Balance Control of a Bipedal Robot Utilizing Intuitive Pattern Generators with Extended Normalized Advantage Functions

Christos Kouppas
Department of Computer Science
Loughborough University
Loughborough, United Kingdom
C.Kouppas@lboro.ac.uk

Mohamad Saada
Department of Computer Science
Loughborough University
Loughborough, United Kingdom
M.Saada@lboro.ac.uk

Qinggang Meng
Department of Computer Science
Loughborough University
Loughborough, United Kingdom
Q.Meng@lboro.ac.uk

Mark King
School of Sport, Exercise and Health Sciences
Loughborough University
Loughborough, United Kingdom
M.A.King@lboro.ac.uk

Dennis Majoe
Motion Robotics LTD
Southampton, United Kingdom
dennis.majoe@motion-robotics.co.uk

***Abstract*—Herein, a combination of Local Pattern Generators (LPG) with reinforcement learning is proposed to balance a bipedal robot using minimal power consumption. This work presents the extension of Normalised Advantage Function (eNAF) algorithm to work with recurrent neural networks without sacrificing time-dependency between data in the same episode. Additionally, a hybrid controller is introduced by combining eNAF algorithm hierarchically with LPGs to provide more robustness with less computational power requirements. The system was asynchronous, as pattern generator had an activation frequency of 100Hz, while eNAF algorithm had only 1Hz and were not synchronised between them. Robot autonomy time was increased through reduction of computational load by introducing variable-ratio activation frequency between the LPGs and the eNAF algorithm. Finally, a new and novel bipedal robot design with non-conventional linear actuators was used as the basis of the simulator model. These experiments were implemented using V-Rep Edu simulator with the industrial Vortex Studio dynamic engine. The results demonstrate a fast and agile recovery by the trained robot after a push in transverse plane.**

***Index Terms*—bipedal robot, deep reinforcement learning, CPG, NAF agent**

I. Introduction

Robust bipedal gait is one of the fundamental abilities for robots that perform dexterous work in non-uniform environments. During the last few decades, scientists sought to develop a stable and robust dynamic algorithm to stabilise walking gait, especially on uneven surfaces [1]. However, even in the current era, there are not many algorithms that can work reliably in different environments. This is mainly caused by the slow and limited adaptability during surface changes, as most approaches are using hard defined dynamics. Two of the most frequent methods of defining balance and locomotion are the trajectory-based Zero-Moment Point (ZMP) and the Centre of Mass (CoM) [2].

In the early 90s, some first steps of combining Neural Networks (NN) with bipedal robots' controllers took place but they lacked computational power to have a sufficiently successful gait cycle [3]. In the last 5 years, the computational power increased exponentially, especially in embedded systems, thus scientists started implementing new and advanced algorithms on bipedal robots [4]. However, these new algorithms mostly used neural oscillators to form a Central Pattern Generator (CPG) which in turn, generated locomotion [5].

The main disadvantage of neural oscillator is the need of high speed NN outputs to synchronise the movement of all joints at a real-time. The pattern generators in this case are actually part of the network and cannot work independently. Additionally, the robot's environment adaptability can be limited in predefined trained function due to the fact that they introduced correlation between the joints.

The approach in this research is inspired by invertebrate and vertebrate animals and humans, with the brain being a higher decision maker and each muscle having a Local Pattern Generator (LPG) to act independently [6], [7]. In nature, organisms use pattern generators to learn a specific gait and then save it in local muscles to increase reaction time and minimize brain use. Locomotion is based on different sequential gaits based on brain decisions, that acts like a higher decision maker.

The higher decision maker, of the examinated robot, was assigned to a Reinforcement Learning (RL) algorithm that utilises an extension of the Normalised Advantage Function (NAF). The extended NAF (eNAF) algorithm offers the ability to use Recurrent Neural Networks (RNN) efficiently without losing information about the previous states of the robot. The LPG were designed to work in an internally closed loop form, based on a simple, intuitive, logic that can be executed with

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

a speed of up to 10kHz. The LPGs had a set of variables that can be tuned from the eNAF agent in order to modulate the gait and, in extend, the locomotion to adapt on the new environment.

The agent was advertising the new values every $1s$ and the LPGs were collecting the values (new or old) in every cycle. For the simulation, LPG speed was limited to $100Hz$ instead of $10kHz$ to reduce computational power and increase the training of RL.

This paper makes the following contribution:

- Combine intuitive, asynchronous, white-box, pattern generators with RL to accelerate reaction time of robots.
- Extend NAF algorithm to handle RNN without losing information about the previous states.
- Enable asynchronous acting-learning as the robot can act with speed of up to 10kHz while eNAF can be much slower learning to make a prediction every 2 seconds.

II. Background

The current work began two years ago with the design of a novel bipedal robot with the aim to solve the speed and power consumption problem faced by other bipedal robots [8]. The new robot was able to use minimal power while standing still because all the actuators were able to power off and lock without consuming energy. The use of these actuators affected the control algorithms, as it removed its ability of making small movements to adjust its position. If the robot was allowed to make adjustments it meant that the actuators were working continuously, consuming energy. The balancing algorithm had a one-shot opportunity to balance the robot and any corrections would have to take place in the next movement.

