Reinforcement Learning of 3D Musculoskeletal Model for Walking or Running with Minimum Efforts

** Vikram Singh Chandel*
*** Subhabaha Pal*

Abstract

In this work we are trying to build a controller for a musculoskeletal model that has the goal of matching a given time-varying velocity vector. The major objective is building a musculoskeletal model which is fully comprehensive and reproduces realistic human movements driven by muscle contraction dynamics. Spectrum of human movement will be generated through variations in the anatomic model.

Keywords : **3D Musculoskeletal Model, Artificial Intelligence, Reinforcement Learning**

I. INTRODUCTION

There has been a lot of progress in technology now which has increased interest in prosthetics for improving human movement when it comes to 3D human model. However, it was difficult for designing such kind of devices as it was taking lot of iterations through many designs. This is further complicated by the large variability in response among many individuals.

Movement of human body depends upon many factors such as geometry of bones, muscles movement, angle between bones, habits, and emotions. Any changes in these anatomical conditions can result in overall movement. Our body has more than 300 muscles which are responsible for movement. Whenever movement of body takes place, the brain sends excitation signals to the nervous system which further control the overall movement of the body. Every time when movement happens, the brain learns some sets of responses.

Recent developments are going on for this particular purpose using Artificial Intelligence techniques to train realistic, biomechanical models which will be key in increasing our understanding of the human-prosthesis interaction and help to accelerate development of this field.

We use Reinforcement Learning (RL) to take care of our issues and promote open source devices in RL to look into the material science test system, the RL condition, and the stage on which we run codes that are for the most part open-source. We additionally empower RL to examine in computationally complex situations with stochasticity and profoundly dimensional activity spaces, pertinent to genuine applications which will connect biomechanics, neuroscience, and software engineering networks.

A. Reinforcement Learning

Reinforcement learning is a training method based on rewarding desired behaviors and/or punishing undesired ones. The learning method has been adopted in artificial intelligence (AI) as a method of directing unsupervised machine learning through rewards and penalties. Reinforcement learning is used in operations research, simulation information theory, game theory, control

Manuscript Received: April 3, 2020; Revised: April 25, 2020; Accepted: May 2, 2020.
* V. S. Chandel, Data Science Student, Manipal University, B-80 Aakriti Gardens, Nehru Nagar, Bhopal - 462 003, India.
(email : vsc8088@gmail.com)
** S. Pal is Senior Faculty, Data Science and Machine Learning, with Manipal ProLearn (Manipal Academy of Higher Education – South Bangalore Campus), 3rd Floor, Salarpuria Symphony, 7, Service Road, Pragathi Nagar, Electronics City Post, Bengaluru – 560 100, India.
(email : subhabaha@gmail.com)
DOI : 10.17010/ijcs/2020/v5/i2&3/152870

theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics, and genetic algorithms. It involves speech and text-based communication. It is a technology behind chatbots, virtual assistants, online translation services and many more.

B. Components of Reinforcement Learning

Environment : Physical world in which the agent operates

State : Current situation of the agent

Reward : Feedback from the environment

Policy : Method to map agent's state to actions

Value : Future reward that an agent would receive by taking an action in a particular state

C. Statement of the Problem

We are going to build a real time controller for simulated agent to walk or run at a particular speed and direction. Here, the agent which is a musculoskeletal model interacts with an environment generally as physics simulator by taking actions on so called muscle excitations based on observations (a function of the internal state of the model) in order to maximize the reward.

Every time when walking is done by the musculoskeletal model, there are certain sets of action which are defined as positive and negative reward. For example, let us suppose the model is walking in back direction or falling apart, then it is a negative reward. If a model is moving forward and taking proper steps, then it is a positive reward.

There are certain sets of actions which define the sets of positive and negative rewards. For example, a model is moving forward but the steps are less in distance or model neck is not properly aligned, which can be the reason of its fall and this can become negative reward.

The model of the specialist included 22 muscles to control 11 degrees-of-opportunity (DOF). At each cycle, the operator got the current watched express, a vector comprising of a set of qualities which incorporate ground response powers, muscle exercises, muscle fiber, lengths, muscle speeds, ligament powers, positions, speeds, and increasing velocities of joint edges and body sections.

Muscle enactments try to create movement as a component of muscle properties. For example, quality, and muscle states, for example, current length, speed, and second arm. A general gauge of muscle exertion will be determined utilizing the total of muscle actuations squared, a regularly utilized measurement in biomechanical considers.

The main goal is to generate the controls such that the model would move forward at 2 m/s. The total reward is calculated as Summation of (4−|vx(st)−3|^2) function from t=1 seconds to t=T.

