Sociedade Brasileira de Automática (SBA)
XXIV Congresso Brasileiro de Automática - CBA 2022, 16 a 19 de outubro de 2022

Simulation of Complex Actuator of a Large Agricultural Robot for Deep Reinforcement Learning

Gabriel A.B. Arias [*] **Arthur J.V. Porto** [*]

[*] *EESC - São Carlos School of Engineering, USP – University of São Paulo, São Carlos, Brazil (e-mail: gabriel.bermudez@usp.br; ajvporto@sc.usp.br).*

Abstract: Testing with robots and especially with large agricultural robots is a task that requires high costs, risks and depends on weather conditions. Although there is a gap between simulation environments and real environments, simulation environments offer the advantages of being totally safe, making it possible to verify the performance of the control algorithm and even have the possibility of simulating sensors and actuators that have not yet been physically implemented. In addition, simulation enable risk-free assessment and adjustment of new control methods before implementation in real systems. In this work, we propose an approach to run experiments with Deep Reinforcement Learning (DRL) algorithms using Robot Operating System (ROS) and Gazebo robotics simulator. For this purpose, we use the `robo-gym` framework to interface between the DRL algorithm and the simulation using an OpenAI Gym environment, and augmented the capacity of Gazebo using the `gazebo-fmi-actuator` plugin that allows co-simulation with Functional Mock-up Unit (FMU). It is also presented an application of the simulation and control of the hydraulic steering system of a large agricultural robot using a Deep Deterministic Policy Gradient (DDPG), Soft Actor Critic (SAC), and Twin Delayed DDPG (TD3) DRL algorithms and comparing they with a PID controller.

Keywords: Simulation, Reinforcement learning control, Functional Mock-up Unit, Robot Operating System

1. INTRODUCTION

Autonomous agricultural robots must move in a dynamic and semi-structured environment, they need to be prepared to operate in open field subjected to different types of terrain, regular and irregular, non-homogeneous, changeable and unpredictable, which presents a difficult challenge for the steering controller design (Taghia and Katupitiya, 2020; Sparrow and Howard, 2021; Oliveira et al., 2021). Concerning Artificial Intelligence, this means dealing with an open world, so the techniques to allow for adaptation during operation rather than in the design phase are crucial. Techniques that allow robots to learn from experience include reinforcement learning, demonstration learning, and transfer learning (Duckett et al., 2018).

Testing in real environment is costly, simulators allow risk-free evaluation and adjustment of new control methods before implementation in real systems. Developing a virtual model for a robot helps in testing and experimenting in different environments, so the risk to the real robot can be mitigated by detecting some possible errors and design flaws before any real world experimentation is conducted, which leads to increased confidence in the robustness of the developed solutions and reduces the time required for integration and failure detection (Young, 2019).

A description and comparison of robotic simulation software between V-REP, Gazebo, ARGos, Actin is presented in Shamshiri et al. (2018), and a comparison in the scope of robotics and RL between Gazebo, MuJoCo, PyBullet and Webbots in presented in Körber et al. (2021), these simulators are based on rigid body physics engines. Although the simulator features are similar, Gazebo integration with ROS is an asset if the control interface for the simulated and real robot is envisaged to be the same. However, some actuators cannot be expressed using the Gazebo physics engine. An approach is to develop a custom gazebo plugin for the actuator, it requires a C++ programming expertise and significant knowledge of the Gazebo simulator (Lange et al., 2021).

Reinforcement learning algorithms require trial and error tests. Therefore, a simulation environment makes the learning process safe, being possible to detect errors and failures before it is implemented in the real robot. Although ROS and Gazebo are excellent tools for developing and simulating robots, they are not designed for running RL experiment, since they do not have sufficient tools to train an RL agent in a complex environment. Furthermore, Gazebo only simulates the rigid body dynamics, but it is not enough to simulate the complexity of some robotics actuators. This paper presents an approach to run DRL

⋆ This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001

experiments in a simulated robot with complex actuators using open-source software. It is also presented an application of the simulation of the steering control of the wheel of a large agricultural robot using a DDPG, SAC and TD3 algorithms and comparing they with a PID controller.

2. METHODOLOGY

We decided to use Gazebo simulator and ROS middleware to develop the simulation environment, due to the compatibility and large community and acceptance in the academic and industrial sector. Furthermore, we use the `robo-gym` framework to develop the environment and train the RL agent (Lucchi et al., 2020) and use the `gazebo-fmi-actuator` plugin for augmenting the capacity of Gazebo simulator allowing to co-simulate FMUs inside Gazebo (Lange et al., 2021). In the following sections, we describe the procedure to develop the simulation environment to run DRL experiments on a simulated robot.