A. Bipedal Robot

Current state-of-the-art humanoids are inspired by humans and that is mainly why researchers designed them to look and walk similarly to humans, despite their low efficiency [9], [10]. However, there are new designs from other universities that are focusing on the efficiency of the walking gait and are inspired by animals [11]. In works that have been published this year, it was demonstrated that the usage of NN and RL could achieve faster walking gait and better stability. These networks were able to update data from the policy framework every $30ms$ with a closed-loop PD-controller acting at a speed of 2kHz [12]. In their experiments, they had the robot requiring predictions from the RL algorithm in each action.

B. Pattern Generators

One way to accelerate the acting speed and limit it to a specific cycle is to use pattern generators. Pattern generators are known for their stability and ability to move the dynamics of the system to a cyclic activation pattern [13]. However, they are not flexible in adapting to new environments. This can be solved by modulating the activation pattern of the LPG using a more powerful algorithm that can change the parameters inside the LPG. This algorithm must be able to learn and correlate gait and locomotion between different environments. Additionally, it must be able to keep track of the previous states of the robot and how they change to understand the dynamic interaction between the robot and its environment.

C. Reinforcement Learning

Neural networks (NN) are a good candidate for modulating LPGs, as they can learn from static or temporal data and keep track of the system's previous states by using RNN. Nowadays, the increase of computational power in embedded systems is at a point that it can run NNs in real-time on robotic platforms without critically affecting power consumption. The main power draw is moved from the computational requirements to the actuation mechanics. RL was chosen, for this project, due to its ability to learn while acting, as the training process involves a real robot producing data in real-time.

The output of the RL was in a continuous action space and NAF algorithm showed major advances in respect of its learning and execution speed in comparison with other algorithms [14], [15]. Choosing RL will make CPG slow, as the robot will have to wait for the RL algorithm to make predictions. In order to overcome this, an asynchronous activation of the two systems was implemented. The NN had to make predictions every second, while the CPG had to use those predictions for multiple concurrent actions until the NN produced newer predictions.

III. Preliminaries

A. Silent Agile Robust Autonomous Host

For this research, a Silent Agile Robust Autonomous Host (S.A.R.A.H.) was designed, with the help of Motion Robotics LTD [16]. The gait of the robot was different in comparison with traditional humanoids to increase the efficiency at the cost of dexterity. The gait was inspired by ostriches, as it was proven in the early $80s$ that animals are in general more efficient than humans [17]. The first prototype of the robot can be seen in Fig. 1.

In order to achieve that efficiency, a new linear actuator [1] was created to replace the mechanical actuation of the robots' knee. The actuator was able to move with a precision of $5mm$, albeit this is not very precise but it did save energy. This is because the actuation lasted for a specific period of time and a mechanical brake held the joint in place instead of wasting energy to correct its position, as with a closed-loop feedback system. Additionally, when the motor is not powered, it can still sustain the weight of the robot without collapsing.

The real robot, as well as its simulated model, had ten Degrees of Freedom (DoF) from which only six were active. The active joints had only one DoF and they were the two rotational abductors, the two rotational hip and the two translational linear actuators. Additional, active joints had a brake mechanism that was energized only to allow a movement of the joint. That means all the actuators could keep a posture without consuming energy. The four passive DoFs were the

[1]Motion Robotics LTD holds all the rights on the details of the actuator.

Fig. 1: First Prototype of S.A.R.A.H. on a treadmill, as was presented at Birmingham in Innovate UK 2017.

joints of calcaneus and metatarsals. Those joints were pushing a pressure sensor to record information about forces that were acted on the feet. Their movement was minimal and limited by the pressure sensors to $3mm \pm 1$.

In respect to computational power, the robot had one Atmel SAMC21 microprocessor [18] for each LPG and one master microprocessor for the coordination and communication with the high-level controllers. For the high-level controllers, one Raspberry Pi [19] was assigned to the neural network predictions and one for data collection and preparation. The computational system was powered by a $12V$, $14Ah$ battery while the actuators had three of those batteries in series, resulting in $36V$. The autonomy of the robot was estimated to five hours of mix-use and over twelve hours of standby use.

B. Low-Level Controlling

The robot was divided into three main areas with one Local Pattern Generator (LPG) controlling each area, representing a muscle. Each LPG was executing a closed cyclic pattern that was moving two joints and it had the ability to inform other joints, if it was active or not. The LPGs were intuitive controllers which were designed as white-box controllers, open so the user can understand its dynamics [20]. The main pattern generator was the one controlling the linear actuator. This LPG was responsible for the lift of the legs and the activation of just those LPG could make the robot rock from one leg to the other with a specific rhythm.

Figure 3 shows the exact flow of the first LPG with the movement starting where the Left|Right feet pressure sensors get activated. If they were activated it means both feet are on the ground. Then, the variable *Factor* (1) takes information from the Inertial Measurement Unit (IMU) about the acceleration and angular velocity of the robot's trunk and calculates a number that is representative of the robot's instability. *Factor* must become greater than 0.07 in order to initiate the movement. Additionally, *Factor* is used to calculate the height of each step (2). After the movement is initiated, the first leg kicks upwards for a fraction of the required height, to move the CoM on the other leg, and then contracts to the required height. The movement stops with the return of the leg to the resting position.

$$Factor = \frac{1}{1+2^{IMU_A-5}} + \frac{1}{1+3^{5-10 \cdot IMU_G}} \quad (1)$$

$$Step = \frac{0.1}{1+5^{1-2 \cdot Factor}} \quad (2)$$

where, IMU_A: Acceleration magnitude of the trunk.
IMU_G: Rotational speed magnitude of the trunk.
$Step$: Target height for each foot (cm).

