Cover Letter

Walk-these-ways

In this project, our aim was to develop a robust navigation policy for animal robots inspired by the locomotion architecture presented in the "Walk These Ways" paper. We sought to implement a new policy termed the navigation policy, which would orchestrate the locomotion policy to enable effective navigation across diverse scenarios. The project unfolded in three distinct phases, each contributing to the evolution of our navigation framework.

Initially, we focused on training the locomotion policy of the robot to achieve stable gaits. This phase laid the foundation for subsequent developments by ensuring that the robot could move effectively under different velocity conditions while maintaining stability. The second phase involved constructing a navigation policy tailored for closed-space environments surrounded by walls on all sides. Our objective was to enable the robot to traverse from one end to the other without colliding with the surrounding walls. To achieve this, we meticulously designed the navigation policy and implemented it within the simulation environment. This phase required rigorous testing and parameter tuning to ensure optimal performance. In the third and final phase of the project, we aimed to enhance the navigation policy to incorporate obstacle avoidance capabilities. We introduced randomly initialized obstacles along the robot's path and tasked the navigation policy with navigating around them to reach the designated goal. We ended up making a robust navigation policy that has produced great results in the gym simulator. A detailed analysis of each phase of the project is given in the report below.

In our collaborative project, my teammate and I were employing diverse approaches and collectively deciding on the most promising ones to pursue. To begin, I crafted a well-organized workflow to comprehend the intricacies of the walk-these-ways architecture and its underlying code structure. Simultaneously, I configured the Isaac Gym environment within my Rutgers iLab workspace. My teammate delved into understanding alternative locomotion papers, broadening our understanding of locomotion principles. Subsequently, I proceeded to construct walls around the robot within the simulation environment, meticulously ensuring that all physical properties of both the walls and the robots remained consistent across various scenarios. Moving forward, I took the lead in developing the code for training the navigation policy, alongside designing the positional and action magnitude reward functions. I rigorously tested these functions across multiple environments within the simulation, refining them iteratively for optimal performance. Concurrently, my teammate devised the code for implementing the goal-body alignment reward, fine-tuning the model parameters to ascertain the optimal configuration of reward weights. As we progressed into the final phase, I focused on augmenting the navigation policy to include obstacle avoidance capabilities. I developed the necessary code to introduce randomly initialized obstacles along the robot's path, enhancing the complexity of the navigation challenge. In tandem, my teammate fine-tuned the model for this phase, ensuring that the navigation policy could adeptly navigate around obstacles while still reaching the designated goal. Additionally, he took charge of crafting the project presentation. Concluding the project, I assumed responsibility for compiling and documenting our work in the project report, encapsulating the insights gained and the accomplishments achieved throughout the various phases.

CS 562 - Walk These Ways

Pradeep Roy Yadlapalli
Rutgers - The State University of New Jersey
py160@scarletmail.rutgers.edu

***Abstract*— This project aims to develop a comprehensive framework for quadruped navigation and obstacle avoidance using reinforcement learning techniques. The project will utilize the Unitree Go1 robot in the Isaac Gym Simulator to train neural network policies for locomotion and navigation tasks. The framework consists of three phases: locomotion training, navigation within walls, and with obstacle avoidance.

In the first phase, the project focuses on training a velocity-conditioned neural network policy for stable gait locomotion. The robot will be commanded to move at specified velocities while avoiding falling or terminating. Phase two involves training a navigation policy to guide the robot through walled corridors without collisions. The third phase extends the navigation policy to incorporate obstacle avoidance capabilities.

The project's feasibility lies in leveraging existing simulators and reinforcement learning algorithms to train robust policies learned from the "Walk These Ways" paper. Each phase will build upon the previous one, culminating in a fully functional system capable of navigating complex environments autonomously.

I. Project Description

Quadruped navigation and obstacle avoidance represent fundamental challenges in robotics with applications across various industries. From search and rescue operations to agricultural automation, legged robots offer versatility and mobility in diverse environments. The Unitree Go1 robot serves as the platform for this project, operating within the Isaac Gym Simulator to facilitate rapid prototyping and experimentation.

The first phase of the project involves training a velocity-conditioned neural network policy for robot locomotion, allowing the robot to walk at commanded speeds while maintaining stability. Subsequent phases extend the capabilities of the navigation policy to navigate within walled corridors and incorporate obstacle avoidance.

By implementing these phases, the project aims to create a versatile framework for quadruped navigation that can adapt to diverse environments and scenarios. The proposed approach offers a feasible solution to address the challenges of robot navigation and obstacle avoidance, with potential applications across various domains.

II. Background

Locomotion policies are adept at adapting to various environments akin to their training settings but often lack swift tuning mechanisms for out-of-distribution scenarios, necessitating laborious retraining cycles. The paper Walk these Ways proposes Multiplicity of Behavior (MoB), encodes diverse locomotion strategies within a single policy, enabling rapid adaptation across unseen environments without the need for extensive retraining. "Walk These Ways," a robust open-source MoB locomotion controller capable of executing versatile gaits with variable footswing, posture, and speed, facilitating diverse tasks such as crouching, hopping, high-speed running, stair traversal, bracing against shoves, rhythmic dance, and more.

