# LegMo: The Legged Momentum Wheel Robot

Jaiveer Singh, Matthew Hallac, Alex Zhou, and Prabhman Dhaliwal

***Abstract*—LegMo is a novel concept robot that combines single-jointed bipedal mechanics with a 3 degree-of-freedom momentum wheel system, capable of achieving robust dynamic stability in a small and cost-effective package. Momentum wheels, also known as reaction wheels, are massive disks that are accelerated to impart torques on a system. Using a set of three reaction wheels on orthogonal axes to complement otherwise under-actuated legs, LegMo exhibits surprisingly effective performance in simple walking tasks, while also expressing a capability for complex actions like pivot turns and fall recovery that traditional bipeds struggle with. Through a reduction in the number of actuators required to just one hip actuator per leg and three reaction wheels, LegMo emerges as a dramatically simpler and more economical alternative when compared against the state-of-the-art. This paper presents an exploratory analysis of the LegMo platform, via the preparation of a PyBullet-based simulation, a comparison between classical and reinforcement learning-based controllers, and a simplified hardware prototype. We also present opportunities for further research into LegMo's unparalleled agility and a true hardware realization of the LegMo concept.**

***Index Terms*—Bipedal robots, momentum wheels, reaction wheels, underactuated dynamics, reinforcement learning for control, dynamical control.**

## I. Introduction

THE development of highly-capable bipedal robots remains an open research problem with significant attention devoted to it in recent years. Prior research has focused on building progressively more complex, biologically-inspired systems with as many as 5 actuators per leg [1]. However, the natural cost associated with sophisticated, actuator-dense robots maintains difficulty in efficiently controlling them. The complex nonlinear dynamics of these robots forces classical control approaches to make significant approximations that negatively affect the versatility of the controller. Learning-based approaches have received much interest as an alternative, but the poor sample efficiency of these approaches generally limits models to simulation. The infamous sim2real gap poses a substantial, though not impossible, obstacle to overcome, particularly when the configuration space of the robot has a high dimensionality [1].

In this paper, we propose LegMo, a novel robot concept that adopts a different approach to the challenge of bipedal design. Specifically, LegMo rejects the humanoid, many-actuator blueprint in favor of a radically simple combination of single-jointed legs and a set of three reaction wheels. The benefits of such simplicity are numerous: the reduced dimensionality makes control easier for both classical and reinforcement learning approaches; the lessened power requirements make it possible for small-scale robots to be powered on standard Lithium Polymer batteries; and the smaller actuator count makes fabrication significantly more economical and accessible. Contrary to what intuition might suggest, the simplified design actually does not present any new tradeoffs. In fact, LegMo's reaction wheels allow for substantially more powerful disturbance rejection in any direction, and the robot also has the potential to perform complex maneuvers like turning about one foot and independently righting itself after a fall.

The authors are enrolled at the University of California, Berkeley. This paper was produced for the final project of EECS C106B, Spring 2021.

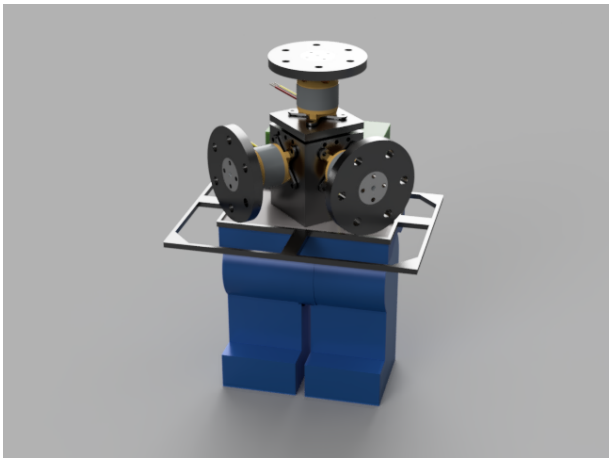

Fig. 1. CAD render of our robot, LegMo.

Given the novelty of the LegMo platform, we have approached the task of exploring the design space from multiple directions. First, we present a justification for the design and an overview of the rotational physics that enables controlled motion of LegMo. Next, we exhibit a comparison between simplified classical control and more advanced reinforcement learning approaches. Finally, we showcase an early hardware prototype of the platform to experimentally validate the science behind LegMo.

