Capstone Project Report

Learning Residual Footstep Policy for Quadruped Locomotion

by

Anirudh Gudi

Submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Robotics

UNIVERSITY OF CALIFORNIA, RIVERSIDE

January 15, 2026

Committee:

Dr. Ioannis Karamouzas, Chair
Dr. Jiachen Li

Abstract

This project investigates a hybrid locomotion framework that augments a classical Raibert-style footstep planner with a deep reinforcement learning (RL) policy for quadruped robots. The core idea is to retain the stability and interpretability of a model-based planner, while allowing a neural network to learn small residual corrections to nominal foothold locations from experience. The policy observes local terrain heightmaps, robot base state, and foot configurations, and outputs 12-dimensional residual adjustments to Raibert's nominal foot placements. These corrected footholds are then tracked by a quad-sdk inverse dynamics controller, whose leg commands are applied to a MuJoCo simulation of the quadruped. Safety is enforced analytically through kinematic workspace limits, terrain and clearance checks, and stability guards based on center-of-mass margin and support polygon geometry, rather than via a learned safety mask.

At the current stage, the end-to-end pipeline comprising the PPO actor–critic policy, the Raibert footstep prior, the quad-sdk-based inverse dynamics controller, and the MuJoCo backend has been implemented and validated on flat terrain, where the residual policy learns to refine but not overwrite the model-based behavior. The long-term objective is to extend training to rough and irregular terrains and to quantitatively evaluate whether the residual RL component improves robustness and stability compared to Raibert-only and quad-sdk's built-in footstep planning. This setting provides a principled testbed for comparing purely model-based footstep planning with a model-based-plus-learning approach under a realistic control stack, and for studying how learned residuals can compensate for modeling errors, complex terrain interactions, and controller limitations.

1 Introduction

Legged robots require careful footstep planning to maintain balance, avoid obstacles, and generate stable locomotion. Unlike wheeled robots, quadrupeds must continuously reason about contact locations with the environment, making foothold selection a critical component of the locomotion pipeline. Classical approaches to legged locomotion rely heavily on simplified models and heuristics, which provide robustness and interpretability but often lack adaptability to new terrains.

One of the earliest and most influential contributions in this space is Raibert's work on dynamically stable legged locomotion, which introduced inverted-pendulum-based heuristics for foot placement [Raibert, 1983]. In this framework, footholds are placed relative to the robot's center-of-mass (COM) velocity to regulate forward motion and maintain balance. Despite their simplicity, Raibert-style heuristics remain widely used due to their efficiency, stability, and intuitive behavior on flat and moderately uneven terrain.

Modern quadrupedal systems extend these ideas using full-body optimization and model predictive control (MPC). For example, IEEE T-RO work on coupled planning, terrain mapping, and whole-body control demonstrates how integrating terrain perception directly into motion planning can enable robust locomotion across non-coplanar terrains [Mastalli et al., 2020]. In addition, fast MPC-style foothold replanning combined with feedback control has been shown to improve robustness of dynamic gaits under disturbances [Xin et al., 2021]. Quad-SDK is a representative example of a full-stack quadrupedal locomotion framework that integrates planning, state estimation, and inverse dynamics control [Norby et al., 2022, Robomechanics Lab, 2025]. Quad-SDK includes a built-in local footstep planner that combines heuristic foot placement with terrain cost maps, and a nonlinear MPC-based controller that computes whole-body motions. While effective, such model-based planners often require manual tuning and assume simplified contact models, which can limit performance when operating on unfamiliar terrain or under modeling inaccuracies.

Recent advances in deep reinforcement learning have demonstrated that policies can learn to adapt foothold placement and gait behaviors from experience, particularly when terrain perception is available. DeepGait is a strong example from IEEE RA-L showing terrain-aware quadrupedal gait planning and control using deep reinforcement learning [Tsounis et al., 2020]. However, purely learning-based approaches often require large amounts of data and may struggle with safety and generalization. SafeSteps proposes a learning-based foothold planner that selects footholds on a discretized grid while enforcing safety through a learned terrain feasibility mask [Omar et al., 2023]. While effective, this approach replaces classical planners entirely and introduces additional learned components that must generalize reliably.

In this project, we pursue a hybrid approach that combines the strengths of classical planning and learning. Rather than learning foothold placement from scratch, we retain a Raibert-style foot-

step planner as a nominal prior and train a reinforcement learning policy to predict small residual corrections to those footholds. The residual policy is trained using Proximal Policy Optimization (PPO) [Schulman et al., 2017] and outputs continuous adjustments to nominal foot targets. Safety is enforced analytically using kinematic workspace constraints and COM support polygon checks rather than learned safety masks.

