Automated Unity Robot Analysis (AURA)

Team name: AURA

Team members:

- Sebastian Soto
- Felix Egan
- Jesus Monroy IV

Scientific Problem

We want to improve the efficiency of the movement of a robotic arm in an industry where the robot has to pick up a box and place it at a specified position. The focus will be in the mechanical power and consumption, where the different torque vectors, velocities, acceleration and position vectors of the joints of the robot would contribute.

Project Overview (Scientific Solution)

We aim to integrate Probability and Statistics & Mechanics into PDGE & Python I & II projects by creating an Automated Unity Robot Analysis (AURA) sim that analyzes robot performance data and provides statistical insights.

We will setup a Unity Project with a Robotic Arm and train it to move an object (box) from point A to point B using Reinforcement Learning (ML-Agents). The robot will perform multiple trials, and we will collect data on its performance (time taken, accuracy, energy consumption, etc). This data will be exported to CSV files.

The three graphs that we would present on the website are: the success rate graph (result vs trial), the inverse kinematic graph (take position of the joints in x, y, z coordinates) Joint angles, angular velocity (3d graph using Threejs) and the energy consumption graph (energy consumed with respect to time)

Afterwards, these three graphs would be cleaned to find the coefficient of determination of the three graphs by the linear regression model and then we would present the accuracy of the study.

Once we have the model trained, we are going to set up an environment for the user to interact with the box to see how the robotic arm was trained. We would set boundaries and show the optimized movement to get the robotic arm to the end effector, which would show the accuracy of the trained robotic arm.

Tech Stack

- **Unity**: For creating the robotic arm simulation and training using ML-Agents and the built-in physics engine.
- **C#**: For scripting within Unity to control the robot and handle data collection and physics related calculations like torque.
- **Unity ML-Agents**: For training the robotic arm using reinforcement learning.

- **Python**: For data analysis and visualization.
- **Pandas**: For data manipulation and analysis.
- **Matplotlib**: For creating visualizations of the data.
- **NumPy**: For numerical computations.
- **SciPy**: For advanced statistical analysis.
- **Jupyter Notebooks**: For interactive data exploration and reporting.

- **HTML/CSS/JavaScript**: For building a simple web interface to display results.
- **Flask**: For serving the web interface and handling backend logic.
- **Unity WebGL**: For deploying the Unity simulation on the web.

- **Vercel**: For deploying the Flask web application and hosting the Unity WebGL build from GitLab.
- **CSV**: Data Storage used for all the trained data found

Science Concepts

- **Physics (Mechanics)**
- To find the power consumption of the three different joints of the robots, we use the concept of inertial dynamics, inverse kinematics and these other concepts:
- We would the concept of center of mass to be able to allow the robot to move correctly to a precise position given its gravity affceted by the different joint links (sum of masses and distances of the joint links of the robot to find the current center of mass)
- We would use inverse kinematics to find the required joint angles to get to the end-effector (the end position of the box) which uses concepts of trigonometry and linear algebra (matrices, sin and cos of angles)
- We would use inertial dynamics to find the torque for each joint using the lagrangian dynamics formula with the inertial matrix, centrifugal matrix, and gravity matrix (this part would be achieved by the built-in physics engine in Unity)
- We would use the formula for mechanical power consumption (P = torque * v) and then the energy formula, which is the integral of power over time to find the energy consumption of each joint at every snapshot trial
- We would use Newton's Second Law in general (F = ma) to find the force required for each joint to be able to move with respect to the mass of the robotic arm
- We would use mechanical efficiency formula with the work input and output of the system to be able to calculate the total mechanical power consumption of the robotic arm

- **Math (Probability and Statistics)**
- We will use linear regression to show the graph of the rate of success of the website
- We would use the coefficient of determination and the standard deviation to access the accuracy of the data trained to create the simulation. The standard deviation would allow to find out if any of the values of the parameters are far from the average of values gotten (for the values of the joint torque for example)

Project Requirements

- **Functional**
- It must display the numerous data taken during the training in a concise way
- The website loading time (and the Unity WebGL) is optimized (2 seconds max)
- The user interface should be modern and reflect our theme of robot simulation (futuristic CSS, blurred backgrounds, etc)
- **Scientific**
- The focus of the scientific requirements would be more on the presentation of data statistics
- The model should present an accurate representation of joint torque of a robotic arm (with the appropriate motor resistance, with reference to a real robotic arm)
- The graph of the data should ressemble as much as possible the theoretical success rate graph (similar to an arctan graph)
- The coefficient of determination should be close to 90% for the data trained, leading to a low percent error on the data trained

Project Structure

CURRENT_FILE_STRUCTURE IDEAL_FILE_STRUCTURE

Core Webpage Features

- **Home Page**: Overview of the project with links to different sections.
- **Unity Simulation Page**: Embed the Unity WebGL build to allow users to interact with the robotic arm simulation directly in the browser by dragging the box into set boundary positions
- **Data Analysis Page**: Summarized statistical analysis results with options to download processed data with HRef To Official Report PDF

Additional Features (Stretch Goals)

- **Interactive 3D WebGL Viewer**: Allow users to manipulate the robotic arm in 3D space, change the box location, etc...
- **Advanced Visualizations**: Implement more complex visualizations like heatmaps of robotic arm movement efficiency.