## II. Related Work

### A. Reaction Wheels

Reaction wheels have already found popular usage in real-world spacecraft and satellites for the purpose of three-axis attitude control in a low-gravity environment, including such notable examples as the Hubble Space Telescope. Given the continued prevalence of reaction wheels in space environments, researchers have used modern reinforcement learning techniques to make these attitude control systems more robust to variations in spacecraft inertia [2]. In a zero-gravity environment, reaction wheels are incapable of producing translational forces; however, there is no such restriction when reaction wheels are deployed in terrestrial locomotive robots. In particular, research has been done into the usage of single-axis reaction wheel systems to increase the stability and efficiency

of planar bipedal walking motion [3]. Furthermore, it has been demonstrated that the usage of a 3-axis reaction wheel system can be used to properly control the translational and rotational movements of a cube on the ground plane [4].

### B. Classical Control

*1) Dynamics:* Classical control for robotics is a well-studied field, even in the context of a novel robot configuration using reaction wheels. The Cubli, a 3-dimensional cube robot that uses reaction wheels for movement, was a significant inspiration for our project. Prior work includes the one-dimensional implementation [5], and the three-dimensional counterpart [4]. From a reaction wheel perspective, the Cubli system and the bipedal robot are very similar dynamically. As a result, the linearized dynamics of the Cubli paper were imported into this project as a good approximation of the LegMo's true dynamics.

*2) Controllers:* The Cubli papers leveraged LQR control approach about a corner setpoint [4]. A cited benefit of this algorithm is that it allows for effectively tuning the relative weights of the state variables and the penalties of input to the system directly. LQR is also a robust control algorithm that serves as a strong baseline to compare with RL-based control, as used in [2].

### C. Reinforcement Learning

In recent years, Deep Reinforcement Learning with neural networks has shown potential to solve problems that were previously intractable for old Reinforcement Learning agents [6]. Some of the new frameworks, like PPO, DDPG, SAC, TRPO, and A2C have even shown promising results in performing feats in robotics control in simulation [7]. Work has even been done for transfer learning from simulation to the real world on a bipedal robot [1].

Since reinforcement learning is often sample-inefficient and requires significant training time, another important area of recent developments is curriculum learning. A good curriculum design gradually exposes the reinforcement learning agent to more and more complex tasks, and accelerates the agent's learning speed by ensuring the current task is not too difficult for it to learn. This student-teacher analogy has been shown to be particularly effective when tackling complex tasks from which the agent will initially struggle to receive a nonzero reward [8].

## III. Methods

### A. Success Criteria

For the purposes of this research paper, we identified three central objectives in exploring the open-ended nature of LegMo's robot design. Firstly, we set out to synthesize a novel bipedal robot configuration that had the potential to change the landscape of the state-of-the-art. Secondly, we intended to explore and compare the performance of both classical and reinforcement learning-based controllers on this new design. Finally, we expected to fabricate a hardware implementation of our robot concept to validate the real-world feasibility of the design.

### B. Theory

*1) LegMo Dynamics:* To better understand the LegMo Design, the reaction wheel dynamics were derived. A reaction wheel uses a high moment of inertia to apply a torque to the body on which it is attached. The equation for torque is

$$\tau = I\ddot{\theta} = F * r \tag{1}$$

where $I$ is the moment of inertia of the rotating object, $\ddot{\theta}$ is angular acceleration, $r$ is the distance from the pivot point, and F is the force component orthogonal to a vector from the pivot point to the point the force is applied. The total torque enacted on the system is equal to the sum of the torque due to gravity ($\tau_g$) and the torque from the reaction wheel ($\tau_{mot}$).

$$\tau_{tot} = \tau_g - \tau_{rw} \tag{2}$$

In our hardware prototype, brushless motors with a torque coefficient of $k_q = 9.54 * 10^-4$ were used. The total torque generated by the motors was therefore

$$\tau_{mot} = I_{rw} * \ddot{\theta}_{rw} = k_q * u \tag{3}$$

where $u$ is the input current. Using the definitions of torque provided above, the angular acceleration of the entire LegMo robot can be derived.

$$I_{sys}\ddot{\theta}_{sys} = mgL * \sin\theta_{sys} - k_q * u \tag{4}$$

$I_{sys}$ is the moment of inertia of the entire robot about the pivot point, $\theta_{sys}$ is the angle between the vertical axis and a line from the pivot point to the center of mass of the robot, $m$ is the mass of the robot, $g$ is gravitational acceleration, $L$ is the distance from the pivot point to the center of mass, $I_{rw}$ is the moment of inertia of the reaction wheel, and $\ddot{\theta_{rw}}$ is the angular acceleration of the reaction wheel.