The second LPG was responsible for the movement of the hip and it was enabled only when the leg was not touching the ground. As can be seen in Fig. 2, the two variables (R_Hip, L_Hip) were calculated first and based on those variables, each leg was moving forwards|backwards with a constant speed. In this LPG, the leg was not guaranteed to return to its initial position as the actuators do not have a closed-loop controller, to reduce power consumption. If the robot cannot be supported by the new angles, the robot will have to make a new step to correct its position.

The last LPG was identical to the previous LPG but it was trained independently with its one variables. This LPG was responsible for the movement of the abductors that were able to move inwards|outwards with a specified speed, and again, the LPG does not guarantee to return the leg to the resting position. Those two LPGs were activated and stopped at the same time based on the pressure sensors.

Table I summarises the training parameters for the LPGs. The last column describes the effects of the control parameters for each LPG. With those parameters, the shape of the stable cyclic pattern can be changed based on the requirements of the environment. It is worth mentioning that the parameters for the LPG of the linear actuator were related only to the forces (acceleration) of the actuator while the step length was predefined by the *Factor* (1). The other LPGs had just one parameter to define the speed but four parameters to calculate the amount of movement of the controlled actuator.

C. High-Level Controlling

In order to increase the adaptability of the LPGs to different environments, a NN was connected hierarchically with them, creating a hybrid controller. The NN was responsible for calculating a set of parameters so that the LPG changed speed and essentially, adjusted its pattern on the new environment. The first LPG had three external parameters that were controlling the force of pushing, lifting and resting motions. The second

TABLE I: Pattern Generators' Parameters

Training session	LPG	Parameter	Actuator	Notes
1	1	P1	Linear	Force/Speed of extending leg
1	1	P2	Linear	Force/Speed of retracting leg
1	1	P3	Linear	Force/Speed of extending leg
2	2	H1-H4	Hip	Used to calculate if and which hip to move
2	2	H5	Hip	Force/Speed of moving hip
3	3	V1-V4	Abductor	Used to calculate if and which abductor to move
3	3	V5	Abductor	Force/Speed of moving abductors

and third LPGs had four parameters that were used in a quadratic equation to decide the direction and the amount of the motion and one parameter that was regulating the force/speed of the motion. On the real robot, the force can be controlled by the amount of voltage that can be applied to the actuators.

Standard RL, without NN, can utilise time-series [21] but

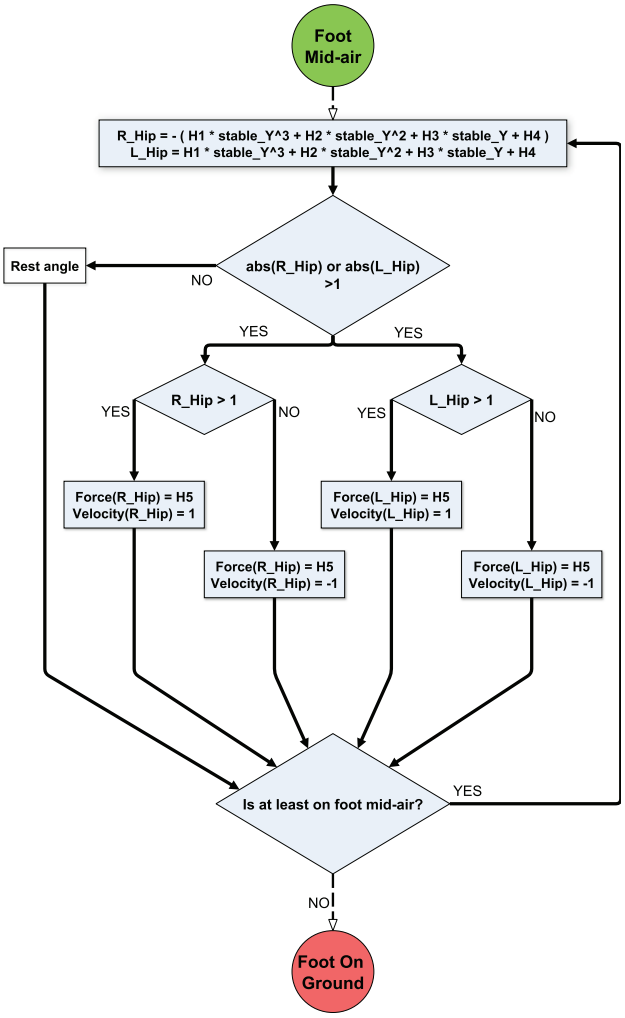


Fig. 2: LPG responsible for the movement of the hip actuators. Red and green circles are input flags from the knee LPG that start and stop this LPG, respectively.

there are not many implementations utilising time-series with NNs in RL. That is acceptable in the majority of problems, as static data at a single instant can describe the possible evolution of a scene. In problems with static data, RNN are used after extruding the features of the network and they are able to track the features' changes and not the actor's changes [22]. Additionally, RL is used mainly in predefined action space with only few algorithms acting with temporal outputs. The main two algorithms that utilise model-free robotic learning with continuous action space are Deep Deterministic Policy Gradient (DDPG) [23] and Normalised Advantage Function (NAF) [14], with NAF outperforming DDPG in real-world robotic tasks [15], [24].

