Overcoming Obstacles With a Reconfigurable Robot Using Reinforcement Learning

LIRAN YEHEZKEL\textsuperscript{1}, SIGAL BERMAN\textsuperscript{2}, (Senior Member, IEEE), AND DAVID ZARROUK\textsuperscript{1}

\textsuperscript{1}Department of Mechanical Engineering, Ben Gurion University of the Negev, Beersheba 8410501, Israel
\textsuperscript{2}Department of Industrial Engineering and Management, Ben Gurion University of the Negev, Beersheba 8410501, Israel
Corresponding author: David Zarrouk (zdavid@bgu.ac.il)

ABSTRACT The reconfigurable robot RSTAR (Rising Sprawl Tuned Autonomous Robot) is a newly developed crawling robot that can reconfigure its shape and shift the location of its center of mass. These features endow the RSTAR with inherent robustness and enhanced ability to overcome obstacles and crawl on a variety of terrains for a vast range of applications. However, defining the trajectories to fully exploit the robot’s capabilities is challenging, especially when complex maneuvers are required. Here, we show how reinforcement learning can be used to determine the optimal strategies to overcome three typical obstacles: squeezing through two adjacent obstacles, ducking underneath an obstacle and climbing over an obstacle. We detail the implementation of the Q learning algorithm in a simulation environment with a physical engine (UNITY\textsuperscript{TM}) to learn a feasible path in a minimum number of steps. The results show that the algorithm successfully charted a feasible trajectory in all cases. Comparing the trajectory found by the algorithm to trajectories devised by 12 human experts with discrete or continuous control showed that the algorithm trajectories were shorter than the expert trajectories. Experiments showing how the physical RSTAR robot can overcome different obstacles using the trajectories found in the simulation by the Q Learning algorithm are described and presented in the attached video.

INDEX TERMS Reinforcement learning, Q learning, crawling robot, reconfigurable robot, sprawl tuning.

I. INTRODUCTION

Miniature search and rescue ground robots that can be used in disaster areas have numerous advantages. Palm-sized search and rescue ground robots such as the MiniWhegs \cite{1}, Dyna-RoACH \cite{2}, DASH \cite{3}, iSprawl \cite{4}, OctoRoACH \cite{5}, RHex \cite{6}, and Sprawlita \cite{7}, STAR \cite{8}, 1STAR \cite{9}, TAYLRoACH \cite{10} and several reconfigurable swarm robotic systems \cite{11}–\cite{13}. They can crawl, run and climb over obstacles. They can be deployed in large numbers to scout vast areas while keeping operators out of harm’s way. Their low weight and small size are crucial to many tasks. However, these features complicate their maneuverability. Although non-orthodox mechanisms can provide advanced solutions to various maneuverability needs, executing trajectories in real world environments remains a challenge \cite{14}. Here, we describe a way to overcome this problem based on autonomous learning.

In a previous work \cite{15}, we developed the Rising STAR robot (RSTAR –Figure 1) a palm- size search and rescue ground robot. The RSTAR is part of the STAR robot family \cite{8}–\cite{10} which is characterized by its ability to rotate the axis of its wheels. RSTAR is fitted with sprawl (which tilts the wheels’ rotating axes) and four bar extension mechanisms. It can change its height, width, and length and shift its center of mass (COM). This considerably enhances the RSTAR’s abilities to overcome obstacles and traverse a wide range of terrains.

Teleoperating a reconfigurable robot like the RSTAR requires a highly trained operator since different mechanisms often need simultaneous actuation. In many applications, it may be imperative to reduce human intervention by facilitating autonomous robot operations. One approach is to teach the robot to autonomously perform common operations. In the case of off-road missions these include climbing over obstacles, and maneuvering around or underneath obstacles. Machine learning, and specifically reinforcement learning (RL), can be used to teach robots how to perform...
such maneuvers offline by running repetitive experiments in a physical or simulated environment.

Considerable efforts have been devoted over the years to devising methods for the autonomous learning of robotic motion trajectories. One of the most popular is reinforcement learning (RL) [17], [18]. However, most researchers have concentrated on devising ways for the robot to learn how to avoid obstacles [21]–[25]. For example, Lee et al. applied RL to navigate between obstacles with a quadruped robot. Similarly, Zhang et al. [22] use deep RL to address navigation issues in cluttered environments on previously unknown rough terrains.

Overcoming an obstacle (e.g., climbing over or ducking under) rather than avoiding one can lead to more efficient operations, and in some cases be critical for mission success. Successful operation typically requires a combination of kinematic and dynamic maneuvers. The motion plan must provide a feasible path as a function of the robot’s dynamic capabilities. The autonomous learning of such trajectories requires taking complex dynamic considerations into account. It must also be done in setups that preserve the mechanical and electronic hardware of the robot during the learning process. To date, few projects have successfully met this challenge. Notable exceptions include Totani et al. [26] who used RL to learn a step climbing method for a crawler-type rescue robot. Dutta et al. [27] used multi-agent learning for locomotion learning in modular self-reconfigurable robots (MSRs).

One of the main drawbacks of RL is that learning times can be prohibitively long when the problem dimensionality is high. In robotic systems, problem dimensionality is a function of the number of controlled dimensions (e.g., actuators). Given the need for multiple learning sessions and exposure of the robotic system to potential hazards during learning epochs, many researchers have opted to use simulation platforms for the learning process.