*2) Recovery angle:* Using the dynamic equations of the LegMo robot and measurements from our hardware prototype, we are able to calculate a maximum recovery angle, or the maximum angle deviation from the vertical axis from which the robot can produce a torque to right itself. This is a critical parameter of the design to know because if this angle is exceeded, LegMo will fall over and not be able to recover. Demonstrations of this phenomenon are provided on our website in the Appendix. With a motor with a sufficiently high torque coefficient, this value would be 90 degrees; in other words, the robot would be able to stand up and recover from any angle. Using the dynamics calculated above, the recovery angle can be calculated as the following:

$$\tau_g \leq \tau_{mot} \tag{5}$$

$$mgl * \sin(\theta) = k_q * u \tag{6}$$

From this equation, we calculate the maximum recovery angle for our hardware prototype to be approximately 0.282 radians, or about 16 degrees. This could be improved by using lighter materials and positioning them closer to the center of mass of the robot, or using better motors with higher torque coefficients. Another potential method to increase the

recovery angle would be to place brakes on the reaction wheels, allowing the robot to produce faster decelerations on the wheels.

### C. Approach

*1) Classical Control:* The first classical controller implemented was an LQR controller fashioned after the Cubli papers. Both these controllers tune specific parameters to balance the reaction wheels on an edge or corner, mimicking Cubli and giving the desired balancing effect for precise control over the robot's motions. For the LQR controller, there are 3 state variables the controller maintains: the angular position of the robot chassis with respect to an origin, the angular velocity of this chassis, and the angular velocity of the reaction wheel. The input was just the current of the motor controlling the reaction wheel. In the 3D case this extends to 3 current inputs to each of the 3 reaction wheel's motors. Constants used including the length, masses, and rotational inertia values for each of the body and reaction wheel, as well as the torque constant of the motor and the acceleration due to gravity, are provided in the below table.

| Constants for LQR Simulation | |
|---|---|
| Constant | Value |
| $l$ | 0.17 m |
| $l_b$ | 0.15 m |
| $m_b$ | 1.0 kg |
| $I_b$ | $3.0 \times 10^{-2}$ kg-m$^2$ |
| $m_w$ | 0.060 kg |
| $I_w$ | $2.0 \times 10^{-4}$ kg-m$^2$ |
| $K_m$ | 0.0083 N-m/A |
| $g$ | 9.81 m/s$^2$ |

The dynamics derivation can be found in the one-dimensional Cubli prototype paper [5], and the end result of the dynamics yields these A and B matrices:

$$A := \begin{bmatrix} ccc0 & 1 & 0 \\ \frac{(m_b l_b + m_w l)g}{I_b + m_w l^2} & -\frac{C_b}{I_b + m_w l^2} & \frac{C_w}{I_b + m_w l^2} \\ -\frac{(m_b l_b + m_w l)g}{I_b + m_w l^2} & \frac{C_b}{I_b + m_w l^2} & -\frac{C_w\left(\left(I_b + I_w + m_w l^2\right)\right.}{I_w\left(I_b + m_w l^2\right)} \end{bmatrix} \quad (7)$$

$$B := \begin{bmatrix} 0 \\ -\frac{K_m}{I_b + m_w l^2} \\ \frac{K_m\left(I_b + I_w + m_w l^2\right)}{I_w(I_b + m_w l^2)} \end{bmatrix} \quad (8)$$

These matrices are used in our LQR controller, which was implemented in Matlab. The data collected in Matlab demonstrates how the LQR controller (in the 1D case) maintains the states at steady values after an initial windup of the reaction wheel. Recall that in LQR, there are Q and R matrices that serve to penalize certain factors to maintain steady state. The Q matrix (which is diagonal 3x3 in the 1D situation) penalizes deviations from the ideal state. The R matrix (which is just a scalar in the 1D situation), penalizes excessive use of energy as an input to the system. The magnitude of penalization for the Q matrix were determined by checking the magnitude of the eigenvalues corresponding to each of the state values. Whichever state value had the highest eigenvalue (as shown by the K matrix returned by the Matlab LQR function), was the one that was penalized the heaviest. We recognized that saturation of the reaction wheels was a more prominent issue as opposed to using too much current, and thus allowed liberal use of motor current in our system; a smaller R value was used. Ultimately, there was some trial and error to tune these values once the magnitudes of the these values were understood.

