Fall 2019 ME751 Final Project Report
University of Wisconsin-Madison

Learning how to walk using PyChrono and Tensorflow

Kieran Nichols

December 18, 2019

Abstract

This project will aim to train robotic devices of a four-legged ant and two-legged human (humanoid) to walk across a flat ground in a straight line. Two powerful tools of PyChono and Tensorflow will used to simulate and train the robots. PyChrono is multi-physics engine for modelling and simulating and Tensorflow is a machine learning platform. Together, I hope to learn how to use the existing code training the ant, adapt and improve the code to create the humanoid, and output the results in the form of pictures, videos, and graphs. The pictures and videos will show the robots' walking from different angles. Whereas, the graphs will display the learning process by showing differences of execution time of two different ways of processing the learning (serial vs. parallel code) and showing how the state (position and velocity) of the robot changes with the learning progression. Overall, this project gives me ample exposure to these powerful packages and helps me as a researcher and scientist be aware of the unique tools that can be used to probe hard scientific questions.

My git repository can be found at https://github.com/kieran-nichols/ME_751_Code.git .

Contents

1. General information

- Mechanical Engineering
- PhD with specialty in Biomechanics
- One-person group: Kieran Nichols
- I release the ME751 Final Project code as open source and under a BSD3 license for unfettered use of it by any interested party.

2. Problem statement

My project centers around utilizing PyChrono and Machine Learning to create a controller to achieve walking for the multi-segmented bodies of a four-legged ant and humanoid (human like body with torso and two legs). PyChrono is a python base for use of Chrono. Chrono is a multi-physics (dynamics and kinematics) engine for modelling and simulation designed to be platform-independent and open source. Chrono is based in C++ object-oriented library and PyChrono places a python wrapper to utilize Chrono's functionalities.

I have struggled to figure out a project that will help me utilize some component of Chrono and give a useful tool for my future work. I've been always interested in Machine Learning and its implementation, so this project seemed a perfect fit for diving deep into the combination.

My goals that I hope to accomplish with this project:

- Set up of Gym-Chrono: operational use of the machine learning branch of PyChrono which allows training a robotic ant to walk.
- Adaptation of Gym-Chrono: creation of new humanoid model and adjustment of cost functions that affect how the ant and humanoid robots learning how to walk
- Scaling analysis to show differences of serial and parallel code

Interestingly, my research involves human standing and walking biomechanics, especially in relation to helping people regain their walking after stroke or other neuromuscular disorders. I have always been fascinated with how we regulate our walking in terms of hip, knee, and ankle joint torque. Walking to humans is a simple task but when engineers attempt to build robotic devices to mimic human walking with then we realize how difficult it is to control especially given perturbations.

This project starts with exploring the abilities of PyChrono and teaching a robotic ant how to walk. It will then take a simple attempt to teach a humanoid robot to walk and explore the effects of cost functions on the walking behavior. Lastly, results of scaling analyses, positions and velocities will be displayed for the robotic ant and humanoid.

3. Solution description

The project is contained in https://github.com/kieran-nichols/ME_751_Code.git. It is a repository that contains much of the necessary files to execute my project goals. Along with these files, a user must have the appropriate bash, python, PyChrono, and Tensorflow installations that is described in the "README.md" file in the "ME_751_Code/chrono-tensorflow".

Project folder structure within ME_751_Code

1) **ME_751_Code**
    2) **chrono-tensorflow**: project folder
        3) **PPO**: this folder contains the files to run the machine learning, graphing, physics engine
        3) **envs**: this folder contains the code for ant and humanoid robotic devices

There are "README.md" files at levels 1, 2, and 3, which describe the general content at those folder levels and how its programs can be used. To execute the machine learning of the humanoid, you must go into "ME_751_Code/chrono-tensorflow/PPO" and run the bash script "run_program.sh".

The flow of programs can be described by the bash file. There are many other ways to run the programs individually or together, but the way I chose to set up the bash file was for the sake of showing the installation process, activation of virtual environment, training serial and parallel machine learning, saving important and relevant info to files, and output the results in the form of graphs.

The machine learning for this project uses coding structures from Tensorflow. Tensorflow is an open-source machine learning platform that allows for easy model building and production of robust machine learning. To solve tasks, neural networks will be used in the process of Proximal Policy Optimization (PPO), which is a form of reinforcement learning. Reinforcement learning is the process promoting actions of a program in order to maximize its specified cumulative reward. PPO further specifies that as the machine learning be based in a stochastic gradient ascent (way of maximizing reward) to perform each set of iterations (in my case; ensuring that the robot stays up).