The goal of this work is to investigate whether such a residual learning formulation can improve robustness and adaptability while preserving the stability and interpretability of model-based planners. The current focus is on flat-terrain locomotion and trajectory tracking, with the long-term objective of extending the system to rough terrain and deploying it on hardware through the Quad-SDK control stack. The simulation backend uses MuJoCo, a widely used physics engine for model-based control and robotics research [Todorov et al., 2012].

2 Related Work

Legged locomotion has been studied extensively through both classical control methods and learning-based approaches. In the context of quadrupedal robots, foothold selection plays a critical role in ensuring stability, robustness, and adaptability to terrain variations. Existing work can broadly be categorized into classical heuristic-based footstep planning, full-stack model-based locomotion frameworks, and learning-based or hybrid approaches.

2.1 Classical Footstep Planning and Heuristic Methods

Early work by Raibert established that dynamically stable legged locomotion can be achieved using simple foot placement heuristics derived from inverted pendulum models [Raibert, 1983]. In this formulation, the foot is placed relative to the robot's center-of-mass (COM) velocity such that forward speed and balance are regulated implicitly. A common expression of the Raibert heuristic places the foot according to

$$\mathbf{x}_f^{\text{nom}} = \mathbf{x}_{\text{hip}} + \frac{T_{\text{step}}}{2}\mathbf{v}_{\text{com}} + k(\mathbf{v}_{\text{com}} - \mathbf{v}_d) \tag{1}$$

where $\mathbf{x}_{\text{hip}}$ is the hip position, $\mathbf{v}_{\text{com}}$ is the current COM velocity, $\mathbf{v}_d$ is the desired velocity, T_{step} is the step duration, and k is a velocity correction gain. This heuristic is attractive due to its simplicity, computational efficiency, and strong stability properties on flat or mildly uneven terrain.

However, Raibert-style planners rely on simplified assumptions about terrain and contact interactions. As a result, they often require manual tuning or additional heuristics to operate reliably on complex or highly irregular terrain. These limitations motivate extensions that integrate terrain awareness or higher-level planning layers.

2.2 Model-Based Quadruped Locomotion Frameworks

Modern quadruped systems often employ full-stack locomotion frameworks that integrate planning, state estimation, and whole-body control. Quad-SDK is a representative example that provides a unified software stack for agile quadrupedal locomotion, combining heuristic footstep planning with nonlinear model predictive control and inverse dynamics [Norby et al., 2022]. Such frameworks are designed to operate across simulation and hardware with minimal changes and offer strong robustness through carefully engineered control pipelines.

Related work in this space emphasizes tight coupling between terrain perception, motion planning, and whole-body control. For example, Mastalli et al. demonstrate that jointly reasoning about terrain geometry, contact constraints, and dynamics enables quadrupeds to traverse non-coplanar terrain more robustly [Mastalli et al., 2020]. While these approaches provide strong guarantees and physical consistency, they typically depend on manually designed cost functions and require retuning when terrain distributions change.

2.3 Learning-Based and Hybrid Approaches

Reinforcement learning has emerged as a powerful tool for legged locomotion, enabling policies to adapt to terrain variations through experience. DeepGait shows that RL can be used to learn quadrupedal gait behaviors that generalize across terrain types, while still leveraging structured control components [Tsounis et al., 2020]. However, purely learning-based methods often suffer from sample inefficiency and pose safety challenges during training.

Hybrid methods that combine learning with model-based control aim to address these limitations. RLOC integrates reinforcement learning with optimal control to achieve terrain-aware locomotion, using learning to adapt higher-level decisions while preserving model-based stability [Gangapurwala et al., 2022]. These approaches highlight the effectiveness of combining structured priors with learning-based adaptation.

SafeSteps proposes a learning-based foothold planner that directly selects footholds on a discretized grid and enforces safety via a learned terrain feasibility mask [Omar et al., 2023]. While effective, this approach replaces classical footstep heuristics entirely and introduces additional learned safety components.