(in this case *st* is the state of the model at time *t*. Similarly, the vx(s) can be described as the horizontal velocity vector of the pelvis in the state. The st can be calculated as M(st−1,a(st1)). It also means the states which follow the simulation and it is given by model M. In this case, we are taking T as the episode termination time step and it is equal to 1000 in case the model does not fall and walks continuously for the full 10 second duration. It can also be taken as equal to the first time point in case when we are finding that the pelvis of our models falls below 0.6 m in order to penalize the fall of the model).

We propose a new family of policy gradient methods for reinforcement learning, which alternate between sampling data through interaction with the environment, and optimizing a "surrogate" objective function using stochastic gradient ascent, whereas, standard policy gradient methods perform one gradient update per data sample. We propose a novel objective function that enables multiple epochs of minibatch updates.

D. Objectives

The aims of this project are to :

- Compute actions according to a Gaussian distribution given by mean and variance.
- Compute the predicted value functions from the critic model.
- Compute the desired function which will be later used as regression target for the critic model.
- Calculates delta terms for every timestep in every trajectory.
- Calculates general advantage estimates for every timestep in every trajectory.
- Update the required parameters for the new actor and the critic models so that model is finally properly trained.

E. Project Goal and Scope

- Use Reinforcement Learning (RL) to solve problems in healthcare.

• Promote open-source tools in RL research, the physics stimulator and the RL environment.

• Encourage RL research in computationally complex environments, with stochasticity and highly-dimensional action spaces relevant to real-life applications.

• Bridge biomechanics, neuroscience, and computer science communities.

F. System Overview

OpenSim environment is used in this project. OpenSim is an open-source stage for demonstrating, recreating, and dissecting the neuromusculoskeletal framework. It incorporates low-level computational devices that are conjured by an application. The module engineering of OpenSim urges clients to broaden usefulness by building up their own muscle models, contact models, controllers, and investigations. For instance, around twelve examination modules composed by various clients are accessible in OpenSim. These examination devices ascertain joint powers, muscle-prompted increasing speeds, muscle powers, and different factors. Despite the fact that these investigations were produced for various musculoskeletal models, they have general appropriateness and can be utilized with any OpenSim model. To make a muscle-driven reenactment of a development, one should initially detail a unique model of the musculoskeletal framework and its co-operations with nature. The components of the musculoskeletal framework are displayed by sets of differential conditions that depict muscle withdrawal elements, musculoskeletal geometry, and body segmental elements.

II. HUMAN MODEL

• 3D musculoskeletal model of healthy adult

• 8 internal degrees of freedom (4 per leg)

hip_abd (+: hip abduction)

hip (+: extension)

knee (+: extension)

ankle (+: plantar flexion)

• 22 muscles (11 per leg)

HAB: hip abductor

HAD: hip adductor

HFL: hip flexor

GLU: glutei (hip extensor)

HAM: hamstrings (biarticular hip extensor and knee flexor)

RF: rectus femoris (biarticular hip flexor and knee extensor)

VAS: vastii (knee extensor)

BFSH: biceps femoris, short head (knee flexor)

GAS: gastrocnemius (biarticular knee flexor and ankle extensor)

SOL: soleus (ankle extensor)

TA: tibialis anterior (ankle flexor)

III. BIOMECHANICS

To summarize briefly, the agent is a musculoskeletal model that includes body segments for each leg, a pelvis segment, and a single segment to represent the upper

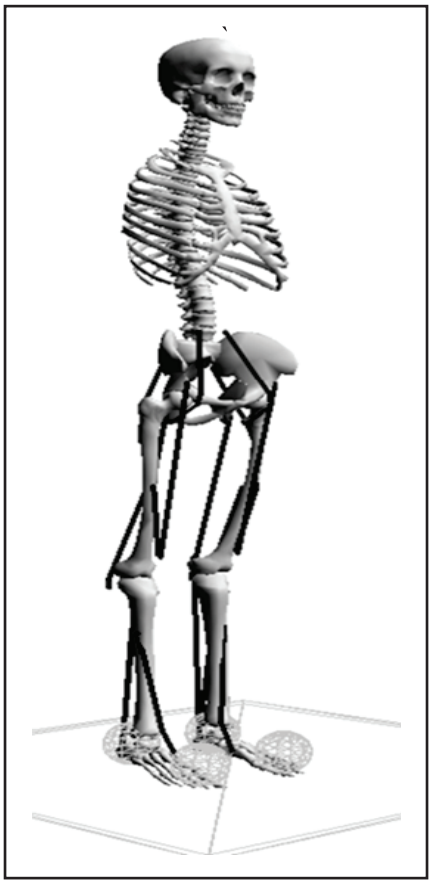

Fig 1. Representiion of 3D Musculoskeletal Model

half of the body (trunk, head, arms). The segments are connected with joints (e.g., knee and hip), and the motion of these joints is controlled by the excitation of muscles. The muscles in the model have complex paths (for example, muscles can cross more than one joint and there are redundant muscles). The muscle actuators themselves are also highly nonlinear. For example, there is a first order differential equation that relates electrical signal the nervous system sends to a muscle (the excitation) to the activation of a muscle (which describes how much force a muscle will actually generate given the muscle's current force-generating capacity).

