



Reinforcement Learning based, Staircase Negotiation Learning in Simulation and Transfer to Reality for Articulated Tracked Robots

Andrei Mitriakov, Panagiotis Papadakis, Jérôme Kerdreux, Serge Garlatti

► To cite this version:

Andrei Mitriakov, Panagiotis Papadakis, Jérôme Kerdreux, Serge Garlatti. Reinforcement Learning based, Staircase Negotiation Learning in Simulation and Transfer to Reality for Articulated Tracked Robots. IEEE Robotics and Automation Magazine, 2021, 28 (4), pp.10-20. 10.1109/MRA.2021.3114105 . hal-03365782

HAL Id: hal-03365782

<https://hal.science/hal-03365782>

Submitted on 5 Oct 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Reinforcement Learning based, Staircase Negotiation Learning in Simulation and Transfer to Reality for Articulated Tracked Robots

Andrei Mitriakov, Panagiotis Papadakis, Jérôme Kerdreux, Serge Garlatti
IMT Atlantique, Lab-STICC, UMR 6285, team RAMBO, F-29238 Brest, France

Abstract—Autonomous control of reconfigurable robots is crucial for their deployment in diverse environments. The development of such skills is however hampered by the diversity in hardware and task constraints. We advocate the use of artificial intelligence-based approaches to improve scalability to different conditions and portability to platforms of comparable traversability skills. In particular, we succeed in tackling the problem of staircase traversal via a reinforcement learning-based control framework applicable to different articulated tracked robots, powerful enough to generalize to varying conditions learnt in simulation and to transfer to reality in a zero-shot setting. Our extensive experiments demonstrate the robustness of the framework in learning tasks with increased risk and difficulty induced by platform diversification and increased control dimensionality.

I. INTRODUCTION

Autonomous 3D navigation has a tremendous potential for robotic applications that range from operation in hazardous environments to providing assistance to humans. In the latter case, a robot may encounter steps and staircases that may have to be avoided or traversed by passive adaptation that palliates minor collisions. Most often, however, ensuring robot and environment safety requires active robot control.

Staircase negotiation has been predominantly addressed using conventional control, assuming proper knowledge of robot kinematics and dynamics. As a result, rudimentary changes of the platform or the task may render the adaptation of such solutions cumbersome if not impossible. Domain expertise is essential yet scarce and hard to obtain in realistic conditions due to the elevated accident risk.

On the other hand, machine learning-based approaches are better suited as they lower the need for ad-hoc solutions customized to specific platforms and endow a robot with the capability of safe and autonomous obstacle negotiation [1]. Using machine learning then raises the question of the type of learning to be sought, such as learning from demonstration (LfD), behaviour cloning [2], reinforcement learning (RL) or their combination (as for example in [3]). For the staircase negotiation task that we consider, it is impractical, if not impossible, to obtain sufficiently rich demonstrations. In this respect, RL-based approaches are particularly well suited while allowing the emergence of novel behaviors.

The work is performed in the context of the project REACT, project VITAAL and is financed by Brest Métropole, the region of Brittany and the European Regional Development Fund.

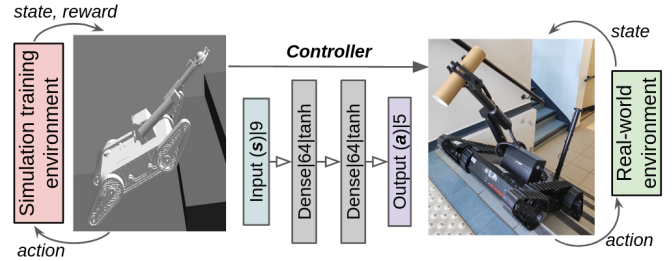


Fig. 1: Pipeline overview: the desired behavior is developed via RL in simulation and transferred in reality

In our current work we pursue a RL-based paradigm for the acquisition of robot controllers with desired behaviour properties such as safety and bumpiness through reward function design, in line with our earlier works, [4], [5]. In particular, we consolidate the training of effective controllers in simulation, by developing and comparing these skills on two distinct articulated, tracked robot vehicles in simulation and further successfully transferring the skill onto one of the real robots. On top of attaining our main goal that consists in successful staircase negotiation by the real robot, we also present new insights that are the product of quantitative as well as qualitative cross-comparison of behaviors among task variants and between robots. Concretely, the current work advances the state-of-the-art in the following points:

- We successfully acquire RL-based staircase negotiation controllers for two robotic platforms in simulation, both exhibiting the desired behavior properties.
- We quantitatively compare controllers obtained for different robots and task variants via Kullback-Leibler (KL) policy divergence.
- The efficacy of the obtained controllers is demonstrated in reality by transfer to a commercial robot (see Fig. 1).

The rest of the paper is organized as follows. Section II presents an overview of earlier approaches for robot control in staircase negotiation by articulated tracked robots. In section III we detail our framework for RL-based staircase negotiation learning in simulation, followed by the procedure for transferring and evaluating the obtained behaviors onto a real robot in section IV. Finally, in section V we present experiments that demonstrate the effectiveness of the framework in simulation as well as in reality through multiple variants of the main task.

II. CONTROL METHODS IN OBSTACLE NEGOTIATION

Robot navigation is dictated by the environment where the robot operates. As obstacle negotiation in outdoor as well as indoor environments requires advanced traversability skills, tracked robots equipped with additional sub-tracks (flippers) have been developed. More recently, the obstacle negotiation potential of such platforms is being studied for improving the autonomy of frail people, either for transporting humans or objects in 3D environments.

Therefore, a strong interest has been shown by the scientific community in modelling the kinematics and dynamics of such robots, in order to exploit their full potential. Although articulated front and/or rear flippers drastically increase the negotiation capability of complex obstacles, they inevitably increase complexity of control, kinematics calculation and estimation of robot dynamics. The presence of a robotic arm further elevates control difficulty due to additional degrees of freedom (DOF) and safety constraints.

Among multiple control methods reviewed by [6], neural-network based control that relies on unsupervised learning or reinforcement learning allows to learn effective controllers for high-dimensional observation and action spaces or decision making from raw input data such as pixels [1] in contrast to learning-free control methods. In the sequel, we review representative works from both categories that treated the problem of indoor navigation of tracked robots, while highlighting limitations and open challenges. For reference, Table I provides a coarse overview of those works.