2.1 Modeling method

The first step is to develop the robot model description in the Unified Robot Description Format (URDF), which is a standard specification for describing robots. The procedure for the development of the URDF can be seen in Fig. 1. The URDF defines the robot model representation in Extensible Markup Language (`XML`) format, using a description of links and joints, where are defined the kinematic, dynamic, visual, and collision properties.

Links The links contain name information, physical mass and moment of inertia parameters, visual and collision properties.

Joints Joints are defined by name, type, parent link, and child link properties. Optionally, joints can be also defined by properties corresponding to each type of joint.

Transmissions Transmissions are an extension of the `URDF` robot description that associates a joint with an actuator, this allows the transmission to have a desired control of the robot links and allows the interface between a real robot and a simulated one in Gazebo.

2.2 `robo-gym` framework

The `robo-gym` framework is an open-source toolkit to run DRL experiments on real ad simulated robots. The framework is shown in Fig. 2 with dash lines, its principal components are the Robot Server (RS), Robot Manager (RM) and the Command Handler (CH). `robo-gym` uses the OpenAI gym interface and allow using open-source implementation of DRL algorithms (Lucchi et al., 2020).

To create a `robo-gym` custom environment, it is necessary to adapt the current environment to send the robot states and receive commands to execute through the RS using Google Remote Procedure Call (gRPC) messages. The RS interacts with the ROS Bridge routine, which is a ROS routine that collects the state of the robot and receive the commands to send to the robot. It is also necessary to create a custom CH routine to adapt the actions command received from the ROS Bridge to as a Markov Decision

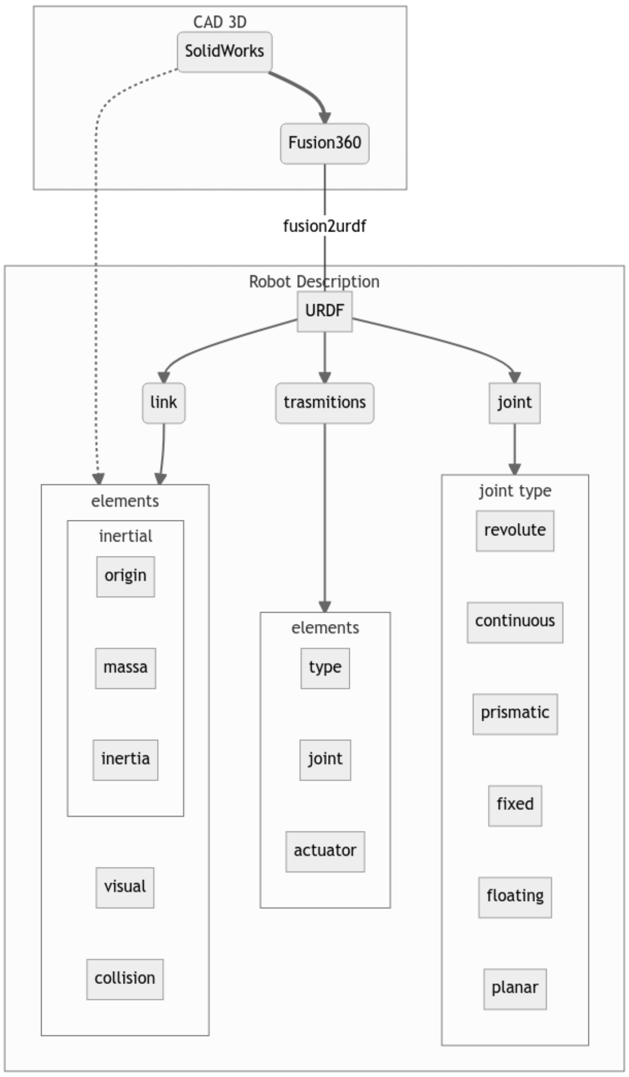

Figure 1. URDF development procedure

Processes to send discrete commands in a controlled period of time. Furthermore, it is recommended to create a routine to publish the state of the robot to a ROS topic that can be accessed by the ROS Bridge. Finally, create a OpenAI gym custom environment to train the RL agent and receive the states and send actions command to the Robot Server. With the custom environment, we can run experiments using an open-source implementation like Stable baselines (Hill et al., 2018) or with customs algorithms.

2.3 `gazebo-fmi-actuator` plugin

The `gazebo-fmi-actuator` is one of the plugins in the `gazebo-fmi`, it permits co-simulation of an actuator dynamic inside Gazebo simulator. With this plugin, it is possible to simulate the actuator dynamics that Gazebo by itself cannot. The `gazebo-fmi-actuator` plugin can be used transparently with any joint specified in the URDF, and it can be controlled using `ros_control` package and `gazebo_ros_pkgs`.