Given the musculoskeletal structure of bones, joint, and muscles, at each step of the simulation (corresponding to 0.01 seconds), the engine computes activations of muscles from the excitations vector provided to the step() function

- actuates muscles according to these activations
- computes torques generated due to muscle activations
- computes forces caused by contacting the ground
- computes velocities and positions of joints and bodies

It also generates a new state based on forces, velocities, and positions of joints in which each action, the following 18 muscles are actuated (9 per leg) :

- hamstrings
- biceps femoris
- gluteus maximus
- iliopsoas
- rectus femoris
- vastus
- gastrocnemius
- soleus
- tibialis anterior. The action vector corresponds to these muscles in the same order (9 muscles of the right leg first, then 9 muscles of the left leg).

The observation contains 41 values:

- position of the pelvis (rotation, x, y)
- velocity of the pelvis (rotation, x, y)
- rotation of each ankle, knee, and hip (6 values)
- angular velocity of each ankle, knee, and hip (6 values)
- position of the center of mass (2 values)
- velocity of the center of mass (2 values)
- positions (x, y) of head, pelvis, torso, left, and right toes, left and right talus (14 values)
- strength of left and right psoas: 1 for difficulty < 2, otherwise a random normal variable with mean 1 and standard deviation 0.1 fixed for the entire simulation.
- next obstacle: x distance from the pelvis, y position of the center relative to the ground, radius.

Open source software system is used for modeling, simulating, and analyzing the neuromusculoskeletal system. OpenSim is built on top of core computational components that allow one to derive equations of motion for dynamical systems, perform numerical integration, and solve constrained non-linear optimization problems. In addition, OpenSim offers access to control algorithms (e.g. computed muscle control), actuators (e.g., muscle and contact models), and analyses (e.g. muscle-induced accelerations). OpenSim integrates these components into a modeling and simulation platform. Users can extend OpenSim by writing their own plug-ins for analysis or control, or to represent neuromusculoskeletal elements (e.g. muscle models). In a graphical user interface, the user is able to access a suite of high-level tools for viewing models, editing muscles, plotting results, and other functions. SimTrack, one of the OpenSim tools enables accurate muscle-driven simulations to be generated that represent the dynamics of individual subjects.

IV. UNDERSTANDING ACTION SPACE

The observation or the input to your controller consists of a local target velocity map V and the body state S.

V is a $2\times11\times11$ matrix, representing a 2D vector field on an 11×11 grid. The 2D vectors are target velocities, and the 11×11 grid is for every 0.5 meter within ±5 meters back-to-front and left-to-side. For example, in fig. 2, the global target velocity map (top-left) shows that the velocity field converges to $(5.4,-1.4)$ and the human model is at (5.4,0.0) (end of the black line). Thus, the local target velocity map (bottom-left) shows that the human model should locomote to the right as the target velocity vector at (0.0,0.0) points towards the right (i.e. close to $[0,-1]$).

S is a 97D vector representing the body state. It consists of pelvis state, ground reaction forces, joint angles and

rates, and muscle states. The keys of the observation dictionary should be self-evident and explain what the values represent.

For example, space [0,1] 22 represents muscle activations of the 22 muscles, 11 per leg.

V. UNDERSTANDING OBSERVING SPACE

The observation can be divided into five components: the body parts, the joints, the muscles, the forces, and the center of mass. For each *body part* component, the agent observes its position, velocity, acceleration, rotation, rotational velocity, and rotational acceleration. Similarly, for each *joint*, the agent observes its position, velocity, and acceleration. For each *muscle*, the agent observes its activation, fiber force, fiber length, and fiber velocity. The *force* component describes the forces acting on body parts. Finally, the agent observes the position, velocity, and acceleration of its *center of mass.*

All the positions in the observation are absolute positions, but we are more in terested in their positions relative to the pelvis for their xx and zz coordinates. If the head is behind the pelvis, the agent will likely fall behind, regardless of its absolute position, whereas, if the head is in front of the pelvis, the agent will likely fall forward or run. Thus, the xx and zz coordinates for each body part should be shifted by that of the pelvis.

