A combined reactive and reinforcement learning controller for an autonomous tracked vehicle

Isabelle Vincent, Qiao Sun

A R T I C L E I N F O

Article history:
Received 17 June 2010
Received in revised form 4 August 2011
Accepted 8 December 2011
Available online 16 December 2011

Keywords:
Control
Learning
Unmanned ground vehicle
Terrain mapping
Tracked robot

A B S T R A C T

Unmanned ground vehicles currently exhibit simple autonomous behaviours. This paper presents a control algorithm developed for a tracked vehicle to autonomously climb obstacles by varying its front and back track orientations. A reactive controller computes a desired geometric configuration based on terrain information. A reinforcement learning algorithm enhances vehicle mobility by finding effective exit strategies in deadlock situations. It is capable of incorporating complex information including terrain and vehicle dynamics through learned experiences. Experiments illustrate the effectiveness of the proposed approach for climbing various obstacles.

1. Introduction

The Autonomous Intelligent Systems Section (AISS) program, at Defence R&D Canada—Suffield, envisions autonomous systems contributing to Canadian Forces operations in the urban battle space. Unmanned ground vehicles (UGVs) are anticipated to access dangerous zones like entering buildings for reconnaissance or surveillance, thus keeping soldiers out of harm's way. Reliable autonomous operations in a complex environment pose many challenges including perception, world representation, navigation, motion planning and machine learning.

Mobility requirements for UGVs are expected to increase considerably as military conflicts shift from open fields to urban settings. Intelligent mobility algorithms need to be developed to allow the UGV to interact closely with its surroundings and successfully cross difficult terrain and obstacles. Such robots usually make use of on-board sensors to investigate the terrain around them and build up an internal representation that enables them to navigate. Despite a large body of research into robotic locomotion, UGVs currently exhibit simple behaviours when compared to the human ability to move in the environment. The problem is to manoeuvre in unknown and complex environments, and this is complicated by a partial comprehension of the world, vehicle speed, various modes of locomotion and mobility objectives.

Human scripted behaviours identify geometric shapes to initiate robotic behaviours. Scripted behaviours may have brittle performance in unknown and complex environments. The UGV must be able to cross obstacles that may degrade the vehicle performance or render the vehicle immobile, as illustrated in Fig. 1. It is impractical to script controllers that can handle highly variable terrain. Learning controllers provide robots the ability of adapting to highly variable environment with a greater chance of success navigating through. Many researchers have studied traversability through learning algorithms. Al-Milli et al. [1] developed an analytical approach to track terrain modelling for traversability prediction. It incorporates vehicle kinematics and dynamics in order to predict the resultant track forces acting between the vehicle tracks and the terrain. However, their work is limited to soft terrain. In general, variables such as soil support, friction, ground contact strength and location cannot be known or measured directly. Machine learning could potentially help associate those variables with sensible quantities, and produce more robust performance when compared to scripted behaviours.

Reinforcement learning aims to learn by exploring different movement strategies, particularly the ones that have successful record. Rather than learning to build a model and then reason for a decision, reinforcement learning combines experience with exploration to reach a decision. It has the potential of offering general solutions to a large variety of tasks performed in different environment.

Crown Copyright © 2011 Published by Elsevier B.V. All rights reserved.
Fig. 1. The UGV must be able to cross obstacles that may degrade the vehicle performance or render the vehicle immobile.

Fig. 2. Variable geometry STRV concept.

Fig. 3. Shape-shifting tracked robotic vehicle.

Online learning and particularly reinforcement learning have been studied by many researchers for mobility control of mobile robots [2–5]. Maes and Brooks [6] used a robot to learn by reinforcement the coordination of its legs to walk forward. Smart and Kaelbling [7] made a mobile robot learn steering commands by reinforcement to execute an obstacle avoidance task and a corridor-following task. Mahadevan and Connell [8] created a behaviour-based robot learning by reinforcement a box pushing task. Mataric [9] tested a group of behaviour-based robots learning by reinforcement to select behaviours in order to find and take home pucks. She demonstrated that a dense reward function with multiple reinforcers and progress estimators accelerate learning. Janusz and Riedmiller [10] applied reinforcement learning, combined with neural networks, to a mobile robot to select the velocity of the wheels in order to avoid collision. Finally, Pomerleau [11] developed a neural network based controller called ALVINN (Autonomous Land Vehicle In a Neural Network) to learn driving on paved roads.