To enhance generalization across scenarios, Mob when given identical observations and a small set of behavior parameters, outputs varied walking behaviors. This approach allows for swift iteration by testing different behaviors via parameter adjustments, circumventing the need for retraining policies and facilitating online demonstrations by human operators.

A. Auxilary Rewards :

Locomotion gaits trained solely on task rewards struggle with real-world transferability. Incorporating auxiliary rewards that emphasize maintaining specific contact schedules, smooth actions, energy efficiency, and foot clearance is essential. [1,2,3,5,6,7]. such auxilary rewards can be biases for generalization

B. Learning Locomotion with Paramterized Style:

Recent works have integrated gait parameters into task specifications, leveraging parameter-augmented auxiliary rewards for bipedal locomotion control.[8,9]. The paper takes inspiration from such approach

C. Methods:

To achieve MoB, They trained a conditional policy $\pi(\cdot \mid \mathbf{c}_t, \mathbf{b}_t)$ capable of completing tasks specified by commands $(\mathbf{c}_t)$ in multiple ways, determined by different behavior parameters $(\mathbf{b}_t)$.

Task Specification: They considered the task of omnidirectional velocity tracking. This task is specified by a 3 -dimensional command vector $\mathbf{c}_t = \left[\mathbf{v}_x^{\mathrm{cmd}}, \mathbf{v}_y^{\mathrm{cmd}}, \boldsymbol{\omega}_z^{\mathrm{cmd}}\right]$ where $\mathbf{v}_x^{\mathrm{cmd}}, \mathbf{v}_y^{\mathrm{cmd}}$ are the desired linear velocities in the body-frame x- and y-axes, and ω_z^{cmd} is the desired angular

velocity in the yaw axis.

Behavior Specification: They parameterized the style of task completion by an 8-dimensional vector of behavior parameters, $\mathbf{b}_t$:

$$\mathbf{b}_t = \left[\boldsymbol{\theta}_1^{\mathrm{cmd}}, \boldsymbol{\theta}_2^{\mathrm{cmd}}, \boldsymbol{\theta}_3^{\mathrm{cmd}}, \boldsymbol{f}^{\mathrm{cmd}}, \boldsymbol{h}_z^{\mathrm{cmd}}, \boldsymbol{\phi}^{\mathrm{cmd}}, \boldsymbol{s}_y^{\mathrm{cmd}}, \boldsymbol{h}_z^{f,\mathrm{cmd}}\right].$$

where $\boldsymbol{\theta}^{\mathrm{cmd}} = \left(\boldsymbol{\theta}_1^{\mathrm{cmd}}, \boldsymbol{\theta}_2^{\mathrm{cmd}}, \boldsymbol{\theta}_3^{\mathrm{cmd}}\right)$ are the timing offsets between pairs of feet which are used to represent gaits. f^{cmd} is the stepping frequency expressed in Hz. As an example, commanding $\boldsymbol{f}^{\mathrm{cmd}} = 3$ Hz will result in each foot making contact three times per second. $\boldsymbol{h}_z^{\mathrm{cmd}}$ is the body height command; $\boldsymbol{\phi}^{\mathrm{cmd}}$ is the body pitch command. $\boldsymbol{s}_y^{\mathrm{cmd}}$ is the foot stance width command; $\boldsymbol{h}_z^{f,\,\mathrm{cmd}}$ is the footswing height command.

Reward Functions: All reward terms are listed in Table 1. Task rewards for body velocity tracking are defined as functions of the command vector $\mathbf{c}_t$. Auxiliary rewards are used constrain the quadruped's motion for various reasons. "Fixed" auxiliary rewards are independent of behavior parameters $(\mathbf{b}_t)$ and encourage stability and smoothness across all gaits for better sim-to-real transfer. During training, one concern is that the robot might abandon its task or choose an early termination when the task reward is overwhelmed by penalties from the auxiliary objectives. To resolve this, as in [7], we force the total reward to be a positive linear function of the task reward by computing it as $r_{\text{task}} \exp\left(c_{\text{aux}}\, r_{\text{aux}}\right)$ where r_{task} is the sum of (positive) task reward terms and r_{aux} is the sum of (negative) auxiliary reward terms (we use $c_{\text{aux}} = 0.02$). This way, the agent is always rewarded for progress towards the task, more when auxiliary objectives are satisfied and less when they are not.