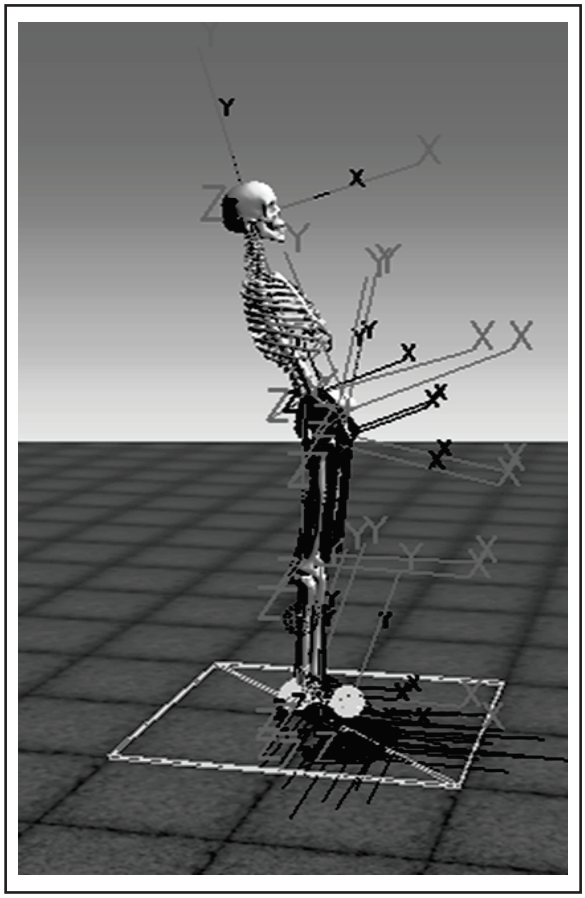

Fig 2. 3D Musculoskeletal Model Muscles with Internal Degrees shown in X, Y and Z Plane

VI. METHODOLOGY

PPO (Proximal Policy Optimization) as an on-strategy solver frequently experiences the issue of bigger change and lower efficiency. To additionally build our efficiency, we applied Deep Exploration with multi-head bootstrapping, which has been demonstrated to unite a lot quicker contrasted and ε-covetous. So as to permit our approach to intently follow the speed target, we infused the speed as a component to the arrangement and worth system. Finally, to address the center issue of nearby ideal, we applied educational plan figuring out how to move efficient and stable strides to different speed extend.

A. Model Architecture

Contrasted with general DDPG (Deep Deterministic policy Gradient) network structures, it has two unmistakable highlights. We infuse the objective speed from the base of the two systems, as the worth capacity needs to assess the present state dependent on track speed, and the strategy needs to make the relating move to arrive at the objective speed. This is like including the objective speed as a component of the perception. Despite the fact that it presents some commotion when the speed is exchanging, it benefits more via consequently sharing the information on various speeds. We likewise utilize different heads for the worth and strategy arrange in our model. It is a comparative design as profound investigation, which re-enacts the troupe of neural systems with lower cost by sharing the base layers.

B. Transfer Learning

We suggest that by sharing information on strolling or running at various paces, it is possible for the agent to learn progressively strong and efficient examples of strolling. We found in our test that the shaky steps gained without

any preparation for low speed strolling don't function admirably for rapid running. We explored on running as quick as conceivable rather than at a specified speed. We acquired an operator that can run quick with sensible and characteristic strides similarly as people. Beginning with the prepared arrangement for quick running, we changed the objective to bring down speed strolling. This procedure looks like exchange realizing, where we need the "information" of step to be kept yet with a more slow speed. Our quickest running has speeds over 4.0 m/s. We moved the arrangement to 1.25 m/s, yet it brings about motions that are as yet not normal enough and were inclined to falling. In any case, we gained ground by moving from a higher speed as the fall rate drops generously.

C. Curriculum Learning

Educational plan learning learns a difficult task continuously by artificially developing a progression of undertakings which increment the difficulty level step by step. As of late, it has been utilized to understand complex computer game difficulties. As the immediate exchange of a higher speed running approach to bring down speed didn't function admirably, we conceived 5 undertakings to diminish the speed directly, with each errand beginning with the prepared arrangement of the previous one. Finally, we have a strategy running at target = 1.25m/s, with common walks that look like a person and low falling rate also.

D. Fine-tuning

In view of the pre-trained strolling model that focused at 1.25 m/s, we fine-tuned the model on the irregular speed condition. Right off the bat, we attempted to compel the arrangement to stroll at 1.25 m/s, given any objective speed between - 0.5 m/s and 3.0 m/s. This was to make a decent beginning for other objective speeds other than 1.2 m/s. We gathered strolling directions at 1.25 m/s, yet changed the highlights of target speed and course to an arbitrary worth. We utilized the gathered directions to re-train the strategy with managed learning. Besides, we utilized the re-prepared model as the beginning point, and fine-tuned it in the randomized objective speed conditions utilizing objective driven DDPG (Deep Deterministic policy Gradient), which gives our final approach.