Robots with variable geometry are able to alter their geometry to produce various modes of locomotion, conforming to the terrain to generate a more desirable locomotion. They use hybrid locomotion concepts such as tracks and legs, which yield dynamic behaviours that outperform conventional vehicles in their ability to overcome obstacles. The Shape-shifting Tracked Robotic Vehicle (STRV) is a variable geometry robot which evolved from AISS investigations [12] into high mobility vehicle designs for autonomous operations in indoor and outdoor environments. Fig. 2 presents the STRV concept of a vehicle capable of changing its geometric configuration to adapt to the terrain and mobility task.

Variable geometry robots have become of interest recently to navigate complex terrain. Several tracked robotic platforms are used for the Robot Rescue Competitions where robotic vehicles must navigate hostile environments. There is also the Defence Industry which is investigating robotic platforms for various urban operations. Most of them are remotely operated or may be semi-autonomous. Very few publications can be found on algorithms to control variable geometry tracked robots for obstacle climbing. Dornhege and Kleiner [13] used a state machine. From an elevation map, a behaviour map is built by classifying the terrain structures. Then, the starting location and orientation of the transition edge between each structure are evaluated. For each transition, the corresponding skill subroutine is performed. It includes drive a ramp, climb up a stair, lift up and drive down from a pallet.

In this research, the exploration of locomotion is addressed by using the STRV, presented in Fig. 3, to climb obstacles autonomously by varying its geometry. The development of control algorithms will provide the STRV with the capability of choosing appropriate geometric configurations to interact efficiently with the surrounding environment.

The objective of this study is to exploit the sensory inputs of the STRV to make decisions on how to control the angular position of the vehicle axles while climbing obstacles at a constant nominal propulsion speed. The obstacles are currently limited to linear shapes such as steps, boxes, ramps and staircases, which are common shapes in urban settings.

This research develops a method to combine a reactive controller with a reinforcement learning controller [14]. The reactive controller offers the advantage of fast execution of tasks under normal conditions while the learning controller allows the ability of dealing with large state space and uncertain environment with an increased chance of success. In addition, reinforcement learning enables progressive actions leading to the achievement of an ultimate goal that is often too difficult to script with a single action. The chance of damaging the vehicle is much reduced. As the control engineer cannot anticipate every situation the robot may encounter, there is a limit to what can be scripted. Learning has the potential to overcome this problem. The reinforcement learning controller provides online adaptation to the reactive behaviour in instances when the robot has become stuck. By penalizing or reinforcing some actions based on progress estimation, the system can adjust the locomotion and overcome difficult situations. This combination of adaptation and reactivity creates a more robust controller.

The organization of the paper is as follows. Section 2 describes the robotic vehicle. Section 3 presents the control algorithms developed to vary the vehicle geometry. Finally, Section 4 reports experiments with the STRV in a simulated environment.
2. Robot description

Previous publications [15–17] have presented the STRV, the variable geometry paradigm and the perception challenges to navigate with this platform. This section describes the vehicle and the sensors chosen for this research project.

The robotic platform (Fig. 3) consists of four independently driven tracks with two solid axles articulating the front and rear track pairs. The vehicle measures 1.32 m long in open configuration and 0.70 m long in closed configuration. It is 0.66 m wide and 0.22 m high. Control algorithms will take advantage of the small size of the vehicle, 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 hybrid ground vehicle 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 reference frame coordinate system used in this paper, sketched in Fig. 4, is 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. Note that the forward velocity $v_x$, used in this document, is along the $x$-axis, which stays horizontal even if the robot pitches. Similarly, the upward velocity $v_z$ is along the gravity axis.

The robot has been outfitted with a Microstrain 3DM-GX1 inertial measurement unit which provides triaxial acceleration, rotational rate and orientation of the vehicle. Also, a Hokuyo URG-04LX scanning laser range finder is mounted at the front of the vehicle to scan vertically and a servomechanism pans the sensor producing a 3D scan of the environment. Finally, six optical encoders, one for each track and one for each axle, provide the track velocities and the axle angular positions.

3. Control algorithms

A box climbing sequence is depicted on Fig. 5 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.

