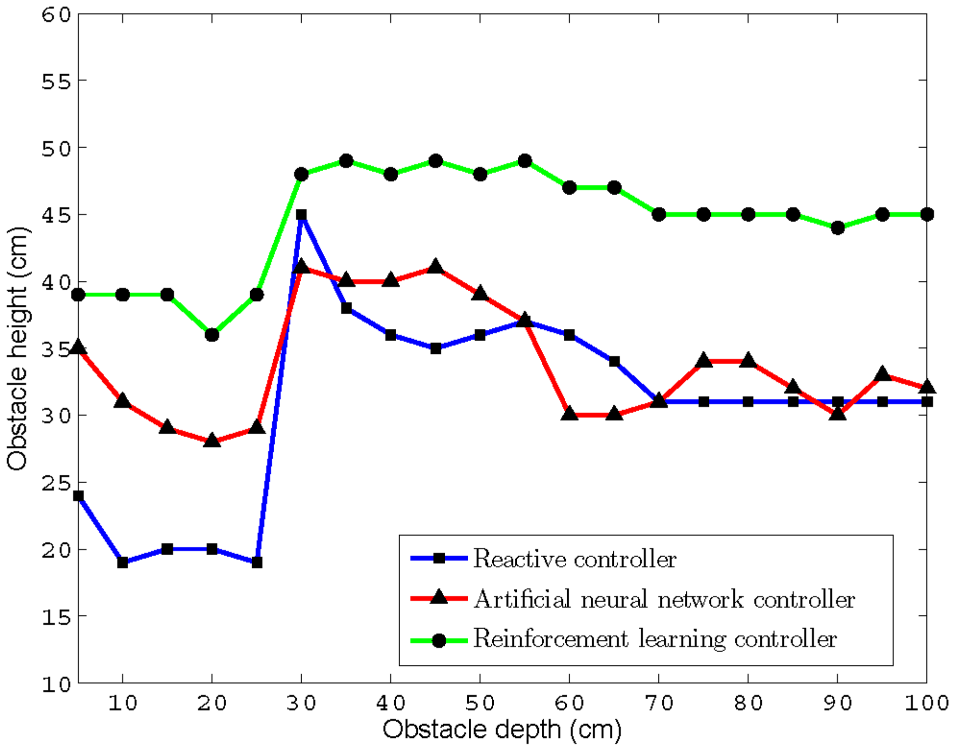


Figure 31: *Plot of the maximum box sizes the controllers traverse driving at 2 km/h nominal speed.*

7.2.1.1 Reactive controller results

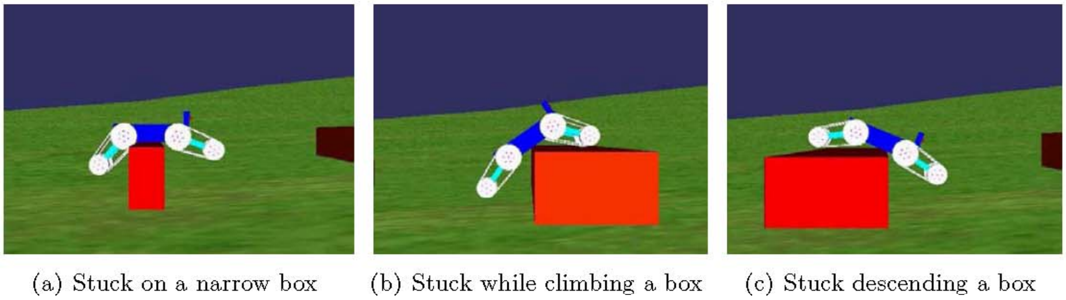

(a) Stuck on a narrow box (b) Stuck while climbing a box (c) Stuck descending a box

Figure 32: *Situations where the robot is stuck when controlled by the reactive controller.*

With the reactive controller, the robot nicely climbs and descends the obstacles. The motion is smooth and predictable. For boxes narrower than the axle span, the robot gets stuck on its belly (Figure 32(a)). It is a very difficult situation to solve and every controller has limited performance with narrow boxes. For deeper obstacles, the vehicle reaches the box height but may not have the back track force to propel the body over the box. In that circumstance, the vehicle gets stuck (Figure 32(b)). Furthermore, the body inertia helps the robot to flip over the box. When the box is too deep, the robot can't balance and it requires more back track pushing force to lift the body. For this reason, the maximum heights reached for deep boxes are lower than those with medium depths. Moreover, the robot becomes quite vertical when attempting to climb tall boxes. Above the limit height, it loses stability and flips back over. Robot and sensors would be damaged if the vehicle flipped over. Finally, the robot may get stuck on its belly when descending a box (Figure 32(c)). This happens when the box edge is in contact with the belly and no track touches the obstacle. In that circumstance, the robot must tilt forward by pushing with its back legs and keeping its front tracks forward to touch the ground.