Task and Behavior Sampling: In order to learn graceful online transitions between behaviors, They resampled the desired task and behavior within each training episode. To enable the robot to both run and spin fast, They sampled the task $\mathbf{c}_t = \left(\mathbf{v}_x^{\mathrm{cmd}}, \mathbf{v}_y^{\mathrm{cmd}}, \boldsymbol{\omega}_z^{\mathrm{cmd}}\right)$ using the grid adaptive curriculum strategy from [3]. Then, they sampled a target behavior $\mathbf{b}_t$. First, they sampled $\left(\boldsymbol{\theta}_1^{\mathrm{cmd}}, \boldsymbol{\theta}_2^{\mathrm{cmd}}, \boldsymbol{\theta}_3^{\mathrm{cmd}}\right)$ as one of the symmetric quadrupedal contact patterns (pronking, trotting, bounding, or pacing) which are known as more stable and which we found a sufficient basis for diverse useful gaits. Then, the remaining command parameters $\left(\mathbf{v}_y^{\mathrm{cmd}}, \boldsymbol{f}^{\mathrm{cmd}}, \boldsymbol{h}_z^{\mathrm{cmd}}, \boldsymbol{\phi}^{\mathrm{cmd}}, \boldsymbol{h}_z^{f,\mathrm{cmd}}, \boldsymbol{s}_y^{\mathrm{cmd}}\right)$ are sampled independently and uniformly.

Policy Input: The input to the policy is a 30 -step history of observations $\mathbf{o}_{t-H\ldots t}$, commands $\mathbf{c}_{t-H\ldots t}$, behaviors $\mathbf{b}_{t-H\ldots t}$, previous actions $\mathbf{a}_{t-H-1\ldots t-1}$, and timing reference variables $\mathbf{t}_{t-H\ldots t}$. The observation space $\mathbf{0}_t$ consists of joint positions and velocities $\mathbf{q}_t, \dot{\mathbf{q}}_t$ (measured by joint encoders) and the gravity vector in the body frame $\mathbf{g}_t$ (measured by accelerometer). The timing reference variables $\mathbf{t}_t = \left[\sin\left(2\pi t^{\mathrm{FR}}\right), \sin\left(2\pi t^{\mathrm{FL}}\right), \sin\left(2\pi t^{\mathrm{RR}}\right), \sin\left(2\pi t^{\mathrm{RL}}\right)\right]$ are computed from the offset timings of each foot: $\left[t^{\mathrm{FR}}, t^{\mathrm{FL}}, t^{\mathrm{RR}}, t^{\mathrm{RL}}\right] = \left[t + \boldsymbol{\theta}_2^{\mathrm{cmd}} + \boldsymbol{\theta}_3^{\mathrm{cmd}}, t + \boldsymbol{\theta}_1^{\mathrm{cmd}} + \boldsymbol{\theta}_3^{\mathrm{cmd}}, t + \boldsymbol{\theta}_1^{\mathrm{cmd}}, t + \boldsymbol{\theta}_2^{\mathrm{cmd}}\right]$, where t is a counter variable that advances from 0 to 1 during each gait cycle and ${}^{\mathrm{FR}}, {}^{1}\mathrm{FL}, {}^{\mathrm{RR}}, \mathrm{RL}^{2}$ are the four feet. This form is adapted from [8] to express quadrupedal gaits.

Policy Architecture: Our policy body is an MLP with hidden layer sizes $[512, 256, 128]$ and ELU activations. Besides the above, the policy input also includes estimated domain parameters: the velocity of the robot body and the ground friction, which are predicted from the observation history using supervised learning in the manner of [7]. The estimator module is an MLP with hidden layer sizes $[256, 128]$ and ELU activations.

III. Our Work

A. *Phase 1 :Training Locomotion Policy*

The initial goal is to train a velocity-conditioned on $\left(\mathbf{v}_x^{\mathrm{cmd}}, \mathbf{v}_y^{\mathrm{cmd}}, \boldsymbol{\omega}_z^{\mathrm{cmd}}\right)$ aka Locomotion Policy denoted as $\pi_\theta(\cdot)$, where θ represents the parameters. This policy takes the sensory data and velocity commands as input and outputs the the joint position commands to the simulation module. These commands are then converted to joint torques by a PD controller, enabling the robot to walk at commanded longitudinal (v_{cmd_x}), lateral (v_{cmd_y}), and angular $(\omega_{\mathrm{cmd}_x})$ velocities. The sensory data comprises of joint positions $q \in R^{12}$, joint velocities $\dot{q} \in R^{12}$, and g representing the orientation of the gravity vector in the robot's body frame. The output (action) $a_t \in R^{12}$ of the policy is the joint position commands. The input to the policy is a history of length H_{loc} of the sensory data and the actions taken, denoted by $o_{t-H_{\text{loc}}:t}$, where $o_t = (q_t, \dot{q}_t, g_t, a_{t-1})$. Since the policy is velocity-commanded, the complete input to the policy is denoted by $x_{t-H_{\text{loc}}:t}$, where $x_t = (o_t, v_{\mathrm{cmd}_t})$. The action is computed as $a_t = \pi_{\text{loc}}(x_{t-H_{\text{loc}}:t})$.

A teacher-student architecture from the works of walk these ways paper , is used to train the policy. The objective in this phase is to train the policy such that the robot can only follow commanded velocities in the range $[-0.5\text{m/s}, +0.5\text{m/s}]$ in any direction (forward, backward, left, right, clockwise, or anti-clockwise).

