Decoding Motor Skills of AI and Human Policies: A Study on Humanoid and Human Balance Control

Kai Yuan, Christopher McGreavy, Chuanyu Yang, Wouter Wolfslag, and Zhibin Li

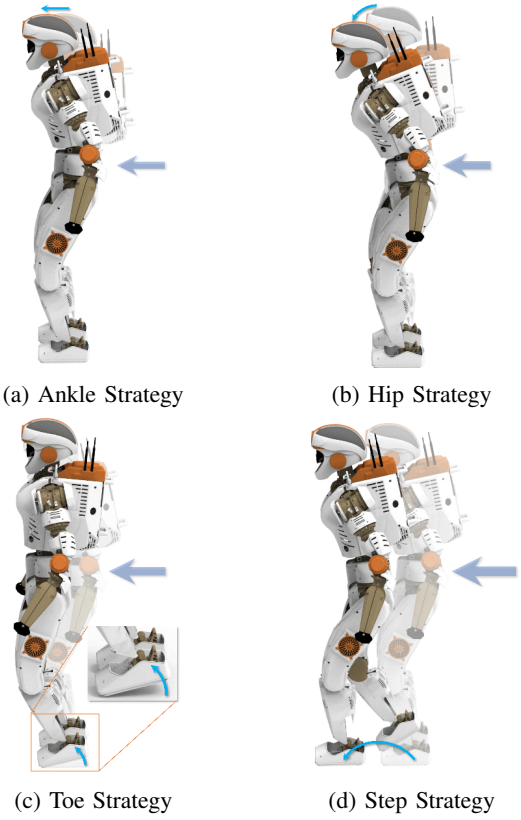
***Abstract*— In this study, we propose a new paradigm of using a machine learning approach to facilitate a quicker, more efficient and effective control development, as a different approach of utilising the power of machine learning in addition to other options that intent to use learning directly in real-world applications. We first develop a DRL-based control framework to learn rich motor skills of push recovery for humanoid robots that exhibit human-like push recovery behaviour. Next, we propose to take advantage of DRL to quickly discover solutions for very difficult problems, and then extract the principles of those policies as guidelines for developing engineered controllers. Furthermore, a comparison between humanoid and human balancing is conducted to show the characteristics of the learned humanoid behaviour. This comparison will show that DRL algorithms can learn a good policy with short development and training time that may require humans years to learn. We analyse input-output data collected from humanoid and human policies and postulate a Minimum-Jerk Model-Predictive Control (MJMPC) Framework that quantitatively reflects both AI and human push recovery policies.**

I. Scientific Motivation

From the advancement in computers, computer-aided design for mechanical and electronic engineering, architecture and many other engineering fields emerged. Foreseeing a similar development curve and technology wave, we forecast a new emerging discipline in the near future that uses learning-aided approaches for catalysing control development, alongside other similar applications such as in medicine discovery. In this study, we propose a new paradigm of using a machine learning approach to facilitate a quicker, more efficient and effective control development, as a different approach of leveraging the power of machine learning in addition to other options that intent to use learning directly in real-world applications.

Machine Learning and Deep Reinforcement Learning (DRL) in particular have reached an advanced stage to produce powerful policies with better autonomous performances than many state-of-the-art control and planning approaches in robot locomotion [1], robotic manipulation [2], and even the control of complex morphological machines [3]. Notably, DRL's ability to solve complex problems with a relatively short development time is especially attractive, which is empowered by training policies that maximise the cumulative reward through the exploration of the action and state space, rather than using prior knowledge of the models about the robot, the world, and their interactions.

To leverage the capabilities of DRL, we first develop a DRL-based control framework to learn rich motor skills of push recovery for humanoid robots. The complexity in whole-body balancing arises in challenges such as multi-contact coordination based on multi-sensory inputs, state transitions between fully- and under-actuated situations, switching policies, and generalising to external disturbances on any body parts, while accounting for all edge cases that a designer has difficulty to consider beforehand. In such a setting, manually designing the individual control strategies and finding a reliable switching mechanism requires both substantial development time, mathematical rigour, and code implementation. On the other hand, through a well-designed DRL framework and task-specific training procedures, a robust policy can be learned automatically by interacting with the environment, requiring only computational power. In particular, as shown in Fig. 1, our *learned policy exhibits human-like push recovery behaviour* with four typical push recovery strategies emerging naturally: ankle, hip, toe, and



(a) Ankle Strategy (b) Hip Strategy

(c) Toe Strategy (d) Step Strategy

Fig. 1: Human-like Push Recovery strategies emerging from Deep Reinforcement Learning. The discovered behaviours serve as a guideline for the design of certifiable and safe controllers that replicate advantageous strategies from AI policies.

IEEE Robotics and Automation Magazine (RAM) paper, presented at IROS 2020. It should be cited as a RAM paper.

stepping strategy.

Though the learned control policy could possibly be deployed on the real robotic system, the lack of explainability and analytical reasoning of the Neural Network makes it unsuitable for safety-critical applications in real world. Furthermore, due to the demand of large data and sample-inefficient nature of DRL algorithms, complex policies are typically trained in simulation, which cannot guarantee the same performance while transferred directly to the real system [1], and the challenge of reality gap raises concerning about both the safety and performance.

To benefit from both the *safety and interpretability for the control policy* and the *versatility and adaptability from learning*, we propose to take advantage of DRL to quickly discover versatile, deployable policies and solutions for very difficult problems, and then study, analyse and extract the principles of those policies as guidelines for developing engineered controllers in a reliable manner. By doing so, we utilise the AI-solutions for rapid control development (Fig. 3) to design safe and certifiable controllers which can be verified and deployed on real-world robots (Fig. 2).

While classical control development is based on gradually building knowledge that increases the performance incrementally, using a template policy will provide disruptive, innovative solutions that will escalate performance (green line, Fig. 3). DRL is able to achieve good performance by a number of iterations in the DRL learning framework. However, the achieved performance is still comparatively low to what tuning in control can do. Combining both approaches to "kick-start" the iteration process helps to design good controllers. After knowing the system and the controller, it is straightforward to improve upon due to the fact that we are then able to understand why the performance is lower than the optimum, whereas in the case of DRL, there is little influence from human engineers to improve the performance but reshaping the reward and/or altering the learning framework, and relying on the exploration being sufficiently large to achieve high performance.

In this paper, we are motivated to study a viable approach to infer underlying principles of an AI policy by studying its perception-action relation, i.e., to some extent, reverse-engineer an equivalent controller in terms of functionality based on a black-box policy. This methodology is not only applicable to AI policies, but also to any black-box policies, such as a human policy. Without knowing exactly how push recovery policies are realised by Artificial Neural Network (ANN) or biological human Neural Network, we can still analyse the behaviour at the functionality level by studying their input-output relationship.

Based on evidence of optimality in human manipulation tasks [4], we hypothesise that policies for push recovery in humans and humanoid are both optimal control process that follows certain optimal criteria that can be quantified. Following this hypothesis, we analyse and utilise input-output data collected from both humanoid and human policies, and propose a Minimum-Jerk Model-Predictive Control (MJMPC) Framework that is able to quantitatively reflect both the AI and human push recovery policies. The engineered controller has high similarity (Coefficient of Determination more than 90%) with the collected data, and also exhibits the same human-like push recovery strategies, which emerge from the proposed MJMPC without the need of manual switching between the strategies.

Furthermore, a comparison between humanoid and human balancing is conducted to show the characteristics of the learned humanoid behaviour. This comparison will show that DRL algorithms are very powerful to learn a policy (e.g., balancing) within a short development and training time that may require humans years to learn. In contrast, in order to design an engineered controller from scratch with similar performance, months or even years are needed for developmental iterations, mainly because of the high-redundancy and a diversity of control actions, which are yet challenging to resolve the physical optimality on a high Degree of Freedom (DoF) robot. In this regard, the learning approach is very attractive because of the significant reduction of manual effort, and the learning architecture requires only the design of input-output and rewards. This article shed some light on a new paradigm: the recent high-profile successes in DRL suggest new high-quality value in learning methods that the discovered policies can be used as a basis for speeding up the development of robotic controllers (Fig. 2). As an outcome in this push recovery study, we obtain a certifiable, analysable optimal controller that does not require any state machine or switching mechanism, while exhibiting human-like push recovery strategies, such as ankle, hip, toe, and stepping strategy all in a coherent optimisation process.

II. Generating Complex Motions for Humanoid Robots through Deep Reinforcement Learning

To use DRL-policies as a basis for analysis, these policies must reach a certain performance threshold that ideally surpasses traditional control approaches both in the types of motions it can generate and the amount of disturbances that it can withstand. DRL has been shown to be capable of learning locomotion and fall recovery policies that surpasses traditional control approaches for quadruped robots in terms of power efficiency, and versatility of motion [1].