7.2.1.2 Artificial neural network controller results

The artificial neural network controller successfully traverses the boxes. The motion is not as predictable and smooth as the reactive controller. However, the vehicle maintains better stability and motion continuity while traversing narrow obstacles, and outperforms the reactive controller for every narrow box depth. More training could improve the predictability of the behaviour and increase the robustness of the controller.

As mentioned previously, the behaviour is less predictable and may be erratic. Although the robot can navigate a terrain in several geometric configurations, some have better traction and stability. A solution is to merge the reactive controller to the neural network controller by averaging their actuation outputs. The resulting behaviour is a good combination of both. However, it would be better to use the neural network behaviour only when the reactive behaviour is less effective, since its progression is less smooth and predictable. Unfortunately, the terrain map may not provide the obstacle depth early enough in the motion process to switch the controller in time for generating the appropriate mobility behaviour. If the algorithm waits for the robot to get stuck before switching the controller, the vehicle may not be able to get out of that situation since inertia may be an important factor in the ability to cross a particular obstacle.

Another improvement would be to train the artificial neural network on the reinforcement learning results. It would provide the neural network with a wider range of data since the reinforcement learning controller is more capable than the reactive controller.

7.2.1.3 Reinforcement learning controller results

The reinforcement learning controller improves considerably the reactive behaviour and outperforms the other controllers. It can traverse boxes roughly 15 cm higher than the reactive controller. Except for 30 cm deep boxes where the difference is about 3 cm. This is the transition from narrow boxes, where the vehicle can't propel itself easily, to boxes deeper than the vehicle span, where inertia and traction contribute to propel the vehicle. All controllers behave the best at 30 cm depth because it is the dimension where the inertia helps the most to flip over the obstacle. For deep boxes, deeper than 70 cm, the reinforcement learning controller crosses boxes up to 45 cm high, compared to the reactive controller navigating boxes up to 31 cm high. Between 30 and 70 cm deep, the limitations of every controller diminish progressively. The reinforcement learning controller has the smallest reduction with 49 to 45 cm, compared to 45 to 31 cm for the reactive controller and 42 to 30 for the artificial neural network controller.

In general, the reinforcement learning controller is the best controller to traverse box-shaped obstacles. It improves considerably the reactive behaviour and increases its abilities over time through online learning.

7.2.2 Test 2: Staircase crossing

The second test consists of series of regular staircases with different step sizes. The test determines the steepest stair inclination each controller climbs and descends. Figure 33 illustrates the staircase parameters. A typical staircase has a 178 to 203 mm riser height and 254 to 304 mm tread depth, giving a 30° to 38° inclination shown by τ. The staircase is located at 2 meters from the robot initial position. It consists of 6 steps up, a flat surface of 1 meter, then 6 steps down. The test fails if the vehicle flips over or gets stuck in the stairs. Figure 34 presents the maximum staircase inclination the robot can traverse for various tread depths. Figure 35 presents the results differently, showing the maximum riser height traversed for different tread depths.