A major challenge in using simulation environments for learning policies (that must be implemented in the physical world) is to reliably ensure the transferability of the learning from the simulated to the physical environment. Much research effort has been exerted for improving the transferability of the simulation to the physical world (sim2real). Researchers have suggested various methods, that largely fall into three categories: bringing the simulation closer to the physical reality e.g., [28], [29], basing training data on data from the physical world e.g., [30] and improving the efficiency of integrating experiments in simulation and in the physical world, e.g., [31], [32]. However, robots notoriously learn policies that take advantages of simulation intricacies and transferability from sim2real remains an open research question [28].

In many systems that apply RL their multiple degrees of mobility interact tightly with the environment, e.g., multirotor drones [34], where complex aerodynamic effects are produced by the interactions between the multi-rotor airflow and the environment. Other systems require intricate parameter tuning, e.g., underactuated legged robots [35], [36] performing highly dynamic movements involving balance. In such cases the feasible solutions require extremely precise determinations. This necessitates searching for solutions in continuous action spaces which constrains the transferability of simulation results. Considerable effort has been devoted in recent years to developing efficient RL algorithms suitable for continuous action environments that are not sensitive to hyperparameters, only need a manageable amount of learning sequences, and whose results are transferable from simulated to physical environments [34]–[37].

Overcoming obstacles (rather than circumventing them) is rarely discussed in literature. Partly, since highly advanced and complex mechanics are required. Robots that have such capabilities, typically require intricate coordination of their degrees of freedom for performing motion. Engaging an obstacle (and especially climbing over an obstacle that is at the mechanical limits of the robot) with no preliminary solution at hand is extremely challenging. It’s a problem with a binary outcome (success/fail) which requires performing multiple mechanical maneuvers that the algorithm must figure out.

The unique design of the RSTAR robot gives it very high maneuverability with a relatively small number of actuators [4]. This makes its solution search space (more) manageable. The inherent stability of the RSTAR and its decoupled degrees of freedom (the mechanisms and wheels can be actuated practically independently) allow for control policies with relatively large tolerances. This makes it possible to use the much simpler discrete RL methods. It also increases the robustness of the solution to small discrepancies in the modeled dynamics, thus enhancing the transferability of the solutions to the physical world. Nevertheless, because motion dynamics are key to mission success, a physical simulation engine is required for meaningful learning. Beyond simplifying the learning process, discretizing the action space simplifies run-time control, since policies can be represented as discrete sequences of actions.
In this article, we show how to apply RL to exploit RSTAR’s advanced reconfiguration capabilities to climb over, duck underneath and squeeze through obstacles. We apply the Q-learning algorithm, an off-policy, temporal differencing model-free RL algorithm [19] for autonomous learning. This model-free algorithm was selected because the formulation of the dynamic equations of RSTAR is complex and prone to error which may impede learning. For deterministic Markov decision processes (MDP), as is the case in the currently simulated environment, Q-learning is guaranteed to converge asymptotically to the optimal solution if each state-action pair is visited infinitely-often [33]. Q-learning has good convergence properties also for general MDPs, even though for the general case convergence is guaranteed only under additional constraints on the learning rate [Sutton & Barto, 2018], and is generalizable across different environments. System operation is also compared to operation by human operators under a similar setup where the artificial learning mechanism does not have more information than the human operator, such that the results hold in the general case.

II. KINEMATICS OF THE RSTAR

The RSTAR is composed of a main rigid body and a pair of arms holding two sets of wheels and whegs (wheeled-legs) [1]. The main body houses a micro controller, the drivers and the batteries. The RSTAR is fitted with a sprawl mechanism and a four-bar extension mechanism which increase the height and width of the robot and rotate the rotation axis of the wheels. Each of these mechanisms is actuated by a single motor and both sides of the robot are phased together and move symmetrically relative to its center. The wheels on each side are actuated by a single motor. In total, the robot has four motors.

A. SPRAWL AND THE FOUR BAR EXTENSION MECHANISMS

The sprawl mechanism makes it possible to change the relative angle between the legs and the main body in the range of \([-90, 90]\) (FIGURE 2). The sprawl angle is used to move the center of mass (COM) of the robot along the fore-aft direction. This feature allows the robot to pitch upwards by moving its COM backward and accelerating or avoid falling on its back by moving the COM to the front when climbing over obstacles. The sprawl also allows to reduce the height of the robot for it to crawl underneath obstacles. The four-bar extension mechanism (FBEM) lengthens or shortens the distance between the body and the legs by changing the FBEM angle in a range of \([-72, 72]\). The FBEM is attached to the sprawl mechanism and rotates together with it. Thus, the combination of the sprawling capabilities and the FBEM allow the RSTAR to raise and lower its center of mass (COM) and move its COM in the fore-aft direction. It also allows RSTAR to increase or decrease its height and width and operate in a variety of planar, upright, or inverted configurations.