In this section, we present a hierarchical learning framework for achieving versatile behaviours during push recovery for humanoid robots as proposed in [5]. The learned policy exhibits a wide range of balancing strategies that are comparable to human push recovery. In particular, the learned policy is able to withstand external disturbances by modulation of the Centre of Pressure (Ankle strategy), Angular Momentum (Hip strategy), Centre of Mass height (Toe strategy), and Support Polygon (Stepping strategy) and surpasses traditional control methods with respect to the disturbances that it can withstand.

All motions are trained for deployment on NASA's humanoid Valkyrie [6], a 44 DoF bipedal humanoid robot designed for operating in disaster response scenarios and extraterrestrial planetary space missions such as unmanned pre-deployments on Mars. Valkyrie is built to operate in human-

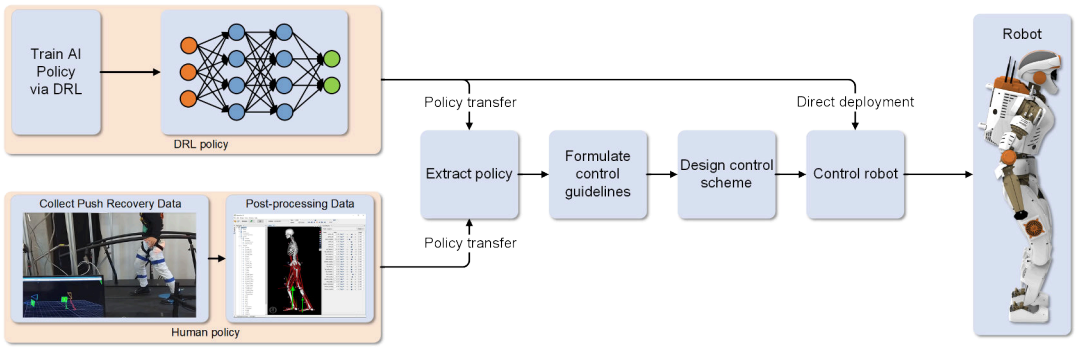


Fig. 2: *Proposed control design approach* by policy transfer from AI policies to real robotic platforms. By understanding the underlying concepts of the learned policies, we can reverse-engineer a controller that closely resembles the behaviour of the AI policy while being transparent and safe.

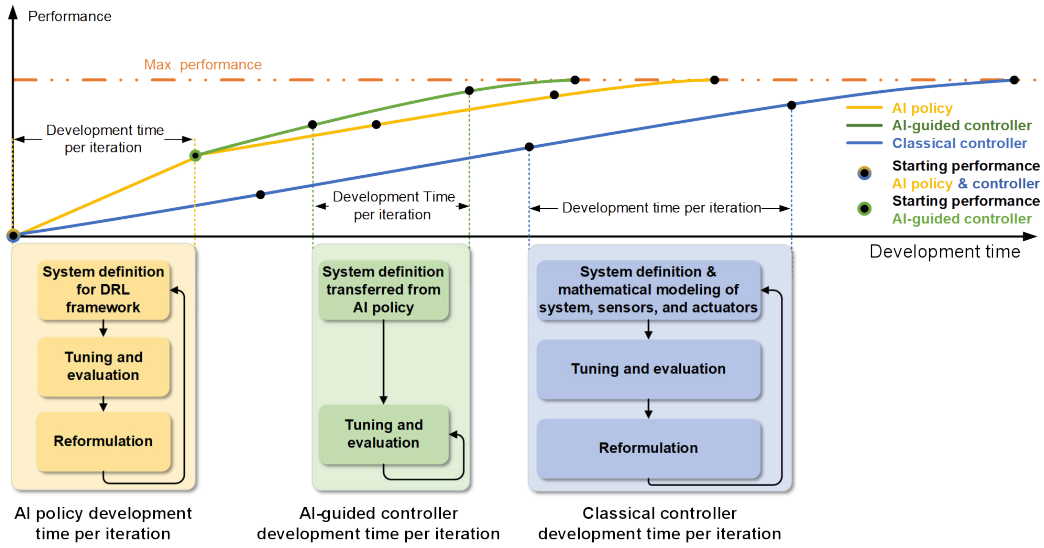


Fig. 3: Qualitative depiction of the performance of the controllers over iterations: leveraging the solution provided by an AI policy decreases the development time of controllers compared to AI policy and traditional controller development while starting at a higher initial performance. The width between every black dot indicates the relative development time of the individual control design approaches with the shortest development time per iteration for the AI-guided controller.

engineered environments standing 1.87m tall and weighing 129kg with ranges of motion similar to humans. Locomotion and manipulation of Valkyrie are enabled through 25 series-elastic actuators in arms, torso, and legs; their respective joint limits are described in Table I. For sensing, Valkyrie has proprioceptive and exteroceptive sensors consisting of a multitude of gyroscopes, accelerometers, load cells, pressure sensors, sonar, LIDAR, depth cameras, and stereo sensors.

For training the policy, the physics simulator PyBullet [7] was used, which is able to simulate physics faster than real-time expediting the training process, and simulates collisions, and soft and rigid dynamics by loading objects defined in the Unified Robot Description Format (URDF). The Valkyrie URDF model closely replicates the real robot and is provided by NASA including physical quantities, such as inertia, mass distribution, sensor noise, friction, and damping [6].

A. *Learning Framework*

Our DRL framework (Fig. 4) has an actor-critic architecture, in which both the actor $\pi_\theta(a|s)$ and the critic $V_\phi(s)$ are Neural Networks parametrised by the weights θ and ϕ respectively.

The AI policy $\pi_\theta(a|s)$ is trained through a policy gradient method called Trust-Region Policy Optimization (TRPO) [8]. Policy gradient methods directly model and optimise the policy that will yield the highest reward $J(\theta)$:

$$J(\theta) = \sum_{s\in\mathcal{S}} d^\pi(s)V^\pi(s) = \sum_{s\in\mathcal{S}} d^\pi(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s)Q^\pi(s,a), \tag{1}$$

with stationary distribution $d^\pi(s)$, state value function $V^\pi(s)$, and action value function $Q^\pi(s,a)$ under policy π_θ. Using the gradient $\nabla_\theta J(\theta)$, gradient ascent is used to find an optimal parameter set θ that maximises the reward.

Because the true value functions $V^\pi(s), Q^\pi(s,a)$ are not known, a critic $V_\phi(s)$ estimating the true value function

TABLE I: Mechanical specifications of Valkyrie.

	Arm				Torso			Leg					
	Shoulder roll	Shoulder pitch	Shoulder yaw	Elbow pitch	Torso roll	Torso pitch	Torso yaw	Hip Roll	Hip pitch	Hip yaw	Knee pitch	Ankle Pitch	Ankle Roll
Lower joint position $[rad]$	-1.3	-2.9	-3.1	-2.2	-0.2	-0.1	-1.3	-0.6	-2.4	-0.4	-0.1	-0.9	-0.4
Upper joint position $[rad]$	1.5	2.0	2.2	0.1	0.3	0.7	1.2	0.5	1.6	1.1	2.1	0.7	0.4
Joint velocity $[rad/s]$	5.9	5.9	11.6	11.5	9.0	9.0	5.9	7.0	6.1	5.9	6.1	11.0	11.0
Joint torque $[Nm]$	190	190	65	65	150	150	190	350	350	190	350	205	205