2.4 Positioning of This Work

In contrast to direct foothold selection, this work adopts a residual learning formulation. A Raibert-style planner provides nominal footholds that encode stable baseline behavior, while a reinforcement learning policy predicts small residual corrections to these footholds. Safety is enforced analytically through kinematic workspace constraints and COM support polygon margins rather than

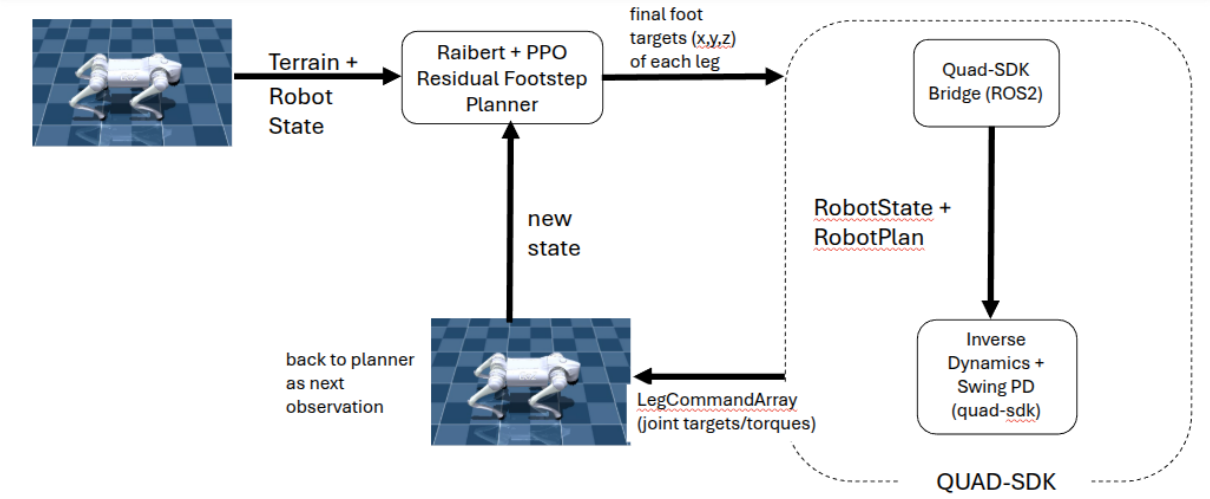


Figure 1: Overall system architecture of the proposed hybrid locomotion framework. Robot state and terrain information are processed by a Raibert-style footstep planner augmented with a PPO-based residual policy. The resulting final foot targets are passed through a ROS2 bridge to Quad-SDK, which computes inverse-dynamics-based leg commands. These commands are executed in a MuJoCo simulation, and the resulting robot state is fed back to the planner, forming a closed loop

learned safety masks. This design preserves interpretability and stability while allowing learning to compensate for modeling errors, controller limitations, and terrain interactions.

By integrating residual learning into an existing Quad-SDK-based control stack, this work aims to bridge the gap between classical footstep planning and learning-based adaptation in a practical and extensible manner.

3 Methodology

This section describes the proposed hybrid locomotion framework, focusing on how a learned residual footstep policy is integrated into an existing Quad-SDK-style control stack. Figure 1 summarizes the closed-loop pipeline: the planner consumes robot state (and optionally terrain), produces final foot targets, Quad-SDK computes leg commands via inverse dynamics and swing tracking, and the simulator returns the next state back to the planner.

3.1 System Overview and Closed-Loop Data Flow (Figure 1)

The pipeline in Figure 1 consists of three conceptual blocks:

(1) Raibert + PPO Residual Footstep Planner At each planning tick, the planner receives the robot base state (position/orientation and velocity), foot configuration (foot positions relative to the base/COM), and a motion intent (desired direction or velocity along a global plan). When ter-

rain sensing is enabled, the planner also receives local terrain information (e.g., heightmap patches around candidate footholds). The planner first computes nominal footholds using a Raibert-style heuristic, then adds a learned residual correction predicted by a PPO policy [Raibert, 1983, Schulman et al., 2017]. The output is a set of *final* foothold targets for all four legs in Cartesian space.

(2) Quad-SDK Bridge (ROS2) The foot targets are injected into the control stack through a ROS2 bridge that provides the interfaces expected by Quad-SDK (e.g., `RobotState` and `RobotPlan`) and calls into the inverse dynamics and swing leg tracking modules [Norby et al., 2022, Robomechanics Lab, 2025]. Conceptually, this block translates "where should the feet go" into "what joint commands/torques should be applied."

(3) Low-Level Execution (Inverse Dynamics + Swing PD) and Simulation Quad-SDK produces a `LegCommandArray` (joint targets and/or torques). These commands are applied to a MuJoCo simulation backend, which advances physics and contact, and the resulting state is fed back to the planner as the next observation [Todorov et al., 2012]. This completes the closed loop shown in Figure 1.