On the original NAF algorithm (3-4), the training occurs after each step only once, which is suitable in situations where the algorithm will change the state of the robot in every single step. As the proposed algorithm has an asynchronous training-acting configuration, the training happens only once, every second (time set by the user) while several simulation steps occur. To improve training, a *training depth* variable was introduced such as, to retrain the network multiple times every second, based on the previous data, but act only once.

$$Q(x,u) = A(x,u) + V(x) \tag{3}$$

$$A(x,u) = -\frac{1}{2}[(u-\mu(x))^T L(x) L(x)^T (u-\mu(x))] \tag{4}$$

where, x: Observations
u: Actions
Q: Q-Learning function
A: Actor function
V: Value function (neural network)
μ: Predicted actions based on actor (neural network)
L: Lower-triangular matrix (neural network)

IV. Extending NAF Algorithm and Replay

In RL, real-time dynamic systems need special treatment in respect of the replay of memory as postural data usually are not enough to give a full picture of the current situation. In a dynamic system, the posture of the robot hides the dynamics of it, which may be catastrophic as momentum can destabilise the system but cannot be detected from postural data. The current algorithms are randomly sampling data from the memory and learn from them which is not ideal. In this work, the episodes were saved individually and data from them were used as a whole and not randomly sampled.

Figure 4 demonstrates how the data are stored for replay learning. At the end of each episode, the *Observations*, *Actions* and *Rewards* are saved individually. When training takes place, the episodes are randomly chosen and passed to the replay memory. Replay memory flattens the timesteps of all the episodes, despite each episode having different amount of timesteps. In order to utilise batch training, the length of the replay memory is trimmed to the requested *training depth*.

To reduce the computational power requirements of the robot, the NNs were making predictions every second and

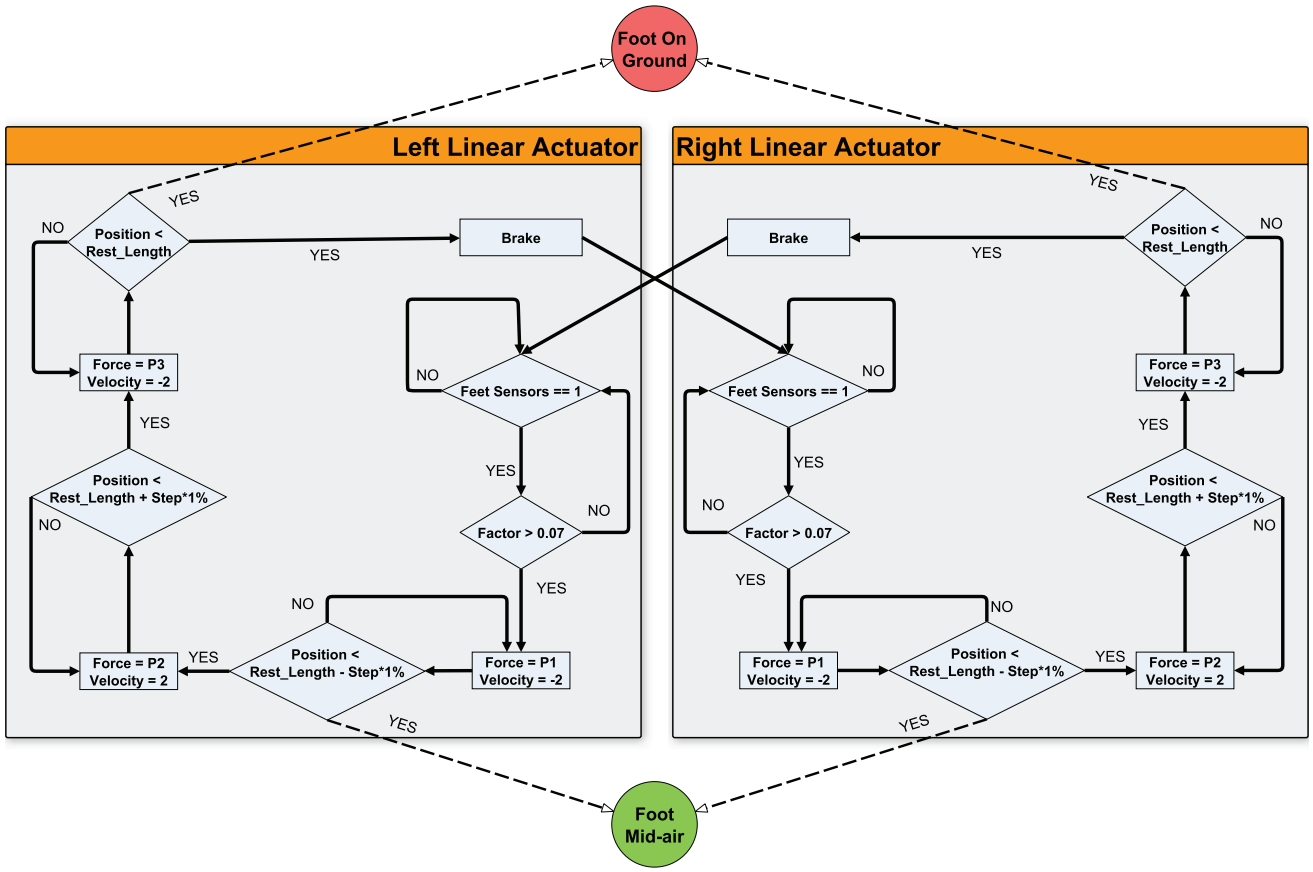


Fig. 3: LPG responsible for the movement of the linear actuators; knee. Red and green circles are output flags that start and stop the other two LPGs, respectively.

the standard NAF algorithm was able to be trained only once per second. This indicates that in a complete episode of 30 seconds, the algorithm will be trained only 30 times, which is not enough to learn the dynamics of the system. In order to increase the training cycles, a variable called *training depth* was introduced. This variable extended the replay memory and therefore caused the training to be repeated *training depth* X *times*. This reduced the amount of simulations needed to take place to learn system dynamics proportionally to the value of the *training depth* variable.