The actuator plugin is limited to one degree of freedom model, this mean that it can only have a single actuator

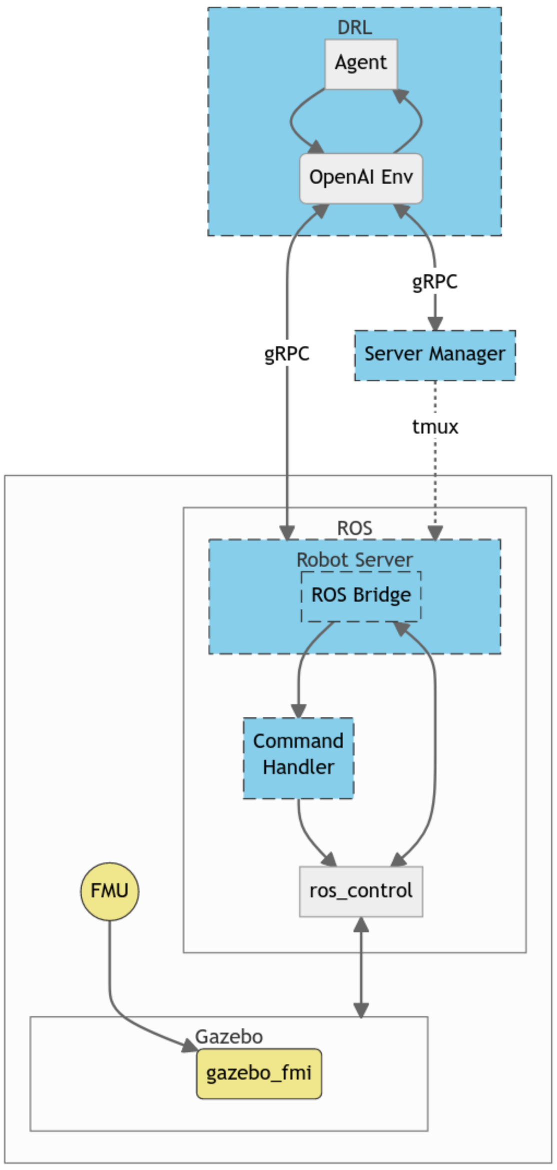

Figure 2. Simulation environment development method

controlled input (`actuatorInput`) and a single actuator output `jointTorque`. The input is the control signal and could have any unit, this input can read the value of the joint force that is set by the `ros_control`. The output is always a torque in the case of a revolute joint or a force in the case of a prismatic joint. The plugin has tree additional inputs that represent the joint position (`jointPosition`), the joint velocity (`jointVelocity`), and the joint acceleration (`jointAcceleration`), this input cannot be controlled and are obtained from the Gazebo joints (Lange et al., 2021).

Co-Simulation Functional Mock-Up Units The Co-Simulation FMU is a file that contains their own numerical solver, specified by the Functional Mock-Up Interface (FMI) standard (Blochwitz et al., 2012). The FMU can be generated using any modeling software that support the FMI standard and that has a tool to export the simulation model to an FMU. Then, we can execute the FMU during

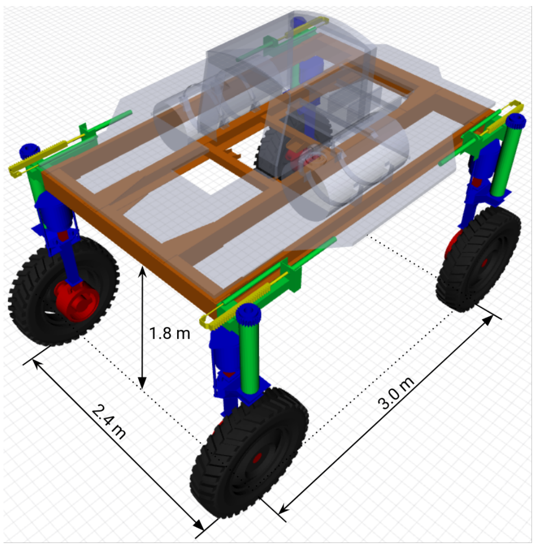

Figure 3. Agribot Dimensions

a simulation run with another software that can import the FMU.