This separation is deliberate: the learned component influences *foothold selection* while the existing whole-body controller handles *dynamic stabilization* and tracking. The learning problem is therefore constrained to a high-level decision (residual foot placement), making the system more interpretable and easier to debug than end-to-end torque policies.

3.2 Nominal Footstep Prior: Raibert-Style Placement

The nominal foothold for leg i is computed using a Raibert-style heuristic, which can be interpreted as a velocity-regulation rule derived from inverted pendulum intuition [Raibert, 1983]. A common form is:

$$\mathbf{x}_{f,i}^{\text{nom}} = \mathbf{x}_{\text{hip},i} + \frac{T_{\text{step}}}{2}\mathbf{v}_{\text{com}} + k_{\text{step}}\left(\mathbf{v}_{\text{com}} - \mathbf{v}_d\right) \tag{2}$$

where $\mathbf{x}_{\text{hip},i}$ is the hip position in world coordinates, $\mathbf{v}_{\text{com}}$ is the planar COM velocity, $\mathbf{v}_d$ is the desired velocity along the plan, T_{step} is the step duration, and k_{step} is a gain. Intuitively:

- The $\frac{T_{\text{step}}}{2}\mathbf{v}_{\text{com}}$ term predicts where the COM will be mid-step and places the foot to "catch" it
- The $k_{\text{step}}(\mathbf{v}_{\text{com}} - \mathbf{v}_d)$ term corrects velocity error by shifting the foothold forward/backward

This prior is strong on flat terrain and provides a stable default gait structure. The learning component is designed to *refine* this plan rather than replace it.

3.3 Residual Footstep Policy

3.3.1 Action Space

The policy outputs a continuous residual for each leg in Cartesian space:

$$\mathbf{a} = [\Delta x_{\mathrm{FL}}, \Delta y_{\mathrm{FL}}, \Delta z_{\mathrm{FL}}, \Delta x_{\mathrm{FR}}, \Delta y_{\mathrm{FR}}, \Delta z_{\mathrm{FR}}, \Delta x_{\mathrm{RL}}, \Delta y_{\mathrm{RL}}, \Delta z_{\mathrm{RL}}, \Delta x_{\mathrm{RR}}, \Delta y_{\mathrm{RR}}, \Delta z_{\mathrm{RR}}] \in \mathbb{R}^{12} \tag{3}$$

The final foot target is:

$$\mathbf{x}_{f,i}^{\text{final}} = \mathbf{x}_{f,i}^{\text{nom}} + \Delta \mathbf{x}_i, \quad \Delta \mathbf{x}_i = [\Delta x_i, \Delta y_i, \Delta z_i]^\top \tag{4}$$

In the current system, $\Delta x_i, \Delta y_i$ are the primary learned corrections on flat terrain; Δz_i is included for a consistent interface and future rough-terrain support (e.g., step-up/step-down), but in early integration it may be applied conservatively to avoid destabilizing vertical oscillations.

3.3.2 Observation Space

The observation is structured to give the policy enough context to refine footholds while keeping the state compact:

- **Base motion:** planar COM velocity (v_x, v_y) and yaw orientation. Yaw is encoded as $(\sin\psi, \cos\psi)$ to avoid discontinuity.
- **Task intent:** a goal-direction unit vector or desired velocity direction along the global plan.
- **Foot configuration:** current foot positions relative to the COM/base (planar components), allowing the policy to reason about stance width and gait phase.
- **Terrain (optional):** local heightmap patches (flattened) around candidate footholds when training for rough terrain.

This design mirrors the hybrid nature of the system: the model-based prior already encodes much of the structure, so the policy can focus on "what correction is needed given the current state and terrain"

3.4 Safety and Feasibility Guards

Unlike approaches that rely on a learned safety mask for footholds [Omar et al., 2023], this project enforces safety analytically at the planner/environment boundary. The goal is to guarantee basic feasibility even during exploration. The main guards are:

Kinematic workspace limits Each leg has a reachable workspace in the body frame. After applying residuals, footholds are clamped to these bounds to prevent unreachable swing targets.

Leg-side constraints and clearance Footholds are constrained to remain on their respective side of the sagittal plane (left legs remain left, right legs remain right) and satisfy minimum inter-foot clearance to reduce self-collisions and entanglement.