A simple PD controller was also implemented. The reaction wheels are controlled using the angular position of the robot and the angular velocity. A trajectory and desired orientation for each time step must first be generated to use this control.

*2) RL Control: PyBulletGym:* For RL control, we first chose to use an out-of-the-box implementation of PPO from Stable Baselines 3 [7] and trained it in PyBulletGym's Walker locomotion environment.

This environment is an infinite, flat plane for the robot to learn to walk on. It gives a constant positive reward for living, reward for making progress in the forward direction, and penalties for the amount of total torque applied to the motors. The end conditions are dependent on the robot's pitch, Z-height, and which bodies of the robot are in contact with the ground.

We chose to limit the pitch of the robot's base to be between positive and negative 75 degrees, constrain its height under 15cm, and ensure that the base link (top body of the robot) was never in contact with the ground, as that would mean the robot had fallen over.

*3) RL Control: Open Source Environment:* We also created our own environment separate from PyBulletGym to test our robot in. We did this for two reasons: first, it was not easy to modify PyBulletGym's existing locomotion environment and model loading scheme; second, PyBulletGym's setup allowed agents to assume total knowledge of their configuration state and global state, which is not possible for most robots in real-world settings.

For this environment, we made the robot's goal slightly more complicated. We aimed to have it be able to achieve any target quaternion orientation along with target leg angles. We also added a heading goal, which described the direction in the X-Y plane we wanted the robot to make progress in. If this was not specified, the robot's goal is simply to reach the target quaternion and leg angles to terminate the episode. When a heading direction was specified, episodes would only terminate when the robot fell over.

Reward was given based on the changes in the robot's distance to its goal, where decreases in distance meant positive reward, and increases in distance meant negative reward. Distance was computed as a weighted sum of the absolute angle difference between current and goal quaternion, absolute difference between current and goal leg angles, and negative of the distance traversed in the goal direction. Furthermore, if no goal direction was given, there was a negative reward for living to incentivize reaching the target orientation as quickly as possible. On the other hand, if goal direction for walking was provided, a living reward was provided to prevent the robot from seeking immediate death.

Just as with the PyBulletGym environment, we trained a PPO agent using the Stable Baselines 3 implementation.

Graphical results are included in the below section.

## IV. Results

### A. Hardware Results

*1) Hardware Prototype:* To test LegMo in the real world, a hardware prototype was developed based on the simulation model. Brushless motors with a torque constant of $9.54 * 10^{-}3$ were used, and reaction wheels had an estimated moment of inertia of $7 * 10^{-}4$. The prototype was 3D printed using PLA, and weighted 1050 grams. The hardware prototype was an effective proof of concept, and a demonstration video can be found on the project website.

### B. Software

*1) Classical Control - LQR:* After setting up all initial values and physical variables, Matlab's LQR() function and system setup functions allowed us to evaluate these results with these parameters:

$$Q = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 10^{-2} & 0 \\ 0 & 0 & 10^{-8} \end{bmatrix} \tag{9}$$

$$R = .00001 \tag{10}$$

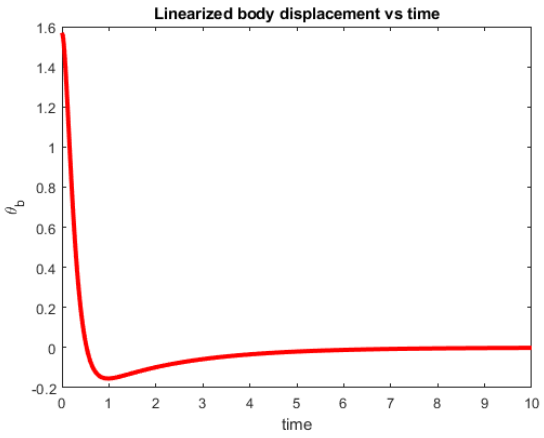


Fig. 2. Linearized Robot Chassis angular displacement vs Time (1D case)

These plots show the stability of the reaction wheels in the 1D case with an LQR controller. The values were selected through trial and error, with the relative magnitudes of the values being figured out through understanding the eigenvalues of the control matrix returned by the $lqr()$ function in Matlab.