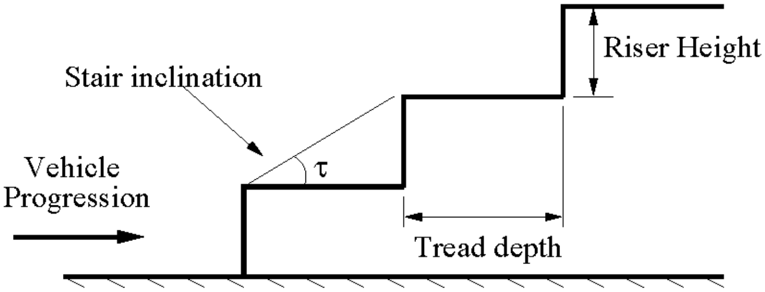


Figure 33: *Staircase parameters.*

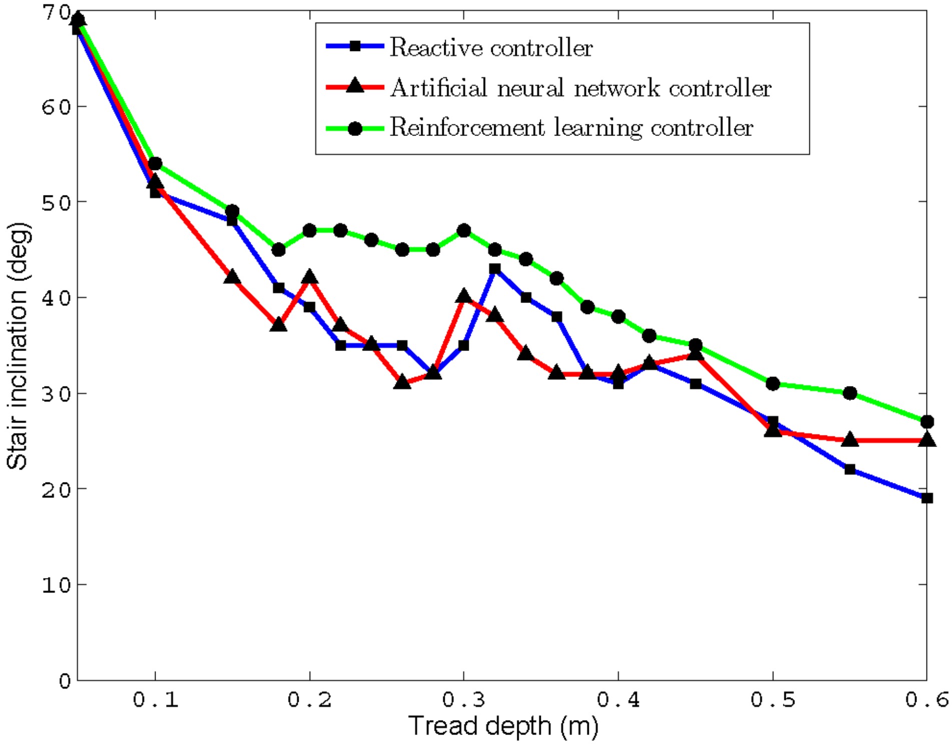


Figure 34: *Stair maximum inclination traversed by the controller for different tread depths, at 2 km/h nominal speed.*

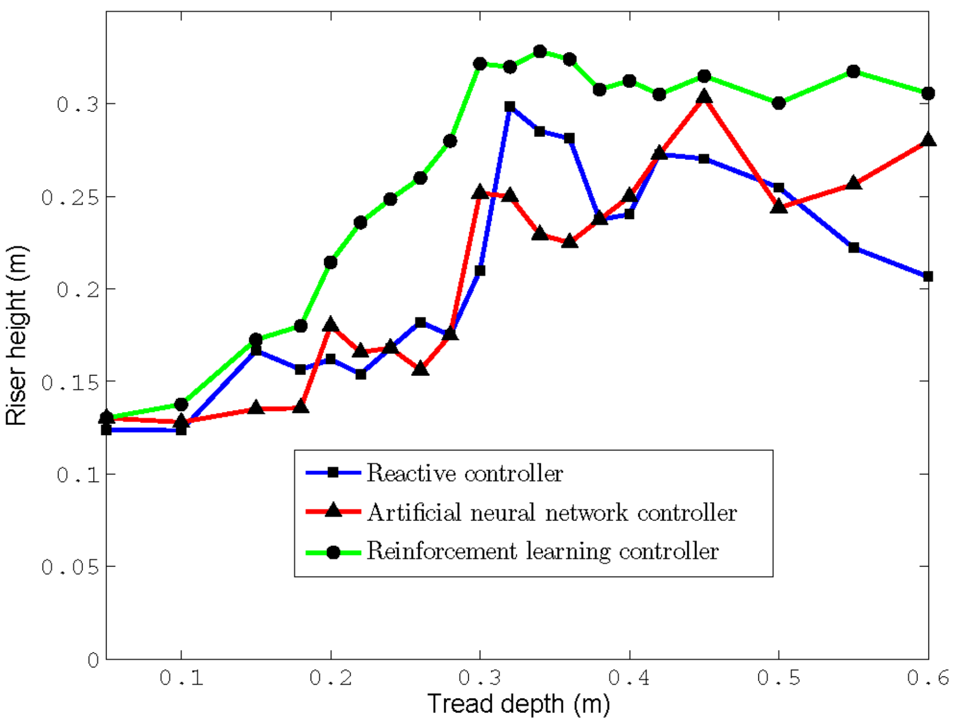
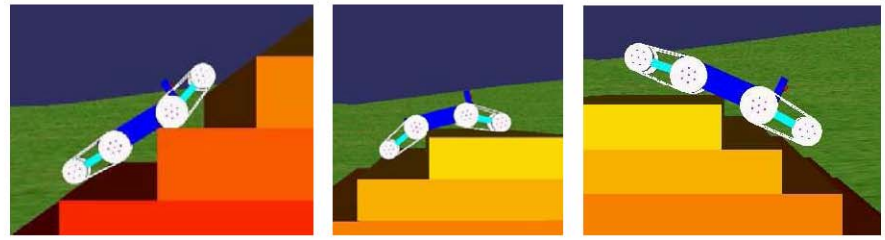


Figure 35: *Staircase maximum riser height traversed by the controller for different tread depths, at 2 km/h nominal speed.*

7.2.2.1 Reactive controller results



(a) Stuck on a tread deeper than the vehicle span.

(b) Stuck when climbing the last stair.

(c) Stuck when descending the first stair.