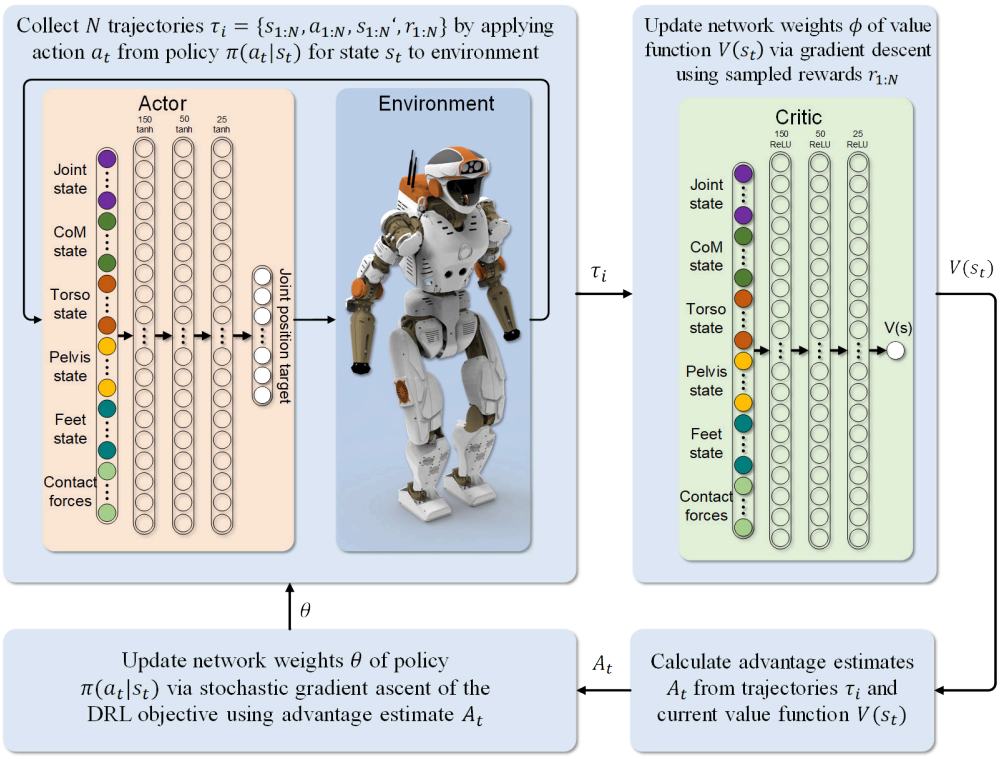


Fig. 4: Learning framework for Deep Reinforcement Learning of a Push Recovery policy.

$V^\pi(s)$ will be trained. The critic $V_\phi(s)$ is updated by minimising the expected loss $L_V(\phi)$ via gradient descent:

$$\min_\phi L_V(\phi) = \min_\phi \mathbb{E}[(V_\phi(s_t) - G_t)^2], \tag{2}$$

with value function estimation $V_\phi(s_t)$ and return G_t. The return $G_t = \sum_{k=t}^{\infty} \gamma^{k-t} r_k$ is the discounted cumulative reward using the discount factor $\gamma \leq 1$ and reward r_t.

Using the value function estimation $V_\phi(s_t)$, the actor's parameters θ are updated by maximising the reward (1) via stochastic gradient ascent. Additionally, TRPO improves the training stability through trust-region optimisation - constraining the KL divergence during a policy update preventing large policy changes that could cause instability - and reduces the variance of the policy gradient estimate using an advantage estimation A_t:

$$\begin{aligned} \max_\theta J(\theta) &= \max_\theta \mathbb{E}[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A_t(s_t)] \\ \text{subject to} \quad & \mathbb{E}[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s)||\pi_\theta(\cdot|s))] \leq \delta, \end{aligned} \tag{3}$$

with updated policy $\pi_\theta(a|s)$, old policy $\pi_\theta(a|s)$, advantage estimate $A_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l (r_t + V(s_{t+1}) - V(s_t))$, discount factor γ, variance/bias trade-off parameter $\lambda \in [0, 1]$, KL divergence $D_{\text{KL}}(\cdot||\cdot)$, and trust region δ.

During the training process, by sampling the action a_t from the stochastic policy $\pi_\theta(a_t|s_t)$ and performing this action, the environment returns the resulting reward r_t, and the corresponding state s_{t+1}. During every update step during training, so-called SARS tuples $\{s_t, a_t, r_t, s_{t+1}\}$ are collected in a batch $\mathcal{D}$. Upon reaching a certain batch size N - the size of the batch is a hyper-parameter trading off the variance of the data against the speed of training - the actor and critic are then updated through gradient ascent and descent respectively. While a stochastic policy provides exploration, additionally, by applying random forces to the pelvis, the policy's ability to withstand large pushes is increased due to a larger coverage of the state space in the stored experience data. With the additional push experience during training, the policy will be able to generalise better to push disturbances during runtime.

B. Inferring actions from the AI policy

By training in the presented learning framework and interacting with the environment, the AI policy experiences

what reward certain actions in different states will yield. Using these experiences in form of SARS tuple trajectories $\tau = \{s_{1:N}, a_{1:N}, r_{1:N}, s'_{1:N}\}$ and reward maximisation through gradient ascent, the policy is incentivised to perform actions that will lead to high-value states while avoiding low-value states. Consequently the AI policy $\pi(a|s)$ learns how to optimally act in state s_t by performing actions a_t to maximise a human-defined reward function r.

The reward function is designed as:

$$r = r_{\text{pose}} + r_{\text{CoM pos}} + r_{\text{CoM vel}} + r_{\text{GRF}} + r_{\text{contact}} + r_{\text{power}}. \quad (4)$$

The reward function is designed such that high rewards are given if torso pose r_{pose}, Centre of Mass (CoM) position $r_{\text{CoM pos}}$, CoM velocity $r_{\text{CoM vel}}$, and ground contact force r_{GRF} (equally distributed ground reaction forces) remain close to the nominal values. A radial basis function $r_i = \exp(-\alpha_i(x_{\text{target}} - x)^2)$ is used for torso pose, CoM position, CoM velocity, and ground contact force to encourage the robot being close to a target position. Furthermore, a penalty (negative value) is given proportional to the power consumption r_{power}, and if the upper body is in contact with ground or the foot is not in contact with the ground r_{contact}. For the contact penalty, a constant value is subtracted if there was no ground contact of the foot or an upper body contact with the ground.

The high-level AI policy generates actions in form of target joint angle references at a frequency of $25Hz$ by forward-propagating through the actor network using the current state as input. The robot performs its motions with upper body joints locked in their nominal position. The action space $\mathcal{A} \in \mathbb{R}^{11}$ thus consists of joint positions for torso pitch, hip roll, hip pitch, knee pitch, ankle roll, and ankle pitch. The target joint angles are given to a low-level Proportional Derivative (PD) controller operating at 500Hz to generate the joint torques that are ultimately applied to control the robot (Top of Fig. 6).

The state space $\mathcal{S} \in \mathbb{R}^{47}$ consists of joint position and velocity of the actuated joints, pelvis states (translational and angular velocity, orientation), CoM states (translational velocity and position in local frame), ground contact force, torso position (in local frame), and foot position (in local frame). The state is sampled at a frequency of 500Hz and filtered by a first-order Butterworth filter with a cut-off frequency of 10Hz.

More details regarding the formulations of the learning framework can be found in [5].

C. Behaviours of the AI Policy

Training a robust push recovery policy in the presented learning framework requires 6-8 hours of simulation time on a commercial desktop PC (Intel i7 6700K, Nvidia Titan X, Tensorflow) and equates to 1-2 days in real time. The learned push recovery policy demonstrates human-like push recovery strategies such as ankle, hip, toe, and stepping strategy which emerge at different levels of disturbance (Fig. 1). The policy performs well in the presence of external disturbances and large sensor noises due to the filtering and sufficient amount of exploration.

Notably, the AI is able to generalise to external disturbances it was not trained for. In Figure 5 snapshots of Valkyrie are shown during which an impulse push on the shin is performed. The policy is able to generalise to this untrained case by generating a stepping behaviour. This generalisation ability further extends to the ability of recovering from an impact during landing from 0.55m height.

The performance of the policy learned by the AI is compared with push recovery controllers in terms of maximum impulse disturbance. The AI demonstrates the ability, comparable to state-of-the-art push recovery controllers, to withstand a wide range of impulse disturbances. The details of the comparison are presented in [5]. In Section IV we will further show that the motions are human comparable both quantitatively and qualitatively.

III. Understanding and Learning the Fundamental Principles From the AI-Policy

The idea of using AI-generated policies as inspiration for solving hard problems has taken flight in other fields, most notably in games such as Go, Chess and Shogi [9] for which the AI policy achieved super-human level performance. Analysing the concepts and solution process of the AI allows humans to construct better solutions and improve the way humans approach these problems: AlphaGo, a super-human policy that beat 18-time world champion Lee Seedol 4:1, has subsequently been analysed to provide humans with insights on why and how AlphaGo won. The release of AlphaGo Teach enabled Go-players to analyse strategies, and changed the way in which humans played the game: AlphaGo Teach helped to quantify the value of starting the game (suggesting that the current compensation of 6.5 points for not starting puts the non-starting player in an advantage), redefined conventional openings, and utilised unconventional moves that went against conventional human wisdom and were previously deemed as disadvantageous.

Inspired by the performance of the AI policy, this section aims to present an approach to reverse-engineer the policy to obtain a unified controller exhibiting the same push recovery strategies as the AI policy. By reverse-engineering the AI policy, its versatility is maintained while enabling us to analyse and modify the controller with respect to stability and safety.