Table I: Overview of related works

Work	Method	DOF	Input data sensors
Mourikis et al. [7]	State-feedback linearization	3	RGB camera, gyroscope
Nagatani et al. [8]	Fuzzy, vision-based	4	Laser sensors
Ben-Tzvi et al. [9]	Fuzzy	3	Motor encoders, compass
Zhang et al. [10]	Fuzzy, resolved motion rate controller	3	Gyroscope
Pecka et al. [11]	Neural, RL	2	IMU, laser sensors
Endo et al. [12]	Proportional-integrative	4	IMU
Ejaz et al. [13]	Neural, RL, end-to-end	2	Depth camera

Standard control Preliminary works on autonomous staircase traversal were devoted to highly customized, standard robot control. One of the first such works was presented in [7] relying on estimation of step edges and gyroscopic data. A highly customized platform was presented in [9], where thanks to a sophisticated track platform the problem of staircase traversal is tackled, further aided by an actively controlled pendulum. Unfortunately, approaches that are highly customized to specific hardware bear limited potential in general.

To palliate the increased cognitive load experienced by users that tele-operate such platforms, the authors of [8] propose a motion planning framework where the robot exploits side laser sensors in order to perceive the geometry of ahead obstacles and adapt flippers so that they are tangential to the surface. Upon the definition of the robot negotiation limits, the authors consider that the system is able to negotiate any uneven terrain,

yet being based on exhaustive sensory data. The framework of [14] is comparatively superior as the system is endowed with passive flippers that push against a traversed obstacle to improve traction, combined with an accident warning system that is based on estimation of the normalized energy stability margin (NESM). Still, the robot is not autonomous and the overall approach is weakly generalizable.

Another approach relying on precise estimation of the staircase configuration was proposed in [12] where the robot could autonomously negotiate a staircase using proprioceptive state estimation. That approach showed prominent performance, however, the system could only negotiate a staircase known beforehand. One of the most elaborate analysis was presented in [10] using a common tracked platform with 2 additional DOF due to a controllable arm. The derived analytical solution is indicative of the high complexity of modelling the interaction between a multi-DOF system and the traversed surface. That system lacked congruent control of tracks, flippers and the arm and needed full knowledge of geometry, kinematics and physical characteristics which can be difficult to acquire.

Learning-based control Learning-based approaches appear more suitable for developing controllers when robot kinematics and dynamics are harder to model. The authors of [15] identify two main categories of learning-based methods: *terrain traversability analysis* and *end-to-end* while [1] presents the diversity of employed learning techniques along with challenges. One of the main requirements is scalability to high-dimensional state and action spaces. Priors could bootstrap learning, still, this demands more attention in the design of the search policy algorithm in order to ignore irrelevant policy properties. Generalization and robustness are relevant for robots that are required to learn various tasks under varying conditions.

To the best of our knowledge, earlier work on learning-based control for actively articulated tracked robots is relatively scarce. At the same time, in staircase ascent and descent where it is hard to model the dynamics of physical interaction, learning-based techniques are particularly promising.

Policy endowment with properties such as bumpiness or stability can be performed in different ways. Constraints may be introduced in a RL optimization problem as in [11] while safety and reward can be associated in a single function [16]. Another more sophisticated approach was demonstrated in [11] where authors incorporated constraints into the optimization problem of relative entropy policy search using RL. Still, results were limited to the traversal of a palette.

Authors of [13] use images for the navigation of a tracked robot in indoor environments, improving dueling double deep Q-network model with layer normalization and noise injection and propose an original algorithm for its training. Still, this approach is only employed for navigation on 2D surfaces.

Deep Q-Network (DQN) was one of the first deep RL algorithms and had shown fascinating results at that time [17]. Over the years, RL algorithms have witnessed a significant evolution with the current state-of-the-art being composed of actor-critic algorithms which benefit from state-action value estimation as in pure value-based DQN and directly optimize policy parameters through gradient ascent.

Indicatively, the off-policy Deep Deterministic Policy Gradient (DDPG) algorithm [18] moved discrete action space to a continuous one as an approximate DQN. Still the major improvement of DDPG, which has introduced Twin-Delayed DDPG (TD3), corresponds to enhanced DDPG with delayed policy improvement, clipped double Q-Learning and target policy smoothing. In parallel to the latter algorithm, Soft Actor-Critic (SAC) was proposed which contains the same features combined with a novel policy entropy based maximization. An earlier on-policy algorithm is Proximal Policy Optimization (PPO) which yields more conservative updates.

III. ROBOT CONTROL FRAMEWORK

Following a typical RL-based problem formalization, given a state $s_t \in S$ the agent selects actions $a_t \in A$ according to a policy π and transits to the next state $s_{t+1} \in S$ receiving the reward $r_t \in R$. The array of consecutive state-action pairs $\tau = \{s_0, a_0, \dots, s_{T-1}, a_{T-1}\}$ creates the trajectory known as roll-out. The set of s_t, a_t, r_t, s_{t+1} observed along a roll-out is used by RL algorithms to perform policy optimization. In the sequel, we detail the chosen action and state spaces, present reward function design and employed controller for the problem of the staircase traversal, the way KL divergence is used to compare policies and finally their transfer to reality.

A. Problem description

We consider a robot that can minimally control 3 DOF corresponding to front and rear flipper angles and linear velocity, occasionally complemented with 2 extra DOF of a robotic arm whose use can prove particularly useful in staircase ascent and descent. Therefore, the entire continuous 5 DOF action space forms the following action vector:

$$a = (\psi_a^{front}, \psi_a^{rear}, v_a, \phi_a^1, \phi_a^2) \quad (1)$$

$$a \in [\psi_a^{min}, \psi_a^{max}]^2 \times [v_a^{min}, v_a^{max}] \times [\phi_a^{min}, \phi_a^{max}]^2$$

where ψ_a^{front} and ψ_a^{rear} are front and rear flipper angles, v_a is the applied main tracks velocity and ϕ_a^1 and ϕ_a^2 are the angles of the two arm joints. Superscripts *min* and *max* denote the kinematic limits of the corresponding DOF. We deliberately exclude yaw control since the traversal of a staircase prescribes that a robot maintains a fixed, orthogonal to the steps orientation along the staircase.