3.1. Perception

The terrain representation used is a terrain elevation map which fuses range data where each terrain scan is acquired at a different vehicle pose. The grid map is defined as 2 m along the $x$-axis, and for 19 vertical laser scan lines per field of view. Fig. 6(a) shows the 3D terrain map obtained for a simulated ascending staircase. The map captures the terrain elevation for each 1-cm grid cell 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. Fig. 6(b) shows one laser scan line of a real staircase without riser. Here laser range scan picks up points behind the stair tread because of open riser. Using the algorithm that retains only the highest elevation...
value per grid cell, the resulting terrain map is shown in Fig. 6(c). This is a good example of how difficult it is to recognize obstacles with laser range data. Without a careful treatment, it would be difficult to recognize the actual terrain.

3.2. 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. This subsection presents a geometric-based reactive controller providing an estimation of a desirable actuation.

The proposed reactive controller orients the bottom of the front tracks with the highest elevation in the next near-range distance $d$ in front of the vehicle as sketched in Fig. 7. A near-range distance of 1.5 times the distance between the track wheels in front of the laser provides a good performance. $\delta$ is the angle between the horizontal and a segment from the big wheel origin to a terrain elevation. $P_0$ is the terrain elevation providing the maximum $\delta$ over $d$. The back track follows the front track motion by executing the front axle commands, but delayed with respect to the vehicle nominal speed. This results in a snake-like behaviour.

The front track origin $P_0$ is set to the base of the front big wheel as presented in Fig. 8. Given the vehicle orientation and axle angular positions, $P_0$ is first located 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 $d$ in front of the robot.

In the terrain map $\text{Grid}_{\text{map}}$, each $(x, y)$ grid location of a data point consists of an index $i$ representing the location along the $x$-axis and $j$ representing the laser sensor pan angle. The projection of a data point location on the $x$ and $y$-axis is expressed by:

\[
(x_i, y_j) = \left( c_1(i - 1), c_1(i - 1) \tan \left( \frac{\pi}{36} (j - c_2) \right) \right),
\]

where $\frac{\pi}{36}$ comes from $\frac{\pi}{2}$ field of view divided in 19 pan angles. The coefficient $c_1$ is the linear $x$-axis grid cell resolution in metres, here $c_1$ equals 0.01 as each grid cell covers a 1-cm region along the $x$-axis. $c_2$ is the middle pan angle index value, here $c_2$ equals 10. The magnitude $z$ of the elevation in each $(x, y)$ location can be expressed as $\text{Grid}_{\text{map}}(i, j) = z$. For $\text{Grid}_{\text{map}}(i, j)$, $\delta_y$ is computed as follows:

\[
\delta_y = \tan^{-1} \left( \frac{\text{Grid}_{\text{map}}(i, j) - P_0(3)}{\sqrt{(x_i - P_0(1))^2 + (y_j - P_0(2))^2}} \right) \tag{2}
\]

for all terrain elevations over $d$, and where $P_0(1)$, $P_0(2)$ and $P_0(3)$ are the track origin position coordinate components in the reference frame. The maximum angle is $\delta_{\text{max}}$ and its corresponding terrain elevation is $P_i$. The line formed from the track origin $P_0$ to $P_i$ represents the desired orientation of the bottom of the front track to cross the terrain over $d$. $\delta_{\text{max}}$ is computed for the bottom of the track instead of the track centreline. $\beta$ represents the angle between the track centreline and the bottom of the track. $\beta$ is computed by:

\[
\beta = \tan^{-1} \left( \frac{R - u}{K} \right), \tag{3}
\]

where $K$ is the distance between the front track wheels, and $u$ and $R$ are the small and big wheel radius respectively, as also illustrated in Fig. 8.

The desired front track orientation $\gamma^d$ is evaluated by subtracting $\beta$ from $\delta_{\text{max}}$ as follows:

\[
\gamma^d = \delta_{\text{max}} - \beta. \tag{4}
\]

The current track orientation $\gamma$ is given by subtracting the pitch angle $\theta$ from the current front axle angular position $\phi_F$.

\[
\gamma = \phi_F - \theta. \tag{5}
\]

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

\[
\Delta \phi_F = \gamma^d - \gamma. \tag{6}
\]

Finally, the desired front axle angular position $\phi^d_F$ is given by the current front axle angular position plus the front track orientation error.

\[
\phi^d_F = \phi_F + \Delta \phi_F. \tag{7}
\]

$\phi^d_F$ is limited to $\pm 180^\circ$. The back axle executes the front axle commands, but delayed with respect to the vehicle nominal speed. Finally, to maintain stability and avoid excessive pitching as much as possible, it is checked that the mass centre of a track does not pass above or below the body mass centre.