B. DRIVING MECHANISM

The RSTAR has a differential driving mechanism, where each side is actuated by a single motor. To improve stability and reduce energy consumption, RSTAR can be fitted
with wheels or spoked legs or a combination of the two, which give it greater ability to engage in different terrains. The round wheels are primarily adapted to smooth surfaces whereas the spoked wheels are more efficient in unstructured environments. The RSTAR can flip its body upside down and drive inverted using dynamic maneuvers to change the type of wheels contacting the ground [15].

C. CONTROL SYSTEM
The RSTAR can be controlled by either a human operator (direct mode) or a micro controller (program mode). In the program mode, the robot can be programmed to perform a consecutive set of actions. The configuration mechanisms (sprawl and FBEM) are controlled in a closed loop with angular potentiometers whereas the driving motors of the wheels are controlled in a closed loop with magnetic encoders. The main components of the control system are illustrated in FIGURE 3.

III. LEARNING PROCESS
This section details the simulated virtual environment used for the learning process, the different obstacles the robot learned to overcome, the discretization of the state space, and the learning algorithm.

A. SIMULATION ENVIRONMENT AND OBSTACLE DEFINITION
The simulations were conducted using a Unity® software environment (real-time engine development platform) [39] and included a (1:1 ratio) model of the RSTAR and different obstacles. We defined the RSTAR’s kinematics (see Section II) and tuned its speed and contact properties with the surface to mimic those of the physical robot. Three common obstacle cases were learned in three separate simulations: (1) A narrow (180 mm wide) channel in which the robot has to reduce its width to pass through (FIGURE 4 - a). (2) A low entry (55 mm high) where the robot has to lower its body to crawl underneath (FIGURE 4- b). (3) A 50 mm step obstacle the robot needs to climb over by shifting its center of mass (FIGURE 4 - c). The geometric properties were modifiable so that the simulation results could be tested on obstacles of different sizes. The simulations were performed on an Intel®Core™i7-3632QM processor 2.2GHz, 8GB RAM memory running a Windows 10 operating system x64 bits.

B. SIMULATION ENVIRONMENT AND OBSTACLE DEFINITION
The Q-learning algorithm was applied using the RSTAR Unity® model. FIGURE 5 illustrates the iterative procedure of the learning process: the RSTAR observes its state, \( s \), performs the action, \( a \), and receives a reward, \( r \). Its next state, \( s' \), is produced by the environment, and finally the \( Q \) value of the related action-value pair is updated.

Each learning iteration ends when one of the following conditions is met:
1) The RSTAR successfully reached its target.
2) The number of actions performed exceeded the maximal number of actions allotted.
3) The RSTAR reached a “dead end” by moving sideways past obstacle borders.
4) The RSTAR flipped over.
All the intermediate learning data, including states, actions, rewards, and Q matrix values, were retained for convergence analysis and validation.

The Algorithm is composed of 4 main partitions:
1. Initialize Q matrix.
2. Set the agent in its initial position in the physical engine and initialize \( \varepsilon \).
3. While:
   (Agent has not reached the target) && (performed less actions than the number of allotted actions) && (not flipped over):
   3.1 Extract the current state, \( s_n \), of the agent from the physical engine.
   3.2 Draw random number
   3.3 If (random number < \( \varepsilon \))
       Choose random action \( a_n \).
       Else
       Choose action \( a_n \) by Q value.
   3.4 Perform action \( a_n a_n \) in the physical engine.
   3.5 Extract the subsequent state, \( s_{n+1} \) from the physical engine.
   3.6 Update (decrease) \( \varepsilon \)
   3.7 Calculate Reward, \( r_n \)
   3.8 Update the appropriate Q value.
4. If: (Iteration number < the maximum allotted iterations)
   Goto 2
Else
   End

C. STATES
The state of the agent (the RSTAR) was based on its position, orientation, and configuration. To reduce the dimensionality of the state space, the position of the robot was limited to the advancing direction (z) alone and roll was neglected. Altogether, the agent’s state was composed of five parameters: a) the position in the advancing direction, b) the yaw, c) the pitch, d) the FBEM angle, and e) the sprawl angle, as illustrated in FIGURE 6.

D. ACTIONS
The action space consisted of five actions (FIGURE 6): 1) rotate the wheels forward by one third revolution (60.7 mm), 2-3) decrease the FBEM angle (2) or increase it (3), 4-5) increase (4) or reduce (5) the sprawl angle. Note
that the robot might be unable to perform a specific action if prevented to do so by an obstacle. For example, in Action type 1, the wheels simply slid if the robot could not advance forward.

**E. Q MATRIX INITIATION**

Each action-state pair was rated based on the expected value when taking that action in that state. The Q matrix was 5 dimensional for the use cases of squeezing through a channel and ducking underneath an obstacle and 6 dimensional for the climbing over the obstacle use case. The number of Q matrix cells was on the order of 10^5 (TABLE 1).