The tasks of the robotic ant and humanoid (environments/env) are to walk as close to a straight line as possible. The env is set up by specifying the physical bodies of the robot with its various segments defined by various constraints and having a floor. The ant is a sphere with four sets of upper and lower legs that is controlled with 8 motors (4 at the sphere-upper leg joints and 4 at the upper-lower leg joints). The humanoid comprises of a sphere with 2 sets of upper and lower leg segments with 4 motors. The motor torques are the actions of the machine learning.

Specific to this project, the good actions are promoted (to keep the robot walking) according to the cost function specified by the env. Basic cost functions used are costs related to use of electricity for the motors, having the joints at its prespecified limits, its sphere touching the floor (which also ends the trial), and it not walking in a straight line. As the robot attempts to walk it will try to minimize these costs by adjusting the motors.

To run the machine learning model, you have to state the number of total iterations and the batch size and if you want to use the serial or parallel code. The parallel code utilizes Python's multiprocessing module which could speed up the learning for trainings that last longer. Both the parallel and serial code should give similar results but may differ in execution time due to how the CPU is utilized.

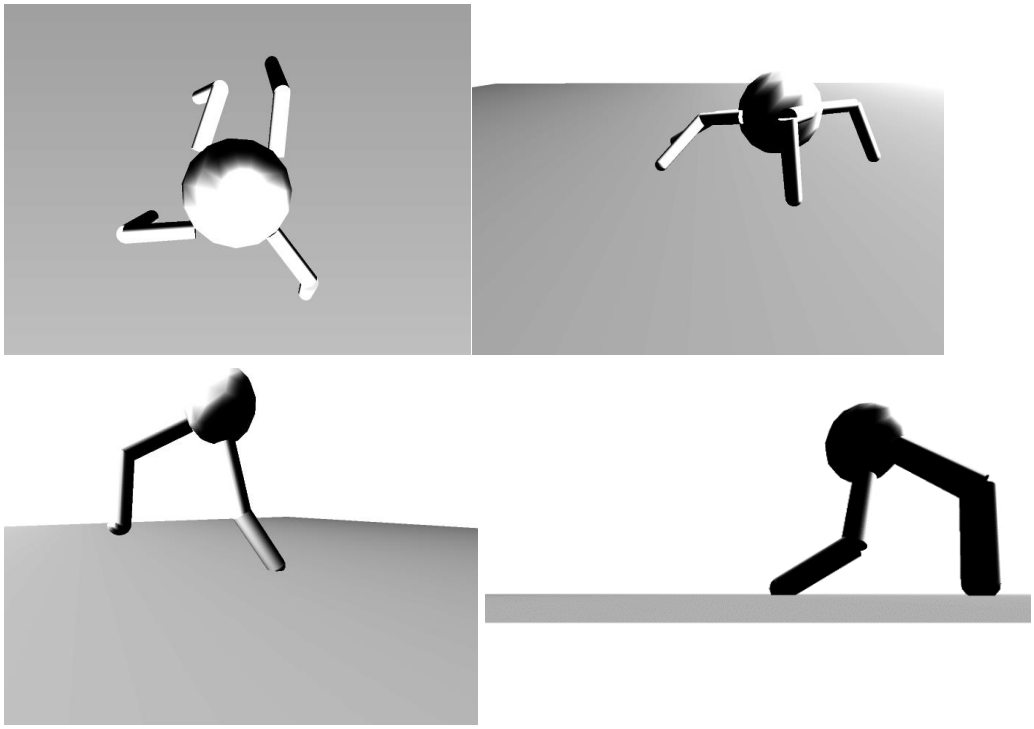
This project's output will comprise of graphs that will include scaling comparison of serial and parallel using the ant with batches of size 20 and variable sizes (1, 10, 100, 1000) of total number of iterations (episodes). Another graph will be a scaling comparison of serial and parallel codewith variable

batch sizes (1, 10, 100, 1000) and set episode size of 1000. To show the progression of learning, I chose to display the y position (forward direction) for different cost functions for the ant and humanoid. Additionally, I saved pictures and videos of both robots.

4. Overview of results. Demonstration of your project

Pictures of ant and humanoid

Figure 1, 2, 3, and 4 (clockwise order) show various pictures of the walking of the robot ant and humanoid



The geometry constraints for the ant were already set up in the PyChrono code. I adapted and added to the ant's code to create the humanoid. To test my geometric constraints, I utilized a freeze function (SetBodyFixed) within the code to I can check my constraints when the program render video.

Videos of successful and unsuccessful walking

The videos of the ant and humanoids walking can be found in my github repository.