Testing has been conducted for climbing boxes and stairs. Fig. 9 shows the movie sequence of the STRV crossing a box. The box was 1 m wide, 1 m deep and 0.3 m high.

The reactive behaviour performs well for various obstacle shapes, and produces a predictable and smooth motion. The algorithm is fast to compute and requires no training or tuning. The main advantage of this control algorithm is the necessity of only one algorithm to navigate all types of obstacles (flats, stairs, ramps, steps up, steps down, and so on). However, its capability to overcome an obstacle depends directly on the propulsion speed and the vehicle dynamics. Even with an appropriate geometric configuration, if the robot does not have enough propulsion power, it will not succeed. Furthermore, in some situations the robot gets stuck while climbing an obstacle. The system is assumed stuck when the vehicle forward velocity is null or negative, $v_x \leq 0$, since only forward movements are considered in this work. The vehicle gets stuck because the reactive controller is non-adaptive and does not take into account what is going on below the body, robot dynamics and uncertainties in perception.
A reactive controller is a good first step. However, human-scripted algorithms prove difficult and time consuming to understand and design, and it is difficult to anticipate all possible situations. The production of mobility behaviours benefits from learning control algorithms to adapt to changing conditions.

3.3. Reinforcement learning controller

To overcome a deadlock situation, a reinforcement learning algorithm is proposed to help the controller select suitable STRV geometric configurations in order to exit the deadlock and continue executing the task. It adjusts the two axle angular positions to make it more fit to progress under the terrain conditions. Although the reactive controller works and produces a predictable and smooth motion, it gets stuck in some circumstances. Reinforcement learning will be useful to search for alternative actuation strategies, when the reactive controller does not have an appropriate solution. Reinforcement learning adapts the robot behaviour to changing environments in real-time and learns actuation online. It learns the best action to perform based on the rewards and penalties incurred from performing particular actions.

3.3.1. Controller summary

Fig. 10 presents the reinforcement learning controller architecture developed. When the robot is stuck using the reactive controller, the reinforcement learning controller takes over control and searches for a suitable behaviour to exit the situation. The production of mobility behaviours benefits from learning control algorithms to adapt to changing conditions.

3.3.2. Reactive controller stuck

The creation of a learning system that would manage the robot navigation like the reactive system does would be very challenging, and not useful since the reactive system already does it. Therefore, the efforts are focused on the situations where the reactive system is not successful. The system is assumed stuck when the vehicle forward velocity is null or negative, since only forward movements are considered in this work. Reinforcement learning kicks in when a stuck situation happens.

3.3.3. Artificial neural network

An artificial neural network stores the knowledge acquired by experience. Fig. 11 presents the artificial neural network architecture. The neural network has 207 nodes in the input layer consisting of the terrain map at 0° pan angle (201 data), forward and upward vehicle velocity $v_x$ and $v_z$, pitch angle $\theta$, current front and back axles angular position $\phi_f$ and $\phi_b$, and $P_f$ 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 and may not provide more useful information to resolve the stuck situation. The velocities and pitch angle give knowledge about the vehicle dynamics. The axle angular positions combined with $P_f$ depicts 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 $\phi_f^d$ and $\phi_b^d$. The artificial neural network is trained online using the Levenberg–Marquardt backpropagation algorithm [18], a second-order nonlinear optimization technique. The training set consists of the sensory inputs fed into the neural network and the target output provided by the update function detailed in Section 3.3.5. The artificial neural network is trained to output this target actuation next time it receives the same sensory inputs.
Artificial neural network architecture for the reinforcement learning controller.

3.3.4. Reward function

Several criteria are considered when building the reward function. During a successful navigation, the robot’s forward velocity is important. It slows down during obstacle ascent and the linear upward velocity increases. A low or negative forward velocity usually signifies a difficult progression. Similarly, a null or small linear upward velocity 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 centre displacement. In most circumstances, when the vehicle is stuck on an obstacle, the robot must move its mass centre forward. This increases the chance of exiting the deadlock situation.