Support polygon and COM margin The stance feet define a support polygon. We compute a stability margin as the distance between the projected COM and the polygon boundary (positive inside). When this margin becomes small, we reduce aggressiveness by scaling commanded motion (and/or residual magnitude), effectively acting as a stability "guardrail." This is consistent with prior work that couples planning with whole-body control and stability constraints [Mastalli et al., 2020].

These checks are intentionally simple but effective: they keep learning within a safe operating region and make failures easier to interpret (e.g., a fall is due to insufficient margin or poor foothold quality rather than an opaque policy decision).

3.5 Reward Function Design

The reward is shaped to (i) move along the commanded trajectory, (ii) maintain stability, and (iii) keep residual corrections small and smooth. At a high level:

$$r_t = w_{\text{prog}}\, r_{\text{prog}} + w_{\text{track}}\, r_{\text{track}} + w_{\text{stab}}\, r_{\text{stab}} - w_{\text{res}}\, \|\mathbf{a}_t\|_2^2 - w_{\text{coll}}\, \mathbb{I}\{e_{\min,t} < e_{\text{coll}}\} + w_{\text{alive}} \tag{5}$$

where:

Progress / trajectory following (r_{prog}) Encourages forward movement along the goal direction. A practical implementation is the dot product of base displacement with the desired direction over the timestep.

Velocity tracking (r_{track}) Encourages matching the desired speed along the plan, e.g.,

$$r_{\text{track}} = -\|\mathbf{v}_{\text{com}} - \mathbf{v}_d\|_2 \tag{6}$$

possibly normalized by a reference speed.

Stability term (r_{stab}) Rewards maintaining COM inside the support polygon with a comfortable margin, and penalizes near-degenerate support polygons (narrow stance, short edges). This term is crucial for turning trajectories (circle/figure-8), where lateral acceleration can reduce margin.

Residual penalty ($\|\mathbf{a}_t\|_2^2$) Explicitly discourages large corrections so the learned policy remains a residual refinement rather than a replacement for the prior. This improves interpretability

and reduces the chance of unstable "overcorrections".

Termination and collision proxy The current Gym environment terminates an episode under three main failure modes. First, if the center-of-mass (COM) margin drops far outside the support polygon ($\texttt{margin} < -0.50$), the state is treated as unrecoverably unstable and the rollout ends. Second, if the minimum support edge becomes critically small ($e_{\min} < 0.03$ m), we terminate to avoid degenerate foothold configurations that typically correspond to leg interference. Third, if the support area collapses below a small fraction of the nominal stance area, we also terminate to prevent learning from physically implausible contact geometry. Separately from termination, we include a dense collision *penalty proxy*: whenever $e_{\min} < 0.05$ m we apply a fixed negative reward, which acts as a soft guard before the hard 0.03 m termination threshold.

3.6 PPO Training Setup

The residual policy is trained using Proximal Policy Optimization (PPO), an on-policy actor–critic method designed for stable updates [Schulman et al., 2017]. PPO optimizes the clipped surrogate objective:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\big(\rho_t(\theta)\hat{A}_t,\ \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\big)\Big] \tag{7}$$

where $\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio, $\hat{A}_t$ is an advantage estimate (e.g., via Generalized Advantage Estimation), and ϵ is the clipping threshold.

Why PPO here The environment is continuous-control with shaped rewards and safety constraints. PPO's clipped updates reduce destructive policy jumps, which is important when exploration can destabilize the robot. This is especially relevant because the low-level controller is not learned; sudden large changes in footholds can produce tracking failures.

Training protocol Training episodes follow commanded trajectories (square, circle, figure-8) with a trotting gait schedule. Rollouts are collected in simulation and used for multiple epochs of minibatch updates. Standard stabilizers (entropy bonus, value loss coefficient, gradient clipping) are used. The policy learns in the presence of analytic guards, which narrows the explored state distribution and improves training stability.

3.7 Quad-SDK Integration Details

The system integrates learned planning with Quad-SDK by *injecting footholds* rather than modifying the controller. The ROS2 bridge:

- collects the robot state and plan
- forwards the final foothold targets (Figure 1) as part of the planning interface

- calls Quad-SDK control routines (inverse dynamics + swing tracking) to produce low-level leg commands [Norby et al., 2022, Robomechanics Lab, 2025]

This preserves the existing execution semantics of Quad-SDK and keeps the learned component modular. In future work, once the MuJoCo coupling is fully consistent with the controller assumptions, the same learned foothold interface can be used for rough terrain training and eventually for hardware deployment, similar in spirit to hybrid learning-control locomotion pipelines [Gangapurwala et al., 2022, Tsounis et al., 2020].