VII. EXPERIMENTS

Our investigations contrasted educational program taking in and gaining, without any preparation in the fine-tuning stage. We utilized a similar model engineering for the two tests. For the on-screen character model, we used tanh as actuation work for each layer. For the pundit model, selu [24] was utilized as enactment works in each layer with the exception of the last layer. The markdown factor for total prize calculation was 0.96. We likewise utilized the edge skip stunt, as each progression of the specialist compares to 4 re-enactment step in the earth with a similar activity. Twelve heads were utilized for bootstrapped profound investigation. This was chosen by considering the exchange off between the different speculations of each head and calculation cost by and by. We indicated the examination of gaining without any preparation and beginning from an approach learned with educational plan learning. Each bend was arrived at the midpoint of on 3 autonomous investigations. Significant enhancements for both execution and security for the educational plan learning can be watched. Further examining the strolling strides shows that educational program learning has a progressively regular strolling signal. Fig. 3 is the learning curve. Averages are computed from a set of 50 episodes. In fig. 3, the agent walks forward while heading at strange directions. (b) The skeleton walks naturally with small steps.

VIII. UNDERSTANDING REWARD

A simulation runs until either the pelvis of the human model falls below 0.60 m or when it reaches 10 s (i=1000). During the simulation, you receive a survival reward every

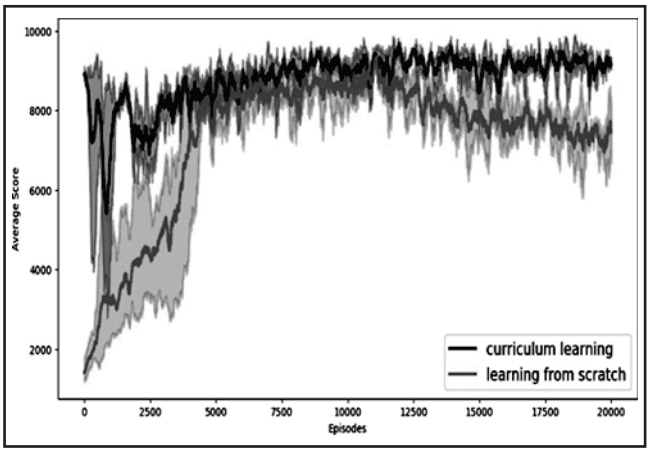

Fig. 3 (a). Model Learning Graph

timestep ii and a footstep reward whenever the human model makes a new footstep step 'i' which mentions steps. The reward is designed so that the total reward J(π) is high when the human model locomotes at desired velocities with minimum effort.

The footstep reward R step is designed to evaluate step behaviors rather than instantaneous behaviors, for example, to allow the human model's walking speed to vary within a footstep as real humans do. Specifically, the rewards and costs are defined as Δti = 0.01 sec is the simulation timestep, vpel is the velocity of the pelvis, vtgt is the target velocity, Am are the muscle activations, and wstep, wvel and weff are the weights for the stepping reward and velocity and effort costs.

$$J(\pi) = R_{alive} + R_{step}$$

$$= \sum_i r_{alive} + \sum_{step_i} \left(w_{step} r_{step} - w_{vel} c_{vel} - w_{eff} c_{eff} \right).$$

$$r_{alive} = 0.1$$

$$r_{step} = \sum_{i \text{ in } step_i} \Delta t_i = \Delta t_{step_i}$$

$$c_{vel} = \| \sum_{i \text{ in } step_i} (v_{pel} - v_{tgt}) \Delta t_i \|$$

$$c_{eff} = \sum_{i \text{ in } step_i} \sum_{m}^{muscles} A_m^2 \Delta t_i.$$

IX. INSTALLATION PROCESS

Anaconda is necessary in order to run the simulations shown in this paper. Anaconda will create a virtual environment with all the necessary libraries to avoid conflicts with libraries in your operating system. Anaconda may be obtained from https://docs.anaconda.com/anaconda/install/. The following instructions have been given assuming that the Anaconda has been successfully installed.