Unlike more common projects, in our work the robot is expected to overcome the obstacle, rather than circumvent it. Interacting with an obstacle is inherently dynamically different from driving around it. The goal state is not a point or a configuration, as is typical in may RL studies, rather it is a zone. With a goal biased Q value initiation [40] and rewards, we direct the robot towards the obstacle but we do not dictate how to overcome it. Therefore, to encourage the robot to advance towards the obstacle and overcome it, the initial Q matrix values were based on the advancing direction and the yaw angle. The values increased as a function of the advancing direction and decreased with the absolute value of the yaw. In all use cases, the workspace was divided into four zones (i-iv) along the advancing direction (z). FIGURE 7-a presents the specific zone borders of the climbing over an obstacle use case. The other use cases had a similar pattern but different border values. Zone (i): located behind the agent’s initial position (z ≤ 3) which was characterized by negative equal values for all yaw angles:

$$Q_i(s_n, a_n) = -100 (Z_{tot} - z)$$  \hspace{1cm} (1)

where $Z_{tot}$ was the total number of partitions in the z (advancing) direction and $z \leq 3$. This zone could be reached if the RSTAR moved its COM backwards using the FBEM.

Zone (ii): located between the agent’s initial position and the proximity line (4 bins before the success line 3 ≤ z < 15). In this zone, the $Q_i$ started with positive values that increased monotonously as the robot advanced towards the success zone. Lower values were assigned to greater yaw angles between 5° to 10° and even lower to values beyond 10° to encourage the robot to maintain a straight orientation.

$$Q_i(s_n, a_n) = \begin{cases} 
100z & \theta_{yaw} \leq 5 \\
50z & 5 < |\theta_{yaw}| < 10 \\
z & 10 \leq |\theta_{yaw}| 
\end{cases}$$  \hspace{1cm} (2)

Zone (iii): in the vicinity of the success zone, with a length of 4 partitions (in the advancing direction 15 ≤ z < 19) and Q values that were substantially higher than zone (ii). In this zone, the Q values were independent of the yaw:

$$Q_i(s_n, a_n) = 1000z$$  \hspace{1cm} (3)

Zone (iv): the success zone, characterized by the highest initial Q values (identical to Eq. (3)) to ensure that the robot was attracted to the success zone and remained in it.

**F. REWARD AND UPDATE**

As in the Q matrix initialization, the workspace was divided into the same four zones (FIGURE 7-b). The rewards were negative in the first three zones to minimize the number of steps and ensure convergence to an optimal solution:

**TABLE 1. Number of states and actions in each simulation.**

| Partition number and size | Through a channel | Low opening | High obstacle |
|---------------------------|-------------------|-------------|---------------|
| Position (z direction)    | 43                | 34          | 24            |
| Size:                     | 18 mm             | 18 mm       | 18 mm         |
| Yaw Angle                 | 3                 | 3           | 3             |
| Range:                    | 0° ≤ ω < 5°, 5° ≤ ω < 10°, and 10° ≤ ω          |
| Pitch Angle               | -                 | -           | 2             |
| Range:                    | 0° ≤ β < 5°, 5° ≤ β < 180°                      |
| FBEM                      | 12                | 12          | 12            |
| Size:                     | 12°               | 12°         | 12°           |
| Sprawl                    | 12                | 20          | 12            |
| Size:                     | 15°               | 9°          | 15°           |
| Possible Actions          | 5                 | 5           | 5             |
| Q matrix cells            | 92,880            | 122,400     | 103,680       |

FIGURE 7. Reward (a) and initial Q value (b) as a function of the advancing direction and yaw angle.
Zone (i): a highly negative rewards zone.
\[ r(s_n, a_n) = -30(Z_{tot} - z) . \] (4)

Zone (ii): the reward in this zone was also negative but increased monotonously to “encourage” the agent to leave the zone and advance forward. The reward was more negative if the yaw angle was larger than 5° to ensure that the agent did not attempt to rotate sideways.
\[ r(s_n, a_n) = \begin{cases} 
-8 (Z_{tot} - z) & |\theta_{yaw}| \leq 5 \\
-12 (Z_{tot} - z) & 5 < |\theta_{yaw}| < 10 \\
-16 (Z_{tot} - z) & 10 \leq |\theta_{yaw}| . 
\end{cases} \] (5)

Zone (iii): the reward function in this zone was similar to zone (ii), and was designed to ensure it continued attempting to advance forward.

Zone (iv): the success zone was highly positive to ensure that the agent advanced towards it and finished its climbing learning.
\[ r(s_n, a_n) = 30000. \] (6)

The initial values of the matrix \( Q_i \) and rewards are summarized in Appendix B.

The \( Q \) matrix was updated using:
\[ Q(s_n, a_n) \leftarrow (1-\alpha)Q(s_n, a_n) + \alpha [r_n + \gamma \max_{a'} Q(s_{n+1}, a')] . \] (7)

where \( r_n \) was the reward for taking action \( a_n \) at state \( s_n \), \( s_{n+1} \) was the state of the agent after taking action \( a_n \). Action \( a' \) was the one associated with the highest possible \( Q(s_{n+1}, a') \) value. The learning rate \( \alpha \) was 0.6 whereas the discount factor \( \gamma \) was 0.9.

G. POLICY

Learning was conducted as an \( \epsilon \)-greedy policy where the probability of taking a random action, \( \epsilon \), decreased by:
\[ \epsilon \leftarrow \epsilon - \frac{\epsilon_i - \epsilon_{\min}}{N_{\max}} . \] (8)

where \( \epsilon_i \) is the initial probability, \( \epsilon_{\min} \) is its minimum value and \( N_{\max} \) is the total number of times random actions are taken. In all simulations \( \epsilon_i = 1 \), \( \epsilon_{\min} = 0 \) and \( N_{\max} = 30,000 \).