4 Results and Evaluation

This section reports the current empirical status of the system shown in Figure 1. At this stage, results are primarily *flat-terrain* and *qualitative* and executed in *simulation*, focusing on validating that (i) the residual policy learns stable walking behavior while keeping residuals small, and (ii) the planner–bridge–controller loop can be exercised end-to-end with the same interfaces that will be needed for rough terrain and hardware deployment.

4.1 Experimental Setup

Training environment Training was performed in a lightweight Gym-style environment with an analytic backend for fast iteration. The control timestep was $\Delta t = 0.05$s and episodes were capped at 3500 steps (175s). A fixed trotting gait schedule was used, with a swing interval of 4 timesteps (0.2s) and step length constraints in the range $[0.15, 0.30]$ m, with a minimum forward step of 0.1 m to avoid stagnation. The nominal desired velocity used for most runs was 0.25 m/s in the forward direction.

Trajectories To evaluate tracking and turning behavior beyond straight-line walking, three canonical reference paths were used:

- **Square:** side length 5 m (1 lap)
- **Circle:** radius 4 m (1 lap), sampled at 48 points
- **Figure-8:** radius 3 m with separation 6 m (1 lap), sampled at 96 points

These paths generate a mix of straight segments and sustained curvature, which is useful for testing whether residual footholds remain stable when lateral demands increase.

PPO training details The residual policy was trained using PPO [Schulman et al., 2017] for up to 1.25m environment steps with rollout length 2048, minibatch size 256, and 8 epochs per update. The discount and GAE settings were $\gamma = 0.985$ and $\lambda = 0.95$, with clipping $\epsilon = 0.2$ and entropy coefficient 0.01. Training was executed on a Laptop GPU (RTX 4060).

Execution stack and visualization For the full pipeline shown in Figure 1, Quad-SDK interfaces and message types were used to keep the planner/controller boundary realistic [Norby et al., 2022, Robomechanics Lab, 2025]. MuJoCo [Todorov et al., 2012] was used as the physics simulation backend for visualization and integration testing of the ROS2 bridge topics.

4.2 Evaluation Metrics

Even though the current results are mainly qualitative, the system logs quantities that directly reflect tracking and stability. The key metrics are:

Trajectory progress Progress is measured as displacement along the local goal direction:

$$r_{\text{prog}}(t) \propto \Delta \mathbf{x}_{\text{com}}(t)^{\top} \hat{\mathbf{g}}(t), \tag{8}$$

where $\hat{\mathbf{g}}(t)$ is the unit vector pointing toward the next target along the trajectory.

Velocity tracking error Tracking quality is captured by the instantaneous error:

$$e_v(t) = \left\| \mathbf{v}_{\text{com}}(t) - \mathbf{v}_d(t) \right\|_2, \tag{9}$$

Stability margin (COM vs. support polygon) A geometric stability proxy is the signed distance from the projected COM to the support polygon boundary. Positive margin indicates that the COM projection lies inside the support polygon. The system logs this margin and uses it as both a reward component and a guard signal.

Support geometry quality To avoid degenerate stances, minimum support edge length and a support-area ratio threshold are tracked. These correlate with robustness to disturbances, especially during turning.

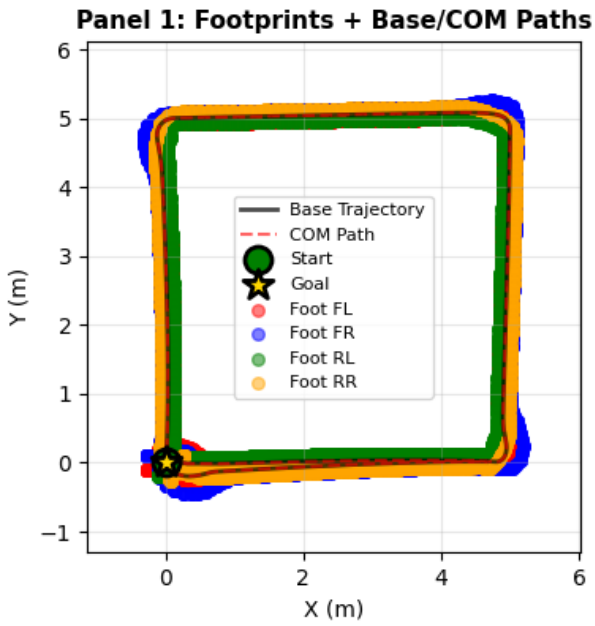
Residual magnitude Because the policy is intended to be a *residual*, the L2 norm of the action is tracked:

