Simulating Robotics Planning Domains with PDSim and ROS

Emanuele De Pellegrin,[1,2] **Ronald P. A. Petrick**[1]

Edinburgh Centre for Robotics
[1]Heriot-Watt University
[2]University of Edinburgh
Edinburgh, Scotland, United Kingdom
ed50@hw.ac.uk, R.Petrick@hw.ac.uk

Abstract

This paper describes work on the Planning Domain Simulation System (PDSim) and its application to robotics. PDSim is a plugin for the Unity game engine for visualising planning domains and plans. PDSim's original system design translates the output of a planner to 2D or 3D animations and effects. PDSim aims to assist users in evaluating the quality of a plan and improve domain and problem modelling. This system demonstration outlines the basic structure of PDSim and how to integrate it with the Robotics Operating System (ROS) to simulate plans in robotics domains. An example with a robotic arm is used to showcase how to interface with ROSbags to be able to visualise sensors from an Internet of Things (IoT) environment used for daily assistive living. Furthermore, the demo also shows how to interact with the robotics PDSim API for plan repair and replanning.

Introduction

Modelling planning domains and ensuring plan accuracy can be difficult, particularly when tackling real-world problems. While plans may be considered valid, relying exclusively on the output of planners may not always enable the identification of mistakes in domain modelling that would become apparent when represented visually, such as through a 2D or 3D visualization (Chen et al. 2020). This is an active research topic, with many approaches and tools available for visualising plans and assisting users in understanding how a plan is generated, in order to detect potential errors in the modelling process (Vaquero et al. 2007; Chen et al. 2020; Tapia, San Segundo, and Artieda 2015; Muise 2016; Le Bras et al. 2020; Roberts et al. 2021; Shah et al. 2021).

The Planning Domain Simulation (PDSim) system (De Pellegrin and Petrick 2022, 2023), introduced a novel method for visualising and simulating planning problems defined in PDDL by harnessing the framework and components of the Unity game engine (Unity Technologies 2022) to deliver a 3D or 2D visualisation of the planning problem. Moreover, PDSim can make use of the inherent modularity of the game engine to interface with the Robotics Operating System (ROS) (Quigley et al. 2009; Maruyama, Kato, and Azumi 2016), enabling it to simulate robotics domains and use the state-of-the-art robotics libraries for motion planning, navigation, and teleoperation, among others.

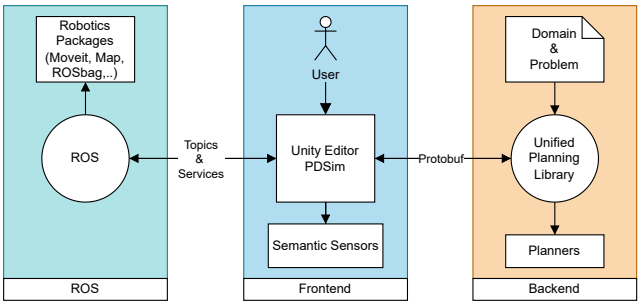

Figure 1: General PDSim system architecture with its three main components: the Unity frontend, the Unified Planning Library backend that connects to external planners, and the ROS connection for integrating robotics packages.

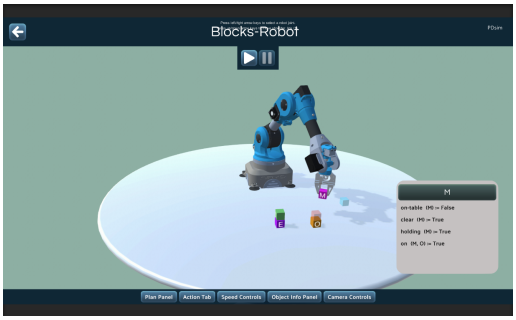

Figure 2: Blocks world domain using a Niryo robot.[1]

Simulators are widely used in robotics applications and can reduce the effort for the development process, letting the user focus on functionality and evaluation of the simulation (Echeverria et al. 2011; Koenig and Howard 2004). In particular, simulators that use game engines such as MORSE (Echeverria et al. 2011) or Drone Sim Lab (Ganoni and Mukundan 2017) are also prevalent in robotics applications. A game engine offers benefits like multiple cameras to follow the simulation, a physics engine, and realistic post-processing effects with no need to implement them from scratch (Ganoni and Mukundan 2017).