Figure 36: *Situations where the robot is stuck when controlled by the reactive controller.*

All of the controllers that were investigated are capable of climbing stairs, however, their performance varies. For the reactive controller, when the distance between two consecutive step edges is bigger than the axle span the vehicle may slip or get stuck in the ascent (Figure 36(a)). That distance increases with the tread depth and the staircase inclination. For this reason, as plotted on Figure 34, for deeper treads, the staircase maximum inclinations traversed are lower. Figure 35 shows that, despite reduced maximum inclinations traversed for deeper treads, the riser heights reached are bigger.

With small tread depths, the reactive controller is very capable. The limits reached are higher than the typical staircase inclination range. At those limits, the vehicle flips back over in the ascent or slips down very fast in the descent, since the stairs are too steep.

Another difficulty occurs at the last ascending step (Figure 36(b)) or the first descending step (Figure 36(c)). The robot may get stuck on its belly. The back legs should push while the front legs flatten their configuration. As the back hip imitates the front command with a delay, it flattens instead of pushing the back hips upward. Even if the vehicle could go up or down the staircase, that stair configuration is considered not traversable since the vehicle is stuck at the last step in the ascent or at the first step in the descent.

7.2.2.2 Artificial neural network controller results

The artificial neural network controller outperforms the reactive controller 40% of the time, and under performs 35% of the time. For treads deeper than 51 cm, the learned behaviour is superior. The neural network generalizes better for circumstances where the vehicle tends to get stuck and successfully exits these. The problem with this controller is that it is not as predictive and smooth as the reactive controller. The generated behaviour is based on what the neural network has learned. Sometimes, it selects geometric configurations that are not appropriate even if the supervisor controller used to train it can achieve the task. More training could improve that aspect. In general, the artificial neural network controller generates a trajectory with more hesitation and erratic movements than the reactive controller. This means a waste of time while navigating and maybe a less desirable controller.

7.2.2.3 Reinforcement learning controller results

The reinforcement learning controller improves considerably the reactive behaviour and outperforms the other two controllers. Its performance curve is smoother. Moreover, it is the only controller that is capable of traversing the full range of typical staircases. In fact, for tread depths ranging from 20 cm to 34 cm, it can climb staircases with inclinations up to about 45°. And as shown on Figure 35, it can traverse staircases with risers up to about 30 cm high for tread depth deeper than 30 cm.

In general, the reinforcement learning controller is the best controller to traverse stairs. It improves considerably the reactive behaviour and increases its abilities overtime through online learning. The more often it climbs a staircase of a specific size, the better is its progression to overcome obstacles with similar dimensions. When the controller is not well trained for a particular staircase, it may take a long time to figure out a good behaviour to accomplish the task, but it will eventually find a way to keep progressing.

7.2.3 Summary

In this section, two control problems are used to demonstrate how learning, and particularly reinforcement learning, can solve complex mobility tasks. Three different ways of controlling the STRV behaviour were applied to two navigation tasks. In the first test, the controllers performances to climb boxes were measured, while in the second test, they were evaluated for staircase navigation. In every test, the reinforcement learning controller proved best for climbing obstacles.

The artificial neural network controller learned successfully to climb obstacles after being trained with the reactive controller as supervisor. The motion it generated, however, was less smooth and predictive than the reactive behaviour. This proves that it is possible to train the robot, to cross obstacles, using simulated or remotely controlled runs instead of scripting all behaviours.

The reactive controller performed extremely well in all tests considering it has no learning capability. Reinforcement learning improved considerably the reactive behaviour based on online adaptation. All tests demonstrate that reinforcement learning can improve mobility in complex environment. The results presented here will serve as evidence of the applicability of reinforcement learning to mobile robot navigation in complex environments.

8 Conclusion

8.1 Main results

The development of an autonomous variable geometry mobile robot for complex terrain navigation opens up many exciting avenues in different aspects of robotics research. The controllers that have been demonstrated in this work will provide a stable base for robot navigation in indoor and outdoor environments. The ability to vary the vehicle geometric configuration to conform to the terrain will increase the UGV mobility autonomy in urban settings.

This report proposed autonomous mobility control to perform obstacle traversal for a shape-shifting tracked robotic vehicle. This will reduce the operator workload while guiding the robot to investigate an area. This kind of robot combines tracked and legged locomotion to successfully negotiate obstacles. Tracked locomotion enables fast motion on open terrain. On the other hand, legged locomotion is suitable for complex terrain to climb over obstacles. The hybrid mechanism combining both locomotions helps in finding suitable solutions to a variety of terrain conditions.

A key issue is the creation of a world representation suitable for mobility control. A terrain mapping algorithm has been developed using a laser range finder merged with inertial measurements to build an egocentric elevation map. It provides obstacle location and shape for planning the robot actuation.