The observation state vector $s \in S$ is defined as follows:

$$s = (p_x^{front}, p_y^{front}, p_x^{rear}, p_y^{rear}, v_s, \psi_s^{front}, \psi_s^{rear}, \phi_s^1, \phi_s^2) \quad (2)$$

where p_x^{front}, p_y^{front} and p_x^{rear}, p_y^{rear} are the local coordinates of the next and previous step edge respectively, v_s is the robot linear velocity, ψ_s^{front} and ψ_s^{rear} are the flipper angles and ϕ_s^1, ϕ_s^2 are the arm joint angles (see Fig. 2).

State parameters were determined as a group rather than distinctively, in order to avoid redundancy and maximize complementarity between state dimensions. As a result, the pitch was excluded from the state because eq. (2) already allows to capture that parameter indirectly. Specifically, as the state contains the flipper angles and the coordinates of the front and rear step edges, this corresponds in reality to a specific pose of the platform onto the staircase at a certain pitch.

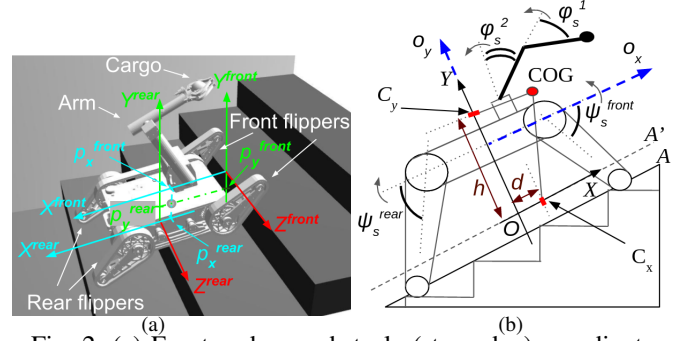


Fig. 2: (a) Front and rear obstacle (step edge) coordinate systems; (b) side schematic view of the robot on a staircase

B. Reward function design

In this section we present reward functions used to obtain policies allowing ascent and descent staircase traversals. To guide learning we assign the following positive reward proportional to the traversed path from start to finish:

$$R^{tr}(\tau) = \frac{\sum_{t=1}^{N_\tau} x_t}{D_{max}} \quad (3)$$

where $R^{tr}(\tau)$ represents the episode return along a trajectory τ whose maximal value can attain 1, x_t is the traversed distance during time step t , N_τ is the number of time steps in the current episode and D_{max} is the total distance between start and finish. While only positive rewards can produce policies that accomplish a given task, those may be sub-optimal in terms of features such as safety. Furthermore, the situations that may compromise the safety of the robot differ between ascending and descending a staircase. To account for robot safety in different contexts we introduce negative penalties alongside with the positive reward.

In detail, to reduce the risk of tip-over during ascent we assign a negative reward proportional to the deviation of the robot centroid. On the other hand, when the robot descends a staircase accidents are mostly due to platform shaking and bumps, therefore we tackle this via a negative reward proportional to platform acceleration. In either case, the scale of a negative reward is normalized to avoid biasing. Independently of the chosen negative reward r_t , the overall time step return R_t then becomes:

$$R_t = \frac{x_t}{D_{max}} + r_t. \quad (4)$$

The next two subsections present the design of the aforementioned negative rewards.

1) *Center of gravity stabilization*: Whenever the center of gravity (COG) destabilizes it may lead to robot tip-over. In detail, during ascent the COG position leads to tip-over only when its projection point C_x crosses the lowermost edge of the robot support polygon, namely, the point of contact of the rear flipper with the staircase (cf. Fig. 2 (b)). In accordance with [5], the COG-based penalty r_t^D is set as:

$$r_t^D = \begin{cases} -1, & \text{if tip over} \\ -K_D * D_t, & \text{otherwise} \end{cases} \quad (5)$$

where K_D is the normalizer and D_t the COG deviation. We assign this penalty only in ascent tasks and when the robot COG projection is located between staircase step edges.

In [4] a robot learnt a stable behavior by favoring poses with low COG with respect to the underlying surface A' that represents the Normalized Energy Stability Margin (NESM) [19]. The presence of an arm nonetheless may place the COG low but also close to the "safety zone" border [10], without violating the NESM. To overcome this, we use the Stability Margin (SM) that estimates the distance between the lowermost robot edge and the COG projection on the ground.

Since it is difficult to estimate where exactly the lowermost robot edge touches the ground, we instead minimize the deviation of the COG projection C_x on the staircase from the projection point O of the robot geometric center. We can thus guarantee that the robot respects the SM criterion. We consider that the arm has to place the COG as close as possible to the point O - the most stable point along both X and Y axes, optimizing both the SM and the NESM. Thus, in eq. (5) the deviation is set as $D = \sqrt{d^2 + h^2}$ where d and h are the distances between O and the projections of the COG on the X and Y axis respectively.

Besides safety, we further observed that the contact of the robot tracks with the surface cannot alone guarantee proper traction. In reality, it is essential that sufficient force is exerted from the robot to the ground which is possible when the projection of the COG C_x is located near the perpendicular projection of the geometrical center O on the staircase. This leads to more uniformly distributed traction along the entire platform. Finally, we further penalize the robot when it loses stability and tips over that happens when its pitch reaches $\pi/2$.

2) *Platform shaking reduction*: The effect of gravity to the movement of the robot switches between ascent and descent, by hindering ascent and accelerating descent that may endanger the robot and lead to tip-overs. Even if no accident happens, every staircase edge traversal causes platform shaking and collisions which influence the robot and potentially an object that is being transported. These events are repetitive when the robot traverses every step, but the greatest impact can be observed in the beginning of the traversal when the robot COG traverses the uppermost stair edge. At that moment the robot velocity can be instantaneously increased leading to a shock, slippage and, at worst, tip-over. The end of traversal is also crucial because if the rear flippers are pushed down the rear robot part just falls down from the step.