We aim to find a controller which is able to stabilise the system, and has a close similarity and fit to the DRL policy. To this end, a choice of system model and the controller type need to be made. In the following, we will show that the DRL policy can be accurately replicated with point-mass dynamics that is controlled by a minimum jerk controller. This indicates that the NN may internally use a point-mass template and try to optimise the jerk.

The design process (Fig. 6) involves three steps: acquiring state-action data from the AI policy, reverse-engineering the policy, and deploying the controller in a whole-body

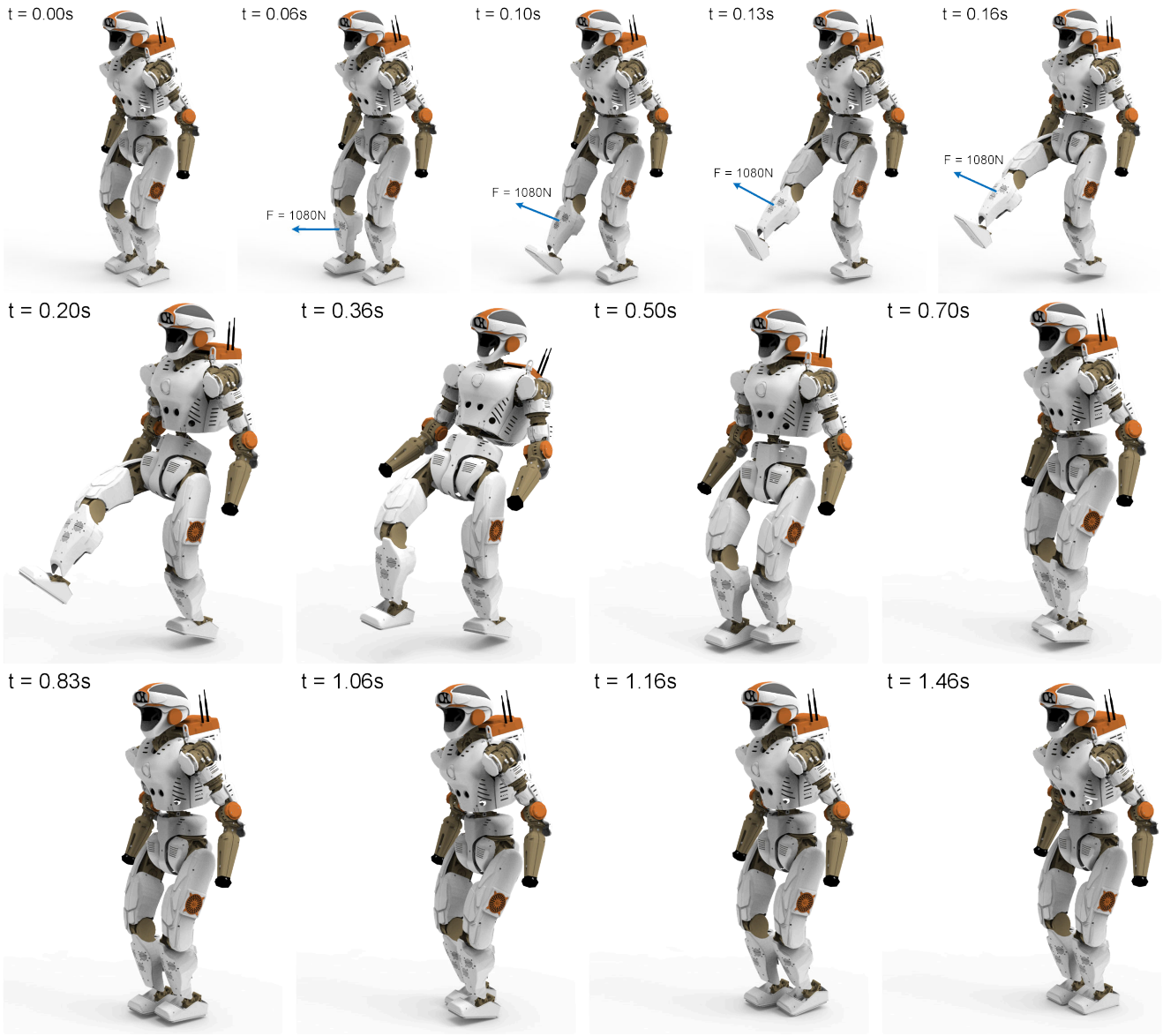


Fig. 5: Valkyrie recovering from an unexpected disturbance in a test scenario which is never encountered during training (impulse at the shin of 108Ns). The learned policy naturally evolves and generates a stepping behaviour. Top row: a nominal starting pose (left), and the motion during the disturbance (being pulled at the shin); two bottom rows: recovery reactions where Valkyrie takes 3 steps backwards and successfully recovers balance.

control framework. First, various state-action pairs are acquired through simulation for different external disturbances. In the reverse-engineering process, based on the criterion of least square error fit, both a suitable system dynamics representation and a controller using these system dynamics will be determined. Lastly, in the deployment phase, the engineered controller will be used to generate step locations and the CoM trajectory which will then be tracked by a whole-body controller.

A. Analysing the AI-policy

The Actor Neural Network can be analysed thoroughly with the goal of finding guidelines and quantities that enable general push recovery. The insights gathered from analysing the actor can be used to improve controller design. Methods for analysing and interpreting the NN are mainly of visual nature and originate from the field of Computer Vision.

In this work, we visualise the NN's activation by using t-distributed Stochastic Neighbor Embedding (t-SNE) to project the activation of NN's onto a 2D plane while preserving neighbourhoods and clusters in projections [10]. T-SNE is performed on the neuron activations to project the high-dimensional NN activation on a 2-dimensional space in order to investigate the generalisation behaviour of the policy. In the final projection, data points that represent similar NN activation are grouped nearby with high probability, whereas non-similar data points are distant from each other. We treat all neuron activations (c.f. Fig. 4) during one time-step as

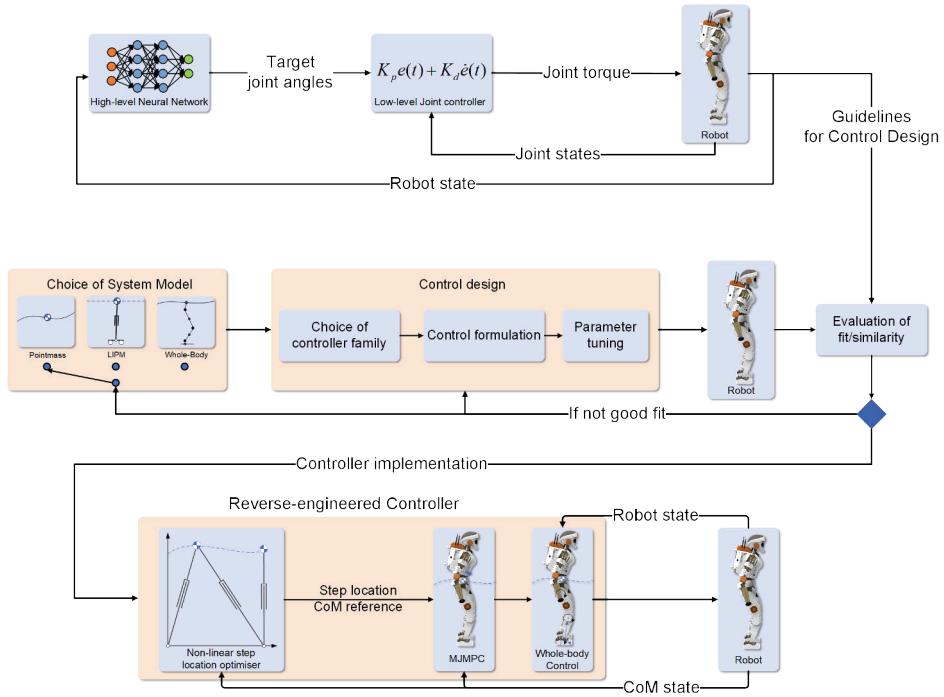


Fig. 6: Process for AI-aided control design. Top: Hierarchical control system for push recovery. The high-level neural network is learned as depicted in Fig. 4. Mid: Design process of the engineered controller. Bottom: Implementation of the designed controller into a whole-body control framework.