$$\|\mathbf{a}(t)\|_2, \tag{10}$$

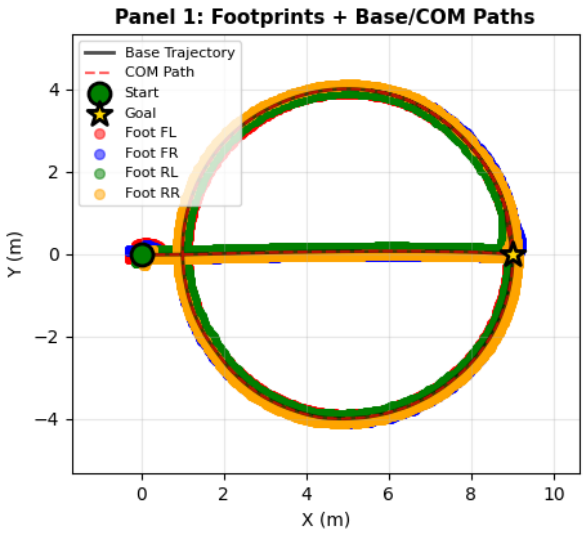
where $\mathbf{a}(t) \in \mathbb{R}^{12}$ is the per-leg residual output. A desirable outcome is stable locomotion with small $\|\mathbf{a}(t)\|_2$.

4.3 Qualitative Results on Flat Terrain

Stable walking with small residuals On flat terrain with a fixed trotting gait, the policy learned to walk stably while keeping residual corrections small. This matches the intended behavior of the hybrid design: the Raibert prior provides the dominant structure and the policy learns refinement corrections rather than rewriting the gait.



(a) Learned residual policy: square trajectory



(b) Learned residual policy: circle trajectory

Figure 2: Qualitative flat-terrain results for the residual footstep policy. The policy produces stable walking behavior while applying small residual corrections on top of the Raibert prior

Trajectory tracking demonstrations The demonstrations include square and circle trajectory executions using the learned policy. These results confirm that the residual planner can operate in a closed loop and produce consistent foothold targets for nontrivial motion intents.

End-to-end pipeline validation A key milestone at this stage is that the planner–bridge–controller–simulation loop can be exercised end-to-end (Figure 1) on flat terrain. In particular, the interface of "final foot targets (x, y, z) per leg" is compatible with the Quad-SDK side of the stack, which is necessary for later rough-terrain and hardware deployment work.

4.4 Baseline Context (Quad-SDK Default Planner)

For context, demonstrations of the existing Quad-SDK planning/control stack in action (including physical robot footage for a circular trajectory). This provides a practical baseline for future evaluation: the goal is not to outperform the controller, but to improve footstep robustness and reduce failure cases on uneven terrain by adding a residual learning layer upstream of the controller.

4.5 Summary of Current Results

Overall, the flat-terrain experiments validate the core hypothesis at a basic level: a residual RL policy can be trained stably to refine Raibert footholds, and the resulting foot targets can be integrated

into a Quad-SDK-style control pipeline (Figure 1) without modifying the low-level controller. The next step is to extend the environment to rough terrain (heightfields, steps, slopes) and to report quantitative metrics such as success rate, stability margin statistics, and failure counts under disturbances.

5 Discussion

This project demonstrates that a residual learning formulation can be integrated cleanly into a classical quadruped locomotion pipeline. On flat terrain, the Raibert prior provides a stable stepping structure while the PPO policy learns small corrections that improve tracking behavior without producing large, unstable foothold shifts. From a software perspective, a major practical outcome is that the learned planner can be inserted upstream of an existing Quad-SDK-style controller (Figure 1) by publishing external foot targets, rather than rewriting the low-level control stack.

At the same time, the current system has clear limitations. The most important open issue is the MuJoCo execution layer. While MuJoCo is used for physics simulation and visualization, the inverse-dynamics control pipeline was originally validated in Gazebo and/or hardware-oriented backends, and the current MuJoCo coupling still requires significant work to match the controller's assumptions. In particular, differences in contact modeling, timing, joint conventions, and actuator interpretation can lead to tracking inconsistencies. As a result, the present results should be interpreted primarily as validation of the learning and planning logic on flat terrain, not as a fully verified control-and-dynamics benchmark.

Another limitation is that reward shaping and weight tuning were performed in a largely manual and exploratory manner. While this is common during early-stage RL development, it makes it harder to claim optimality or generality. The policy was also trained mainly in steady walking conditions, so initialization transients are handled through gating and warmup logic in the ROS policy node rather than being fully learned.