B. *Phase 2: Training Navigation Policy*

Once the locomotion policy π_{loc} is obtained, it will be used for robot navigation. A policy $\pi_{\text{nav}}(\cdot)$ will be trained using reinforcement learning for navigating a walled corridor from start position A to goal position B at a distance of K from A within a time limit T_{max}, ensuring the robot does not collide with the wall or the obstacles. The velocity

Term	Equation	Weight
$r_{v_{x,y}^{\text{cmd}}}$: xy velocity tracking	$\exp\{-\lvert\mathbf{v}_{xy} - \mathbf{v}_{xy}^{\text{cmd}}\rvert^2/\sigma_{vxy}\}$	0.02
$r_{\omega_z^{\text{cmd}}}$: yaw velocity tracking	$\exp\{-(\boldsymbol{\omega}_z - \boldsymbol{\omega}_z^{\text{cmd}})^2/\sigma_{\omega z}\}$	0.01
$r_{c_f^{\text{cmd}}}$: swing phase tracking (force)	$\sum_{\text{foot}}[1 - C_{\text{foot}}^{\text{cmd}}(\boldsymbol{\theta}^{\text{cmd}}, t)]\exp\{-\lvert\mathbf{f}^{\text{foot}}\rvert^2/\sigma_{cf}\}$	−0.08
$r_{c_v^{\text{cmd}}}$: stance phase tracking (velocity)	$\sum_{\text{foot}}[C_{\text{foot}}^{\text{cmd}}(\boldsymbol{\theta}^{\text{cmd}}, t)]\exp\{-\lvert\mathbf{v}_{xy}^{\text{foot}}\rvert^2/\sigma_{cv}\}$	−0.08
$r_{\boldsymbol{h}_z^{\text{cmd}}}$: body height tracking	$(\boldsymbol{h}_{\boldsymbol{z}} - \boldsymbol{h}_z^{\text{cmd}})^2$	−0.2
$r_{\boldsymbol{\phi}^{\text{cmd}}}$: body pitch tracking	$(\boldsymbol{\phi} - \boldsymbol{\phi}^{\text{cmd}})^2$	−0.1
$r_{\boldsymbol{s}_y^{\text{cmd}}}$: raibert heuristic footswing tracking	$(\mathbf{p}_{x,y,\text{foot}}^f - \mathbf{p}_{x,y,\text{foot}}^{f,\text{cmd}}(\boldsymbol{s}_y^{\text{cmd}}))^2$	−0.2
$r_{\boldsymbol{h}_{\boldsymbol{z}}^{f,\text{cmd}}}$: footswing height tracking	$\sum_{\text{foot}}(\boldsymbol{h}_{z,\text{foot}}^f - \boldsymbol{h}_z^{f,\text{cmd}})^2 C_{\text{foot}}^{\text{cmd}}(\boldsymbol{\theta}^{\text{cmd}}, t)$	−0.6
z velocity	$\mathbf{v}_z^2$	−4e−4
roll-pitch velocity	$\lvert\boldsymbol{\omega}_{xy}\rvert^2$	−2e−5
foot slip	$\lvert\mathbf{v}_{xy}^{\text{foot}}\rvert^2$	−8e−4
thigh/calf collision	$\mathbb{1}_{\text{collision}}$	−0.02
joint limit violation	$\mathbb{1}_{q_i > q_{max} \lvert\lvert q_i < q_{min}}$	−0.2
joint torques	$\lvert\boldsymbol{\tau}\rvert^2$	−2e−5
joint velocities	$\lvert\dot{\mathbf{q}}\rvert^2$	−2e−5
joint accelerations	$\lvert\ddot{\mathbf{q}}\rvert^2$	−5e−9
action smoothing	$\lvert\mathbf{a}_{t-1} - \mathbf{a}_t\rvert^2$	−2e−3
action smoothing, 2nd order	$\lvert\mathbf{a}_{t-2} - 2\mathbf{a}_{t-1} + \mathbf{a}_t\rvert^2$	−2e−3

Task

Augmented Auxiliary

Fixed Auxiliary

Fig. 1. Reward structure: task rewards, augmented auxiliary rewards, and fixed auxiliary rewards.

commands provided by $\pi_{\text{nav}}(\cdot)$ will facilitate motion, where $\hat{v}_{\text{cmd}_t} = \pi_{\text{nav}}(\cdot)$ is the command velocity used as part of the input in x_t for the underlying lower level Locomotion Policy $\pi_\theta(\cdot)$

1) Creation of a navigator class in a `navigator.py` file, along with its configuration file `navigator_config.py`, which behaves like a gym and contains the step function that commands the locomotion policy to move the robot. The distance between A and B is defined as $K = 3$m.
2) Setting up the environment with walls. The walls are 4m long and are at a distance of -1m and $+1$m from the robotâs origin in the y-direction and obstacles are randomly rendered per scenario.
3) Implement the Actor-Critic Proximal Policy Optimization (PPO) algorithm, defining the input space, designing a reward function, and training the robot to navigate without colliding with the walls. Training will be conducted on many environments in parallel.
4) Training a new policy with a randomized initial position and orientation of the robot over specified ranges. This policy ensures that the robot can navigate to the goal from any random location and orientation within the given ranges.