Three controllers were designed to autonomously climb obstacles by selecting appropriate vehicle geometric configurations. A reactive controller successfully performed simulated box and staircase navigation. It has the advantage of being completely understood by the control engineer and behaves adequately for an important range of situations. A reactive algorithm, however, is arduous to design and it is difficult to script for every possible circumstance.

To facilitate the controller design process, and adapt in real-time to unforseen conditions, machine learning offers interesting solutions. Offline learning using an artificial neural network proved capable of copying a supervisor to navigate complex terrain. In this work, the reactive controller was used to supervise the neural network training process. Similarly, the system could be trained successfully using remotely controlled or simulated robot runs. This would facilitate considerably the controller design and tuning.

Reinforcement learning proved capable and very attractive to overcome the non-adaptive aspect of the reactive controller. In this work, reinforcement learning adapted the reactive behaviour online when undesirable situations occurred. It found appropriate geometric configurations to progress forward based on experience, rewards and progress estimation. The resulting autonomous mobility adjusted to changing conditions, terrain mapping uncertainties and actuator imperfections. It broadens the applicability of the variable geometry robotic vehicle to complex terrain navigation. Table 6 summarizes the controller advantages and disadvantages.

Reactive controller	Advantages:	1) Predictive 2) Smooth 3) Fast to compute 4) No training required
	Disadvantages:	1) Non-adaptive 2) Gets stuck easily 3) Difficult to anticipate all possible situations when designing the controller
Artificial neural network	Advantages:	1) Can be trained by copying another controller 2) Generalizes well inside the training range
	Disadvantages:	1) Large training data required 2) Non-adaptive after training 3) Generates erratic behaviour sometimes 4) Motion less smooth and predictive than the reactive controller 5) Long tuning process
Reinforcement learning	Advantages:	1) Real-time tuning process based on experience 2) Adaptive 3) Behaviour improves every time it deals with similar situations 4) Continuous learning process
	Disadvantages:	1) Reward and exploration functions difficult to design 2) It may not find an appropriate solution 3) Learns but may forget over time

Table 6: *Summary of the controllers advantages and disadvantages.*

8.2 Future research directions

The next research step is testing the controllers on the real robot. It will be interesting to verify the controllers reliability and robustness to real environments, sensors and actuators.

A second research direction will be the improvement of the terrain understanding. This includes obstacle shape modeling for more complex terrain navigation. Also, track re-engineering for ground tactile sensing would provide information about the terrain surface below the robot. Lauria et al. [6] present tactile sensing concepts developed for wheels. Dornhege and Kleiner [4] uses touch sensors in tracked flippers.

Design of a traversability map would facilitate STRV control. This map would identify the traversable and untraversable areas in the terrain map. The traversability depends on the robot inherent limitations (clearance, width, span, dynamics, etc.), and its behavioral and learning capabilities. The more behaviours developed, the greater the ability to traverse a variety of regions. Dornhege and Kleiner [4] introduce a planning framework which classifies the terrain based on specific skills of the robot *Lurker* and builds the corresponding traversability map.

Finally, for a safe stair ascent, the STRV should autonomously align to the stair edges to provide good traction and stability. It would steer the vehicle to align with the edges. This would permit the robot to climb circular stairs for instance. Furthermore, it should autonomously center the robot relatively to the edges width. This would avoid hitting a wall. Xiong and Matthies [46] have elaborated a vision-guided controller aligning and centering a tracked vehicle climbing stairs.

The controllers presented in this technical report show the promise of using learning techniques on mobile robots. It incorporates adaptation abilities into the system, and produces improved UGV locomotion for complex environments.

References

[1] Mitchell, T.M. (1997), *Machine Learning,* WBC McGraw-Hill, New York.

[2] (2007), The Family of Future Combat Vehicles, Technical Report Directorate of Land Concepts and Doctrine.

[3] Office of the Secretary of Defense, (2007), Unmanned Systems Roadmap: 2007-2032, Department of Defense, Washington, DC.

[4] Dornhege, C. and Kleiner, A. (2007), Behavior maps for online planning of obstacle negotiation and climbing on rough terrain, In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3005–3011.

[5] Kleiner, A. (2006), RescueRobots Freiburg 2006. (Online) Institut fur Informatik Freiburg. http://gkiweb.informatik.uni-freiburg.de/ rescue/robots/robots.php (26 June 2008).

[6] Lauria, M., Piguet, Y., and Siegwart, R. (2002), Octopus: An autonomous wheeled climbing robot, In *Proceedings of the 5th International Conference on Climbing and Walking Robots*, Bury St Edmunds and London, UK: Professional Engineering Publishing Limited.

[7] Blackburn, M. (2003), Learning Mobility: Adaptive control algorithms for the Novel Unmanned Ground Vehicle (NUGV), Technical document Space and Naval Warfare Systems Center, San Diego, CA.