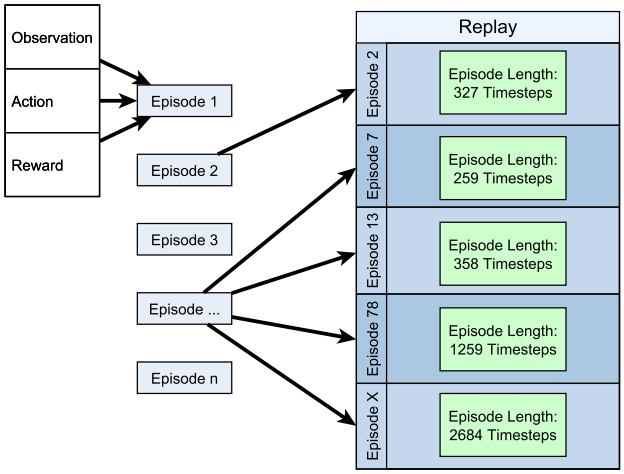


Fig. 4: Continuous time episodic replay.

Repeated training under the same epoch, with the use of a high *training depth* value, may lead to the overfitting of data and not allow the NN to get trained in a wide range of input values. To eliminate that possibility, a random value was added to the actor's actions based on OrnsteinUhlenbeck process. The process was converging the random values slowly to zero and that is ideal as the NN gets trained further needs less exploration. To make things more robust, the randomness was randomly enabled based on a binary variable at the beginning of the simulation. This guaranteed that some recorded data in the replay memory had unbiased data from the actor's NN. Double randomness is beneficial in systems that interact with the real world as they include learning data with realistically expected values that are coherent between them, without any type of noise.

V. Experiments

The robot was designed with the help of Motion Robotics LTD and the basic skeleton was imported in V-Rep Edu. The specific simulator was chosen as it was offering simulations in three-dimensional space with imported designs from computer-aided design (CAD) software. Additionally, it was offering several dynamics engines with *Vortex dynamic*

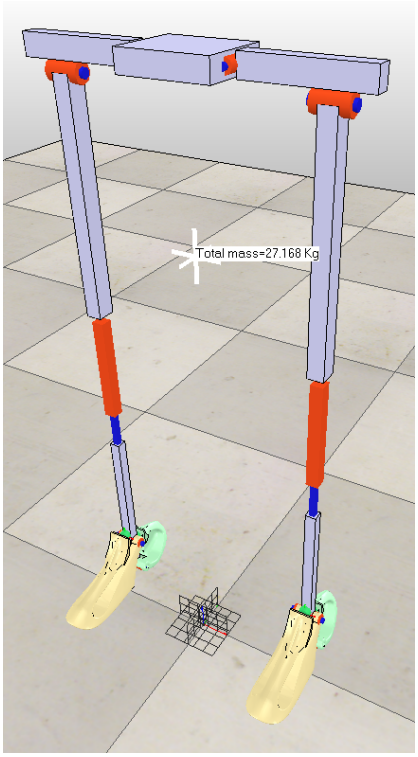


Fig. 5: Robot simulated in V-Rep Edu. White mark demonstrate the center of mass. The actuators are illustrated with red color and blue rods in the inside.

engine being the only industrial evaluated one [25]. As such, *Vortex dynamic engine* was used for the experiments. After the import of the model, the main code of the CPG was implemented using Lua scripting language inside the simulator. The independently run RL algorithm, remotely controlled the CPG through a client-server remote API. This enabled the RL algorithm to run using a different programming language (python) at a different speed to that of the CPG.

Figure 5 shows the simulated robot and its center of mass. The robot was simplified but the weight distribution was kept the same as the actual robot. Also, the modelled actuation system was similar with the robot and could stop in a particular position but could not perform micro-adjustments. That was done deliberately in order to preserve energy and push the robot to learn one-shot movements.

The experiments began by pushing the robot randomly in order to unbalance it. The main purpose of training was to keep balance and not fall while performing 3 steps per second. By the end of a training cycle, the output of the RL can be hardcoded into the LPGs, then the robot could balance in that specific training environment without the supervision of any NNs. This provided stability to the controlling loop of individual actuators in scenarios where the top NN was not ready to provide information or where it got disconnected abruptly. That is the key difference between the suggested methodology and other NN based balancing algorithms [26], [27]. Others use neuron oscillators which require values from a NN for each step that they made.

