Python argument types in
SemanticsAPI.Get(str, str)
did not match C++ signature:

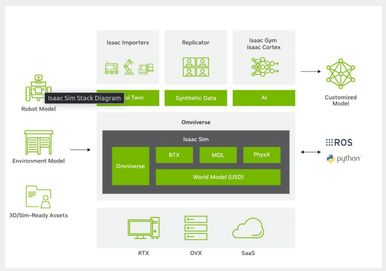
Futures of Training a JetBot in Virtual Environments: Unity and Isaac Sim

Students: Ramiah Curry (Morehouse College), Darius Chao (UCSD)
Mentors: Dr. Kwai Wong (UTK)

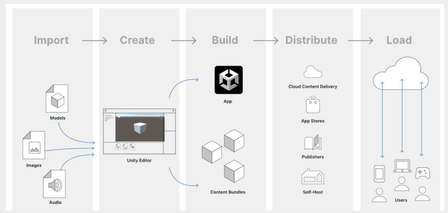
Background

Unity 3D is a widely-used game development engine, supporting virtual reality platforms. It offers an assortment of tools and packages for creating simulations and applications. Among its standout features is ML-Agents, an open-source plugin integrating machine learning into Unity projects which we used for this project. In addition, we also tested out the Perception Package, a synthetic data generation tool in Unity.

NVIDIA Omniverse is a platform with all sorts of tools and software related to 3D modeling and simulations of robotics in virtual environments. For this project, we focused on exploring the Isaac Sim toolkit which is known for its capabilities related to highly-realistic 3D simulations and robotics developments. We looked into OmniIsaacGym, Isaac Sim's machine learning library and Replicator, Isaac Sim's synthetic data generator.



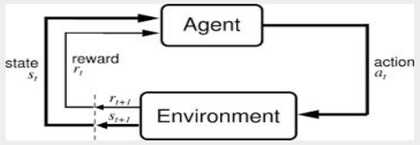
Isaac Sim Stack Diagram



Implementation

Unity's Perception package was utilized by scripting variations in background, lighting, camera angles, and object placements via the UI. The scene included "distractions" in the background with target road signs scattered throughout.

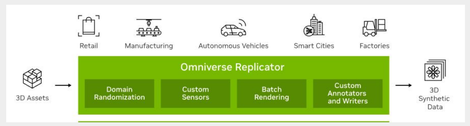
Using Unity's ML-Agents, we designed a virtual training environment to simulate our office building hallway. Setting up the ML-Agents package involved defining learning algorithms and reward systems in C# scripts. During training, agents interacted with the environment, learning through trial and error, resulting in improved decision-making and behavior.




```
behaviors:
  SignTrainer:
    trainer_type: ppo
    hyperparameters:
      batch_size: 128
      buffer_size: 2048
      learning_rate: 0.0003
      beta: 0.005
      epsilon: 0.2
      lambd: 0.95
      num_epoch: 3
      learning_rate_schedule: linear
    network_settings:
      normalize: false
      hidden_units: 256
      num_layers: 2
      vis_encode_type: simple
```

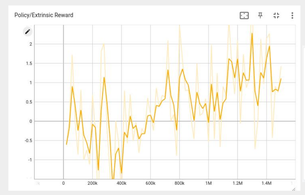
Isaac Sim's Replicator offers increased customization compared to Unity's Perception Package but requires more setup time. We utilized both Python scripts and the UI to configure Replicator and exported annotated images in a custom format.

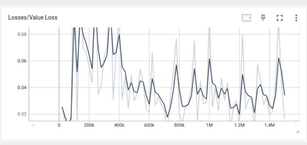
For OmniIsaacGym, we employed a basic environment with the JetBot and a red cube, focusing on object following. Python scripts defined the environment, reward system, learning algorithms, and functions.