[8] (2004), Novel Unmanned Ground Vehicle (NUGV). (Online) SPAWAR Systems Center San Diego. http://www.nosc.mil/robots/land/nugv/nugv.html (26 June 2008).

[9] Lewis, P., Flann, N., Torrie, M. R., Poulson, E. A., Petroff, T., and Witus, G. (2005), Chaos an Intelligent Ultra-Mobile SUGV: Combining the Mobility of Wheels, Tracks, and Legs., In *SPIE Defense and Security Symposium, Unmanned ground vehicle technology VII*, pp. 427–438, Vol. 5804, Orlando.

[10] (2008), Chaos Datasheet. (Online) Autonomous Solutions Inc. http://www.autonomoussolutions.com/products/products-main/chaos-home-3.html (26 June 2008).

[11] Smart, W.D. and Kaelbling, L.P. (2002), Effective Reinforcement Learning for Mobile Robots, In *Proceedings of the International Conference on Robotics and Automation*, pp. 3404–3410, Piscataway, NJ: IEEE Press.

[12] Conn, K. and Peters, R.A. (2007), Reinforcement Learning with a Supervisor for a Mobile Robot in a Real-world Environment, In *Proceedings of the 2007 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 73–78.

[13] Faria, G. and Romero, R.A.F. (2000), Incorporating Fuzzy Logic to Reinforcement Learning, In *Proceedings of the Ninth IEEE International Conference on Fuzzy Systems*, Vol. 2, pp. 847–852.

[14] Jaksa, R., Majernik, P., and Sincak, P. (1999), Reinforcement Learning Based on Backpropagation for Mobile Robot Navigation, In *Proceedings of Computational Intelligence for Modelling, Control and Automation*, pp. 46–51.

[15] Maes, P. and Brooks, R.A. (1990), Learning to Coordinate Behaviors, In *Proceedings of the 8th National Conference on Artificial Intelligence*, pp. 796–802, San Mateo, CA: Morgan Kaufman.

[16] Janusz, B. and Riedmiller, M. (1995), Self-learning neural control of a mobile robot, In *Proceedings of the IEEE International Conference on Neural Networks*, Vol. 5, pp. 2358–2363.

[17] Pomerleau, D. (1995), Neural Network Vision for Robot Driving, In Arbib, M., (Ed.), *The Handbook of Brain Theory and Neural Networks.*

[18] Zavlangas, P.G., Tzafestas, S.G., and Althoefer, K. (2000), Fuzzy obstacle avoidance and navigation for omnidirectional mobile robots, In *Proceedings of the third European Symposium on Intelligent Techniques*, pp. 375–382.

[19] Saffiotti, A. (7 Aug. 1997), Fuzzy logic in Autonomous Robot Navigation: a case study. (Online) University of Orebro. http://www.aass.oru.sa/Agora/FLAR/HFC (3 Apr. 2006).

[20] Yung, N. H. C. and Ye, C. (1996), Self-learning Fuzzy Navigation of Mobile Vehicle, In *Proceedings of the International Conference on Signal Processing*, pp. 1465–1468, Beijing, China: ICSP 1996.

[21] Iagnemma, K. and Dubowsky, S. (2004), Mobile Robots in Rough Terrain: Estimation, Motion Planning, and Control with Application to Planetary Rovers, Vol. 12 of *Springer tracts in Advanced Robotics (STAR) Series.*

[22] Kaelbling, L.P. (1993), Learning in Embedded Systems, The MIT Press.

[23] Sutton, R.S. and Barto, A.G. (1998), Reinforcement Learning: An Introduction, Bradford Books.

[24] Watkins, C.J.C.H. and Dayan, P. (1992), Q-learning, In *Machine Learning*, Vol. 8, pp. 279–292.

[25] Mataric, M.J. (1994), Interaction and Intelligent Behavior, Ph.D. thesis, Massachusetts institute of technology.

[26] Fair, M. (2000), Autonomous stairclimbing with a mobile robot, Ph.D. thesis, University of Oklahoma.

[27] Smart, W.D. and Kaelbling, L.P. (2000), Practical Reinforcement Learning in Continuous Spaces, In *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 903–910, San Francisco, CA: Morgan Kaugmann.

[28] Mataric, M.J. (1994), Reward Functions for Accelerated Learning, In *Proceedings of the eleventh International Conference on Machine Learning*, pp. 181–189, San Francisco, CA: Morgan Kaufmann.

[29] Lin, L.-J. (1993), Scaling-up reinforcement learning for robot control, In *Proceedings of the 10th International Conference on Machine Learning*, pp. 182–189, Amherst, MA: Morgan Kaufmann.

[30] Humphrys, M. (1997), Action Selection methods using Reinforcement learning, Ph.D. thesis, University of Cambridge.