Such behaviours lead to distinctive pitch velocity peaks, therefore the controller should mitigate such events. Letting W_t denote pitch velocity and K_W the normalization coefficient, the corresponding negative reward is:

$$r_t^W = \begin{cases} -1, & \text{if tip over} \\ -K_W * W_t, & \text{otherwise} \end{cases} \quad (6)$$

This penalty is assigned only in descent tasks and when the robot traverses the staircase, namely, when the COG projection on the staircase is located between the staircase nosing limits.

C. Controller learning details

In designing a controller learnt via RL there are several alternatives that can be considered [17]. Approximators such as tile or coarse coding tend to suffer from the curse of dimensionality and do not generalize well due to sensitivity to grain size. The main advantage of artificial neural networks over other function approximators is their generalization capability and straightforward increase of depth which was well studied.

Another common choice is radial basis function neural networks (RBF-NN) which were widely applied before the emergence of deep neural networks. RBF-NN converges quickly to global optima with less trials and errors, still, they have proved of limited use due to absence of tractable and stable integration into more complex deep neural network architectures.

Shallow networks with one hidden layer tend to have more parameters to approximate the same function in contrast to deep networks with less parameters and the additional ability to learn different representations at intermediate levels. Thus, we have opted for a multi-layer perceptron (see Fig. 1). The first layer is composed of 9 neurons on which the state vector s is received every time step. In the sequel, the input is forwarded through 2 dense layers to the output layer which forms the action vector a .

Following up on the discussion at section II, SAC and TD3 are considered to be more sample efficient than PPO, since they are off-policy and converge faster to higher returns. Still, we use PPO in our work to optimize the parameters of the policy because it is more straightforward to implement, less sensitive to hyper-parameter changes and shown the most favorable results in deep RL.

D. KL divergence-based policy comparison

We can quantitatively compare a pair of trained policies via a commonly used probability distribution divergence measure, namely, the KL divergence. Often, RL algorithms exploit the divergence as metric to be optimized or a constraint of policy updates as for example in PPO where updates of the policy parameters are constrained through a penalty on KL divergence.

We denote the reference policy for a given task as π_q and we wish to find how the policy π_p diverges from π_q (assuming same observation and action spaces). The process of learning a policy amounts to learning the mean and variance statistics of the underlying probability distribution, rather than the exact probability distribution for each possible state-action pair. Making the hypothesis of a normally distributed policy with mean and variance that depend on the state, a policy can then be sampled randomly over the total space of observations with actions chosen according to that policy. Thus, once the total set of observations has been sampled, the corresponding actions are prescribed by one of the two policies being compared, namely, the reference policy π_q . After having determined the state-actions pairs over which the policies will be compared, we then calculate the divergence as:

$$D_{KL}(\pi_p, \pi_q) = \sum_{s,a} \pi_p(s,a) \log \frac{\pi_p(s,a)}{\pi_q(s,a)} \quad (7)$$

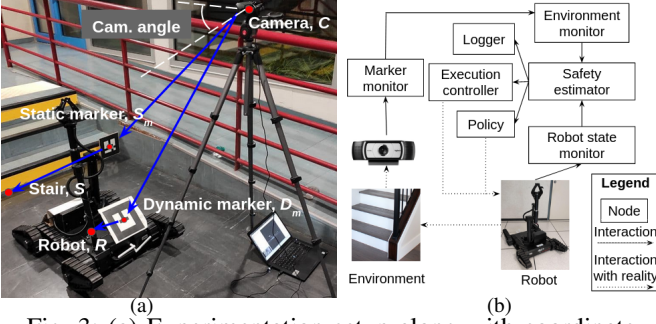


Fig. 3: (a) Experimentation setup along with coordinate frame hierarchy; (b) architecture of deployed robot system

Training a policy is always a stochastic procedure and as such a policy is a random variable itself, together with the number of state-action pairs that we choose to calculate divergence. We currently determine empirically the number of times needed to train a policy as well as the number of samples taken in the calculation of the divergence, though a more consistent estimation is possible if required.

IV. CONTROLLER DEPLOYMENT ON REAL ROBOT

System description We begin by presenting the experimentation set-up used to obtain the ground-truth robot and environment state. This allows to establish an upper bound on the effectiveness of a controller in reality with moderate state errors while allowing the development and evaluation of alternative robot localization approaches later on. On the other hand, we opted for injecting noise in the state during policy learning in simulation (cf. [4]) to make it more robust to errors in general as well as to reduce the risk of accidents.

To obtain the robot pose we employed visual markers whose pose can be reliably obtained via calibrated cameras and standard toolkits. We distinguish markers into *static* and *dynamic* with associated coordinate frames S_m and D_m (see Fig. 3). We use the notation jT_i for the transformation between coordinate frames, denoting translation and rotation of the coordinate frame i within the coordinate frame j . We can then obtain transformations ${}^C T_{S_m}$ and ${}^S T_{D_m}$, the latter being measured manually. The dynamic marker is placed onto the robot and used to locate its base coordinate frame with respect to camera C , via a constant transformation ${}^R T_{D_m}$ set manually.

The camera was placed in front of the operational space of the robot, ensuring proper coverage of the robot as well as the staircase. Its purpose is to associate the robot and the staircase into a common hierarchy of coordinate frames, allowing at any moment to retrieve the coordinates of the front and rear step edges with respect to the robot (recall eq (2)). The staircase is then represented as a series of static transformations ${}^{S_i} T_S$ where S_i is the coordinate frame associated to a step edge.

The staircase perception task could also be performed by a robot using its on-board sensors but we refrain to do so to keep our set-up generic and independent of the robot. Perception of a staircase before descent is more challenging but this problem resides out of our current scope.

Policy deployment from simulation to reality In the beginning of an experiment the robot faces the staircase and

the controller is activated upon the approaching of front and rear step edges. For the task of staircase negotiation, it turns out that only a subspace of the entire action space spanned by the flipper and arm joints is useful. This is something that we take into account by constraining the action space from which actions can be sampled. As a direct consequence of the fact that joints are allowed to move in a significantly smaller space, this provides sufficient time to perform a necessary action and adaptation of low-level controllers from the simulated platform to the real one and makes this transfer zero-shot. Crucially, constraining the action space further serves in ensuring safety of the platform, since learning is stochastic and non-previously encountered conditions may lead to accidents. For example, overly raising rear flippers while ascending or front flippers while descending, combined with acceleration can provoke a tip-over. To prevent such behaviours, the limits of rear flippers while ascending and front flippers while descending were set as $\psi_s^{rear}, \psi_s^{front} \in [-\pi/4, 0]^2$ respectively.