There are two main limitations to modeling the FMU to use with the `gazebo-fmi-actuator` plugin. The first is the FMU Co-Simulation solver, that could be not powerful enough to solve very complex models, like OpenModelica uses the Euler solver as default. The second is the Gazebo physics `max_step_size` parameter, that limits the FMU integration step size. The `max_step_size` parameter default value is 0.001 second, high step size could cause high FMU integrations errors that could cause a poor simulation or even the FMU couldn't solve the model. On the other hand, low size step values increase the simulation time, that is a major factor in the DRL agent training.

3. APPLICATION

As application, we developed the simulation environment for the Agribot, the control task is a servo problem for steering the wheel to a given reference. The Agribot is a large modular robotic platform capable of moving in typical environments in the agricultural area for data acquisition and research. As show in Fig. 3, the Agribot has a structure with a gantry format with 1.8 m between the ground and the base of the chassis, it also has an adjustable gauge system that can be adjusted from 2.25 m to 2.4 m that serves to adjust the front distance between vehicle axles. It has a mechanical system consisting of a diesel engine that provides power for the hydraulic propulsion system and hydraulic steering system. (Tabile, 2012; Torres, 2014; Archila, 2016).

The development of the simulation environment, start with the input information of previous works, the Agribot 3D `SolidWorks` model, and the Hydraulic Steering System circuit drawings. With the 3D CAD model, we create the URDF, and with the hydraulic steering circuit drawings, we develop the simulation model to export to FMU. Finally, we use the framework `robo-gym` to create the simulation to train the RL agent in the simulated environment. The next sections will describe in more detail each part of the followed procedure.

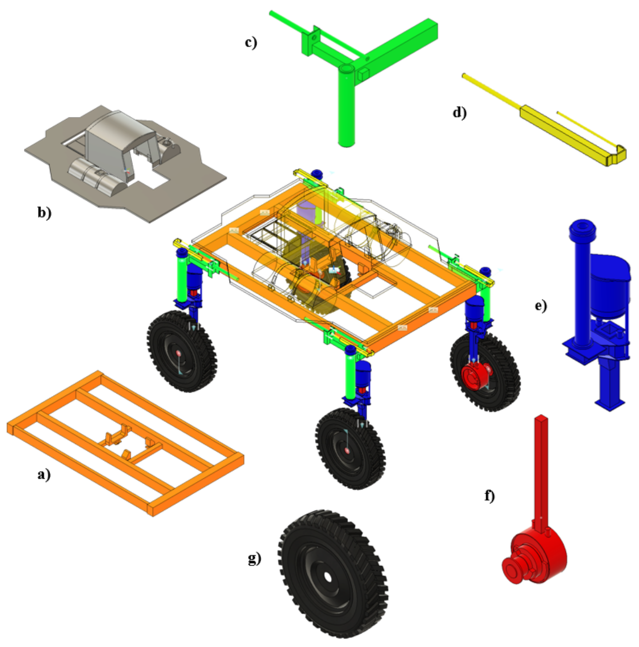

Figure 4. Agribot Links

3.1 Agribot Robot Description modeling

The Agribot was divided into 22 links, Fig. 4 shows the links by color where: a) Main Chassis ; b) Upper Chassis ; for each of the steering modules (front left, front right, rear left and rear right) c) Steering Module Support; d) Hydraulic Cylinder ; e) Steering Module; f) Suspension Module and g) Wheel.

The diagram shown in Fig. 5 shows the hierarchy of links and joints simplified for the general case of a wheel. The a) Main Chassis is the base link that has as child link the b) Upper Chassis links and the c) Steering Module Support link, with fixed joints each one. The Steering Module Support includes a hydraulic cylinder that has two children link, the d) Hydraulic Cylinder Rod link with a prismatic joint, and the e) Steering Module link with a revolute joint. There is a rack and pinion mechanism between the d) Hydraulic Cylinder Rod (rack) and e) Steering Module (pinion), the hydraulic cylinder in the c) Steering Module Support moves the rack that transmits the movement to the pinion to steer the wheel. There is also a closed kinematic chain between the links c) Steering Module Support, d) Hydraulic Cylinder and e) Steering Module, to represent this joint, a mimic property is created inside the e) Steering Module joint that mimics the movement of the d) Hydraulic Cylinder Rod joint by a conversion factor, in this case, it transmits the movement of the rack linear with rotational movement of the pinion. The e) Steering Module has as a child link the f) Suspension Module with a prismatic joint, and the f) Suspension Module has as a child link the g) Wheel, which moves the robot due to the friction of the wheel with the floor.

For the simplicity of the model, and due to the mimic joint only work for visualization in RViz and not in Gazebo, the prismatic joint of the e) Steering Module Support and d) Hydraulic Cylinder Rod links and the prismatic joint of the e) Steering Module and f) Suspension Module links are set to fixed joints. Thus, the revolute joint between the c) Steering Module Support and the e)Steering Module, and the continuous joint between the f) Suspension Module and the g) Wheel, are the two only mobile joints in the URDF model.