Every deadlock 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, traction quality, obstacle dimensions, and vehicle dynamics. How much to rotate the axes, in which sequence and without flipping over, is a very challenging problem. Reinforcement learning may help to solve that problem.

The construction of the reward function is a difficult and very important task since it controls the learning process. The system will not learn the task if the reward function is not designed properly. Through experimentation, it was determined that different stuck situations require different reinforcements to successfully resolve the situations. For this reason, the reward function is split in three components. The classification consists of (1) narrow boxes where $P_0 < 0.15 \text{ m}$ below the body centre and $\theta > -5^\circ$, (2) upward stairs where $P_0 > 0$ and $\theta < 0^\circ$ and (3) steps or any other situations.

Eq. (8) presents the reward function. As recommended in Ref. [9], the developed reward function consists of several rewards and progress estimators to orient and accelerate learning. It consists of six elements.

1. $P_0$ is a progress estimator. It equals 1 when $v_F > 0.01 \text{ m/s}$, otherwise $P_0$ equals $-1$. It gives an important reward when the vehicle gets some forward velocity to reinforce the behaviour.

2. The normalized forward velocity, $\frac{\dot{v}_F}{v_{\text{max}}} \text{,}$ reinforces forward motion. $v_{\text{max}}$ is the maximum or nominal propulsion speed.

3. The normalized upward velocity, $\frac{\dot{v}_x}{v_{\text{max}}} \text{,}$ reinforces upward motion.

4. The normalized pitch angle, $\frac{\dot{\theta}_b}{\theta_{\text{max}}} \text{,}$ reinforces forward pitch to keep the vehicle closer to the obstacle and avoid flipping over.

5. Finally, it is important to consider the mass centre displacement. Reinforcing a forward mass centre relative displacement, $\frac{d_{p} - D_{p_{t-1}}}{\mid d_{p_{t-1}} \mid}$, where $D_{p_t}$ is the mass centre location along the $x$-axis at time $t$ with respect to the robot body centre, usually reduces the risk of flipping over. The mass centre relative displacement is computed by dividing the displacement of two 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.

6. Similarly, reinforcing an upward mass centre displacement, $\frac{D_{z_p} - D_{z_{p_{t-1}}}}{\mid D_{z_{p_{t-1}}} \mid}$, where $D_{z_p}$ is the mass centre location along the $z$-axis at time $t$ with respect to the robot body centre, usually lowers the body near the obstacle and often increases the traction surface. In other words, if the distance between the mass centre location and the robot body centre along the gravity axis is greater at time $t$ than at time $t - 1$, there is an upward mass centre displacement and in many situations this results in a body position nearer the obstacle and should facilitate vehicle propulsion. This should facilitate vehicle propulsion. Only the sign of the mass centre displacement is considered, $\text{sgn}(D_{z_p} - D_{z_{p_{t-1}}})$.

Each reward coefficient $a$, $b$ or $c$ expresses the strength of the $i$th component on the total reward. The coefficients were tuned manually to reinforce some behaviours in specific situations. The numeric value for each reward coefficient is presented in Table 1.

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.

3.3.5. Update function

\[ \dot{\phi}_F \leftarrow \phi_F + \alpha S_F \]

\[ \dot{\phi}_B \leftarrow \phi_B + \alpha S_B. \]

Eq. (9) presents the update function used. $\phi_F$ and $\phi_B$ represent the current front and back axle angular positions. $\dot{\phi}_F$ and $\dot{\phi}_B$ are the desired axle angular position estimations used to train the artificial neural network. The learning rate $\alpha$ is set to 1. It represents how much of the reward $r$ 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 axle must take to realize the actuation. Suppose the front legs are at $5^\circ$ and the neural network simulates $10^\circ$, 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 $S_B$.

3.3.6. Exploration function

The exploration is very important in reinforcement learning to acquire knowledge over the full action space and therefore make better action selection [19]. In this algorithm, the exploration function explores new avenues when the reinforcement learning function acts poorly. That is, when $r_1 - r_{t-1} \leq \text{threshold}$. Here, the threshold was tuned manually to 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

<table>
<thead>
<tr>
<th>Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reward function coefficient numeric values.</strong> The row indicates the coefficient letter and the column the coefficient subscript.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>$a$</td>
</tr>
<tr>
<td>$b$</td>
</tr>
<tr>
<td>$c$</td>
</tr>
</tbody>
</table>
that the robot is stuck longer before deciding to look for new avenues. Usually a random function is used to explore the action space. However, it would be dangerous to operate randomly the actuators while climbing obstacles. The robot could fall from the obstacle and be damaged. For that reason, the exploration function proposed here favours a smooth progression of the actuators by exploring at proximity of the current actuation. The exploration function switches the sign of the update function by inverting the sign of $S_F$ or $S_B$ in Eq. (9) 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.