The deployed system architecture is shown in Fig. 3 (b). The *Marker monitor* detects markers in the camera image. The *Environment monitor* builds and maintains the geometric relationships between the staircase and the robot, within a single transformation hierarchy and provides the front-most and rearmost step coordinates (recall eq. (2)). The *Robot state monitor* provides information about the robot flippers and arm configuration, its velocity and IMU data. The *Safety estimator* receives the output of *Environment monitor* and *Robot state monitor*, evaluates safety metrics such as the COG deviation and angular velocity and its output is concatenated with data provided by *Environment monitor*. That output is fed to *Execution controller* which decides whether to block motors in the case of upcoming accidents or halt the system when the experiment terminates. Otherwise, state output data pass to *Policy* which samples the action vector \mathbf{a} which is sent to the robot actuators. The latter executes those actions and *Marker monitor* observes changes in the environment. Finally, the output of *Safety estimator* is logged via *Logger*.

V. EXPERIMENTS

In this section we present the experiments that we performed with two different platforms, namely, **Absolem**¹ [11] and **Jaguar V4 Manipulator**². Extensive simulated and real-world experiments were performed with varying degree of difficulty for the task of staircase negotiation, allowing us to successfully deal with the associated challenges (cf. video³).

A. Description of platforms and associated tasks



In Table II we juxtapose the two articulated tracked robots for which we trained the various controllers. Their simulated models matched the geometric and mass characteristics of the real ones, making use of vendor provided geometric models. The two robots possess similar obstacle negotiation properties thanks to the presence of flippers, with the main difference being that Absolem can independently move each flipper in

¹bluebotics.com, github.com/mariogianni/trav_nav_indigo_ws

²jaguar.drrobot.com/specification_V4Arm.asp

³partage.imt.fr/index.php/s/LDyp6QRp4nGGKd2/download

TABLE II: Robot characteristics and task variants

Description	Jaguar	Absolem
Photo		
Mass, kg	41	29
Length, m	0.98	1.196
Width, m	0.7	0.61
Height, m	0.4	0.46
Task	Task id	Task id
Ascent, 3DOF, Default	Jag-Asc-3-Def	Abs-Asc-3-Def
Ascent, 3DOF, COG	Jag-Asc-3-COG	Abs-Asc-3-COG
Descent, 3DOF, Ang.	Jag-Des-3-Ang	Abs-Des-3-Ang
Ascent, 5DOF, COG	Jag-Asc-5-COG	None
Descent, 5DOF, Ang.	Jag-Asc-5-Ang	None

contrast to Jaguar where front and rear flippers are coupled. To make consistent comparisons, we therefore chose to couple Absolem flippers as well.

It follows that each robot has at most 4 DOFs corresponding to linear velocity, angular velocity, front and rear flipper angles, with Jaguar having 2 additional DOFs due to its arm. For the staircase traversal task though we have deemed that yaw angular velocity control was unnecessary during traversal after the following observations; (i) a robot can easily align itself with the staircase before traversal and (ii) the robot remains aligned during traversal thanks to the coupled front and rear flippers grip on the steps. Thus, the maximum number of DOF is 3 for Absolem and 5 for the Jaguar.

Each task of Table II is thus characterized by the direction of traversal (Ascent or Descent), the number of DOF of the action space and the type of reward used to guide learning (*Default* refers to the use of eq. (3), *COG* refers to the use of eq. (4) and (5) and *Ang* refers to the use of eq. (4) and (6)).

B. Simulation environment

The simulation environment used for training is shown in Fig. 4. The Gazebo simulator (gazebo.org) is employed using the Contact Surface Motion (CSM) model of tracks [20] for both robots. Adding a new robot has no impact on the overall training time since each robot controller can be trained independently and all of them in batch, server-side simulations. The simulation environment further allows to vary staircase size in ranges of real-world staircases (cf. [4] for details). We note that this is an essential step in succeeding to transfer the policies trained in simulation into the real-world and can be thought of as a data augmentation step, that favors generalization over multiple obstacle shapes and further deals with the influence of noise in the state estimation process.

The robot starts off at the usual location at either the ground floor or the first floor depending on traversal direction, aligned with the staircase and has to achieve the goal position by respecting the preset criteria (both positive and negative rewards are assigned during traversal). The episode ends if the robot tips over, exits the training area, reaches the goal or exceeds the maximal episode time step length.

We evaluate 8 task variants as listed in Table II differing by traversal direction, DOF variation, applied criterion and

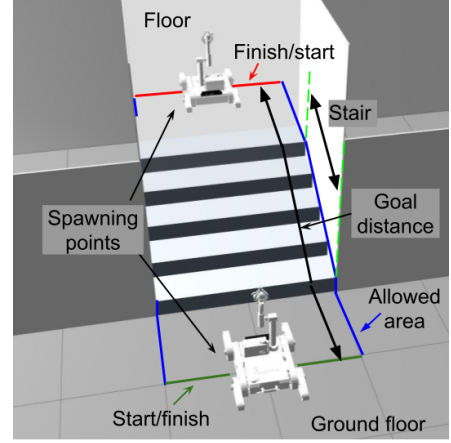


Fig. 4: Simulation environment

robotic platform. The Jaguar robot model starts always with a vertically stretched arm that is fixed in 3 DOF tasks, flippers of both platforms are parallel to the floor in the beginning and $\phi_a^1, \phi_a^2 \in [-90^\circ, 90^\circ]^2$. Finally, penalties are evaluated and assigned only when robots reside on the staircase.

C. Results

1) *Performance for ascent tasks:* Fig. 5 (a) shows the mean smoothed episode return over independent learning trials with min-max bands, for (Jag-Asc-3-COG) and (Abs-Asc-3-COG).

Mean episode return starts from -0.23 and -0.44 for the two tasks, because of the negative penalty and the failure of the robots to advance to the goal, and converges to 0 by the end of learning for both policies. The max bands can attain the values of 0.26 and 0.28 episode return respectively. This means that the robots learn to control flipper angles and the linear velocity. The episode return for the policy (Abs-Asc-3-COG) is lower in the beginning because the Absolem COG is higher and car tip-over more easily. None of the two learning curves reaches the maximum return 1 as it is impossible to put the COG closer to the point O (see Fig. 2) than a certain distance. This is normal since the robots cannot be entirely parallel to the ground while they traverse the staircase.