C. *Phase 3: Navigation with Obstacle Avoidance*

In this phase, a new navigation policy $\pi_{\text{obsnav}}(\cdot)$ with obstacle avoidance capabilities will be built. This policy includes information about the obstacleâs location and size in the entire environment, enabling the policy to learn to avoid obstacles present.

1) Using the existing environment defined in Phase 2, we add to this an obstacle that cannot be moved when a force is applied to it. The obstacle location and size is varied for each environment.
2) Implement the Actor-Critic Proximal Policy Optimization (PPO) algorithm, defining the input space, designing a reward function, and training the robot to navigate without colliding with the walls and obstacles. Training will be conducted on many environments in parallel.
3) In the input observational space we mention the location and size of the obstacle. Thereby enabling the policy to understand the location and size of the obstacle such that the policy can learn to avoid the obstacle.

D. *Simulation of Scenarios with walls and Obstacles*

In the second phase of the project, we embarked on the task of training our robot to navigate through a diverse range of environments. To simulate real-world scenarios effectively, we developed a total of 1024 environments for training purposes. Each environment features a robotic dog amidst four surrounding walls as in Fig 2, providing a confined space that challenges the robot's navigation abilities. Additionally for the third phase of the project, we introduced a dynamic element to the environments by incorporating randomly sized rectangular blocks, strategically placed as obstacles within the robot's path as in Fig 3. These obstacles serve as formidable challenges, requiring the robot to adeptly maneuver around them to reach its destination. When the robotic dog comes in contact with any of these obstacles it terminates the simulation and resets itself to a new initialized step. The initialization process of the robotic dog is meticulously designed to inject randomness and diversity into training scenarios. Each robotic dog begins its journey at a randomized location along the x-axis, constrained

within the interval [0m, 0.8m], and along the y-axis, ranging from [-0.5m, 0.5m]. Moreover, the orientation of the dog, crucial for its spatial awareness, is set within the range [$-\pi$, π] relative to the environmental base position. This comprehensive setup not only enriches the training process but also reflects real-world scenarios, where robotic dogs must adeptly navigate through constrained environments while encountering unforeseen obstacles.

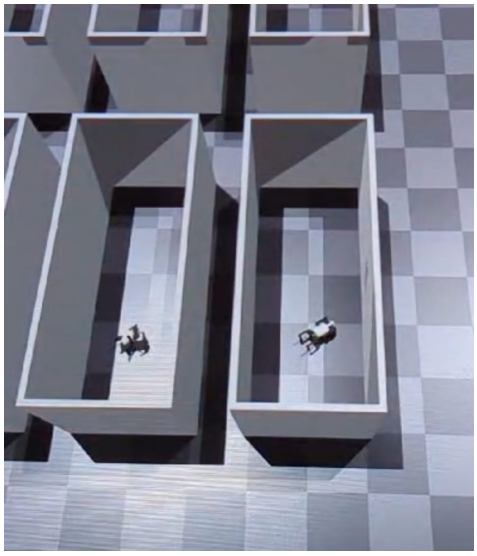

Fig. 2. Phase 2 Simulation

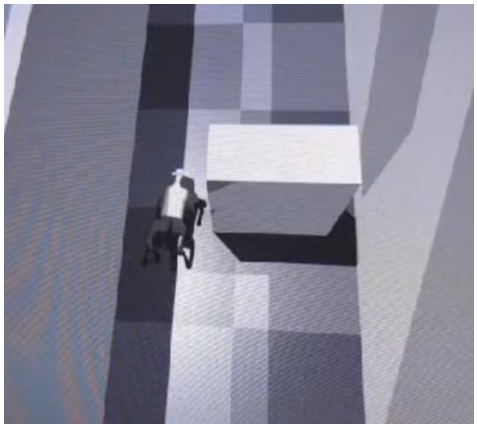

Fig. 3. Phase 3 Simulation

*E. **Reward Functions***

Four Key reward functions were implemented to improve the navigation policy.

Linear position tracking: We track the position of the robot and reward it based on the distance that has travelled along the x axis. We calculate r_{p_x} : x position tracking reward with weight 2.

$$r_{p_x} = \exp\left\{-\frac{|p_x - p_{\text{cmd}_x}|^2}{\sigma_{p_x}}\right\}$$

Goal body alignment: We track the orientation of the robot and reward it based on whether it is pointing towards the goal position. We calculate the cosine similarity with weight 0.1

$$\text{Cosine Similarity} = \frac{\mathbf{v}_{\text{goal}} \cdot \mathbf{v}_{\text{body}}}{\|\mathbf{v}_{\text{goal}}\| \cdot \|\mathbf{v}_{\text{body}}\|}$$

Action smoothness : Robots should be penalized for taking adverse actions with high magnitudes resulting in termination or collision with obstacles. We penalize the resultant magnitude of the velocity outputs as

$$|v_t - 1|^2$$

with weight 0.001 resulting in safer locomotion velocities

Destination Push: We observed that that the robots rotate in circles when it nears the goal position but doesn't cross the 3 meters line from the start. So we gave a destination reward of 400 units when it does cross the final line reaching the goal state.