[31] Qiao, Junfei, Hou, Zhanjun, and Ruan, Xiaogang (2007), Q-learning Based on Neural Network in Learning Action Selection of Mobile Robot, In *Proceedings of the IEEE International Conference on Automation and Logistics*, pp. 263–267.

[32] Trentini, M., Beckman, B., and Vincent, I. (2005), Intelligent Mobility Algorithm Research and Development, (Technical memorandum TM 2005-243) DRDC Suffield.

[33] Trentini, M., Beckman, B., Digney, B., Vincent, I., and Ricard, B. (2006), Intelligent Mobility Research for Robotic Locomotion in Complex Terrain, In *SPIE Defense and Security Symposium, Unmanned Systems Technology VIII*, pp. 21–38, Vol. 6230, Orlando.

[34] Trentini, M., Collier, J., Beckmann, B., Digney, B., and Vincent, I. (2007), Perception and mobility research at Defence R & D Canada for UGVs in complex terrain, In *SPIE Defense and Security Symposium, Unmanned Systems Technology IX*, Vol. 6561, Orlando.

[35] Vincent, I. (2007), Shape-shifting Tracked Robotic Vehicle for complex terrain navigation, (Technical memorandum TM 2007-190) DRDC Suffield.

[36] Broten, G., Giesbrecht, J., and Monckton, S. (2005), World representation using terrain maps, (Technical report TR 2005-248) DRDC Suffield.

[37] Monckton, S., Collier, J., Giesbrecht, J., Broten, G., Mackay, D., Erickson, D., Vincent, I., and Verret, S. (2006), Staged Experiments in Mobile Vehicle Autonomy II: Autonomous Land Systems Demonstration Results., (Technical memorandum TM 2006-243) DRDC Suffield.

[38] Lacroix, S., Mallet, A., Bonnafous, D., Bauzil, G., Fleury, S., Herrb, M., and Chatila, R. (2000), Autonomous Rover Navigation on Unknown Terrains Demonstrations in the Space Museum “Cité de l’espace” at Toulouse, *7th International Symposium on Experimental Robotics, Honolulu, HI (USA).*

[39] Belluta, P., Manduchi, R., Matthies, L., Owens, K., and Rankin, A. (2000), Terrain Perception for DEMO III, In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 326–331, Dearborn, MI.

[40] Bonnafous, D., Lacroix, S., and Simeon, T. (2001), Motion generation for a rover on rough terrains, In *Proceedings of the International Conference on Intelligent Robots and Systems*, Vol. 2, pp. 784–789.

[41] Jazayeri, A., Fatehi, A., and Taghirad, H. (2006), Mobile Robot Navigation in an Unknown Environment, In *9th IEEE International Workshop on Advanced Motion Control*, pp. 295–299.

[42] Janglova, D. (2004), Neural Networks in Mobile Robot Motion, *International Journal of Advanced Robotics Systems*, 1(1), 15–22.

[43] Jaksa, R., Sincak, P., and Majernik, P. (1999), Backpropagation in Supervised and Reinforcement Learning for Mobile Robot Control, *Journal of Electrical Engineering*, 50(7-8).

[44] Hagan, M.T., Demuth, H.B., and Beale, M. (1996), Neural Network Design, PWS Publishing Company, Boston, MA.

[45] (2008), Vortex. (Online) CMLabs Simulations Inc. http://www.cm-labs.com/software/vortex/Vortex4-brochure.pdf (11 Sep. 2008).

[46] Xiong, Yalin and Matthies, L. (2000), Vision-Guided Autonomous Stair Climbing, In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pp. 1842–1847.

Annex A: Mass center location

To evaluate the mass center location, we assume a 2D system in the xz-plane. The vehicle is simplified as a body with two tracks. It is sketched in Figure A.1.

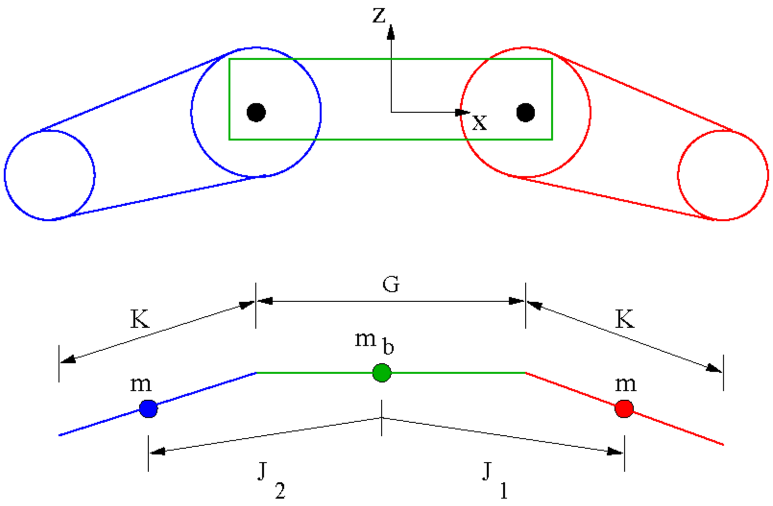


Figure A.1: *Simplified sketch of the STRV.*

The parameters are described in the following list.