For the RL, the reward was decoupled from the input sensors and involved the position of the robot's trunk and the amount of steps per second that the robot was performing (5). Each LPG was optimised to deal with specific aspects of the reward. More specifically, the first LPG was focusing on step frequency as it was training only the activation of the linear actuators. The second LPG was changing the forwards|backwards movements in the x-axis. The last LPG was responsible for side movements in the y-axis. The maximum reward score could be achieved while the robot was not moving away from its origin and while performing 3 steps per second. The reward was not depended on the RL states as the purpose of this method is to be as open to the user as possible.

$$reward = (1-Main_x)\cdot(1-Main_y)\cdot Main_z\cdot 2^{-(Frequency-3)^2} \quad (5)$$

where, $Main_{[x|y|z]}$: Movement of robot's trunk in x|y|z axis.
$Frequency$: Steps per second.

As can be seen in Fig. 6, the linear actuator's LPG parameters were merging to specific values, which were maximizing the reward and it can be cross-referenced with the reward plot in Fig 7. In the reward plot, there are several peaks that do not relate to peaks of predicted parameters. That can be attributed to the introduction of exploration randomness, which was not recorded in Fig. 6. Additionally, the robot had asymmetric dynamics as it exhibited different behaviours when it was pushed from different sides. The reason is that the robot's foot looked like a human's foot, with the metatarsal part being longer than the calcaneus. Due to the length limitations of this paper, only the linear actuators' parameters are presented however, H and V parameters followed similar trends.

In order to evaluate the results, the trained parameters for all three LPGs were implemented in V-Rep Edu simulator manually and can be found in Table II. The robot was disturbed randomly and the disturbance of its trunk was recorded. The validation experiments started with the robot free falling from $10cm$ and stabilizing during the first second. Afterwards, a

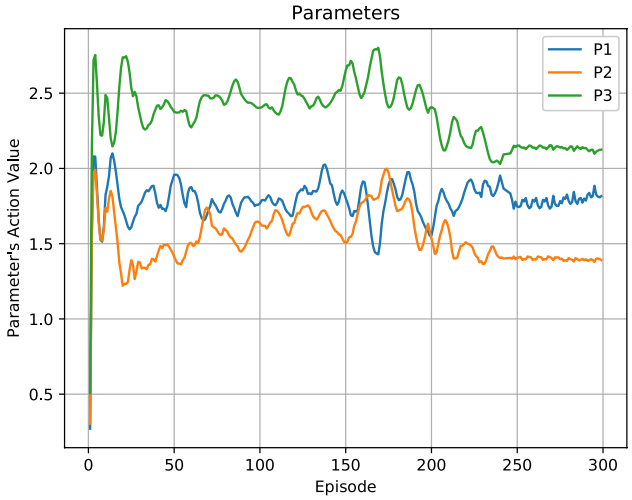


Fig. 6: Linear actuator parameters results in respect of the training episodes.

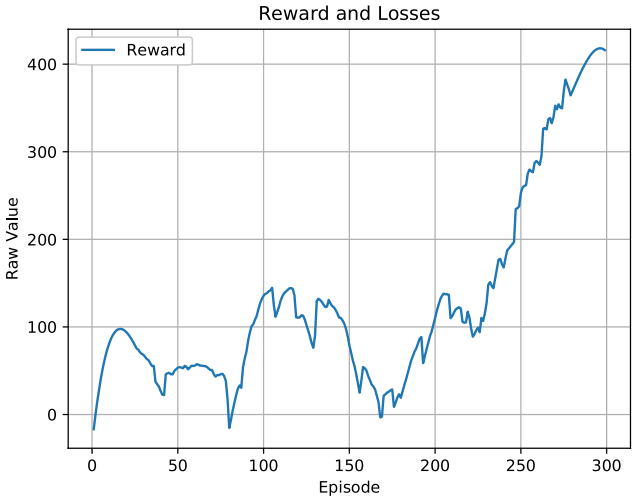


Fig. 7: Reinforcement Learning reward during training.

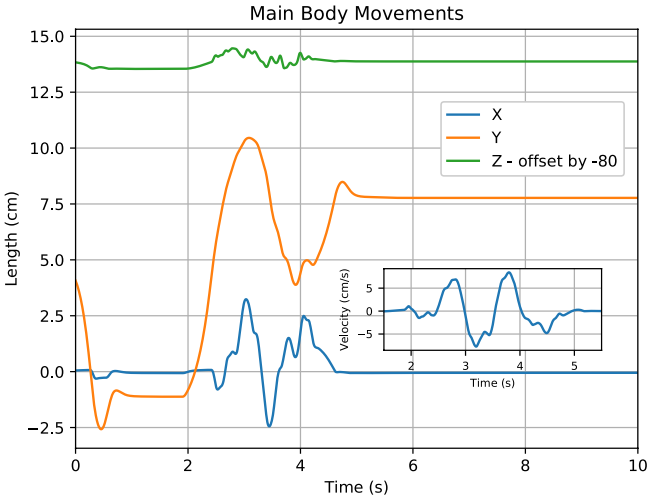


Fig. 8: Trunk Movements in 3D space.

random force was applied in the transverse plane in order to destabilise the robot and make it initialise its CPG. The robot then gained stability after making few stationary steps.