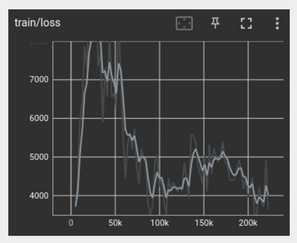

```
model = PPO(
    "CnnPolicy",
    my_env,
    policy_kwargs=policy_kwargs,
    verbose=1,
    n_steps=2560,
    batch_size=2560,
    learning_rate=0.0001,
    n_epochs=50,
    device="cuda:0",
    tensorboard_log=log_dir,
)
```

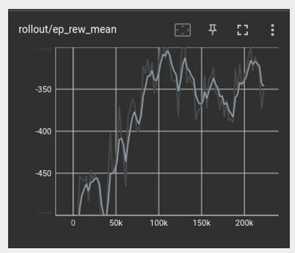
Results



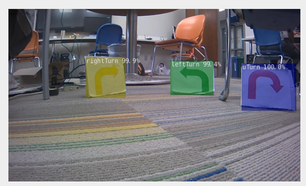


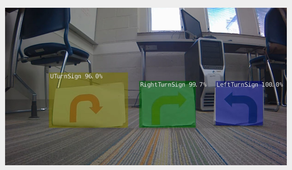
The graph (left) is the result of the cumulative reward generated from training the JetBot on a virtual track in Unity. The graph (right) shows the loss decreasing over time.





Similarly, the loss graph (left) decreases as the JetBot learns over time in Isaac Sim. The mean reward graph (right) therefore increases since it learns to do its task more effectively.



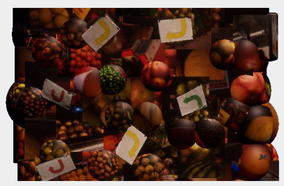


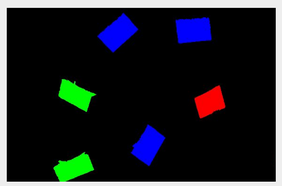
Object detection model trained on Replicator's synthetic data (left) and object detection model trained on Unity's Perception Package (right). Both are very accurate.

Objective

Our goal for this project was to see if we could train JetBots to drive autonomously within a virtual environment and then port the trained models to the real JetBots. We also wanted to compare the two software and figure out the pros and cons of both.

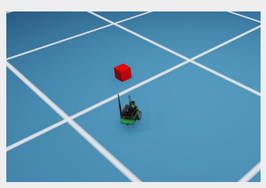
Testing/Training

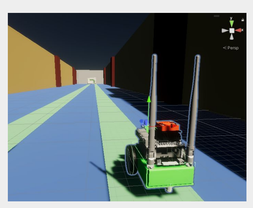




Unity Perception Package synthetic data (left) and its annotated result (right).

Isaac Sim Replicator synthetic data (left) and its annotated result after being passed into Roboflow (right).





Isaac Sim object following environment with OmniIsaacGym (left) and Unity corridor navigation with ML-Agents Package (right).

Analysis

Reinforcement learning is not as effective as object detection for autonomous driving, but with a well-tuned reward system and sufficient training time, it can work. However, identifying objects using the camera sensor in reinforcement learning is challenging and not always accurate. Porting these models to a JetBot proved difficult as well and was not able to be completed. On the other hand, the synthetic data generation part of the project was more successful. The Perception Package and Replicator generated valuable data for training a model to run on a real JetBot, streamlining the data collection process.

References

[1] U. Technologies, "Digital Twins," Unity, https://unity.com/solutions/digital-twins (accessed Jun. 30, 2023).
[2] "Isaac Sim Introduction," What Is Isaac Sim? - Omniverse Robotics documentation, https://docs.omniverse.nvidia.com/app_isaacsim/app_isaacsim/overview.html (accessed Jun. 30, 2023).
[3] NVIDIA-Omniverse, "Nvidia-Omniverse/omniisaacgymenvs: Reinforcement learning environments for omniverse isaac gym," GitHub, https://github.com/NVIDIA-Omniverse/OmniIsaacGymEnvs/tree/main (accessed Jun. 30, 2023).
[4] A. Juliani et al., "Unity: A general platform for intelligent agents," arXiv.org, https://arxiv.org/abs/1809.02627 (accessed Jun. 30, 2023).
[5]"Core [Omni.Isaac.Core]." Core [Omni.Isaac.Core] - Isaac_sim 2022.2.1-Beta.29 Documentation, 17 Mar. 2023, docs.omniverse.nvidia.com/py/isaacsim/source/extensions/omni.isaac.core/docs/index.html?highlight=seman#module-omni.isaac.core.objects.

Acknowledgments

This project was sponsored by the National Science Foundation through Research Experience for Undergraduates (REU) award, with additional support from the Joint Institute for Computational Sciences at University of Tennessee Knoxville.