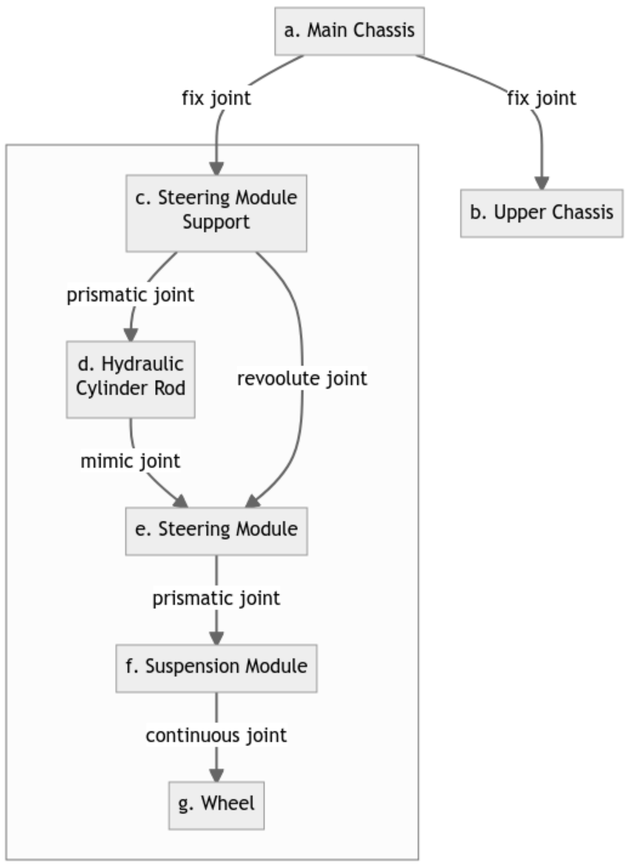

Figure 5. Diagram of Agribot links and joints

The Agribot URDF model was created based on the Agribot 3D detailed model provided in `SolidWorks` 3D model. The 3D `SolidWorks` model was exported to `Autodesk Fusion 360` software to remove all unnecessary details for the simulation, merge the fixed parts, and set the joints for mobile links. Fig. 4 shows the simplified 3D model, which is composed of only 22 parts out of the 948 parts of the original model. The initial URDF was created thanks to the `Fusion2URDF` script (Kitamura, 2020) from the 3D simplifies model, then it was modified to have all Agribot properties described above. The inertia and mass properties of the URDF were taken from the `SolidWorks` 3D detailed model.

3.2 Hydraulic Steering System Modeling

The hydraulic steering system is responsible for the movements of the wheels through a hydraulic cylinder, using a closed-loop control. It has as a control signal the duty cycle of a PWM (0% to 100%) in the proportional valve of the actuator and the signal from the linear potentiometer as feedback. The parameters of the hydraulic model were obtained from the manuals of the hydraulic pump, proportional valve Hydraforce (2008) and hydraulic actuator manufacturers and the data provided in the works of Torres (2014).

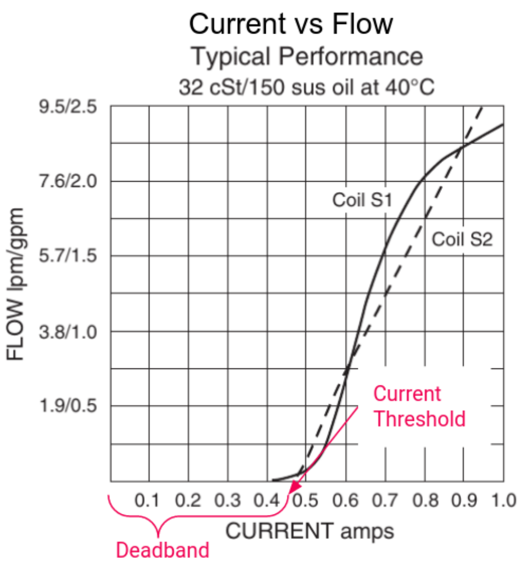

Figure 6. Diagram typical valve Performance

In the work of Tabile (2012) describes some problems related to the steering hydraulic system that hinder the development of the control of the steering system. The principal issues are the delay of the hydraulic system due to the slow response time and high inertia of the hydraulic actuators; the non-linearity and saturation limit of the electrohydraulic cartridge proportional valve solenoids as shown in Fig. 6; difference between the area of the piston cylinder and the rod; and the inertia of the steering system due to friction in different terrains.