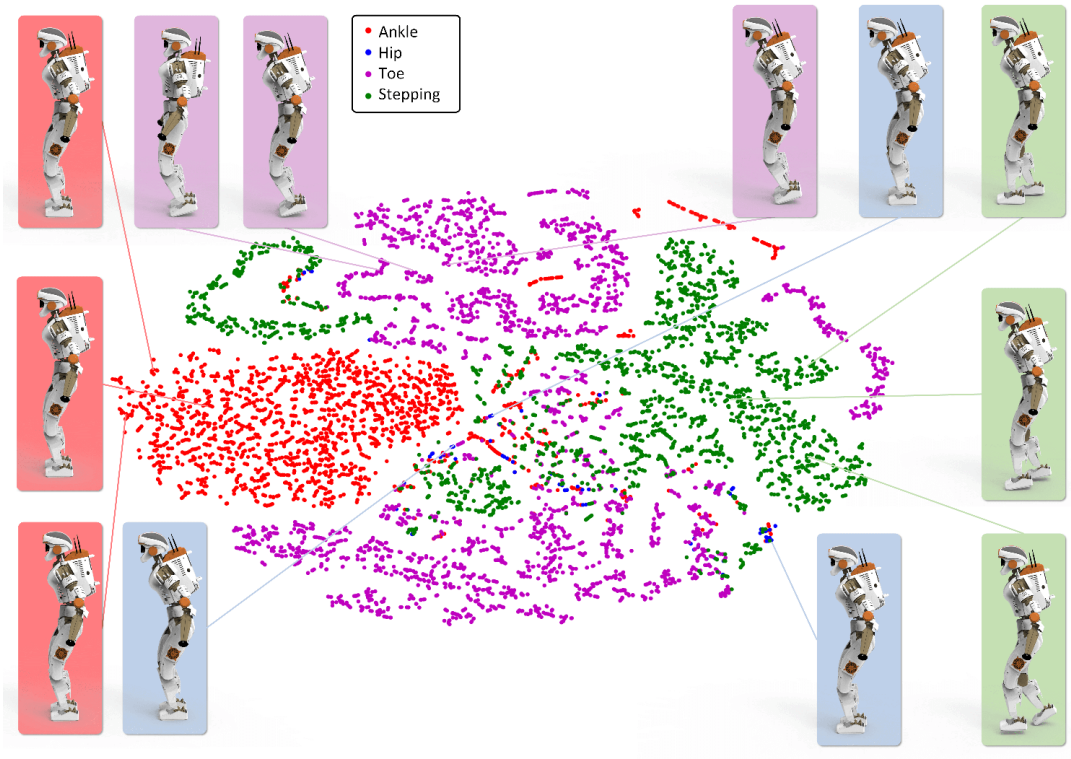


Fig. 7: T-SNE with every dot indicating the activation of all 175 neurons during one time step. The classification of every dot is determined by whether they are non-zero (exceed a threshold) for foot velocity (step), angular momentum (hip), vertical CoM velocity (toe), and is Ankle Strategy otherwise.

one high-dimensional data point for the t-SNE analysis. For every time step of a trial we collect all neuron activations across disturbances ranging from $0\text{Ns} - 210\text{Ns}$. During these disturbances, all four push recovery strategies (Fig. 1) occurred and are labelled by colour in Fig. 7.

As can be seen in Figure 7, the distinct strategies are cover separate regions and thus partly explain the ability of the AI policy to generalise across different disturbances, as the activation is similar for the same strategy. NN representations that are labelled as the same strategy are perceptually similar and are projected close to each other. The points that represent ankle strategy cluster well with few outliers and is visually distinct from other strategies. The toe strategy and stepping strategy are also quite distinct from each other. Furthermore, the lack of activations corresponding to the hip strategy (blue points in Fig. 7) indicate that the AI policy does not utilise the Angular Momentum as much as the other strategies and could suggest that the hip strategy should be used less than other strategies, such as the toe strategy. The hip strategy does not cluster as clearly as the other strategies, thus we hypothesise that it is an artefact of other motions rather than an intentional motion. The effectiveness of the hip strategy is further questioned in the literature [11], and should be kept in mind when reverse-engineering the AI policy.

B. Incorporating AI-policy inspired principles in the control design

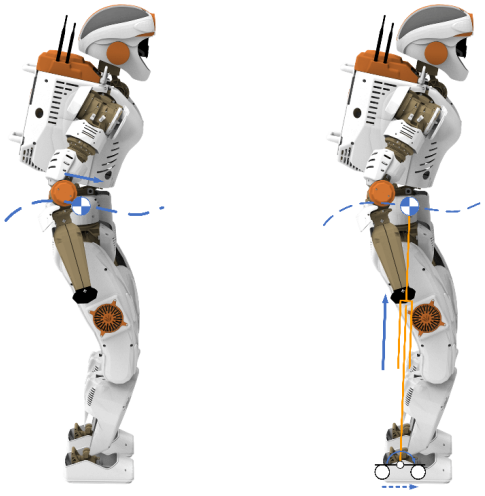


Fig. 8: System models used for control design. Left: Point-mass model with free moving CoM independent on actuation. This model is used for our proposed Minimum Jerk Model-Predictive Controller. Right: Inverted-Pendulum model with actuated ankle torque and kick-force and free moving CoM according to the Inverted Pendulum motions. This model is used to validate and verify that the controller generated a feasible and implementable trajectory.

The close resemblance between the DRL (dashed lines in Fig. 10) and human data (dashed lines in Fig. 13), for which strong evidence of being minimum jerk exists [4], [12], along with the typical smoothness characteristics of minimum jerk trajectories motivates the design of a minimum jerk control scheme. Inspired by this, reverse-engineering the AI policy involved investigating whether it is also minimum jerk[1]. We designed an MPC controller that is using point-mass dynamics to minimise the jerk of the CoM state for motion generation and feedback control. In order to prevent that arbitrarily large motions are generated that violate the actuation constraints of the robot, the CoM state is further constrained in the MPC scheme.

The idea of MPC lies in solving an optimisation problem at every time-step. By using the feedback of the current state, a closed-loop control behaviour can be achieved. The objective function $J = \frac{1}{2}\int_0^{t_f}\left(\frac{\mathrm{d}^3x(t)}{\mathrm{d}t^3}\right)^2 \mathrm{d}t = \frac{1}{2}\int_0^{t_f} u(t)^2 \mathrm{d}t$ is designed to minimise jerk $\dddot{x}$ (the input u of the system) with final time t_f. The MPC solves the following constrained optimisation problem:

$$
\begin{aligned}
\min_{u(t)} \quad & \frac{1}{2}\int_0^{t_f} u(t)^2 \mathrm{d}t \\
\text{subject to} \quad & \frac{\mathrm{d}^3x(t)}{\mathrm{d}t^3} = u \\
& [x(0), \dot{x}(0), \ddot{x}(0)] = [x_0, \dot{x}_0, \ddot{x}_0] \\
& [x(t_f), \dot{x}(t_f), \ddot{x}(t_f)] = [x_f, \dot{x}_f, \ddot{x}_f] \\
& [x_{\min}, \dot{x}_{\min}, \ddot{x}_{\min}] \leq [x, \dot{x}, \ddot{x}] \leq [x_{\max}, \dot{x}_{\max}, \ddot{x}_{\max}], \qquad (5)
\end{aligned}
$$

with initial condition $[x_0, \dot{x}_0, \ddot{x}_0]$, and terminal condition $[x_f, \dot{x}_f, \ddot{x}_f]$. The upper and lower inequality constraints of the optimisation problem prevent the Minimum Jerk Model-Predictive Control (MJMPC) scheme outputting trajectories that the subsequent whole-body controller cannot track. We constrain the maximum vertical displacement, velocity, and acceleration of the CoM, as well as the maximum jerk the point mass can be exposed to. For the final state two quantities need to be determined: the final time t_f at which the system should reach state $\boldsymbol{x}_f$, and the final CoM state itself which is provided by non-linear step optimiser [13] and thus evokes stepping behaviour if the push gets too large.

1) Determining final time via data fitting: While the final time t_f, at which the robot should come to a rest, could also be optimised, this would introduce non-linearity to the optimisation problem. Thus, we treat the final time t_f as an open parameter: too short and it violates the physical capabilities, too long and it may get unstable or use too much energy. We determine an appropriate value for t_f via least square error fitting between collected DRL data and the MJMPC. In Figure 9 it can be seen that both a piece-wise linear and a quadratic approximation are able to fit the data.

[1]For the control design process of reverse-engineering the AI policy, we followed the processes as in Fig. 5. Our study eventually found that applying the controller proposed in our previous work [12] leads to the best fitting results, and strong resemblance between the AI policy and the reconstructed controller. In contrast, the further tested different models (point-mass, Inverted Pendulum, Whole-Body Model), and controllers (PD control, Linear Quadratic Regulators, and MPC) have worse fitting to the data.