Finally, evaluation is still mostly qualitative. Although the system logs velocity error, residual magnitude, stability margin, and support polygon metrics, these have not yet been reported as a complete quantitative benchmark with success rates across many randomized trials, nor compared formally against strong baselines such as the default Quad-SDK footstep planner on the same terrain distribution.

6 Conclusion and Future Work

This work presented a hybrid quadruped locomotion framework that augments a Raibert-style footstep planner with a PPO-trained residual policy. The policy outputs small per-leg Cartesian

residuals that refine nominal footholds, while safety is enforced analytically through workspace limits and stability guards. The architecture preserves interpretability and leverages an existing Quad-SDK-style inverse dynamics controller, allowing the learned component to remain modular and easy to integrate.

On flat terrain, the current implementation validates the end-to-end planning loop: the residual policy learns stable behavior that refines but does not overwrite the model-based prior, and the ROS integration demonstrates that learned footholds can be injected into the existing control stack shown in Figure 1. However, full MuJoCo-based execution of the inverse dynamics pipeline is still pending and remains the main engineering task required before reporting stronger quantitative results.

If more time were available, the next steps would be:

- **Fix and validate the MuJoCo–Quad-SDK coupling:** ensure consistent joint conventions, actuator interfaces, timestep synchronization, and contact behavior so that inverse dynamics and swing tracking behave as intended
- **Rough terrain training** enable heightmap observations, train on slopes/heightfields, and report quantitative metrics including success rate, stability margin statistics, and failure modes
- **Stronger baseline comparisons:** compare against Raibert-only footholds and the default Quad-SDK planner under identical conditions and disturbances
- **Hardware deployment:** once simulation alignment is stable, deploy the residual planner using the same interface on a physical quadruped to test robustness under real-world modeling errors

Overall, this project establishes a practical foundation for studying how residual learning can improve foothold planning while preserving the stability guarantees and engineering reliability of a model-based quadruped control stack.

References

Sapan Gangapurwala, Matthias Geisert, Alexander Mitchell, and Ioannis Havoutis. Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control. *IEEE Transactions on Robotics*, 2022. doi: 10.1109/TRO.2022.3172469.

Carlos Mastalli, Ioannis Havoutis, Michele Focchi, Darwin G. Caldwell, and Claudio Semini. Motion planning for quadrupedal locomotion: Coupled planning, terrain mapping, and whole-body control. *IEEE Transactions on Robotics*, 36(6):1635–1648, 2020. doi: 10.1109/TRO.2020.3003464.

Joseph Norby, Yanhao Yang, Ardalan Tajbakhsh, Jiming Ren, Justin K. Yim, Alexandra Stutt, Qishun Yu, Nikolai Flowers, and Aaron M. Johnson. Quad-sdk: Full stack software framework for agile quadrupedal locomotion. In *ICRA Workshop on Legged Robots*, 2022. URL https://www.andrew.cmu.edu/user/amj1/papers/Quad_SDK_ICRA_Abstract.pdf.

Shafeef Omar, Lorenzo Amatucci, Victor Barasuol, Giulio Turrisi, and Claudio Semini. Safesteps: Learning safer footstep planning policies for legged robots via model-based priors, 2023. URL https://arxiv.org/abs/2307.12664.

Marc H. Raibert. *Dynamically Stable Legged Locomotion*. PhD thesis, Carnegie Mellon University, 1983. URL https://www.ri.cmu.edu/pub_files/pub3/raibert_marc_h_1983_1/raibert_marc_h_1983_1.pdf.

Robomechanics Lab. robomechanics/quad-sdk (github repository). https://github.com/robomechanics/quad-sdk, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Vassilios Tsounis, Mitja Alge, Joonho Lee, Farbod Farshidian, and Marco Hutter. Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):3699–3706, 2020. doi: 10.1109/LRA.2020.2979660.

Guiyang Xin, Songyan Xin, Oguzhan Cebe, Mathew Jose Pollayil, Franco Angelini, Manolo Garabini, Sethu Vijayakumar, and Michael Mistry. Robust footstep planning and lqr control for dynamic quadrupedal locomotion. *IEEE Robotics and Automation Letters*, 6(3):4488–4495, 2021. doi: 10.1109/LRA.2021.3068695.

Appendix

A Additional Figures or Code

Supplementary Material: Video demonstrations are available here
Supplementary Material: Github repo is available here.