TABLE II: Pattern Generators Final Parameters

Training 1		Training 2		Training 3	
P1	1.81 ± 0.05	H1	0.05 ± 0.01	V1	0.07 ± 0.01
P2	1.45 ± 0.05	H2	1.13 ± 0.03	V2	1.06 ± 0.03
P3	2.09 ± 0.05	H3	2.31 ± 0.03	V3	2.54 ± 0.03
		H4	1.72 ± 0.05	V4	1.85 ± 0.05
		H5	1.51 ± 0.05	V5	2.77 ± 0.05

In figure 8, the movements in three-dimensional space can be observed, while the IMU data can be seen in Fig. 9. The first disturbance measurements were due to the fall $(0.5 - 1s)$ but the second disturbance measurements $(2s - 5s)$ were due to the steps that the robot took, in order to get stabilised. The robot made six steps in three seconds, as can be determined by the change in direction of the derivative of the X-axis movement in Fig. 8. Moreover, in those three seconds the robot moved only $7.5cm$ in the Y-axis. The aim of the training session was to achieve 3 steps/s with minimal movement in transverse plane. By the end of the validation episode, all the measurable variables were stabilised which shows that the robot was stabilised.

VI. Discussion

Herein, a new approach that utilises reinforcement learning for dynamic balance control is presented. Neural networks might not be robust and fast enough to solely control a robot that has to adapt to different dynamic environments. Therefore, the addition of pattern generators can guarantee that the robot will always act in a stable, closed cyclic pattern. Additionally, in real-time applications, power consumption efficiency is a very important factor that can be achieved using reduction of resources without compromising the stability of the system.

In this work, power consumption reduction has been addressed in two ways. Firstly, by designing the robot to use minimal power by powering off its actuators while not being in use. Secondly, by introducing a hybrid, asynchronous controller that utilised fast, yet power-efficient pattern generators that were connected with slow but adaptable recurrent neural networks. In addition, the neural networks offered the capability of learning and transferring knowledge between different environments.

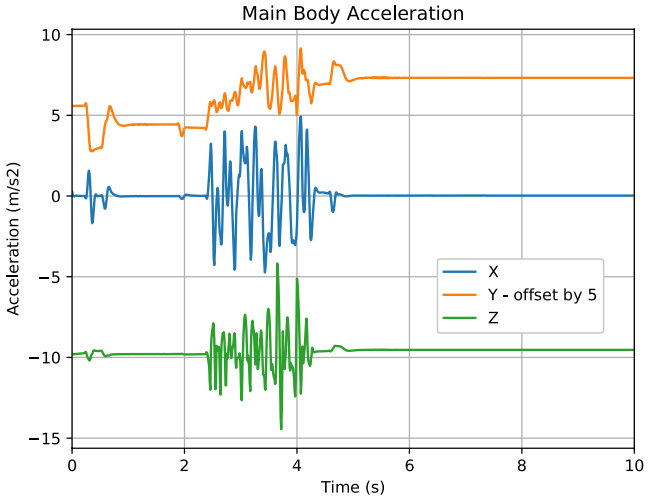
The presented work has focused on the integration of reinforcement learning with local pattern generators, and not into pattern generators, to create a central pattern generator. The process of training reinforcement learning algorithms with recurrent neurons is challenging, as the memory of previous episodes must be called randomly without breaking the coherence between the time-steps of each episode. Also, decisions from reinforcement learning were executed every 100 steps, and that introduced a challenge as the algorithm must be able to learn and adapt to the dynamics of a system that was not able to act on.

The results demonstrated that the robot achieved stability in less than 300 episodes. The validation experiments also demonstrated that the robot could act without the continues supervision of the reinforcement learning algorithm. The results showed minimal movement in the transverse plane and a speed of 2 steps/s, out of the ideal 3 steps/s that it was trained for.

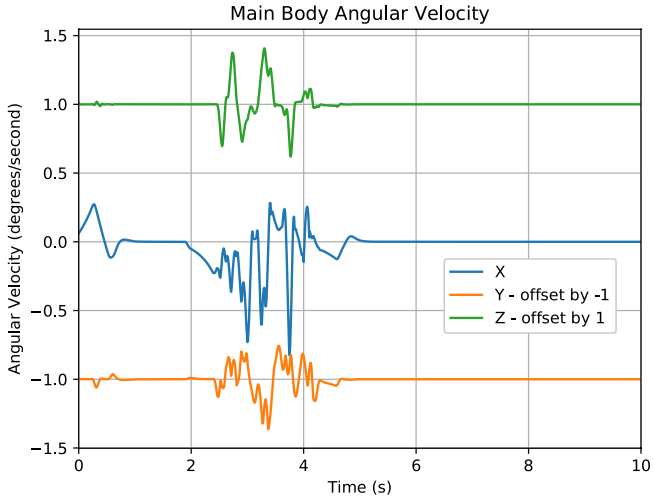
The natural evolution of this work is to be implemented on the actual robot, and then evaluate the response in different environments. At the same time, different real-time systems will be tested and evaluated to explore the full potentials and limitations of the eNAF algorithm.

Acknowledgement

The authors gratefully acknowledge the financial support of the Engineering and Physical Sciences Research Council (EPSRC) Center for Doctoral Training in Embedded Intelligence under grant reference EP/L014998/1. Additionally, to pay special regards to Motion Robotics LTD for their support and input to this research.



(a) Acceleration of robot's trunk.



(b) Angular velocity of robot's trunk.

Fig. 9: IMU data from robot during evaluation.

REFERENCES

[1] Honda Motor Co., "Robot Development History, Honda Motor Co." [Online]. Available: https://global.honda/innovation/robotics/robot-development-history.html

[2] H. Van der Kooij, R. Jacobs, B. Koopman, and F. Van der Helm, "An alternative approach to synthesizing bipedal walking," *Biol. Cybern.*, vol. 88, no. 1, pp. 46–59, 2003.