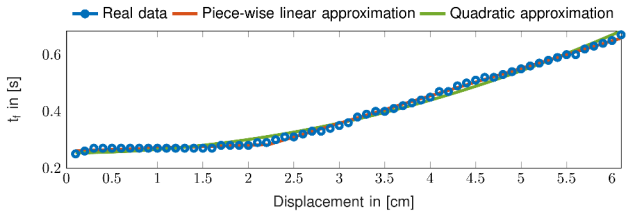


Fig. 9: CoM displacement from nominal pose over t_f for the robot.

TABLE II: Average Coefficient of Determination (R^2) over 2000 trials between DRL and robot trajectories for Ankle, Hip, Toe, and Step Strategy.

	Mean R^2				
Axis	Ankle	Hip	Toe	Step	Total
X	0.97	0.94	0.93	0.96	0.95
Z	0.90	0.91	0.96	0.88	0.91

2) Non-linear Step Optimisation: In an outer loop, a non-linear step optimiser [13] provides the MJMPC scheme with a reference point. The position x_f in the final state $\boldsymbol{x}_f = [x_f, 0, 0]$ is determined by the non-linear step location optimiser, and the velocity $\dot{x}_f$ and acceleration $\ddot{x}_f$ are set to zero. The step optimiser is able find a step location and timing within 3ms due to the specific formulation of the optimisation problem. Consequently, we are able to run the optimiser at real time at 100Hz. Furthermore, the non-linear optimiser considers the kinematic constraints of the robot to determine the maximal step length, while also constraining the minimal and maximal step velocity of the robot.

3) Emerging Strategies and Quality of Fit: From the x and z CoM position trajectories in Figure 10 it can be seen that MJMPC generates trajectories from which the push recovery strategies (Fig. 1) emerge naturally. From the data of the DRL policy we found that very little Angular Momentum was generated (Section III-A), and consequently do not regulate the angular momentum of the humanoid. Typical for the ankle strategy (blue solid line), the Centre of Pressure (CoP) is moved within the Support Polygon to move the CoM position. The CoM height modulation emerges from regulating the CoM height to its nominal position. For large pushes, the step optimiser sets a new reference position, and thus stepping behaviour emerges.

To show the quality of fit, we identify the open parameters by optimising these via least square error between the controller and the DRL policy. We apply k-fold cross validation across 2000 trials of the robot, and show the mean and standard deviation in Table II with an average coefficient of determination of 0.95 and 0.91 for x and z direction respectively.

As can be seen in Figure 10, by using the methodology described in Section III-B.1, a final time t_f can be chosen such that the engineered policy fits the DRL, indicating that MJMPC is a suitable controller for resembling the AI policy. The fit for the vertical component is slightly worse due to the fact, that the approximation of t_f for the CoM height was not as well as for the sagittal component.

C. Realisability of MJMPC on Real Systems

In the following, we propose a framework for real world deployment of the generated push recovery motions and demonstrate the feasibility of the motions generated from MJMPC.

In order to deploy the push recovery motions on a real robot while guaranteeing stability and implementability, a whole-body controller is included in the control framework (Fig. 11). To this end, Quadratic Programming (QP) based whole-body controllers can be leveraged to track reference motions while providing stability and were successfully deployed on humanoids such as Valkyrie [6], Atlas [14], and HRP-2 [15]. Beside the ability to incorporate stability and feasibility guaranteeing constraints in the optimisation problem formulation, QP problems can be solved extremely fast with off the shelf QP solvers enabling loop closure at over $500Hz$. Furthermore, in practise, whole-body QP controllers exhibit robustness to model uncertainties due to fast frequent loop closure, and have been shown to reliably work on the real Valkyrie platform.

After the MJMPC generates a reference CoM trajectory y_{ref}, a whole-body controller is minimising the tracking error $J_{\text{task}} = \frac{1}{2}\|y - y_{\text{ref}}\|$ while guaranteeing physically feasible torques realising the push recovery motion. Reformulation of the tracking tasks of $J_{\text{task}} = \frac{1}{2}\|y - y_{\text{ref}}\|^2 = \frac{1}{2}\|AX - b\|^2$ with state $X = [\ddot{q}, \tau, \lambda]^T$ consisting of torque commands τ, joint accelerations $\ddot{q}$ and contact wrench λ yields the matrices $H = A^T A, f = -A^T b$. The QP problem is formulated as:

$$\begin{aligned} &\min_X X^T H X + f^T X \\ &s.t. \quad A_{eq} X + B_{eq} = 0 \\ &\quad A_{ineq} X + B_{ineq} \geq 0. \end{aligned} \tag{6}$$

Further tasks, e.g., tracking feet trajectories from the non-linear step location optimiser, regularisation terms, or other tasks benefiting the stability of the controller, can be stacked in $A = [w_0 A_0, w_1 A_1, ..., w_n A_n]^T, b = [w_0 b_0, w_1 b_1, ..., w_n b_n]^T$ and task priorities are represented by the weights w_i, where $i = 0, ..., n$ is the i-th task (for details, c.f., [14]).

The equations of motion guaranteeing physical coherence between the states form the equality constraints of the QP problem (6):

$$\begin{bmatrix} M(q) & -S & -J^T(q) \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \tau \\ \lambda \end{bmatrix} + h(q, \dot{q}) = 0, \tag{7}$$

with inertia matrix $M(q)$, selection matrix S, Jacobian matrices $J^T(q)$ of the contact links, and nonlinear effects $h(q, \dot{q})$.

In addition to physical coherence of the solution, locomotion specific constraints are formulated guaranteeing that the robot will not fall over. This is achieved by imposing inequality constraints in the QP problem on the contact

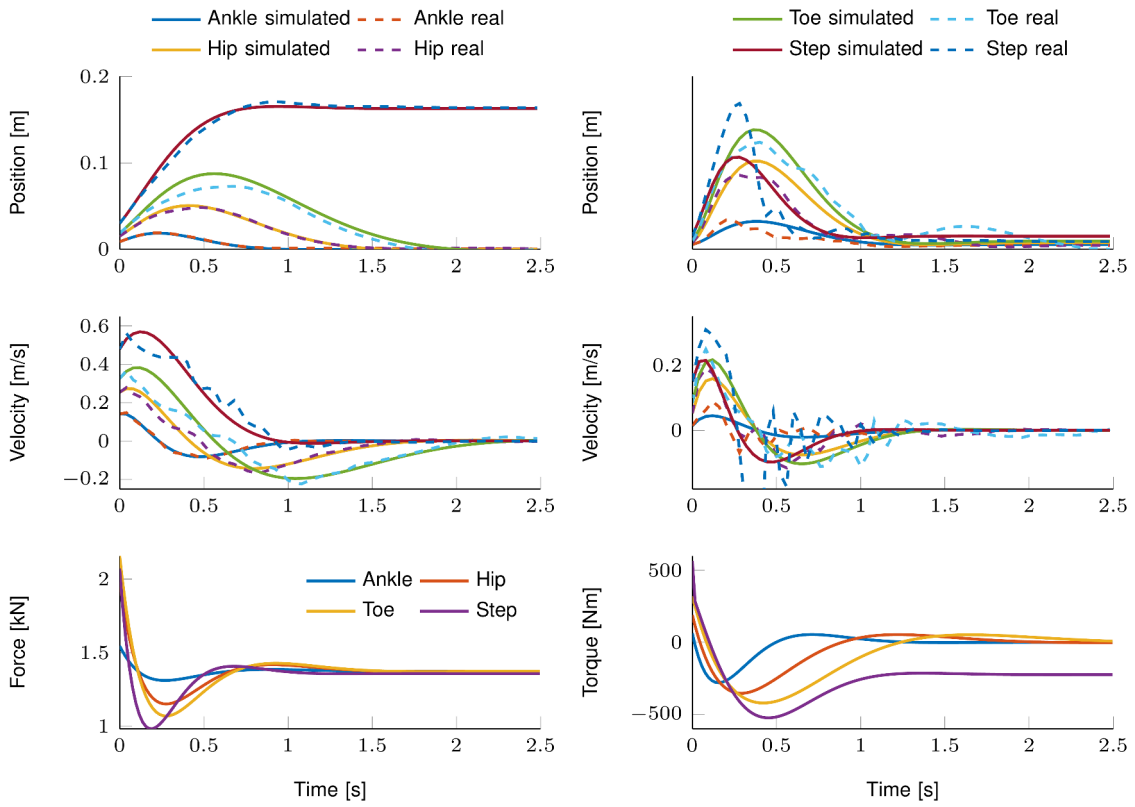


Fig. 10: Fit between robot data and reverse-engineered controller. Left column: x position, velocity, and required force. Right column: z position, velocity, and required torque