IV. RESULTS

In the initial phase of our project, we focused on training 1024 environments of the robotic dog for 30000 iterations with multiple gaits using Proximal Policy Optimization algorithm to execute a comprehensive locomotion policy, encompassing forward and backward movement as well as angular rotation. Through extensive training and evaluation, we achieved significant milestones in enhancing the robot's locomotion capabilities. The results indicate that the robot successfully learned to navigate forward and backward with remarkable stability and precision. Moreover, the implementation of angular rotation was executed with finesse, enabling the robot to pivot effectively in various directions.

Quantitative analysis revealed that the robotic dog consistently achieved high accuracy rates in executing locomotion commands, with minimal deviations from desired trajectories. The forward and backward movements exhibited minimal oscillations, highlighting the effectiveness of the learned locomotion policy in maintaining stability during motion. Furthermore, angular rotation maneuvers were executed with agility and precision. We can see the plot with command velocities given to the robot at certain time intervals and the simulation velocities attained by the robot with wave functions in Fig 4.

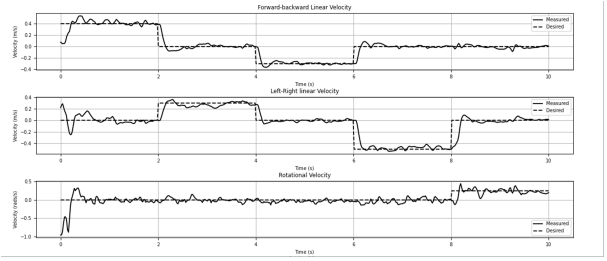

Fig. 4. Locomotion Policy : Robot Simulation Velocities with respect to command velocity Inputs a) Forward and backward velocities b) Left and Right velocities d) Angular velocities

In the second phase of the project , Similar to the phase one , we trained 1024 environments for 4000 iterations with surrounded walls around the robotic dog.Upon training the navigator policy $\pi_{\text{nav}}(\cdot)$ using the Actor-Critic Proximal Policy Optimization algorithm, the performance of the robot in navigating the walled corridor from start position A to goal position B has been evaluated.

The navigator policy which orchestrates the locomotion policy has demonstrated efficient navigation capabilities, successfully guiding the robot from the starting position to the goal position within the specified distance of 3 meters. Through reinforcement learning, the robot has learned to avoid collisions with walls present within the environment. This indicates the effectiveness of the reward function design and the training process in promoting safe navigation. In Fig 5 , we see the plots of all the rewards attained per iteration for the second phase . We also observed that the number of successful simulations across the environments converges around 150 per iteration as shown in Fig 6. Fig 9 Visualizes three simulated paths.

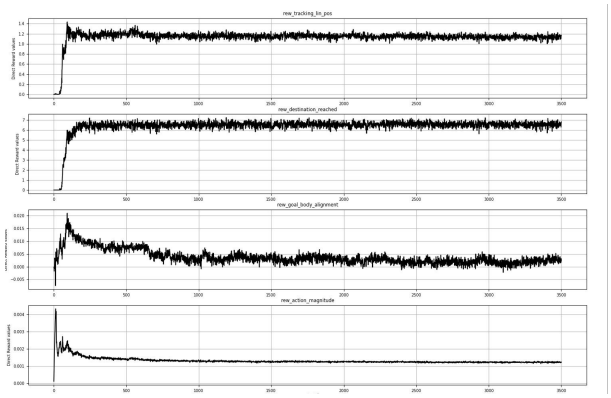

Fig. 5. Plots of Rewards accumulated over iterations for a) Linear Position Reward b) Destination Push c) Goal Body Alignment d) Action Magnitude

In the third phase of the project , we have randomly initialized an obstacle in the path during navigation and trained the model as we did for the second phase. The presence of the

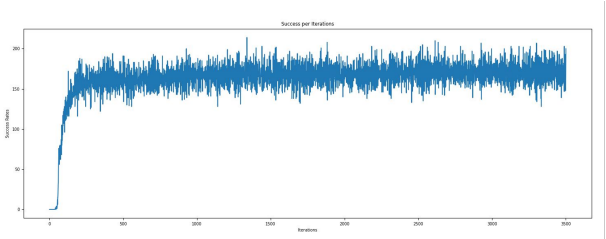

Fig. 6. Plot of the number of successful simulations per iteration

obstacle introduced an additional layer of complexity to the navigation task, requiring the robot to perceive, evaluate, and react to dynamic environmental changes. The trained navigator policy has demonstrated effective obstacle avoidance behavior in response to the presence of the randomly initialized obstacle. The robot successfully navigated around the obstacle while maintaining progress towards the goal position. In Fig 7 , we see the plots of all the rewards attained per iteration for the third phase . We also observed that the number of successful simulations across the environments converges around 120 per iteration as shown in Fig 8. Fig 10 visualized three simulated paths.