IV. SIMULATION RESULTS

This section presents the convergence rate of the algorithm followed by an analysis of the solution and its validity over a range of parameters. The last section compares the simulation to human operators.

A. CONVERGENCE RATE

The convergence rate of each simulation was tested using three parameters: the sum of reward over the path at each iteration, the number of actions per iteration, and the success rate of the iterations. The results of all three use cases were qualitatively similar. FIGURE 8 presents the results of squeezing between the walls of a channel.

The algorithm started to find feasible solutions after 100 iterations (60% success) (FIGURE 8 top). Subsequently, the number of actions in the found solution decreased and after 600 iterations, the success rate was nearly 100%. The quality of the solution (the number of actions) improved with the number of iterations and reached an average of 20 actions per solution. The best solution (the smallest number of actions) involved 17 actions. Section IV.E discusses the fact that some of the solutions with a larger number of actions had advantages when overcoming challenging obstacles not previously learned. The iterations started with a highly negative sum of rewards (FIGURE 8 bottom). The number of iterations increased, along with the sum of the rewards (the absolute number decreased) as the robot successfully found feasible solutions after nearly 200 iterations and then continued to slowly improve.

B. SQUEEZING THROUGH A CHANNEL

The simulation converged to a successful solution (with 17 actions) after about 600 iterations (24 hours running time). The simulation presented in FIGURE 9 shows that starting in (a) for a channel width of 240 mm, the simulated robot performed four actions to lower its sprawl from 48 to 18 degrees and decrease its width from 240 mm to 163 mm. When the width of the robot was smaller than that of the channel (180 mm), the simulated robot continued advancing to reach its target. (see video)

C. DUCKING UNDERNEATH AN OBSTACLE

In this use case the simulation converged after 700 iterations to a solution with 14 actions (24 hours running time). Starting from an initial configuration in which the simulated RSTAR’s
FIGURE 9. The simulated RSTAR crawling between two walls. Starting at (a), the robot increased the sprawl to reduce its width (b-c) and continued advancing toward its target (see video).

FIGURE 10. Starting from its initial configuration (a), the simulated RSTAR moved its FBEM forward (b), then lowered its sprawl (c) and continued advancing forward (see video).

height was 100 mm (compared to the 55 mm clearance of the obstacle), the simulated RSTAR first moved its FBEM forward (b) and then lowered its sprawl until reaching a height of 51 mm (c), then continued to advance (d) to reach its target.

D. CLIMBING ON TOP OF AN OBSTACLE

The RSTAR was trained to climb from the ground as in FIGURE 11 (a) to the top of a step as in FIGURE 11 (f). Given the fact that the obstacle was substantially higher than the radius of the wheels, this use case was complex (even for experienced human operators). It required more complex dynamic maneuvers relative to the previous two use cases and its solution required more actions to perform. In order to reduce the learning time and speedup solution convergence, the learning process was divided into two epochs. In the first epoch, the agent was initially placed with its wheels on the tip of the step (FIGURE 11 (d)) and learned to climb as illustrated in FIGURE 11 (d)-(f). After the convergence of the first epoch, the Q values were saved and were set to be the initial values of the second learning epoch (the random action factor $\epsilon$ was reset to 0.6).

In the second learning epoch, the agent learned how to start climbing from the ground (a) until it placed its wheels at the tip of the step (c). Together with the first learning epoch, the robot successfully learned how to fully climb over the step starting from the ground. In total, the algorithm required 26 hours to converge to the solution.

The algorithm produced multiple solutions to climb over the obstacle, with a range of 24 to 27 actions. The difference between the different solutions is analyzed in Section (IV.E).

FIGURE 11 illustrates how the simulated RSTAR climbed over a step obstacle. The step was 50 mm high compared to the 58 mm diameter of the RSTAR’s wheels. Starting in (a), the simulated robot moved its COM backward using its FBEM (b) and advanced forward to pitch upwards and place its front wheels on the edge of the obstacle (c). Note that dynamically, moving the COM was essential to allow it to pitch upwards. After placing its front wheels and leaning on the step, it sprawled down to lower it COM (d) and then advanced its COM forward using the FBEM (e). Both actions were also critical to climbing, since otherwise the robot would pitch on its back when it attempted to advance forward. The robot continued advancing using its wheels to finish climbing over the obstacle (f). (See video)

E. SOLUTION SUITABILITY FOR UNTRAINED OBSTACLE SIZES

To examine the level of compatibility of the solutions to obstacle size variation, the learned solutions were tested on obstacles of different sizes (TABLE 2). To squeeze through a channel, the solution learned for the 180 mm wide channel emerged as suitable for narrower channels up to 164 mm wide. For ducking underneath an obstacle, the solution for
TABLE 2. Versatility of the simulation results.

| Use Case                        | Actions | Solution Compatibility |
|--------------------------------|---------|------------------------|
| Squeezing through a 180 mm channel. | 17      | 164 mm and wider       |
| Ducking underneath a 55 mm opening. | 14      | 51 mm and higher       |
| Climbing over a 50 mm obstacle.   | 24      | Up to 50 mm            |
|                                 | 26      | Up to 51 mm            |
|                                 | 27      | Up to 55 mm            |

A clearance of 55 mm was appropriate for a 51 mm clearance. To climb over an obstacle, the solution with the smallest number of actions (24) was only suitable for climbing over the obstacle it was trained on (50 mm). However, solutions with more actions were more versatile. Two solutions with 26 and 27 actions allowed the robot to climb on top of obstacles that were 51 and even 55 mm high.