Fig. 5 (b) shows the evolution of the COG during learning. Both policies reduce the COG deviation at the same pace and reach minimal values after 8000 time steps, or 120 episodes. The Absolem policy exhibits higher COG deviation equal to 0.19 m due to the higher COG placement in the initial configuration where flippers are extended, still, it decreases the initial COG by 17%. The Jaguar policy decreases the COG deviation by 36% down to 0.075 m that approximately represents the COG deviation along the Y axis.

The learning process for two robots is fairly similar. Furthermore, the same value of episode return is achieved by the end of training. The COG curves exhibit similar behaviour and converge at different values, due to differences in mass distributions. Those differences also change robot control features, still both robots achieve the goal by minimizing the COG deviation. Note, however, that the absolute COG deviation difference between Absolem and Jaguar noticed at

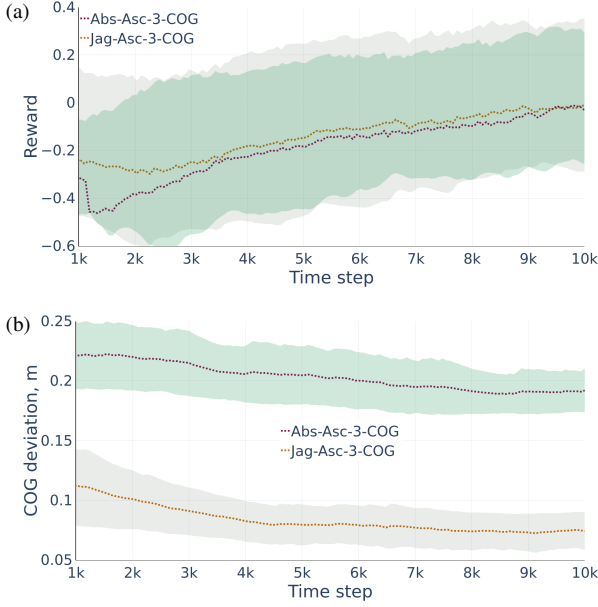


Fig. 5: Ascent task learning analysis

the end of learning in Fig. 5 (b), is not transposed to an equivalent absolute difference in the total reward in Fig. 5 (a), thanks to the normalization of each negative penalty. These results strongly indicate that the framework is applicable to different robot models for the ascent task.

2) *Performance for descent tasks:* Figure 6 (a) presents learning curves for tasks (Jag-Des-3-Ang) and (Abs-Des-3-Ang). The firsts starts at -0.1 episode return and converges to the value 0.1 . The second policy convergences similarly but starts at -0.3 episode return and ends at -0.1 . Both platform policies exhibit similar learning behaviour separated by around 0.2 that can be explained by robot dynamics, because the Absolem is exposed to higher pitch angular velocity. This is seen in Fig. 6 (b) where the mean angular velocity of a platform during an episode is higher for the Absolem, due to a higher centroid position over the surface in the resting state. The Absolem policy decreases its angular velocity by the end of learning by 13% , while the Jaguar policy drops it by 10% . Minimal angular velocity values are attained by 5000 steps, but from this moment onward the learning curves slight increase. This may happen due to optimization of traversal time as the robot spends less time in staircase traversal. As before, these results suggest that the framework is effective for both robots.

3) *KL divergence between policies:* Figure 7 presents KL divergences for learned controllers in the 3 DOF tasks. We omitted 5 DOF tasks since the simulated model of Absolem robot does not include an arm. An $(i, j)^{th}$ cell of the heat map represents the divergence between a controller with row index i and a controller with column index j , namely, to $D_{KL}(\pi_i, \pi_j)$. If the KL divergence is low for two policies, then this implies that very similar actions are chosen for the same observations.

The heat map allows the extraction of some very useful insights. For example, for a given staircase traversal direction and total reward design, the two robots develop different policies. This can be particularly seen in divergences (1,4) and

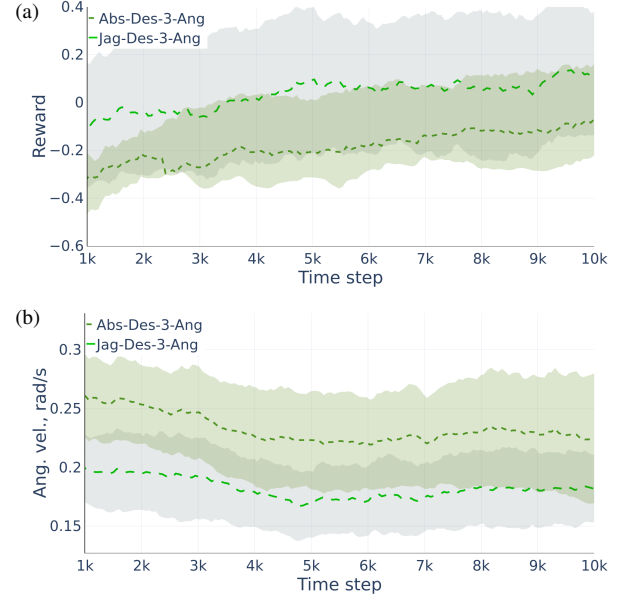


Fig. 6: Descent task learning analysis

(2,5) and demonstrates the importance of separate controller training for the different robots. Despite the fact that, in simulation, the robots share the same motors, the same observations and the same action space, the differences in geometry and mass distribution result in different necessary flipper actions throughout the traversal. It further suggests an operator that is sufficiently skilled to operate a given platform, will not be necessarily able to operate a slightly different platform. We can further observe that for a given robot and traversal direction, alternating the total reward function also induces a quantitative change in the employed actions, for example in divergences (1,2) and (4,5). This justifies our interest for optimizing staircase negotiation with more elaborate criteria than merely arriving to the goal.