The `gazebo-fmi-actuator` plugin is used transparently with the steering joints of the Agribot, and is called in the start of the simulation. The CH sent the action to the joint effort controller managed by the `ros_control` package, then `gazebo_ros_control` set as an effort to gazebo. The action is not necessary an effort command it could be voltage in case of electric motors, in this case we use the duty cycle percent of PWM of the directional valve of the actuator. The `gazebo-fmi-actuator` plugin reads the effort value that was set, and it passes to the FMU as the actuator Input, it then runs the FMU simulation, and returns the result effort to the joint to continue the Gazebo simulation. The flowchart in Fig. 3 shows the proposed solution using robo-gym framework for a simulated environment using gazebo and the `gazebo-fmi-actuator` plugin. We use the OpenModelica modeling tool to develop the hydraulic steering system, using lookup tables to emulate the non-linearity of the proportional valve, then export the model to a FMU.

3.3 The Learning algorithm

To showcase the difference between using the simulation of the FMU and to compare DRL algorithm with a classical PID control, we run multiple experiments with six configurations:

(1) no FMU + PID
(2) no FMU + DDPG
(3) FMU + PID
(4) FMU + DDPG
(5) FMU + SAC

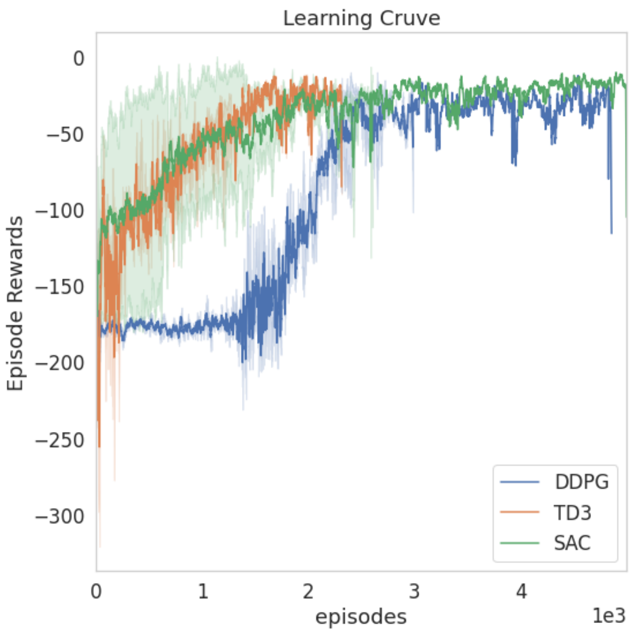

Figure 7. Learning Curve

(6) FMU + TD3

We use the DDPG, SAC and TD3 algorithm proposed by Lillicrap et al. (2019), Haarnoja et al. (2018) and Fujimoto et al. (2018) respectively. We also use the Stable baseline open-source standard implementation of the algorithms to run the experiments. The hyperparameter using in the experiments are the default, and we use the policy `LnMlpPolicy` that implements actor critic, using an MLP (2 layers of 64), with layer normalization (Hill et al., 2018).

The observation space was the current angle of joint (θ_{joint}), the reference angle (θ_{ref}) and the joint velocity (ω_{joint}) . The action space was the duty cycle percent of PWM of the proportional valve between -1 to 1. The reward function is shown in equation (1). We train the agent using the FMU simulation with 5000 episodes of 100 time steps.

$$r(s, a, s') = -(|\theta_{ref} - \theta_{joint}| + 0.01\omega_{joint}^2) \qquad (1)$$

4. EXPERIMENTAL RESULTS

The agent learning curve is shown in Fig. 7 for the three algorithms. The best agent model was saved to be used for comparative with a PID controller for the FMU experiments. The Fig. 8 shows the result for the simulation without FMU for the DDPG agent, in these case the performance of PID controller is slightly better than the DDPG algorithm, worth noting that the DDPG agent was trained only using the FMU simulation. In Fig. 9 is shown the result of the test using the FMU simulation, in this case we use the DDPG, SAC and TD3 algorithms. Further work is necessary to enhance hydraulic steering system model to include more components, in the current simulation only was included the proportional valve solenoids non-linearity using lookup tables, if we include more complexity in the hydraulic model is expected that DRL agents greatly exceed the PID controller.