PDSim provides users with the ability to create real-world scenes that reflect the execution environment of the planning problem, exploiting the functionality of the Unity game engine as an enhancement to existing automated planning tools. The key difference between PDSim and other simulation tools lies in its primary focus on automated planning. In PDSim, both the physics and animations represent changes in the semantic state of the simulated 3D world. This unique feature enables PDSim to serve as a tool for visualising and inspecting plans and act as a digital twin for robotics applications that rely on planning for goal-directed decision making. Furthermore, PDSim can be used as a simulator for learning action models in robotics. This is achieved through its customizable interface, which allows users to define the simulated outcome of action effects within the virtual environment. Additionally, by attaching virtual sensors, PDSim enables modification of the semantic meaning of physical interactions with the environment.

PDSim Overview

PDSim extends the Unity game engine editor (Unity Technologies 2022) and is able to use components offered by the engine such as a path planner, lighting system, and scene management, among others. The general PDSim architecture diagram in Figure 1 shows the structure of PDSim and its main three components: the Unity front end, the Unified Planning Library (UPL) back end, and the ROS middleware. The front end is responsible for handling the rendering and for mapping planning actions and fluents to animations. The back end runs the Unified Planning Library (UPL), a Python library provided by the AIPlan4EU project.[2] Finally, ROS communicates with Unity to provide robot-related information, such as joint angles, transform data, and camera feedback, depending on the sensors available to the robot being simulated. Through Unity, the user can use and customise PDSim's 'semantic sensors' which can be attached to planning objects. For instance, Figure 4 shows a ray sensor that checks if a cube is colliding with the table during planning time by modifying the 'on-table' predicate if a collision occurs. This is particularly useful for the simulation to provide feedback if something goes wrong during action execution: a user could use this information together with PDSim's API as input to a replanning or plan repair process.

PDSim Demo

In this system demonstration, we will present a set of simulations and visualisation projects to illustrate the main features of PDSim. In particular, the presented projects will include: how to set up a visualisation environment starting from the domain and problem definition and its pipeline with Unity, how to set up a robotics simulation in PDSim for solving a robotics planning problem, and how to use PDSim as a digital twin in an IoT assistive environment.

For example, Figure 2 shows how PDSim can use the high-level blocks world planning domain with the low-level actions of a robotic arm to perform a pick and place task. Figure 3 shows how PDSim's predicate animation system is used in a digital twin environment, where predicates are defined as sensors in an IoT assistive daily living lab. This example explores how to use PDSim's semantic mapping between a recorded ROSbag file and its Unity 3D visualisation in a real-world IoT environment.

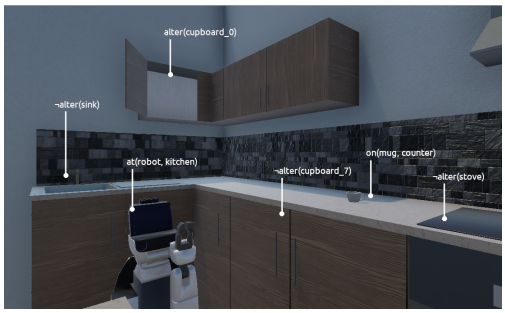

Figure 3: PDSim for real-world robotics: HSR robot (Yamamoto et al. 2019) representing semantic sensor data in a ROSbag recording can be replayed as a 3D animation.

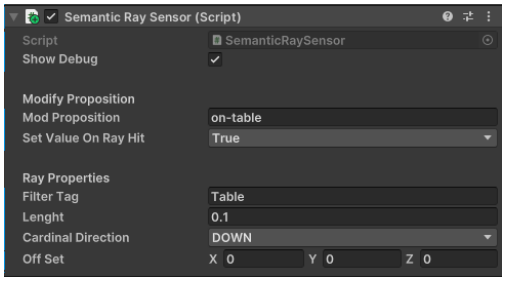

Figure 4: A semantic sensor component attached to a planning object in Unity using a configurable ray sensor.