Scaling analysis graphs using humanoid

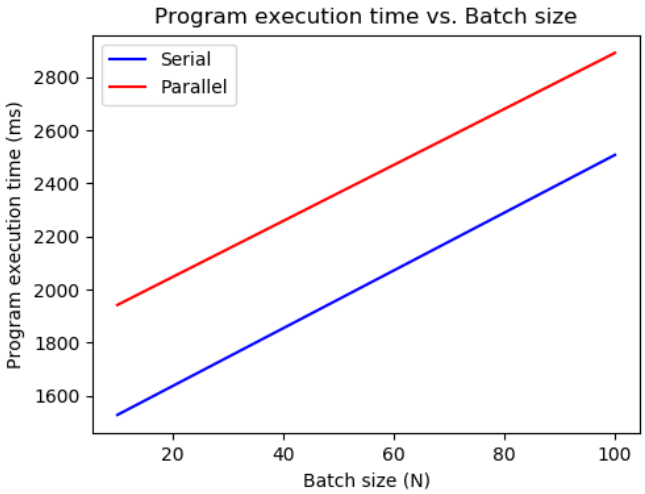


Figure 5 shows the program execution time vs batch size for the ant robot. There are constant slopes for the serial and parallel codes and a constant vertical offset between the two. This offset indicates that the parallel code consistently takes longer than the serial code. I originally expected the parallel code to have lower execution times for higher batch sizes but since execution time for an episode may still be too low (it was less than 3 seconds).

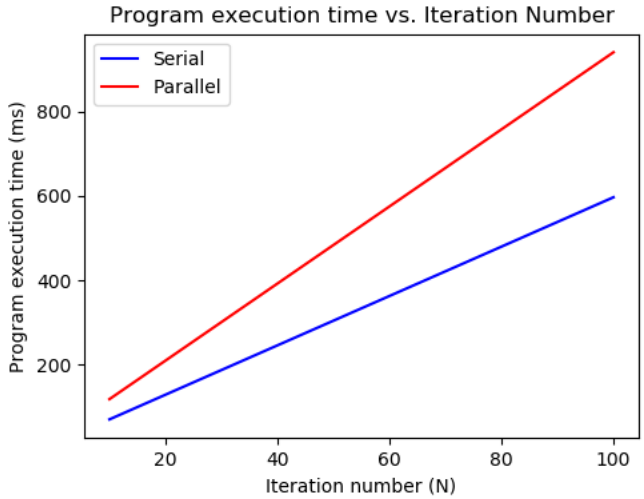


Figure 6 shows the program execution time vs Iteration Number (number of episodes) for the ant robot. There are constant slopes for the serial and parallel codes and an increasing vertical rise in execution time for lager iteration numbers.

State graphs

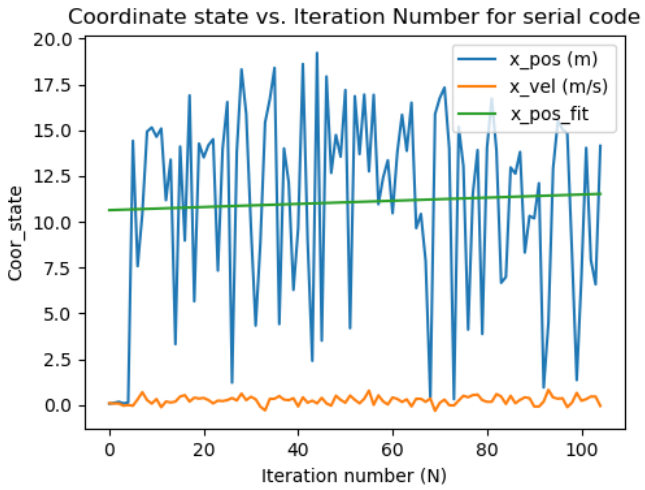


Figure 7 shows the x_pos, x_vel, and x_pos_fit of the ant robot. The x axis is the forward direction of the robot. The y position flucatuates for each iteration (episode) but looking at x_pos_fit, you can see that there is a general trend to move further forward. The goal of the humanoid is to move forward in a straight line.

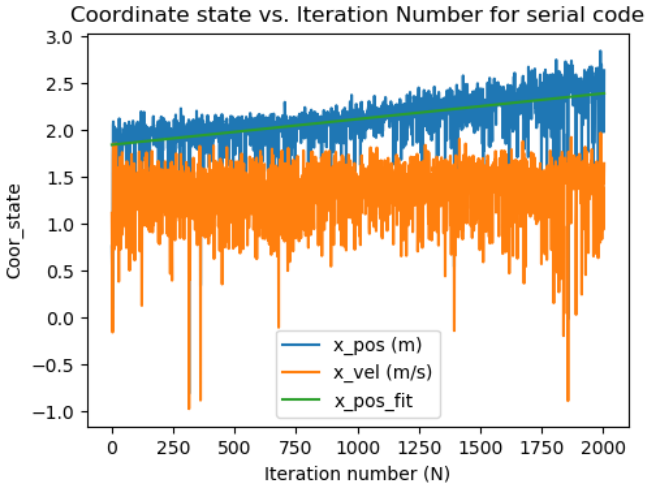


Figure 8 shows the x_pos, x_vel, and x_pos_fit of the humanoid robot. The x axis is the forward direction of the robot. The y position flucatuates for each iteration (episode) but looking at x_pos_fit, you can see that there is a more pronounced general trend to move further forward. A slightly curved green line would have been a better way of fitting the x_pos (blue). The goal of the humanoid is to move forward in a straight line.