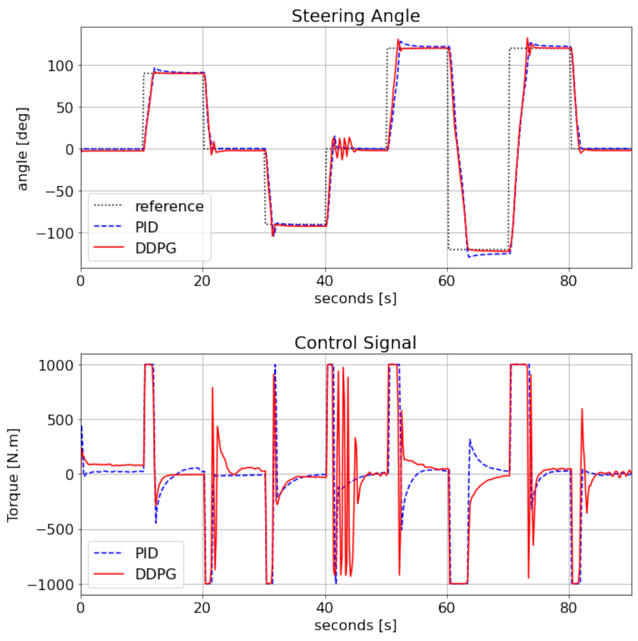

Figure 8. Test with no FMU

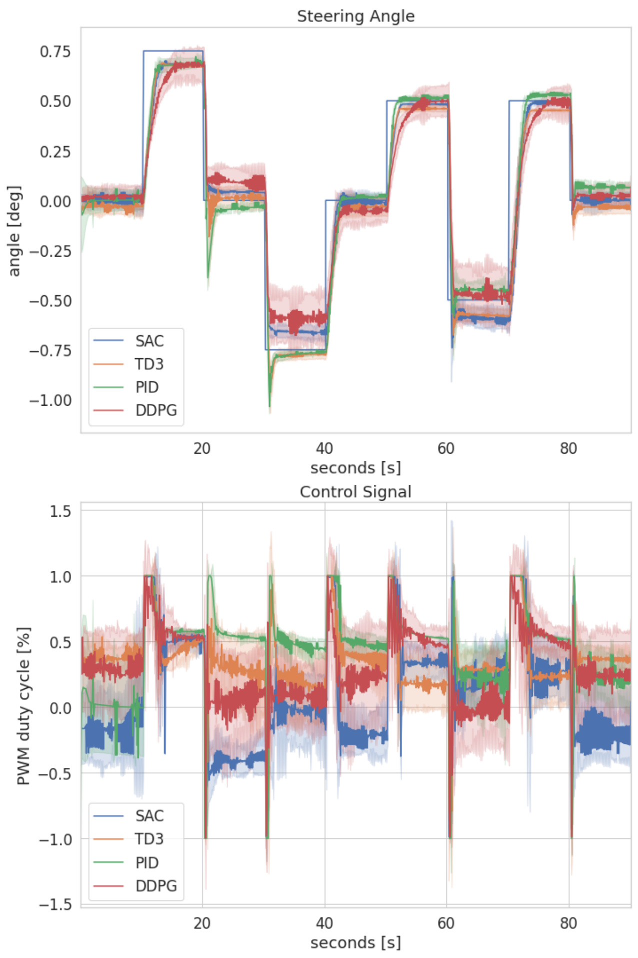

Figure 9. Test with FMU

5. FUTURE WORK

The present work proposes an approach to simulate complex actuators inside a simulation environment to run DRL experiments applied to control of a hydraulic actuator to steering the wheel of a large agricultural robot. Further investigation is required to validate the simulation model, especially the FMU of the hydraulic actuator. Furthermore, it is necessary to evaluate the performance of the DRL algorithm, since robo-gym permit to train a simulated and real robots using a common interface and allows running parallel instances.

ACKNOWLEDGMENT

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001

REFERENCES

Archila, J.F. (2016). *Design of a Rover to Precision Agriculture Applications*. Text, Universidade de São Paulo. doi:10.11606/T.18.2017.tde-21112017-160424. URL `http://www.teses.usp.br/teses/disponiveis/18/18149/tde-21112017-160424/`.

Blochwitz, T., Otter, M., Akesson, J., Arnold, M., Clau&szlig, Christoph, Elmqvist, H., Friedrich, M., Junghanns, A., Mau&szlig, Jakob, Neumerkel, D., Olsson, H., and Viel, A. (2012). Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. URL `https://ep.liu.se/en/conference-article.aspx?series=ecp&issue=76&Article_No=17`.