Finally, the highest divergences are noticed between policies where the staircase traversal is different, namely for (1,6) and (2,6). This clearly suggests that staircase ascent and descent require independent treatment, as dynamics and risks signifi-

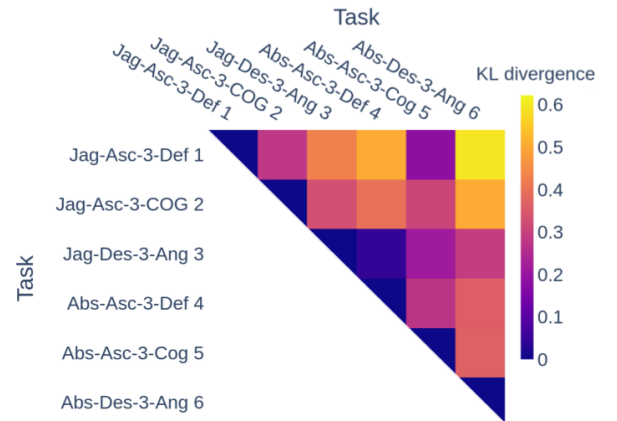


Fig. 7: Mean Kullback-Leibler divergence between policies

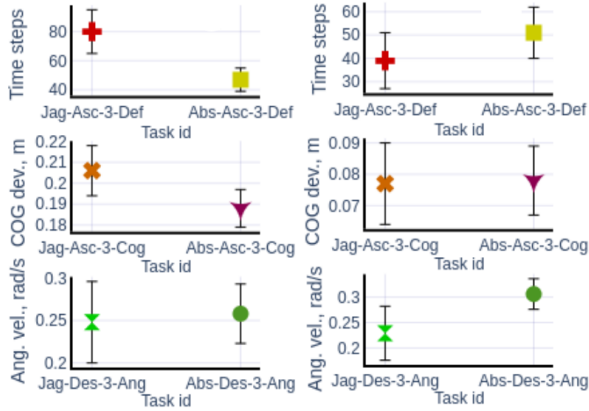


Fig. 8: Performance evaluation of policies when deployed on Absolem (left) and Jaguar (right) robots.

cantly differ. Two pairs of policies were found unexpectedly similar however, i.e. (1,5) and (3,4), an event that can be attributed to the stochasticity of training.

We finally performed a quantitative analysis of policy adequacy on different platforms provided in Fig. 8. This experiment is destined to assess how well a policy trained on one robot would perform when deployed to another robot. We performed 10 such trials (rollouts) for a given task. Policies (Jag-Asc-3-Def) and (Abs-Asc-3-Def) are evaluated on the basis of time steps spent in a rollout. As we can see, if we deploy a policy on a platform for which it was trained, then the robot spends less episodes until task completion. Characteristically, rollouts tested on Absolem last around 47 time steps if we use the (Abs-Asc-3-Def) policy against 80 time steps if (Jag-Asc-3-Def) policy is used where performance is significantly degraded. Similarly, deployment of the (Jag-Asc-3-Def) on Jaguar requires on average 39 time steps as opposed to 51 when the (Abs-Asc-3-Def) policy is applied.

Deployment of (Jag-Asc-3-Cog) on Absolem further reveals a slight performance degradation in terms of COG deviation measured at 0.206 against 0.188 when (Abs-Asc-3-Cog) is deployed. In contrast, the deployment of these two policies on Jaguar seems equally performant.

With respect to descent, the deployment of the (Jag-Des-3-Ang) and (Abs-Des-3-Ang) policies on the Jaguar robot clearly favors the robot specific policy, but the same policies perform equally well on Absolem. We can conclude that a policy trained and deployed on the same robot provides the best performance, otherwise performance can only be degraded.

4) *Real-world performance:* Policies (Jag-Asc-5-COG) and (Jag-Des-5-Ang) were tested in two staircases presented in Table III (we recall that the arm is actively involved in the optimized controllers). The respective experiments are included in the supplementary video. Overall, we have performed 10 trials for each task id (5 trials per staircase) discussed in this section, all of which were successful.

Policy (Jag-Asc-5-COG) was tested on the big staircase, the robot achieving the task by keeping the rear flippers pushed down, the front flippers up, the first arm joint was inclined clockwise and the second arm joint counter-clockwise. The

TABLE III: Staircases configurations

Name	Number of steps	Height	Length
Big	5	0.195	0.275
Small	3	0.17	0.305

TABLE IV: Observed mean target values on real staircases

Task id	Stair	C_y , m	D , m	Ang. vel., rad/s
Jag-Asc-5-COG	Big	0.11 ± 0.01	0.12 ± 0.01	0.33 ± 0.06
Jag-Asc-5-COG	Small	0.1 ± 0.01	0.11 ± 0.01	0.36 ± 0.09
Jag-Des-5-Ang	Big	0.16 ± 0.02	0.09 ± 0.02	0.23 ± 0.06
Jag-Des-5-Ang	Small	0.12 ± 0.02	0.05 ± 0.02	0.23 ± 0.09

flipper configuration ensures that the robot has the contact points with rear and front stair steps. The entire arm is moved forward to decrease COG deviation, which minimizes tip-over risks or getting stuck. A point of ambiguity emerges for the arm as certain configurations decrease C_y and increase C_x at the same time, or vice versa, hence minimizing COG deviation D is not straightforward. Remarkably, the obtained results of the learnt controller show that the robot learnt to incline forward the first arm link and backward the second link, which is indeed the best configuration. For reference, Table IV provides average observed values for some key parameters.

The accomplishment of the descent task (Jag-Des-5-Ang) on the big staircase merits more attention. One of the most important aspects is the linear velocity control (see provided video), where the robot moves smoothly and adapts its velocity very attentively that leads to increased time in the staircase traversal. Naturally, this greatly reduces the mean perceived angular velocity by 0.1 rad/s, as opposed to the ascent task.

The ascent and descent tasks were further successfully accomplished on the small staircase composed of 3 steps, each of height 0.17 m and length 0.305 m. The values reported in Table IV allow us to draw the same conclusions between ascent and descent, as for the big staircase.

VI. CONCLUSION

We have presented an effective RL framework for control learning of staircase negotiation in multiple task variants. In particular, we trained controllers for two robotic platforms in simulation with reward function designs suited for ascent and descent and different staircases. Results show learning convergence and optimization of targeted behaviour features for both platforms within 150 episodes. We employed KL divergence to quantitatively compare the obtained policies, allowing us to consolidate earlier empirical findings. Most importantly, we succeeded in deploying the controllers learned in simulation to a commercial robotic platform in firstly encountered real-world staircases. The robot was able to successfully ascend and descend while respecting the underlying criteria.