5. Deliverables:

The following list will emphasize the importance of each of my deliverables:

1) Git Repository: my project will be publicly available
2) README.md files that help guide a user through my repository
3) This Project Report: a pdf copy will be placed in the project folder

To run my program, you can use the README.md files as guidance. Below is a summary of my README.md files:

Program Execution for ME_751_Code Project

1) Prerequisites

PyChrono/Chrono,
Tensorflow,
Bash,
Python

2) Installing

Bash: Use linux system (this project was tested on Ubuntu)
Python: Anaconda (conda)
PyChrono: Refer to http://projectchrono.org/pychrono/
Tensorflow:Refer to https://docs.anaconda.com/anaconda/user-guide/tasks/tensorflow/

3) Environments

ChronoAnt
ChronoHumanoid

4) Basic program execution (enter folder of ME_751_Code/chrono-tensorflow/PPO)

'python ./train_serial.py ChronoAnt -n 20000 -b 100 --renderON'

1. Use python or python3 command in a bash terminal
2. This command specifies that the serial learning will be utilized. `./train_parallel.py` can also be used for parallel learning. `./tester.py` can be used for running the episodes without further improving the learning.
3. 'ChronoAnt' will use the Ant robot. 'ChronoHumanoid' will use the humanoid robot. The robot are also called env (environment)
4. `-n 20000` says that 20000 episodes will be used
5. `-b 100` says that batches of 100 will be processed at a time for learning.
6. `--renderON` displays the robot and floor as it learns how to walk or fails to walk.

`Ctrl-C` doesn't usually work for stopping the python/bash program when using PyChrono and TensorFlow. `pkill -9 python` will kill the python kernel.

5) Other commands

Discount factor: `-g`,
Lambda: `-l`,
Kullback Leibler divergence target value: `-k`,
Help: `-h`

6) Project's high level program

This project has a shell script that can be run in bash to have outputs that log checkpoints for saving, save pictures of rendering, processing videos from pictures, scaling analysis of the serial and parallel codes, and create state graphs that describe the position and velocity of the robots.

To execute program, use `bash run_program.sh`

6. Conclusions and Future Work

This project gave me the ability to work with the robotic devices of ChronoAnt and ChronoHumanoid in a simulated environment whose kinematics and kinetics are governed by the Pychrono engine. Machine Learning using Tensorflow allowed me to train these robots to learn how to walk. I investigated the various properties of the serial and parallel versions of the learning and reported the various effects of the cost functions.

Admittedly, there are many parts of this project that I didn't fully understand but only learnt how to use for the sake of getting PyChrono and Tensorflow to work with the learning. Since this project was for the ME 751 class project that focuses on Computational Kinematics and Dynamics, I tried to focus the results around

If someone were to continue this project, I hope the future work will revolve around developing more suitable cost functions for each environment, setting appropriate low velocity and low acceleration costs and inclusion of costs that promote stepping behavior that mimic ant or mammal. The humanoid could have added segments like arms, and more representative mass and inertia. Humanoid could have costs that minimis high velocities and accelerations.

I appreciated playing with the abilities of PyChrono and Tensorflow and creating a humanoid within the Chrono environment. Machine Learning offers amazing tools that can help solve complex problems. I hope researchers in the future take advantage of these software packages.

References

1) https://en.wikipedia.org/wiki/Reinforcement_learning
2) https://www.tensorflow.org/
3) https://medium.com/@iKhushPatel/convert-video-to-images-images-to-video-using-opencv-python-db27a128a481
4) https://git.cae.wisc.edu/git/me759-knichols4
5) https://git.cae.wisc.edu/git/me459-knichols4
6) https://github.com/projectchrono/gym-chrono
7) http://api.projectchrono.org/development/tutorial_pychrono_demo_tensorflow.html
8) https://arxiv.org/abs/1707.06347
9) http://api.projectchrono.org/development/tutorial_table_of_content_pychrono.html
10) https://gist.github.com/PurpleBooth/109311bb0361f32d87a2
11) https://docs.anaconda.com/anaconda/user-guide/tasks/tensorflow/