m Track mass

m_b Body mass

M System mass

J_1 Front track mass center location vector from the body mass center

J_2 Back track mass center location vector from the body mass center

J_3 Body mass center location vector from the body mass center

J_{sys} System mass center location vector

K Distance between the wheels on a track

G Axle spread

θ Pitch angle

ϕ_F Curent front axle angular position

ϕ_B Current back axle angular position

The mass center of each component is assumed in the middle of that component. Taking the body mass center as reference, each component mass center location vector is determined with the following equations.

$$\begin{aligned} J_{1_x} &= \tfrac{1}{2}G\cos(\theta) + \tfrac{1}{2}K(\cos(\phi_F)\cos(\theta) - sin(\phi_F)\sin(\theta)) \\ J_{1_z} &= \tfrac{1}{2}G\sin(\theta) + \tfrac{1}{2}K(\cos(\phi_F)\sin(\theta) + sin(\phi_F)\cos(\theta)) \end{aligned} \tag{A.1}$$

$$\begin{aligned} J_{2_x} &= -\tfrac{1}{2}G\cos(\theta) + \tfrac{1}{2}K(-\cos(\phi_B)\cos(\theta) + sin(\phi_B)\sin(\theta)) \\ J_{2_z} &= -\tfrac{1}{2}G\sin(\theta) + \tfrac{1}{2}K(-\cos(\phi_B)\sin(\theta) - sin(\phi_B)\cos(\theta)) \end{aligned} \tag{A.2}$$

$$\begin{aligned} J_{3_x} &= 0 \\ J_{3_z} &= 0 \end{aligned} \tag{A.3}$$

Then, the mass center of the system is located by summing the products of each component mass by its mass center location vector. The system mass center location J_{sys} relatively to the body center is given as:

$$J_{sys} = \frac{mJ_1 + mJ_2 + m_bJ_3}{M}, \tag{A.4}$$

where

$$M = 2m + m_b \tag{A.5}$$

is the mass of the entire system.

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when document is classified)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence R&D Canada – Suffield
Box 4000, Station Main, Medicine Hat, Alberta,
Canada T1A 8K6

2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)

UNCLASSIFIED

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

Control algorithms for a Shape-shifting Tracked Robotic Vehicle climbing obstacles

4. AUTHORS (Last name, followed by initials – ranks, titles, etc. not to be used.)

Vincent, I.

5. DATE OF PUBLICATION (Month and year of publication of document.)

December 2008

6a. NO. OF PAGES (Total containing information. Include Annexes, Appendices, etc.)

80

6b. NO. OF REFS (Total cited in document.)

46

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence R&D Canada – Suffield
Box 4000, Station Main, Medicine Hat, Alberta, Canada T1A 8K6

9a. PROJECT NO. (The applicable research and development project number under which the document was written. Please specify whether project or grant.)

9b. GRANT OR CONTRACT NO. (If appropriate, the applicable number under which the document was written.)

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)

DRDC Suffield TR 2008-123

10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

(X) Unlimited distribution
( ) Defence departments and defence contractors; further distribution only as approved
( ) Defence departments and Canadian defence contractors; further distribution only as approved
( ) Government departments and agencies; further distribution only as approved
( ) Defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11)) is possible, a wider announcement audience may be selected.)

Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

Research in mobile robot navigation has demonstrated some success in navigating flat world while avoiding obstacles. However, algorithms which analyze complex environments in order to climb three-dimensional obstacles have had very little success due to the complexity of the task. Unmanned ground vehicles currently exhibit simple autonomous behaviours compared to the human ability to move in the world.

This research work aims to design controllers for a shape-shifting tracked robotic vehicle, thus enabling it to autonomously climb obstacles by adapting its geometric configuration. Three control algorithms are proposed to solve the autonomous locomotion problem for climbing obstacles. A reactive controller evaluates the appropriate geometric configuration based on terrain and vehicle geometric considerations. As a scripted controller is difficult to design for every possible circumstance, learning algorithms are a plausible alternative. A neural network based controller works if a task resembles a learned case. However, it lacks adaptability. Learning in real-time by reinforcement and progress estimation facilitates robot control and navigation. This report presents the reinforcement learning algorithm developed to find alternative solutions when the reactive controller gets stuck while climbing an obstacle. The controllers are validated and compared with simulations.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Mobile robot
Navigation
Control
Perception
Learning

Defence R&D Canada

Canada's Leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

DEFENCE R&D DÉFENSE

www.drdc-rddc.gc.ca