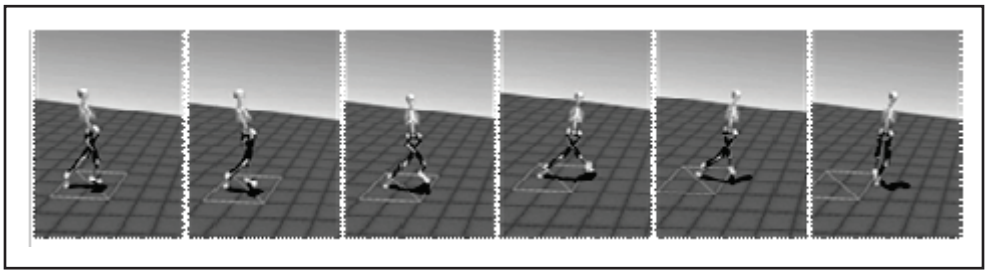

Fig. 3 (b). Model Learning From Scratch

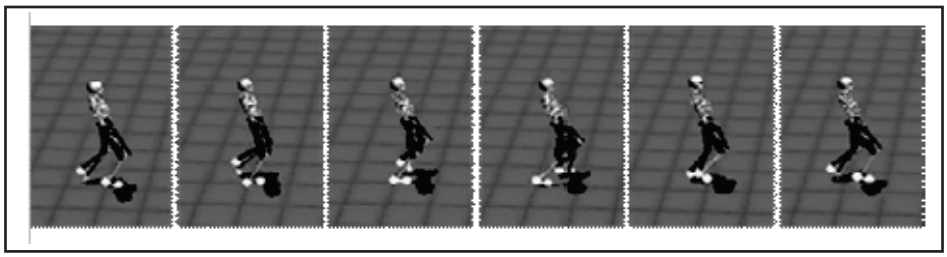

Fig. 3 (c). Curriculum Learning

Fig. 3. Model Learning

1. Getting Started

On **Windows**, open a command prompt and type :

conda create -n opensim - rl -c kidzik opensim python=3.6.1 activate opensim-rl

On Linux/OSX, run :

conda create -n opensim-rl -c kidzik opensim python=3.6.1 source activate opensim-rl

These commands will create a virtual environment on your computer with the necessary simulation libraries installed. Next, we need to install our Python reinforcement learning environment. Type (on all platforms) :

```
conda install -c conda-forge lapack git
pip install osim-rl
```

If the command python -c "import opensim" runs smoothly, you are done! Otherwise NOT.

It is to be noted that the source opensim-rl activates the anaconda virtual environment. It is needed to type it every time Anaconda is opened.

2. Basic Usage

To execute 200 iterations of the simulation enter the python interpreter runs the following :

```
from osim.env import L2M2019Env

env = L2M2019Env(visualize=True)
observation = env.reset()
for i in range(200):
observation, reward, done, info =
env.step(env.action_space.sample())
```

The function env.action_space.sample() returns a random vector for muscle activations as shown in Fig. 4, So, in this example, muscles are activated randomly (muscles which belong to centre part from waist to leg are active muscles and outer to centre muscles which are covering inner muscles are inactive muscle). Clearly, with this technique we won't go too far.

Our goal is to construct a controller, that is, a function from the state space (current positions, velocities and accelerations of joints) to action space (muscle excitations), that will enable to model to travel as far as possible in a fixed amount of time. Suppose you train a neural network by mapping observations (the current state of the model) to actions (muscle excitations), that is, you have a function action = my controller(observation), then

```
total_reward = 0.0
for i in range(200):
# make a step given by the controller and record the state and the reward observation, reward, done, info
= env.step(my_controller(observation))
total_reward += reward
if done:
break

# Your reward is
print("Total reward %f" % total_reward)
```

X. ABOUT POLICY

A. Representation

Stochastic policy was used. Both the function Pi, the value function, V_{φ} were implemented using feed-forward neural networks with two hidden layers, with 256 tanh units each.

The network input consisted of all 406 features provided by the environment. The joint positions (x, and z's) were made relative to the position of the pelvis. In addition, the coordinate system was rotated around the vertical axis to zero out the z component of the target velocity vector. All the features were standardized with a