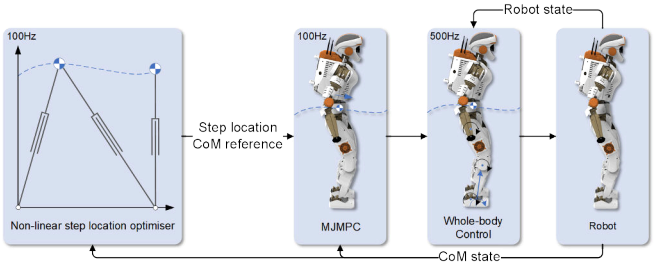


Fig. 11: Control diagram for the robot.

wrench $\lambda = [f_x, f_y, f_z, \tau_x, \tau_y, \tau_z]$ between feet and the ground. Slippage in x, y direction is prevented by constraining the force in the respective direction $|f_x| \leq \mu f_z, |f_y| \leq \mu f_z$ for a given friction coefficient μ. Unilateral forces (no suction of the feet to the ground) are guaranteed by $f_z > 0$. A stability constraint is achieved by constraining the CoP within the Support Polygon $|\tau_x| \leq \mathcal{Y} f_z, |\tau_y| \leq \mathcal{X} f_z$ with dimensions $\mathcal{X}, \mathcal{Y}$ of the Support Polygon. Lastly yaw slippage is prevented by $\tau_{min} \leq \tau_z \leq \tau_{max}$ with $\tau_{min} = -\mu(X+Y)f_z + |\mathcal{Y} f_x - \mu\tau_x| + |\mathcal{X} f_y - \mu\tau_y|, \tau_{max} = +\mu(X+Y)f_z - |\mathcal{Y} f_x + \mu\tau_x| - |\mathcal{X} f_y + \mu\tau_y|$.

The physical feasibility of the motions on the actuators during push recovery motions can be further shown using an Inverted Pendulum Model. The Inverted Pendulum (IP) Model [16] simplifies the dynamics of a robot into an IP whose length can be extended via a "kick-force" and a torque applied to the pivot point corresponding to the ground reaction force and torque of the real robot. Using the torque limits of the actuators, the robot can produce a ground contact force of $f_{max} = 2500\text{N}$ and torque of $\tau_{max} = 1000\text{Nm}$. Using the IP model, the required ground contact forces and torques for tracking the CoM reference can be calculated and are shown in Figure 10. For all four push recovery scenarios, the required force and torques are well below the maximal generatable ground contact force and torque of $f_{max} = 2500\text{N}$ and $\tau_{max} = 1000\text{Nm}$ respectively.

Summarised, feasible and stable motions are enabled using this proposed control framework of first generating push recovery motions via MJMPC and then tracking the motions with a whole-body controller. In the inequality constraint (5) of the MJMPC optimisation problem it is explicitly considered at which speed the whole-body controller can track a reference motion. Therefore, trackability of the reference motion is guaranteed. Furthermore, for the whole-body controller, the CoM Height Modulation does not necessarily need to come from a toe lift, but can come from any leg length extension increasing the CoM height. This is possible if the gait is not performed with stretched knees which is mostly the case in robot control tasks that aim to prevent singularities in a stretched knee pose.

IV. Benchmarking Between Human- and AI-policies

To this point, we have shown that methods of policy extraction can be used to estimate and reconstruct control policies from DRL. Interestingly, when these methods are applied to human movement similar strategies are revealed [12]. In this section we compare the findings from our previous work with those based on the AI policy to highlight the similarities between policies extracted from humans and DRL. This will show that even though training in DRL can take just hours, the emergent behaviours are similar to human behaviours which are highly successful only after being refined over months. These similarities show that DRL can be a high quality source of inspiration for control design since it can closely match human level behaviour.

A. Data Collection

To compare human and DRL push recovery policies, human data was recorded in 60 trials while short impulses were applied close to their CoM (Fig. 12 left), analogous to those applied to robots during DRL testing. After the push ends (Fig. 12 middle) the subject takes a recovery action (e.g., stepping in Fig. 12 right). Different impulse magnitudes were applied to obtain a wide range of push recovery behaviour. Impulses were measured by a Force/Torque sensor. Movement was measured via a VICON motion tracking system and the data was post-processed using OpenSim and gait analysis tools as presented in [17].

B. Push Recovery Strategies in Humans and AI policies

Analysis of the human data [12] shows that various strategies are used for push recovery as shown in Fig. 1. Each strategy is associated with a control action such as the modulation of (CoP), Angular Momentum, CoM Height, or Support Polygon. Interestingly, a very similar structure also emerges in the AI policy, which demonstrates the similarities between DRL and human policies.

C. Control Strategy

We find that a single controller implementing the MJMPC scheme in Section III can explain human data across all strategies (Fig. 13) as was the case for DRL. Comparing these to the DRL results (Fig. 10, Table II) and it becomes apparent that the trajectories are similar in each strategy. In conclusion, applying policy extraction methods to humans and DRL revealed that they are generalisable to different sources and shows that the respective policies are very similar, as evidenced by the MJMPC controller.

D. Why Learn From DRL Policies Instead of From Humans?

This section showed that the policies of DRL and humans and their derived, reversed-engineered controllers are similar, which raises the question: why use an AI policy instead of learning from a human policy? The answer lies in the logistics and applicability of both methods for control design. While humans could be used as template to gain insights on how to improve the target system (humanoid robots), learning from an AI policy yields three main advantages:

1. Access to all internal states: by deploying DRL on the same system as the target system, the state space is identical. Thus, data can be collected from DRL learned policies as soon as the learning process is complete and is immediately available for analysis.
   In the case of human policy analysis, data post-processing is highly labour intensive and requires expensive data collection equipment in order to infer accurate joint angle, joint torques and CoM positions [17]. Data collection required around two weeks of work, including setting up experiments, collecting subjects, carrying out experiments and processing data.
   These labour intensive logistics further motivate the use of DRL to transfer policies over the collection of human data since a lot more data can be extracted in a shorter time from DRL trials.
2. Policy transfer to non-humanoid robots: using humans as template policy only allows a policy transfer to humanoids. In contrast, DRL policies can be applied to system which are non-humanoid, such as quadrupeds or multi-legged robots, or where the template is not available, such as extinct vertebrates [18].
3. Analysis of the internal mechanisms of the policy: For analysis of the policy beyond its input-output relations, e.g., t-SNE analysis (Section III-A), the AI policy is more accessible than the human policy. The AI policy is represented as NN and analysis tools outlined in Section III-A can be leveraged to gain insights in the mechanisms of the AI policy. Analysing the mechanisms of the human policy on the other hand require a neuroscientific understanding of the brain and other involved components of the humans.

The summarised strengths and limitations of DRL in a number of different areas with respect to classical control and human control can be found in Table IV.

TABLE III: Coefficient of Determination (R^2) between human and robot trajectories for Ankle, Hip, Toe, and Step Strategy.

	Mean R^2				
Axis	Ankle	Hip	Toe-Lift	Step	Total
X	0.76	0.84	0.95	0.89	0.86
Z	0.95	0.92	0.86	0.59	0.83

V. Discussion and Conclusion

In this work, we presented an alternative application of DRL: instead of directly deploying the AI policy, we aim to formulate control guidelines from the AI policy. As a result, we bypass major drawbacks of learned policies - unsafety, and stability issues - and obtain a certifiable, optimal controller that demonstrates human-like behaviours (Fig. 1). These results were obtained for the challenging task of Push Recovery.

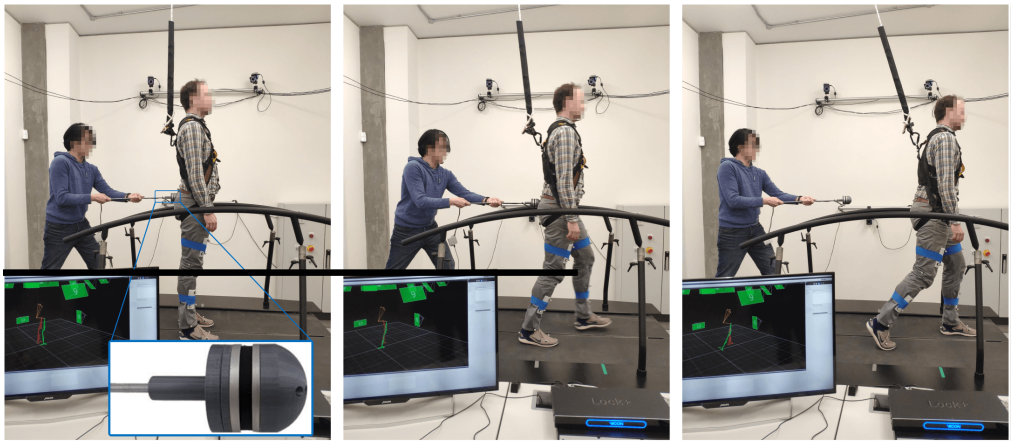


Fig. 12: Example of one experiment trial. Left: subject stands upright during the beginning of the push. Middle: subject transits into stepping strategy after being pushed. Right: subject comes to a halt after stepping action.