F. OUTPERFORMING HUMAN EXPERTS

To estimate the quality of the learning process, we compared the algorithm’s results to human solutions. This comparison was conducted for the most challenging case of climbing over a step. Human solutions were proposed by two groups of six M.Sc. and fourth year undergraduate students from the Bio-Inspired and Medical Robotics Lab. who were familiar with the physical RSTAR robot and its kinematics. Both groups were asked to solve the climbing use case using the simulation. Each student was given an explanation of the action possibilities and then allotted one full hour to find a solution. The students worked separately and were not allowed to collaborate. The first group was asked to find a solution by listing a sequence of actions to perform. The students were told how many actions the simulation needed (24) but not the action set.

Interestingly, and surprisingly at least to the authors of this manuscript, the Q learning algorithm outperformed all the “human experts”. None of the students was able to find a solution with a smaller number of actions. Two students were not able to find a climbing solution at all, and the four others found solutions ranging from 25 to 28 actions compared to the Q learning’s solution of 24 actions.

The students in the second group were asked to control the simulated RSTAR using a joystick (continuous operation) and their performance was recorded, using the Unity software, in terms of motor actuation. They were asked to stop when they believed they had found a good solution but were not given any feedback on the quality of their solution during the trial. All the students of this group succeeded in climbing over the step using the joystick but their solutions demanded more time (+47%), and more actuation of the FBEM (+220%) and driving motors (+188%). However, on average, they used less sprawling (-40%). A summary of their solutions is presented in TABLE 3. For this small number of samples (6), we compared the results of the algorithm versus the students for three different parameters, the time required for the operation, the sum of sprawl and FBEM actuation, and the total amount of driving forward. In all three measures, the algorithm outperformed the students with a 95% confidence probability.

Given the robust and parsimonious design of the RSTAR, a user can quickly learn a feasible control policy. However, optimizing the steps appears to be very challenging since the user must make two concurrent control decisions with different objectives (driving the robot or changing the robot’s configuration using the sprawl or the FBEM).

V. HARDWARE IMPLEMENTATION AND VALIDATION

This section presents the implementation of algorithm’s results on an actual RSTAR prototype. In each experiment, the initial conditions of the RSTAR (position and configuration) and the obstacle’s geometric properties were (nearly) identical to the learned dimensions.

Although the robot can calculate its own configuration using its encoders and potentiometers, it does not have a camera or range sensor (for example) to detect its location or orientation. In each case, a sequence of actions was uploaded to the RSTAR’s control system. The uploaded sequence was the set learned using the Q learning algorithm. To squeeze through a channel the robot reduced its width and advanced between the two walls (FIGURE 12). The robot

FIGURE 12. The RSTAR robot (a) side view, (b) isometric view, (c) front view, and (d) view of the robot when it is flat.
also successfully lowered its body and ducked underneath the obstacle (FIGURE 13). In both cases, the robot successfully performed its task in the physical environment as of the first attempt.

The robot was able to climb over obstacles measuring 45 mm in height but not obstacles 50 mm high using its minimal 24 action sequence. This 10% difference between the simulation and the actual robot can be attributed to minor differences in the definition of mechanical and geometric properties and the accuracy of the solution of the physical engine.

Interestingly, the robot successfully climbed over the 50 mm high obstacle using the 27 actions solution (which according to the simulation would allow it to climb over 55 cm steps). This result is consistent with the Q learning solution which suggested that the 27 actions solution would increase its climbing capability by 5 mm. The results of the three experiments and their comparison to the simulation are summarized in table 4. In all the experiments, the physical robot repeatedly performed its task successfully. There were no unsuccessful attempts.

VI. CONCLUSION

In this article, we applied a Q learning algorithm to a reconfigurable robot learning autonomously to overcome three typical obstacles use cases. These included squeezing through a channel, ducking underneath an obstacle, and climbing over an obstacle (which requires a very skilled operator). To simplify the process, safeguard the robot, and reduce the learning time, all the learning took place in a simulated environment using Unity software. The algorithm found solutions to all the use cases. Given that our problem is overcoming an obstacle rather than circumventing it, we directed the robot using goal-biased Q matrix initiation and rewards to advance towards the obstacle and we rewarded the robot for successfully overcoming the obstacle but did not dictate how to overcome the obstacle. We used a uniform discretization along all the DOF which was sufficient to find solutions for each problem.
TABLE 5. Summary of the initialization of the Q matrix and the reward values.