*2) Classical Control - PD:* A video of the PD controller attempting to pivot on one leg can be found on our website in the Appendix. The PD controller had several downsides. First, to achieve a desired orientation or perform a desired trajectory required finding the required torque values for the reaction wheels in three dimensions. This is difficult for complex trajectories, and defining the trajectory for a simple walking motion was a challenge. The second downside of PD control was that for each new trajectory, the $k$ values had to be re-tuned, leading to a slower implementation that also had errors that could compound at each time step. Again, trial and error was employed to find proper k values.

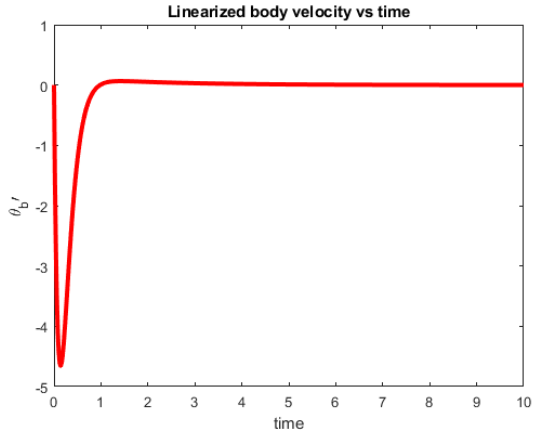


Fig. 3. Linearized Robot Chassis Angular Velocity vs Time (1D case)

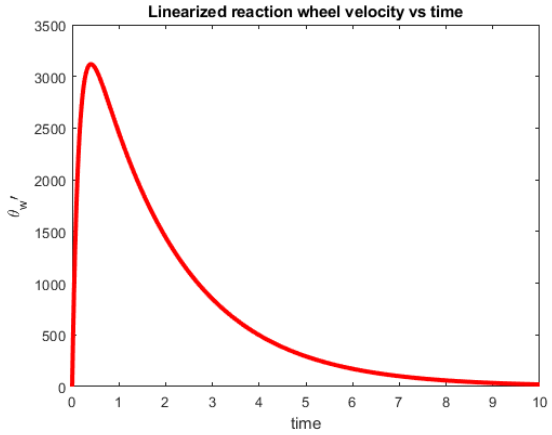


Fig. 4. Linearized Wheel Angular Velocity vs Time (1D case)

*3) Reinforcement Learning: PyBulletGym Results:* After 500,000 timesteps of training, we were able to get a working RL agent that skipped forward very quickly. A snapshot of its motion can be seen in Figure 5.

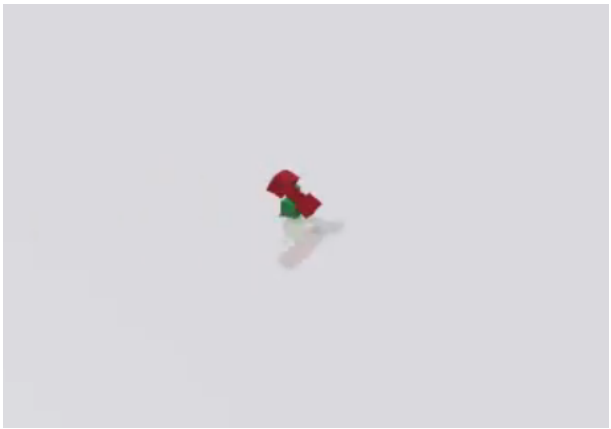

Fig. 5. Snapshot of PyBulletGym RL agent walking.

We can also see this success reflected in the reward accrued (Fig. 6 and time spent alive (Fig. 7) by the agent over the

course of its learning.

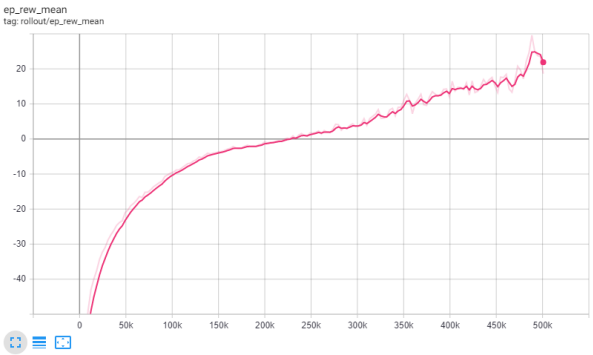


Fig. 6. Reward accrued by PyBulletGym RL agent during rollout.