### 3.3.7. Training

The system was trained to climb boxes and stairs. The training included boxes from 5 cm to 1 m deep, by 1-cm increments. For each depth, every height was tried, starting at 5 cm, by 1-cm increments. The system was trained for each box depth and height until successful (5 times or less); otherwise, the training skipped to the next box inclination. At the end, the system was trained a second time for the same situations.

The reinforcement learning controller has the advantage of being a real-time learning process based on experience. It adapts and optimizes the robot behaviour continuously. The behaviour improves every time the robotic platform deals with similar situations. Even if the situation is new, the controller will figure out a solution, but it may take some time.

### 4. Simulations

This section presents tests using a simulator and compares the performance of the reactive and the reinforcement learning controllers. Two tests were designed to evaluate the performance and limitations of the controllers. They consist of box crossing and staircase climbing.

#### 4.1. Simulator

The simulator Vortex is provided by CMLabs Simulations Inc. [20], a physics based engine for real-time simulation. The Vortex simulator models 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 logistics of control algorithms.

#### 4.2. 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. Fig. 12 illustrates the box parameters and Fig. 13 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. The box is located at 2 m from the robot initial position and the robot nominal propulsion speed is 2 km/h. For each box depth corresponding to a data point 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. Note that the vehicle dynamics influence the results.

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 (Fig. 14(a)). It is a very difficult situation to resolve 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 (Fig. 14(b)). Furthermore, the body inertia helps the robot to flip over the box. When the box is too deep, the robot cannot 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 over. Finally, the robot may get stuck on its belly when descending a box (Fig. 14(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.

The reinforcement learning controller outperforms the reactive behaviour. 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 cannot 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 (vehicle length in closed configuration), the reinforcement learning controller crosses boxes up to 45 cm high (twice the robot’s height), compared to the reactive controller navigating boxes up to 31 cm high. Further examination of detained controller command indicates that the reinforcement learning controller is able to search for different track orientations when the vehicle gets stuck. It uses different angular positions of the tracks that can result in a resultant force on the vehicle enough to move the vehicle forward. 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. 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.

4.2.1. Test 2: staircase crossing

The second test consists of regular staircases with different step sizes. The test determines the steepest stair inclination each controller climbs and descends. Fig. 15 illustrates the staircase parameters. The staircase is located at 2 m from the robot initial position. It consists of 6 steps up, a flat surface of 1 m, then 6 steps down. The nominal propulsion speed is 2 km/h. The test fails if the vehicle flips over or gets stuck in the stairs. Fig. 16(a) presents the maximum staircase inclination the robot can traverse for various tread depths. Fig. 16(b) presents the results differently, showing the maximum riser height traversed for different tread depths. For each tread depth, several trials are run for different increasing riser heights until the system fails. Note that the vehicle dynamics influence the results.

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 (Fig. 17(a)). That distance increases with the tread depth and
the staircase inclination. For this reason, as plotted on Fig. 16(a), for deeper treads, the staircase maximum inclinations traversed are lower. Fig. 16(b) 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, which is 178–203 mm riser height and 254–304 mm tread depth, giving a 30°–38° inclination. However, at the limits, the vehicle flips 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 (Fig. 17(b)) or the first descending step (Fig. 17(c)), where 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.

The reinforcement learning controller outperforms the reactive behaviour. 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 to 34 cm, the reinforcement learning controller can climb staircases with inclinations up to about 45°. And as shown on Fig. 16(b), it can traverse staircases with risers up to about 30 cm high for tread depth deeper than 30 cm (depth corresponding to the axle span). 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.

5. Conclusion

This paper proposed autonomous mobility controllers to perform obstacle crossing for a tracked robotic vehicle. Two 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 difficult to script for every possible circumstance.

To facilitate the controller design process, and adapt in real-time to unforeseen conditions, machine learning offers interesting solutions. 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.

References