| Zone 1 | Zone 2 | Zone 3 | Zone 4 |
|--------|--------|--------|--------|
| **Between Walls** | | | |
| Initial $Q(s_n, a_n) = -100(Z_{tot} - z)$ | $100z$ | $100z$ | $1000z$ |
| | $5z$ | $5z$ | $1000z$ |
| | $z$ | $z$ | $1000z$ |
| **Reward** | | | |
| $r(s_n, a_n) = -30(Z_{tot} - z)$ | $-8(Z_{tot} - z)$ | $-8(Z_{tot} - z)$ | $-30000$ |
| | $-12(Z_{tot} - z)$ | $-12(Z_{tot} - z)$ | |
| | $-16(Z_{tot} - z)$ | $-16(Z_{tot} - z)$ | |
| **Under obstacle** | | | |
| Initial $Q(s_n, a_n) = -100(Z_{tot} - z)$ | $100z$ | $100z$ | $1000z$ |
| | $5z$ | $5z$ | $1000z$ |
| | $z$ | $z$ | $1000z$ |
| **Reward** | | | |
| $r(s_n, a_n) = -30(Z_{tot} - z)$ | $-8(Z_{tot} - z)$ | $-8(Z_{tot} - z)$ | $-30000$ |
| | $-12(Z_{tot} - z)$ | $-12(Z_{tot} - z)$ | |
| | $-16(Z_{tot} - z)$ | $-16(Z_{tot} - z)$ | |
| **Climbing** | | | |
| Initial $Q(s_n, a_n) = -100(Z_{tot} - z)$ | $100z$ | $100z$ | $1000z$ |
| | $5z$ | $5z$ | $1000z$ |
| | $z$ | $z$ | $1000z$ |
| **Reward** | | | |
| $r(s_n, a_n) = -30(Z_{tot} - z)$ | $-8(Z_{tot} - z)$ | $-8(Z_{tot} - z)$ | $-30000$ |
| | $-12(Z_{tot} - z)$ | $-12(Z_{tot} - z)$ | |
| | $-16(Z_{tot} - z)$ | $-16(Z_{tot} - z)$ | |

A finer discretization may lead to better results. However, a finer discretization will require a longer learning time.

The results were implemented on a physical RSTAR. The robot was placed in an environment reminiscent of the simulated environment and preprogrammed to perform the learned set of actions. With no human intervention, the RSTAR successfully overcame all three obstacles on its first attempt.

These learned solutions could also be implemented on similar obstacles with different geometries. For instance, the solution of squeezing through a channel was developed for a width of 180 mm but was also shown to be suitable for a narrower channel 164 mm wide. The same sequence for climbing over a 50 mm obstacle was also proven compatible for steps with lower heights. To climb over an obstacle, the algorithm provided multiple solutions with different numbers of actions. An analysis of these solutions showed that a larger number of actions were suitable for climbing over higher obstacles. For example, the simulated robot climbed over a 50 mm step with 24 actions, and over a 55 mm step with 27 actions. This result was also validated in the experiments.

The climbing use case was especially challenging. Although the algorithm had no prior knowledge of the kinematics of RSTAR or the obstacles, it generated mechanically intelligent results for climbing on the step. The result included moving the COM backward and then forward and changing its height to overcome the obstacle. This solution outperformed solutions devised by a group of human experts, and used fewer actions.
Based on these encouraging results, we are now working on including perception capabilities in the setup to learn action sequences directly based on perceptual input.

**APPENDIX A**

Figure 16 compares the algorithm to human experts. Human experts needed more time, and activated the driving wheels and the FBEM more often than the algorithm. By contrast, they tended to actuate the sprawl less often than the algorithm. One expert did not actuate the sprawl at all.

**APPENDIX B**

See Table 5.

**REFERENCES**

[1] J. M. Morrey, B. Larribrecht, A. D. Horchler, R. E. Ritzmann, and R. D. Quinn, “Highly mobile and robust small quadruped robots,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Dec. 2003, pp. 82–87.

[2] A. M. Hoover, S. Burden, X.-Y. Fu, S. Shankar Sastry, and R. S. Fearing, “Bio-inspired design and dynamic maneuverability of a minimally actuated six-legged robot,” in *Proc. 3rd IEEE RAS EMBS Int. Conf. Biomed. Robot. Biomechatronics*, Sep. 2010, pp. 869–873.

[3] P. Birkmeyer, K. Peterson, and R. S. Fearing, “DASH: A dynamic 16g hexapedal robot,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2009, pp. 2683–2689.

[4] S. Kim, J. E. Clark, and M. R. Cutkosky, “iSprawl: Design and tuning for high-speed autonomous open-loop running,” *Int. J. Robot. Res.*, vol. 25, no. 9, pp. 903–912, Sep. 2006.

[5] A. O. Pullin, N. J. Kohut, D. Zarrouk, and R. S. Fearing, “Dynamic turning of 13 cm robot comparing tail and differential drive,” in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 5083–5089.

[6] K. C. Galloway, G. C. Haynes, B. D. Ilhan, A. M. R. J. Knopf, G. Lynch, B. Piotnick, M. White, and D. E. Koditschek, “X-RHex: A highly mobile hexapod robot for sensorimotor tasks,” *Dept. Mech. Eng.*, Univ. Pennsylvania, Philadelphia, PA, USA, Tech. Rep., 2010.

[7] J. G. Chalm, S. A. Bailey, J. E. Clark, R. J. Full, and M. R. Cutkosky, “Fast and robust: Hexapod robots via shape deposition manufacturing,” *Int. J. Robot. Res.*, vol. 21, nos. 10–11, pp. 569–882, Oct. 2002.