[3] D. Lee and W. ElMaraghy, "A neural network solution for bipedal gait synthesis," in *[Proceedings 1992] IJCNN Int. Jt. Conf. Neural Networks*, vol. 2. IEEE, jun 1992, pp. 763–768.

[4] H. Shahbazi, K. Jamshidi, A. H. Monadjemi, and H. Eslami, "Biologically inspired layered learning in humanoid robots," *Knowledge-Based Syst.*, vol. 57, pp. 8–27, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.knosys.2013.12.003

[5] S. F. Rashidi, M. R. S. Noorani, M. Shoaran, and A. Ghanbari, "Gait generation and transition for a five-link biped robot by Central Pattern Generator," *2014 2nd RSI/ISM Int. Conf. Robot. Mechatronics, ICRoM 2014*, pp. 852–857, 2014.

[6] J. Massion, "MOVEMENT , POSTURE AND EQUILIBRIUM : INTERACTION AND COORDINATION," *Prog. Neurobiol.*, vol. 38, no. 1, pp. 35–56, 1992.

[7] J. B. Nielsen, "How we Walk: Central Control of Muscle Activity during Human Walking," *Neurosci.*, vol. 9, no. 3, pp. 195–204, jun 2003. [Online]. Available: http://journals.sagepub.com/doi/10.1177/1073858403009003012

[8] C. Kouppas, Q. Meng, M. King, and D. Majoe, "S.A.R.A.H.: The Bipedal Robot with Machine Learning Step Decision Making," *Int. J. Mech. Eng. Robot. Res.*, vol. 7, no. 4, 2018.

[9] American Honda Motor Co. Inc., "ASIMO Specifications — ASIMO Innovations by Honda," 2018. [Online]. Available: asimo.honda.com/asimo-specs/

[10] Boston Dynamics, "ATLAS - The World's Most Dynamic Humanoid," 2018. [Online]. Available: www.bostondynamics.com/atlas

[11] Agility Robotics, "Cassie." [Online]. Available: http://www.agilityrobotics.com/robots

[12] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Iterative Reinforcement Learning Based Design of Dynamic Locomotion Skills for Cassie," mar 2019. [Online]. Available: http://arxiv.org/abs/1903.09537

[13] R. L. Calabrese, "Cellular, synaptic, network, and modulatory mechanisms involved in rhythm generation," *Curr. Opin. Neurobiol.*, vol. 8, no. 6, pp. 710–717, dec 1998.

[14] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous Deep Q-Learning with Model-based Acceleration," *Transplant. Proc.*, vol. 32, no. 5, pp. 932–934, mar 2016. [Online]. Available: http://arxiv.org/abs/1603.00748

[15] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine, "Composable Deep Reinforcement Learning for Robotic Manipulation," *Proc. - IEEE Int. Conf. Robot. Autom.*, no. 1, pp. 6244–6251, 2018.

[16] C. Kouppas, M. Rodosthenous, N. Sagyndyk, Q. Meng, M. King, and D. Majoe, "Designing a novel bipedal Silent Agile Robust Autonomous Host (S.A.R.A.H)," in *UK-RAS Conf. 'ROBOTS Work. AMONG US'*, Bristol, 2017, pp. 114–117.

[17] C. R. Taylor, N. C. Heglund, and G. M. Maloiy, "Energetics and mechanics of terrestrial locomotion. I. Metabolic energy consumption as a function of speed and body size in birds and mammals." *J. Exp. Biol.*, vol. 97, no. 1, pp. 1–21, 1982.

[18] Microchip Technology Inc., "Microchip SAM C, ATSAMC21J18A," 2019. [Online]. Available: https://www.microchip.com/wwwproducts/en/ATSAMC21J18A

[19] Raspberry PI Foundation, "Raspberry Pi 3 Model B," 2019. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[20] J. Pratt and G. Pratt, "Intuitive control of a planar bipedal walking robot," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 3, no. May 1998, pp. 2014–2021, 1998.

[21] K. Doya, "Reinforcement Learning In Continuous Time and Space," *Neural Comput.*, vol. 12, no. 1, pp. 219–245, 2000.

[22] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, J. Quan, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. van Hasselt, D. Silver, T. Lillicrap, K. Calderone, P. Keet, A. Brunasso, D. Lawrence, A. Ekermo, J. Repp, and R. Tsing, "StarCraft II: A New Challenge for Reinforcement Learning," 2017. [Online]. Available: http://arxiv.org/abs/1708.04782

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

[24] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE Int. Conf. Robot. Autom.* IEEE, may 2017, pp. 3389–3396.

[25] CM Labs Simulations, "Theory Guide: Vortex Software's Multibody Dynamics Engine," Tech. Rep., 2016.

[26] T. Matsubara, J. Morimoto, J. Nakanishi, M. aki Sato, and K. Doya, "Learning CPG-based biped locomotion with a policy gradient method," *Rob. Auton. Syst.*, vol. 54, no. 11, pp. 911–920, 2006.

[27] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng, "Learning CPG-based biped locomotion with a policy gradient method: Application to a humanoid robot," *Int. J. Rob. Res.*, vol. 27, no. 2, pp. 213–228, 2008.