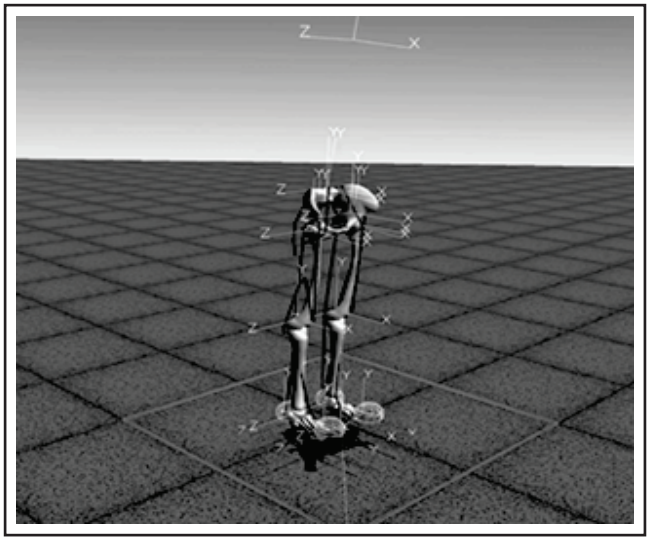

Fig. 4. Initial State of a Muscuskeleton Model

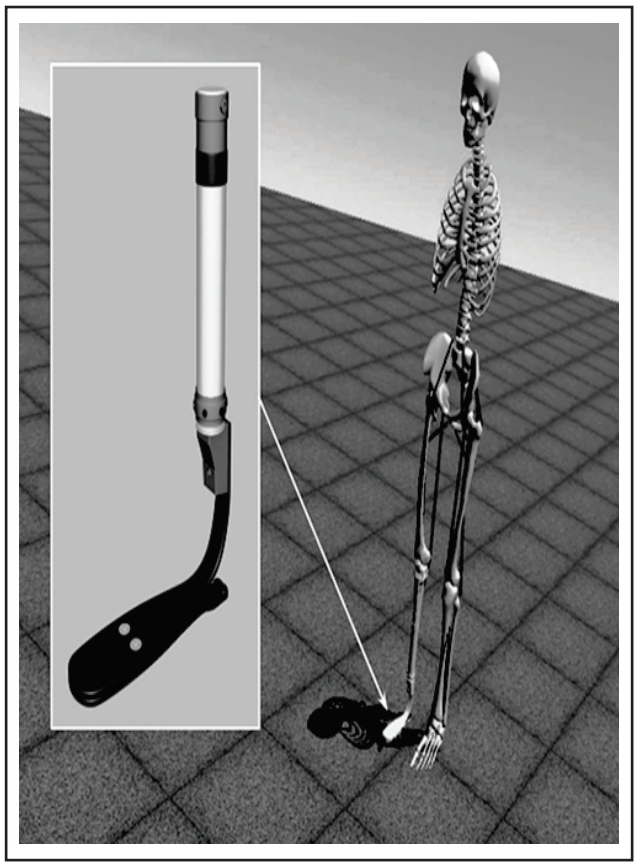

Fig. 5. Leg that can be Changed as per Requirement

running mean and variance.

Our network outputs a Bernoulli policy, which gives samples from[0,1]. It was found that this policy was leading to better result.

B. Policy Training

The arrangement parameters θ, φ, and ψ were scholarly with Proximal Policy Gradient with the Generalized Advantage Estimator as the objective for the bit of leeway work. An objective favorable position remedy was applied so as to manage the non-stationarity of the earth brought about as far as possible. The amendment avoids the operator the end of the scene that is brought about as far as possible by utilizing the worth gauge. As a result, it improves the accuracy of the worth capacity, thereby diminishing the fluctuation of the inclination estimator.

C. Training Regime

The philosophy applied comprised of three phases: i) worldwide instatement, ii) policy refinement and last strategy calibrating.

PPO (Proximal policy Optimization) is vulnerable to nearby minima. Being an on-approach calculation, each cycle improves the strategy by a smidgen. As a result, it is probably not going to roll out enormous conduct improvements of the specialist. When the operator begins to display a specific step, PPO (Proximal policy optimization) can't change to a totally extraordinary method of strolling later. To lighten this issue, during the worldwide instatement stage, 50 runs were executed in equal. After around 1000 cycles, two steps were chosen depending on their presentation and conduct difference to be improved in the ensuing stage.

The subsequent stage, arrangement refinement, included more number of tests per run and went on until an intermingling was observed. After the steps, parameter decreased to 1.

In the last stage, strategy adjusting, all the investigation motivators were in the long run killed and the approach was divided into two sub-approaches, one for each assignment modes: i) prepared set-go, utilized for the initial 100 timesteps; and ii) ordinary activity for the remainder of the scene.

XI. CONCLUSION

At long last, in the wake of doing the way toward expanding likelihood of positive prize, the model had the option to walk and run appropriately.

The depicted preparing strategy brought about two particular strides of comparative normal execution. Both walks have fascinating qualities. The marginally better approach begins forward with his prosthetic leg. At that point pivots on initial stage scarcely proceeds with its walk in reverse form in this particular time interval. It appears that the preparation found that strolling in reverse was a progressively proficient approach to manage the progressions of the speed vector.

The other strategy begins with lifting his prosthetic leg; he hops on this sound leg for the entire scene utilizing the prosthetic leg to keep balance. This is, unquestionably, not the most common method of strolling.

We have likewise presented proximal strategy streamlining, a group of arrangement advancement strategies that utilization different ages of stochastic angle climb to play out every approach update. These strategies have the steadiness and unwavering quality of trust-locale techniques, yet are a lot more straightforward to execute,

requiring just barely any lines of code change to a vanilla strategy inclination usage, material in increasingly broad settings (for instance, when utilizing a joint design for the approach and worth capacity), and have better generally execution.

XII. LIMITATIONS

This project can only be done when we have high processing machines like GPU because training of data is the toughest part and is not possible in normal processing speed systems.