- **Real-time Data Updates**: If feasible, allow real-time updates of data as the robot performs tasks in the Unity sim.
- **Downloadable Reports**: Generate comprehensive PDF reports summarizing the analysis with charts and statistics.

Ideal Timeline

TIMELINE

- (check down for the tasks and mnilestones with respect to the timeline)

Team Agreement

- Our website will not collect any sensitive personal information from the users
- We will use AI as a tool, not as an author
    - We will use AI as a guide to understand all the logic of the website
- Mode of communication: Discord. Respond by 24 hours
- We will create our own feature branches to work on the project to avoid merge conflicts and to keep everything organized
- Document all the code and the different functions
- If the feature is too close to the deadline, we reserve the right to change the person in charge of that feature in order to finish it
- **Meetings**: Tuesdays 10-11:30 and/or 20-22:00 (depending on the tasks for the week)

Reference Examples (WIP)

- Unity ML-Agents Documentation
- Prob and Stats Textbook & Formulas
- Mechanics Textbook & Formulas
- Online tutorials and resources for Unity, ML-Agents, Flask, and data analysis
- Reliable YouTube videos on Robotics concepts
- Robotics textbooks and research papers on robotic arm kinematics and dynamics
- GitHub repositories and open-source projects related to Unity ML-Agents and robotic simulations
- **Links:**
- View of a Review on Linear Regression Comprehensive in Machine Learning | Journal of Applied Science and Technology Trends. (n.d.). https://jastt.org/index.php/jasttpath/article/view/57/20
- Energy consumption in Robotics: A simplified modeling approach. (2024, 05 Nov.). https://arxiv.org/html/2411.03194v1
- [1] M. Ali, S. Giri, S. Liu, and Q. Yang, "Digital Twin-Enabled Real-Time Control in Robotic Additive Manufacturing via Soft Actor-Critic Reinforcement Learning," arXiv preprint arXiv:2501.18016, Jan. 29, 2025. [Online]. Available: https://arxiv.org/abs/2501.18016

- Engineering Simplified. (2022, July 23). Inverse Kinematics of Robots | Robotics 101 [Video]. YouTube. https://www.youtube.com/watch?v=1-FJhmey7vk

Annexe

The current deadlines added are preliminary and are subject to change

- **Setup Tasks**:
    - Finish the flowcharts and design plan of the website - February 6
    - Finish the project proposal draft - February 6th
- **Main Timeline Milestones**:
- Milestone 1: Training the Data (March 6th)
    - Setting up the design of the robotic arm in Unity - February 26
    - Setting the joint vectors to get input - February 20
    - Add a reward system with respect to the velocity inputs of each joint
    - Start the intensive training (5 million trials) - March 6
- Milestone 2: Data Export and power consumption logic (March 27th)
    - Do the calculations to find the power consumption of each of the joint positions
    - Filter and clean the noisy data - March 13
    - Export the data in C# through CSV - March 27

(rest of the dates to be determined)

- Milestone 3: Parsing Data (April 2nd)
    - Parse the data for all three graphs that we would need - March 31st
    - Use JUpyter to create different snapshots of the data created to be able to add the data required to create the 3D graph of box position with respect to time - April 2nd
- Milestone 4: Vizualising data (with Pandas) (April 17th)
    - Create the graph with all of the data of the training cleaned up - April 10
    - Optimize the graph visualization for performance - April 17
- Milestone 5: Website Design (April 20th)
    - Add all the Flask routes to each of the pages - March 4
    - Do the Home Page - April 3rd
        - Adding a href to all pages of the website - March 10
    - Do the About Page - March 22nd
        - Add all citations and references - April 2nd
    - Do the Analysis Page - April 20
        - Downloadable files used - April 20

    - Do the Dashboard Page
        - Graphs to present : - May 1st
            - Success rate
            - IK graph
            - Power efficiency
            - Box position with respect to time (3D graph)
    - use WebGL to add the Unity environment with the robotic arm inside the Simulation Page - March 30
    - Add the animations and styling for the pages - April 17

- Milestone 6: User Interactivity (May 2nd)

    - Do the scrollable animation to show the result of the 100 best trained robots and their data - April 3
    - Create the trained environment to let the user interact with the box - April 30th
        - Set the three boundaries (two circle bounds and the end effector angle bound)
        - Add the transparent material to show the green area where the user can drag the box
        - Render the animation of the robot movement to get to the box

- Milestone 7: Testing and finising the website - May 2nd

    - Do some debugging and website runtime optimization if required

Documentation

- Unity 3D Robot Arm Model
- Flask
- Jinja
- Matplotlib
- Pandas
- NumPy
- SciPy
- Vercel Deployment Docs