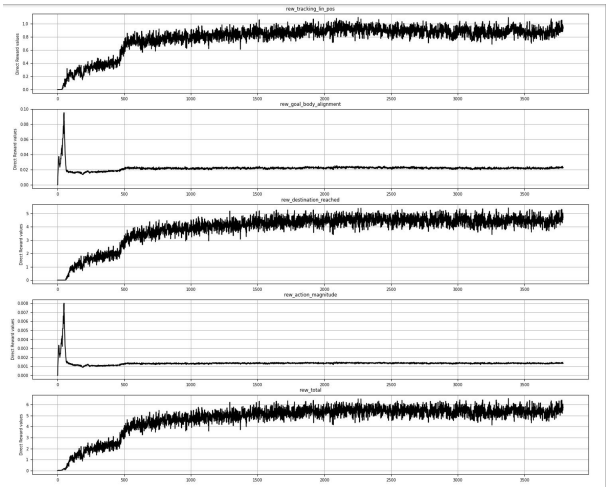

Fig. 7. Plots of Rewards accumulated over iterations for a) Linear Position Reward b) Destination Push c) Goal Body Alignment d) Action Magnitude

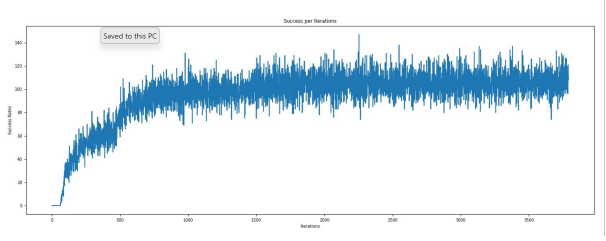

Fig. 8. Plot of the number of successful simulations per iteration

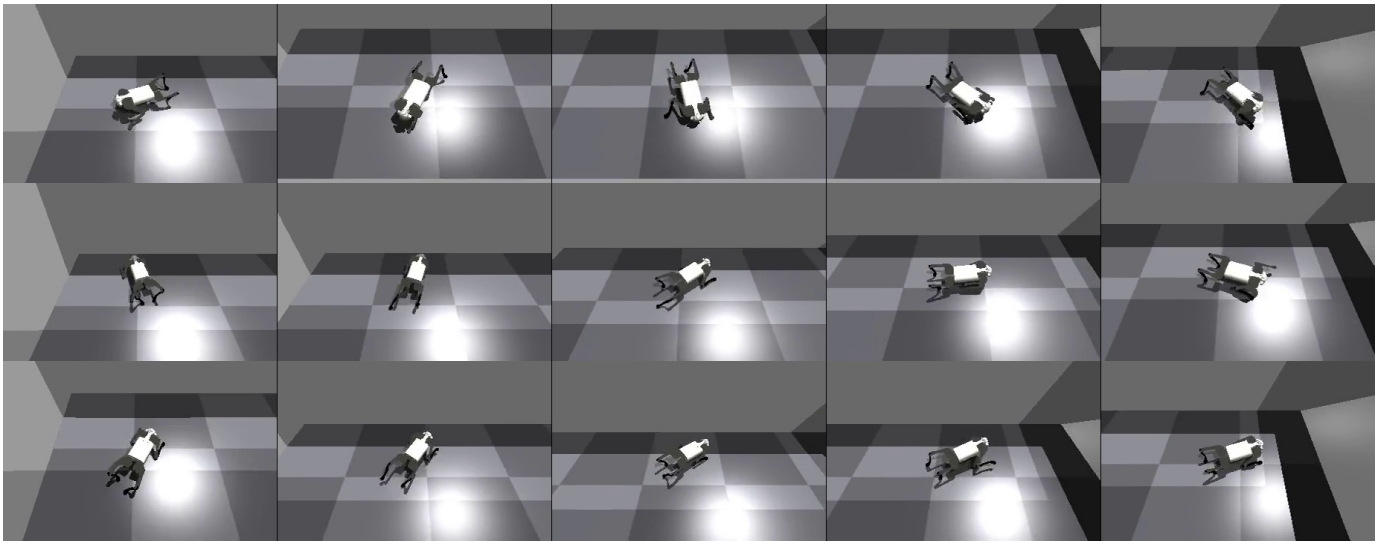

Fig. 9. Three Simulated paths from Phase 2

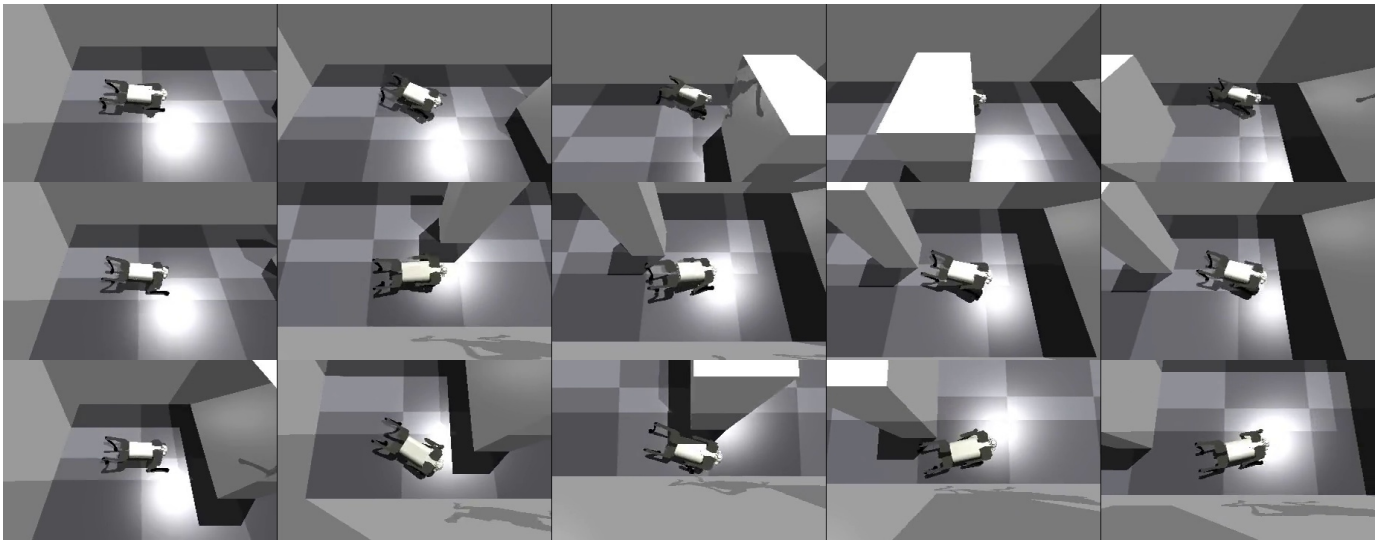

Fig. 10. Three Simulated paths from Phase 3

V. Conclusion

In conclusion, this project presents a comprehensive framework for quadruped navigation and obstacle avoidance using Actor-Critic Proximal Policy Optimization (PPO) algorithm. The framework consists of three phases: locomotion training, navigation within walled environments, and navigation with obstacle avoidance capabilities.

In the initial phase, a velocity-conditioned neural network policy for stable gait locomotion was trained, allowing the robot to move at specified velocities while maintaining stability. Subsequent phases extended the capabilities of the navigation policy to navigate within walled corridors and incorporate obstacle avoidance.

Through extensive training and evaluation, significant milestones were achieved in enhancing the robot's locomotion capabilities. The robot successfully learned to navigate forward and backward with remarkable stability and precision, while also executing angular rotation maneuvers effectively. The implementation of reward functions facilitated safe navigation, ensuring the robot avoided collisions with walls and obstacles while progressing towards the goal position.

The results demonstrated the effectiveness of the reinforcement learning approach in training policies for versatile navigation tasks. The trained policies exhibited efficient navigation capabilities, successfully guiding the robot through diverse environments while avoiding obstacles and maintaining stability.