Also, on the off chance we have a go at gaining without any preparation to accomplish some particular speed, our initial examination uncovered that the skeleton strolls with an assortment of signals that bring about practically a similar exhibition in remunerations. The operator either strolls a horizontal way (crab-like strolling), knocking, or hauling one of its leg. While none of those strolling steps is characteristic, they are almost indistinct in the prizes.

Nonetheless, in spite of the fact that we found that those unreasonable steps can sensibly create static speed strolling, they perform ineffectively concerning solidness. Moving the model to other indicated speeds turns into an issue, and the framework is inclined to fall, particularly at the time of exchanging speeds.

This project can only be done with high processors and heavy GPU system because training of model is a very difficult task as it requires huge number of iteration in order to get positive rewards. There are various sets of combination which is finally implemented with the help of Bernoulli in order to get different sets of positive response.

REFERENCES

[1] K. Arulkumaran, N. Dilokthanakul, M. Shanahan, and A.A. Bharath, "Classifying options for Deep Reinforcement Learning," 2016. Imperial College, London. [Online]. Available: https://arxiv.org/pdf/1604.08153.pdf

[2] P-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," Reasoning and Learning Lab, McGill University, 2016. [Online]. Available: https://arxiv.org/pdf/1609.05140.pdf

[3] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018. [Online]. Available: https://arxiv.org/pdf/1801.01290.pdf

[4] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, "Distributed prioritized experience replay," 2018. [Online]. Available: https://arxiv.org/pdf/1803.00933.pdf

[5] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, Snapshot ensembles: Train 1, get m for free," 2017. [Online]. Available: https://arxiv.org/abs/1704.00109

[6] Z. Huang, S. Zhou, B. Zhuang, and X. Zhou, "Learning to run with actor-critic ensemble," 2017. [Online]. Available: arXiv:1712.08987

[7] I. Osband and C. Blundell, "A.P.B.V.R.: Deep exploration via bootstrapped dqn," 2016.

[8] W. Jas´kowski, O. R. Lykkebø, N. E. Toklu, N.E., F. Trifterer, Z. Buk, J. Koutník, and F. Gomez, Reinforcement Learning to Run... Fast. In: S. Escalera, M. Weimer (eds.) NIPS 2017 Competition Book. Springer, 2018.

[9] C. T. John, F. C. Anderson, J. S. Higginson, and S. L. Delp, "Stabilisation of walking by intrinsic muscle properties revealed in a three-dimensional muscle-driven simulation," *Comput. Methods in Biomechanics and Biomedical Eng.*, vol. 16, no. 4, 2013. Doi: https://doi.org/10.1080/10255842.2011.627560

[10] L. Kidzin´ski, S. P. Mohanty, C. Ong, Z. Huang, S. Zhou, A. Pechenko, A. Stelmaszczyk, P. Jarosik, M. Pavlov, and S. Kolesnikov et al., "Learning to run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments," 2018. [Online]. Available: https://arxiv.org/pdf/1804.00361.pdf

[11] L. Kidzin´ski, S. P. Mohanty, C. Ong, J. Hicks, S. F. Carroll, S. Levine, M. Salathé, and S. L. Delp, "Learning to run challenge: Synthesizing physiologically accurate motion using deep reinforcement learning," in S. Escalera, M. Weimer (eds.) NIPS 2017 Competition Book. Springer, Springer 2018. [Online]. Available: https://arxiv.org/abs/1804.00198

About the Authors

Vikram Singh Chandel is pursuing P.G. Diploma in Data Science from Manipal University. He completed B. E. (Information Technology) from University Institute of Technology – RGPV in 2011. He has worked in the IT industry as a software engineer from 2012 to 2018.

Dr. Subhabaha Pal is a well-acclaimed Data Science & Analytics academician whose name had been included in the list '20 Most Prominent Analytics & Data Science Academicians in India 2018' published by Analytics India Magazine. He was also honoured as the 'Most Supportive Faculty' by Data Science Society, Bulgaria. He is Ph.D. and M.Sc. in Statistics from the University of Calcutta and has taught in well-known institutions like Manipal University, T. A. Pai Management Institute & International Institute of Digital Technologies among many. He has 40 research paper publications to his credit in well-known national & international indexed (SCOPUS and ABDC) journals and has also published multiple books on Statistics & Machine Learning with well-known publishers . Dr. Subhabaha Pal worked in different domains and verticals apart from Data Science which include SAP Implementation & Corporate Risk Management in well-known organizations like Manipal Global Education Services & Kuwait Petroleum Corporation. He was also instrumental in founding an Analytics Company InstaDataHelp Analytics Services (www.instadatahelp.com) which provides academic and consultancy services in analytics.