Video example

Two video examples are available online: (1) a video showing how to use PDSim to set up a visualization from the ground up,[3] and (2) a video showing the blocks world example simulation using ROS.[4]

Conclusion

This paper presents an overview of the PDSim system, an extension to the Unity game engine to simulate planning domains and plans. This system demonstration will showcase the main PDSim Unity interface and its integration with ROS for robotics and automated planning simulation. The demo will walk through all the requirements and steps required to create a simulation from scratch, and how to extend PDSim's capabilities by using its API to access semantic information from the simulation.

[1] https://niryo.com/

[2] https://www.aiplan4eu-project.eu/

[3] https://drive.google.com/file/d/1AHlcYkadRa1ndJp7sxpC2VE0OTEZh0ii/view?usp=sharing

[4] https://drive.google.com/file/d/1SFz1UKWtNG1Mszs0ZDUe5CwjBapfwbm4/view?usp=sharing

References

Chen, G.; Ding, Y.; Edwards, H.; Chau, C. H.; Hou, S.; Johnson, G.; Sharukh Syed, M.; Tang, H.; Wu, Y.; Yan, Y.; Gil, T.; and Nir, L. 2020. Planimation. *ICAPS 2019 System Demonstration, arXiv:2008.04600*.

De Pellegrin, E.; and Petrick, R. 2022. What Plan? Virtual Plan Visualization with PDSim. In *Proceedings of the ICAPS 2022 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*.

De Pellegrin, E.; and Petrick, R. 2023. PDSim: Planning Domain Simulation and Animation with the Unity Game Engine. In *Proceedings of the ICAPS 2023 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*.

Echeverria, G.; Lassabe, N.; Degroote, A.; and Lemaignan, S. 2011. Modular open robots simulation engine: MORSE. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 46–51.

Ganoni, O.; and Mukundan, R. 2017. A framework for visually realistic multi-robot simulation in natural environment. *arXiv:1708.01938*.

Koenig, N.; and Howard, A. 2004. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2149–2154.

Le Bras, P.; Carreno, Y.; Lindsay, A.; Petrick, R.; and Chantler, M. J. 2020. PlanCurves: An Interface for End-Users to Visualise Multi-Agent Temporal Plans. In *Proceedings of the ICAPS 2020 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*.

Maruyama, Y.; Kato, S.; and Azumi, T. 2016. Exploring the performance of ROS2. In *Proceedings of the ACM SIGBED International Conference on Embedded Software (EMSOFT)*, 1–10.

Muise, C. 2016. Planning.domains. ICAPS 2016 System Demonstration.

Quigley, M.; Gerkey, B.; Conley, K.; Faust, J.; Foote, T.; Leibs, J.; Berger, E.; Wheeler, R.; and Ng, A. 2009. ROS: an open-source Robot Operating System. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Workshop on Open Source Robotics*.

Roberts, J. O.; Mastorakis, G.; Lazaruk, B.; Franco, S.; Stokes, A. A.; and Bernardini, S. 2021. vPlanSim: An Open Source Graphical Interface for the Visualisation and Simulation of AI Systems. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 486–490.

Shah, N.; Verma, P.; Angle, T.; and Srivastava, S. 2021. JEDAI: A System for Skill-Aligned Explainable Robot Planning. *AAMAS 2022 Demonstration Track, arXiv:2111.00585*.

Tapia, C.; San Segundo, P.; and Artieda, J. 2015. A PDDL-based simulation system. In *Proceedings of the IADIS International Conference Intelligent Systems and Agents*.

Unity Technologies. 2022. Unity.

Vaquero, T. S.; Romero, V.; Tonidandel, F.; and Silva, J. R. 2007. itSIMPLE2.0: An Integrated Tool for Designing Planning Domains. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 336–343.

Yamamoto, T.; Terada, K.; Ochiai, A.; Saito, F.; Asahara, Y.; and Murase, K. 2019. Development of human support robot as the research platform of a domestic mobile manipulator. *ROBOMECH journal*, 6(1): 1–15.