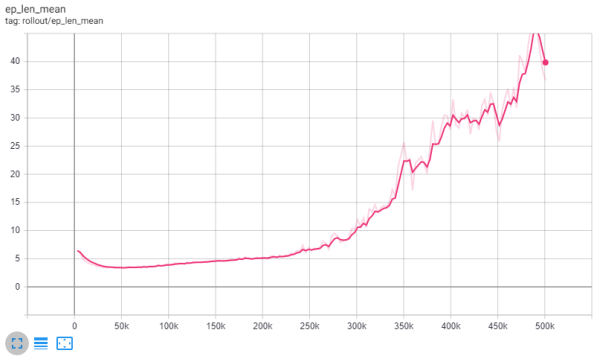


Fig. 7. Time spent alive by PyBulletGym RL agent walking in PyBullet.

These results are promising, as it seems it is relatively easy for an RL agent to learn to control our reaction wheel bipedal robot design. Since our robot has very few joints in the legs, if we can transfer this success to a real-world prototype, this could lead to simpler, more efficient bipedal robots through the use of reaction wheels.

*4) Reinforcement Learning: Open Source Environment Results:* After timesteps of training, we were able to get a working RL agent that could reach the target quaternion orientation and leg angles most of the time, as can be seen in Figure 8. However, we were not able to get a working model for walking yet. There were some cases in which the robot could turn or wobble a few steps forward, but these movements did not look very well-controlled nor stable.

The training numbers for training without goal walking direction from the evaluation steps can be seen in Figures 9 and 10. In particular, we can see that as training goes on, the robot learns to quickly reach the target orientation and leg angles within 10,000 timesteps, and we can see this reflected in the episode lengths as well.

Despite being unable to get walking working, we still believe that these results show promise for our open-source environment. Given more time or fine-tuning, we believe that we should be able to get walking working even with the limited observation-space provided to the robot.

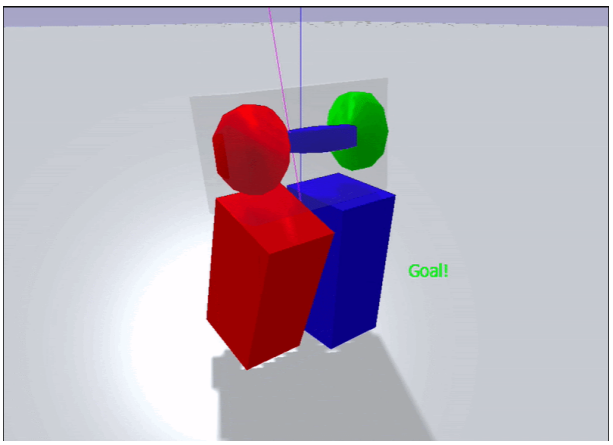


Fig. 8. Snapshot of RL agent posing in our open source environment.

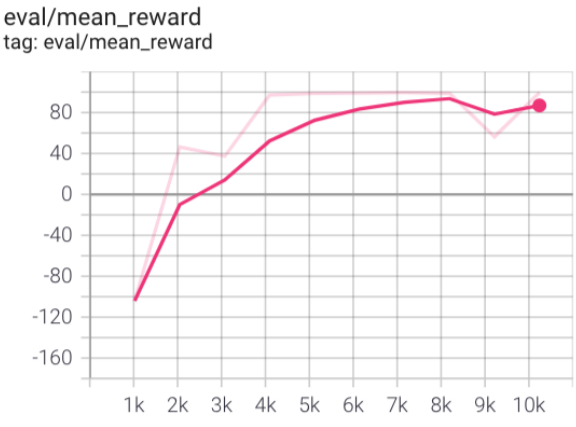


Fig. 9. Reward accrued by RL agent during evaluation in the open source environment.

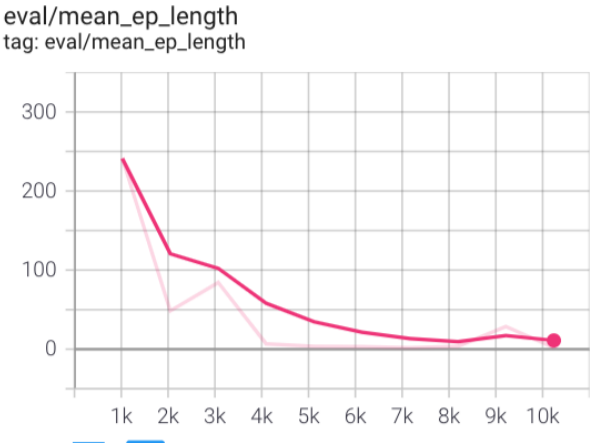


Fig. 10. Time spent to reach goal orientation by RL agent in the open source environment.