[8] D. Zarrouk, A. Pullin, N. Kohut, and R. S. Fearing, “*STAR*, a sprawl tuned autonomous robot,” in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 20–25.

[9] D. Zarrouk and R. S. Fearing, “Controlled in-plane locomotion of a hexapod using a single actuator,” *IEEE Trans. Robot.*, vol. 31, no. 1, pp. 157–167, Feb. 2015.

[10] K. Karydis, D. Zarrouk, I. Poulakakis, R. S. Fearing, and H. G. Tanner, “Planning with the *STAR*(s),” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 3033–3038.

[11] H. Ahmadzadeh and E. Masehian, “Modular robotic systems: Methods and algorithms for abstraction, planning, control, and synchronization,” *Artif. Intell.*, vol. 223, pp. 27–64, Jun. 2015.
B. A. Kadian, J. Truong, A. Clegg, E. Wijmans, S. Lee, L. Savva, S. Cernova, and D. Batra, “Sim2Real predictivity: Does evaluation in simulation predict real-world performance?” IEEE Robot. Autom. Lett., vol. 5, no. 4, pp. 6670–6677, Oct. 2020.

J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, “Sim-to-real: Learning agile locomotion for quadruped robots,” in Proc. Robot., Sci. Syst. (RSS), Pittsburgh, PA, USA, Jun. 2018.

A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus, “Learning robust control policies for end-to-end autonomous driving from data-driven simulation,” IEEE Robot. Autom. Lett., vol. 5, no. 2, pp. 1143–1150, Apr. 2020.

A. Marco, F. Berkenkamp, P. Hennig, A. P. Schoellig, A. Krause, S. Schaal, and S. Trimpe, “Virtual vs. Real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2017, pp. 1557–1567.

Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, “Closing the sim-to-real loop: Adapting simulation randomization with real world experience,” in Proc. Int. Conf. Robot. Autom. (ICRA), May 2019, pp. 8973–8979.

N. Shimkin, Reinforcement Learning—Basic Algorithms, Learning in Complex Systems Spring 2011 Lecture Notes. Haifa, Israel: Technion Institute of Technology, Jan. 2020. [Online]. Available: https://webee.technion.ac.il/people/shimkin/LCS11/ch4_RL1.pdf

G. Shi, X. Shi, M. O’Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, “Neural lander: Stable drone landing control using learned dynamics,” in Proc. Int. Conf. Robot. Autom. (ICRA), May 2019, pp. 9784–9790.

A. Nagabandi, G. Yang, T. Asmar, R. Pandya, G. Kahn, S. Levine, and R. S. Fearing, “Learning image-conditioned dynamics models for control of under-actuated legged millirobots,” 2017, arXiv:1711.05253. [Online]. Available: http://arxiv.org/abs/1711.05253

J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, “Sim-to-real: Learning agile locomotion for quadruped robots,” 2018, arXiv:1804.10332. [Online]. Available: http://arxiv.org/abs/1804.10332

T. F. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” 2015, arXiv:1509.02971. [Online]. Available: http://arxiv.org/abs/1509.02971

R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

L. A. Matignon, G. Laurent, and L. N. Fort-Piat, “Reward function and initial values: Better choices for accelerated Goal-directed reinforcement learning,” in Artificial Neural Networks—ICANN 2006 (Lecture Notes in Computer Science), vol. 1, no. 4131. Berlin, Germany: Springer, 2006, pp. 840–849.

LIRAN YEHEZKEL received the B.Sc. degree from the Ben Gurion University of the Negev, Israel, in 2017, where he is currently pursuing the M.Sc. degree with the Department of Mechanical Engineering. His research interest is in imparting autonomous abilities for reconfigurable robots using reinforcement learning.

SIGAL BERMAN (Senior Member, IEEE) received the B.Sc. degree in electrical and computer engineering from the Technion Israel Institute of Technology and the M.Sc. degree in electrical and computer engineering and the Ph.D. degree in industrial engineering from the Ben Gurion University of the Negev. She is currently an Associate Professor with the Department of Industrial Engineering and Management, Ben Gurion University of the Negev, Beersheba. She leads the Intelligent Systems Engineering Laboratory, where her research focuses on analysis and engineering of intelligent systems capable of dexterous motion. She develops deterministic and stochastic models for synthesis of robotic motion and for analysis of human motion.

DAVID ZARROUK received the B.Sc., M.Sc., and Ph.D. degrees from the Technion Israel Institute of Technology, Israel, in 2005, 2008, and 2011, respectively. From 2011 to 2013, he was a Postdoctoral Fellow with the Biomimetic Milisystems Laboratory, University of California at Berkeley. He is currently an Assistant Professor with the Department of Mechanical Engineering, Ben Gurion University of the Negev. His research interests are in medical robotics, robotics in flexible and slippery surfaces interactions, biomimetics, and minimally actuated mechanisms.

Dr. Zarrouk received multiple prizes for excellence in research and teaching, including the Fulbright Postdoctoral Fellowship in 2011, the Fulbright-Ilan Ramon Postdoctoral Fellowship for most prominent Israeli Fulbright Fellow in 2011, the Hershel Rich Innovation Award, the Aharon and Ovadia Barazani prize for excellent Ph.D. thesis, and the Alfred and Yehuda Weisman prize for consistent excellence in teaching.