Overall, this project provides a promising framework for autonomous quadruped navigation in complex environments, with potential applications in various industries such as search and rescue operations, agricultural automation, and more. The robust policies learned through reinforcement learning offer adaptability and versatility, making them suitable for real-world deployment in diverse scenarios.

REFERENCES

[1] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Sci. Robot.*, 5(47):eabc5986, Oct. 2020. doi:10.1126/scirobotics.abc5986.

[2] A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: Rapid motor adaptation for legged robots. In *Proc. Robot.: Sci. and Syst. (RSS)*, Virtual, July 2021. doi:10.48550/arXiv.2107.04034.

[3] G.B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid locomotion via reinforcement learning. *Proc. Robot.: Sci. and Syst. (RSS)*, June 2022. doi:10.48550/arXiv.2205.02824.

[4] P. Agrawal. The task specification problem. In *Conference on Robot Learning*, pages 1745-1751. PMLR, 2022.

[5] Z. Fu, A. Kumar, J. Malik, and D. Pathak. Minimizing energy consumption leads to the emergence of gaits in legged robots. In *Proc. Conf. Robot Learn. (CoRL)*, pages 928-937, London, UK, Nov. 2021. doi:10.48550/arXiv.2111.01674.

[6] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Proc. Conf. Robot Learn. (CoRL)*, pages 91-100, London, UK, Nov. 2021. doi:10.48550/arXiv.2109.11978.

[7] G. Ji, J. Mun, H. Kim, and J. Hwangbo. Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion. *IEEE Robot. Automat. Lett. (RA-L)*, 7(2):4630-4637, Apr. 2022. doi:10.1109/LRA.2022.3151396.

[8] J. Siekmann, Y. Godse, A. Fern, and J. Hurst. Sim-to-real learning of all common bipedal gaits via periodic reward composition. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 7309-7315, Xiâan, China, June 2021. doi:10.1109/ICRA48506.2021.9561814.