Our framework can accommodate further improvements that would further increase the autonomy of the deployed system. Although we managed to train effective controllers for normal situations, accidents might still happen as the robot may encounter new situations not explored in training, hence incremental learning is an interesting perspective. Also, full 3D navigation autonomy implies the development of multiple individual controllers that need to be coordinated with each

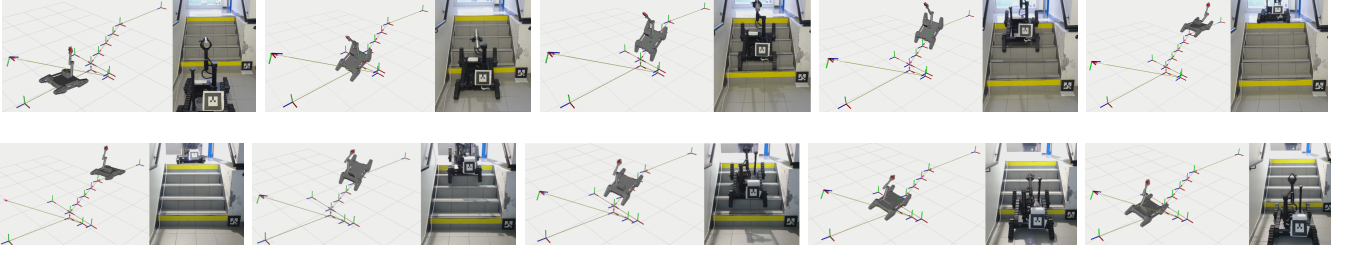


Fig. 9: Snapshots of staircase negotiation with congruent control of 5 DOF. Top row; ascent by minimizing COG deviation. Bottom row; descent by minimizing pitch angular velocity

other. We seek to address these challenges in the future as well as explore the possibility to use raw sensory data as input to the learnt controllers.

REFERENCES

- [1] K. Chatzilygeroudis, V. Vassiliades, F. Stulp, S. Calinon, and J. Mouret, "A survey on policy search algorithms for learning robot controllers in a handful of trials," *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 328–347, 2020.
- [2] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," in *In Proc. of the 27th Int. Joint Conf. on Artif. Intell.*, 2018, pp. 4950–4957.
- [3] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, vol. 29. Curran Associates, Inc., 2016.
- [4] A. Mitriakov, P. Papadakis, S. M. Nguyen, and S. Garlatti, "Staircase traversal via reinforcement learning for active reconfiguration of assistive robots," in *IEEE Int. Conf. on Fuzzy Systems*, 2020.
- [5] A. Mitriakov, P. Papadakis, S. M. Nguyen, and S. Garlatti, "Staircase negotiation learning for articulated tracked robots with varying degrees of freedom," in *Int. Symposium on Safety, Security and Rescue Robotics*, 2020.
- [6] S. K. Tzafestas, V. Vassiliades, F. Stulp, S. Calinon, and J. Mouret, "Mobile robot control and navigation: A global overview," *J. of Intelligent Robotic Systems*, vol. 91, pp. 35–58, 2018.
- [7] A. I. Mouriakis, N. Trawny, S. I. Roumeliotis, D. M. Helmick, and L. Matthies, "Autonomous stair climbing for tracked vehicles," *The Int. J. of Robotics Research*, vol. 26, no. 7, pp. 737–758, 2007.
- [8] K. Nagatani, A. Yamasaki, K. Yoshida, T. Yoshida, and E. Koyanagi, "Semi-autonomous traversal on uneven terrain for a tracked vehicle using autonomous control of active flippers," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2008.
- [9] P. Ben-Tzvi, S. Ito, and A. A. Goldenberg, "A mobile robot with autonomous climbing and descending of stairs," *Robotica*, vol. 27, no. 2, p. 171–188, 2009.
- [10] H. Zhang and A. Song, "System centroid position based tipover stability enhancement method for a tracked search and rescue robot," *Advanced Robotics*, vol. 28, no. 23, pp. 1571–1585, 2014.
- [11] M. Pecka, V. Šalanský, K. Zimmermann, and T. Svoboda, "Autonomous flipper control with safety constraints," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2016.
- [12] D. Endo, A. Watanabe, and K. Nagatani, "Stair climbing control for 4-dof tracked vehicle based on internal sensors," *J. of Robotics*, pp. 1–18, 10 2017.
- [13] M. M. Ejaz, T. B. Tang, and C. Lu, "Vision-based autonomous navigation approach for a tracked robot using deep reinforcement learning," *IEEE Sensors J.*, vol. 21, no. 2, pp. 2230 – 2240, 2021.
- [14] S. Suzuki, S. Hasegawa, and M. Okugawa, "Remote control system of disaster response robot with passive sub-crawlers considering falling down avoidance," *ROBOMECH J.*, vol. 1, 2014.
- [15] D. C. Guastella and G. Muscato, "Learning-based methods of perception and navigation for ground vehicles in unstructured environments: A review," *Sensors*, vol. 21, no. 73, 2021.
- [16] K. Zimmermann, P. Zuzanek, M. Reinstein, and V. Hlavac, "Adaptive traversability of unknown complex terrain with obstacles for mobile robots," in *IEEE Int. Conf. on Robotics and Automation*, 2014.
- [17] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2018.
- [18] A. Azar, A. Koubaa, N. Ali Mohamed, H. Ibrahim, Z. Ibrahim, M. Kazim, A. Ammar, B. Benjdira, A. Khamis, I. Hameed, and G. Casalino, "Drone deep reinforcement learning: A review," *Electronics*, vol. 10, 2021.
- [19] E. Garcia, J. Estremera, and P. Gonzalez-de-Santos, "A classification of stability margins for walking robots," *Robotica*, vol. 20, no. 6, pp. 595–606, 2002.
- [20] M. Pecka, K. Zimmermann, and T. Svoboda, "Fast simulation of vehicles with non-deformable tracks," *CoRR*, vol. abs/1703.04316, 2017.