Duckett, T., Pearson, S., Blackmore, S., Grieve, B., Chen, W.H., Cielniak, G., Cleaversmith, J., Dai, J., Davis, S., Fox, C., From, P., Georgilas, I., Gill, R., Gould, I., Hanheide, M., Hunter, A., Iida, F., Mihalyova, L., Nefti-Meziani, S., Neumann, G., Paoletti, P., Pridmore, T., Ross, D., Smith, M., Stoelen, M., Swainson, M., Wane, S., Wilson, P., Wright, I., and Yang, G.Z. (2018). Agricultural Robotics: The Future of Robotic Agriculture. *arXiv:1806.06762 [cs]*. URL `http://arxiv.org/abs/1806.06762`.

Fujimoto, S., van Hoof, H., and Meger, D. (2018). Addressing Function Approximation Error in Actor-Critic Methods. doi:10.48550/arXiv.1802.09477. URL `https://arxiv.org/abs/1802.09477v3`.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. doi:10.48550/arXiv.1801.01290. URL `https://arxiv.org/abs/1801.01290v2`.

Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. (2018). Stable baselines. `https://github.com/hill-a/stable-baselines`.

Hydraforce (2008). *Electro-Hydraulic Proportional Valves*. URL `https://www.hydraforce.com/globalassets/forms/proportional-manual.pdf`.

Kitamura, T. (2020). Fusion2URDF. GitHub. URL `https://github.com/syuntoku14/fusion2urdf`.

Körber, M., Lange, J., Rediske, S., Steinmann, S., and Glück, R. (2021). Comparing Popular Simulation Environments in the Scope of Robotics and Reinforcement Learning. *arXiv:2103.04616 [cs]*. URL `http://arxiv.org/abs/2103.04616`.

Lange, R., Traversaro, S., Lenord, O., and Bertsch, C. (2021). Integrating the Functional Mock-Up Interface with ROS and Gazebo. In A. Koubaa (ed.), *Robot Operating System (ROS): The Complete Reference (Volume 5)*, Studies in Computational Intelligence, 187–231. Springer International Publishing, Cham. doi:10.1007/978-3-030-45956-7_7. URL `https://doi.org/10.1007/978-3-030-45956-7_7`.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2019). Continuous control with deep reinforcement learning. *arXiv:1509.02971 [cs, stat]*. URL `http://arxiv.org/abs/1509.02971`.

Lucchi, M., Zindler, F., Mühlbacher-Karrer, S., and Pichler, H. (2020). Robo-gym – An Open Source Toolkit for Distributed Deep Reinforcement Learning on Real and Simulated Robots. *arXiv:2007.02753 [cs]*. URL `http://arxiv.org/abs/2007.02753`.

Oliveira, L.F.P., Moreira, A.P., and Silva, M.F. (2021). Advances in Agriculture Robotics: A State-of-the-Art Review and Challenges Ahead. *Robotics*, 10(2), 52. doi:10.3390/robotics10020052. URL `https://www.mdpi.com/2218-6581/10/2/52`.

Shamshiri, R.R., Hameed, I.A., Pitonakova, L., Weltzien, C., Balasundram, S.K., Yule, I.J., Grift, T.E., and Chowdhary, G. (2018). Simulation software and virtual environments for acceleration of agricultural robotics: Features highlights and performance comparison. *International Journal of Agricultural and Biological Engineering*, 11(4), 15–31. doi:10.25165/ijabe.v11i4.4032. URL `https://ijabe.org/index.php/ijabe/article/view/4032`.

Sparrow, R. and Howard, M. (2021). Robots in agriculture: Prospects, impacts, ethics, and policy. *Precision Agriculture*, 22(3), 818–833. doi:10.1007/s11119-020-09757-9. URL `https://doi.org/10.1007/s11119-020-09757-9`.

Tabile, R.A. (2012). *Desenvolvimento de uma plataforma robótica modular e multifuncional para aquisição de dados em agricultura de precisão*. Tese de Doutorado, Universidade de São Paulo. doi:10.11606/T.18.2012.tde-05072013-101540. URL `http://www.teses.usp.br/teses/disponiveis/18/18145/tde-05072013-101540/`.

Taghia, J. and Katupitiya, J. (2020). *Applied Guidance Methodologies for Off-road Vehicles*. Springer Tracts in Advanced Robotics. Springer International Publishing. doi:10.1007/978-3-030-42359-9. URL `https://www.springer.com/gp/book/9783030423582`.

Torres, C.J. (2014). *Arquitetura supervisória aplicável na robótica agrícola móvel*. Dissertação de Mestrado, Universidade de São Paulo, TESE 8776. doi:10.11606/D.18.2017.tde-16112017-113717. URL `http://www.teses.usp.br/teses/disponiveis/18/18145/tde-16112017-113717/`.

Young, H. (2019). *ARDEE: A General Agricultural Robotic Development and Evaluation Environment*. Ph.D. thesis.