## V. Conclusion

### A. Major Contributions

This paper presents LegMo, a novel robot design that combines a set of three reaction wheels with a simple pair of legs. This robot's versatility is argued mathematically and in reference to prior work with reaction wheels, and is demonstrated via results in simulation. The core concept of transferring angular momentum into spatial movement via legs is also validated through a hardware prototype, whose CAD and specifications are made freely available. The authors have also published a PyBullet-compatible OpenAI Gym environment with this new robot design, opening the door to other researchers interested in deploying reinforcement learning techniques on this design.

### B. Difficulties Encountered

By virtue of the multifaceted nature of this project, we encountered numerous problems across both the engineering and scientific fronts.

With regards to the classical controllers, the math for extending the dynamics to 3D was incredibly challenging to extend our system to, so an approximation by just using the 1D model 3 times was proposed instead for our classical LQR controller.

Deciding on an appropriate architecture for the reinforcement learning models was a particularly challenging issue, since we needed to balance the improved learning capability of a larger neural network with the corresponding increase in training exposure necessary. Significant effort was expended to minimize total training time via curriculum learning approaches, though the results of this effort were mixed.

In order to facilitate a hypothetical transfer of our reinforcement learning techniques from simulation to the real world, we went to great lengths to ensure our URDF model of the robot was as accurate as possible. While a Domain Randomization approach may have been more effective at enabling this sim2real transfer, it would have required multiple magnitudes more training to achieve a robust agent, which was prohibitive given the compressed timeline of this project.

Unsurprisingly, the hardware implementation presented a host of engineering challenges. Working with only hobbyist-grade tools made the manufacturing of low-tolerance, well-balanced reaction wheels exceptionally difficult; the final prototype still exhibits undesirable oscillations as a result. Due to budgetary and time constraints, the brushless motors ultimately used to power the reaction wheels were limited to cheap and readily-available drone motors, whose limited torque prevented our hardware prototype from fully matching simulation performance.

### C. Future Work

Given the novelty of the LegMo design, we believe there is still significant unexplored potential in the merits of this kind of reaction wheel-equipped bipedal robot. In order to validate the basic concept of such a design, we have focused our efforts on establishing baseline walking and mobility in simulation. However, by virtue of the 3 degree-of-freedom reaction wheel setup, we understand that LegMo is capable of practically any sequence of movements, including sequences that were previously impossible for bipedal robots to carry out. One can imagine the utility of a robot capable of rapidly pivoting on a single foot to change walking direction, running and jumping up steps, and recovering after being knocked down. Continuing our work on the curriculum learning-inspired Reinforcement Learning agent seems like a profitable approach for achieving these same maneuvers.

An additional area of investigation is in preparing a feature-complete hardware rendition of the LegMo system. While our work in simulation and early-stage prototyping are both promising, we believe a complete physical model will best convey the effectiveness of our new robot design, while also proving that the sim2real transfer of our reinforcement learning model is possible.

## Appendix

### A. Demonstrations of Results

For photos and videos of our project's performance both in and out of simulation, as well as the underlying code and CAD, please see our website: LegMo Homepage

## Acknowledgment

The authors would like to thank Professors Sastry and Ma, along with the other members of the EECS C106B course staff. Additional thanks are due to Michael McNabb for his inputs on the mechanical design and fabrication of the hardware prototype.

## References

[1] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for robust parameterized locomotion control of bipedal robots," 2021.

[2] J. Allison, M. West, A. Ghosh, and F. Vedant, "Reinforcement learning for spacecraft attitude control," 10 2019.

[3] T. L. Brown and J. P. Schmiedeler, "Reaction wheel actuation for improving planar biped walking efficiency," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1290–1297, 2016.

[4] M. Gajamohan, M. Muehlebach, T. Widmer, and R. D'Andrea, "The cubli: A reaction wheel based 3d inverted pendulum," in *2013 European Control Conference (ECC)*, 2013, pp. 268–274.

[5] M. Gajamohan, M. Merz, I. Thommen, and R. D'Andrea, "The cubli: A cube that can jump up and balance," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 3722–3727.

[6] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, p. 26–38, Nov 2017. [Online]. Available: http://dx.doi.org/10.1109/MSP.2017.2743240

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

[8] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," 2020.