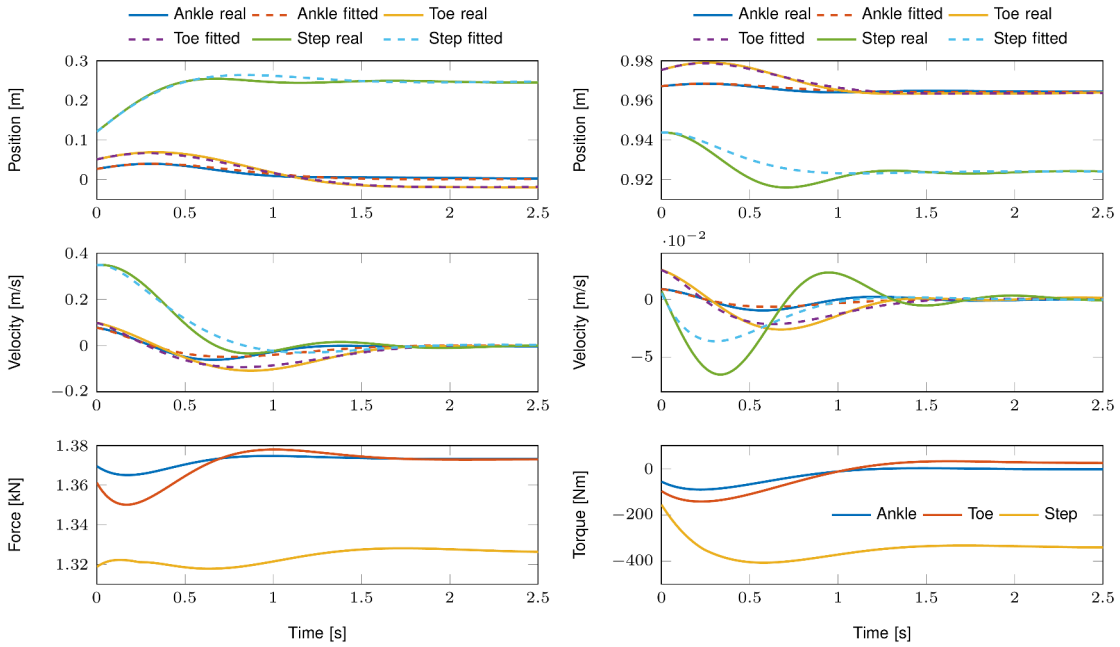


Fig. 13: Fit between human data and reverse-engineered controller. Left column: x position, velocity, and required force. Right column: z position, velocity, and required torque.

TABLE IV: Comparison between various control paradigms. Note that the limitations of DRL in optimality, robustness, and safety are cancelled out by the strengths of classical control in these areas, and vice versa. The "Control + DRL" paradigm can overcome the difficulties which are encountered in human-inspired control (last column).

Attribute	Control	DRL	Humans	Control + DRL	Control + Human
Optimality	Optimal	Sub-Optimal	Near-Optimal	Optimal	Optimal
Robustness	High	Low	High	High	High
Behaviour Emergence Time	Weeks	Weeks	Months	Weeks	Weeks
Generalisability	High	High	Low	High	Low
Data Collection Time	N/A	Low	High	Low	High
Prior Knowledge Required	High	Low	High	Low	High
Safety/Accountability	High	Low	High	High	High

A. Results

We showed that DRL is powerful enough to learn complex motions that resemble those of humans. The learned policy demonstrated the same push recovery strategies that can be observed in humans, and exhibit similar robustness as state-of-the-art control algorithms. After analysing the learned AI policy we use it as a guideline and template for control design.

The engineered controller is able to reproduce the same strategies as human and AI policies with close quantitative fit to the collected data. As observed in humans [12], the policy minimises jerk and implements feedback control via Model-Predictive Control. Furthermore, analysis of the required torques and forces on the system show that the trajectories provided by the engineered policy are realisable on the real system.

We further compared the decoded DRL and human push recovery policies, and surprisingly found that the AI policy has strong similarity with human policies. We hypothesise that both the AI and humans are able to identify the key features in the problem, as required for high quality performance. This finding is interesting due to the time for the DRL agents to acquire human-comparable push recovery abilities: learning a good AI policy requires 6-8 hours; human infants require 10-18 months to learn the ability of locomotion [19].

While MJMPC was able to reproduce both policies, and both human and AI policy can be used as template model for control design, we found the usage of AI policies for template models more advantageous. This was due to the data of DRL being immediately available - in contrast to the human data which required time-intensive post-processing -, and having direct access to the policy in form of a Neural Network allowing analysis of the Neural Network in addition to merely the input-output behaviour. We performed a t-SNE analysis on the AI policy (Fig. 7), and found that the DRL agent - through interaction with the environment - decided to not use the Angular Momentum modulation due to its little effectiveness.

B. Challenges and Outlook

In this study, we showed that through analysis of template models from human and AI policies for push recovery a certifiable safe controller for humanoid robots can be engineered. Currently, all analysis and implementations were conducted in a high-fidelity physics-simulator which, despite our best efforts, differs from reality. To mitigate this problem, we proposed a control framework for real-world deployment that combined the proposed control law with a whole-body controllers in a cascaded manner. Due to the shown feasibility of the motions and the reliable stability of whole-body controllers in real-world applications, we anticipate a seamless deployment of the controller onto a real system.

Our future work will investigate approaches in directly bridging this reality gap and use interactions with the real environment to close this reality gap in the DRL process. Furthermore, we aim to apply this principled approach of designing controllers from template models to other systems (e.g., quadrupeds) and tasks (locomotion and manipulation).

VI. Acknowledgement

This work was supported by the EPSRC CDT in Robotics and Autonomous Systems (EP/L016834/1), and EPSRC Future AI and Robotics for Space (EP/R026092/1).

References

[1] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[2] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *International conference on robotics and automation*. IEEE, 2017.

[3] M. Zhang, X. Geng, J. Bruce, K. Caluwaerts, M. Vespignani, V. SunSpiral, P. Abbeel, and S. Levine, "Deep reinforcement learning for tensegrity robot locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 634–641.

[4] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *Journal of neuroscience*, 1985.

[5] C. Yang, K. Yuan, W. Merkt, T. Komura, S. Vijayakumar, and Z. Li, "Learning whole-body motor skills for humanoids," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 270–276.

[6] N. A. Radford, P. Strawser, K. Hambuchen, J. S. Mehling, W. K. Verdeyen, A. S. Donnan, J. Holley, J. Sanchez, V. Nguyen, L. Bridgwater *et al.*, "Valkyrie: Nasa's first bipedal humanoid robot," *Journal of Field Robotics*, vol. 32, no. 3, pp. 397–419, 2015.

[7] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation in robotics, games and machine learning." [Online]. Available: https://pybullet.org/

[8] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.

[9] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[10] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[11] P. Zaytsev, W. Wolfslag, and A. Ruina, "The boundaries of walking stability: viability and controllability of simple models," *IEEE Transactions on Robotics*, 2018.

[12] C. McGreavy, K. Yuan, D. Gordon, K. Tan, W. Wolfslag, S. Vijayakumar, and Z. Li, "Unified push recovery fundamentals: Inspiration from human study," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[13] W. Hu, I. Chatzinikolaidis, K. Yuan, and Z. Li, "Comparison study of nonlinear optimization of step durations and foot placement for dynamic walking," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 433–439.

[14] S. Feng, X. Xinjilefu, W. Huang, and C. G. Atkeson, "3d walking based on online optimization," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, 2013, pp. 21–27.

[15] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014.

[16] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi, *Introduction to humanoid robotics*. Springer, 2014, vol. 101.

[17] D. Gordon, G. Henderson, and S. Vijayakumar, "Effectively quantifying the performance of lower-limb exoskeletons over a range of walking conditions," *Frontiers in Robotics and AI*, 2018.

[18] J. A. Nyakatura, K. Melo, T. Horvat, K. Karakasiliotis, V. R. Allen, A. Andikfar, E. Andrada, P. Arnold, J. Lauströer, J. R. Hutchinson *et al.*, "Reverse-engineering the locomotion of a stem amniote," *Nature*, 2019.

[19] H. Forssberg, "Ontogeny of human locomotor control i. infant stepping, supported locomotion and transition to independent locomotion," *Experimental Brain Research*, vol. 57, no. 3, pp. 480–493, 1985.