A Details of Low-level Controller

A.1 Notation

We represent the base pose of the robot in the world frame as $\boldsymbol{q} = [\boldsymbol{p}, \boldsymbol{\Theta}] \in \mathbb{R}^6$. $\boldsymbol{p} \in \mathbb{R}^3$ is the Cartesian coordinate of the base position. $\boldsymbol{\Theta} = [\phi, \theta, \psi]$ is the robot's base orientation represented as Z-Y-X Euler angles, where ψ is the yaw, θ is the pitch and ϕ is the roll. We represent the base velocity of the robot as $\dot{\boldsymbol{q}} = [\boldsymbol{v}, \boldsymbol{\omega}]$, where $\boldsymbol{v}$ and $\boldsymbol{\omega}$ are the linear and angular velocity of the base. We define the control input as $\boldsymbol{f} = [\boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{f}_3, \boldsymbol{f}_4] \in \mathbb{R}^{12}$, where $\boldsymbol{f}_i$ denotes the ground reaction force generated by leg i. $\boldsymbol{r}_{\text{foot}} = (\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3, \boldsymbol{r}_4) \in \mathbb{R}^{12}$ represents the four foot positions relative to the robot base. $\mathbf{I}_n$ denotes the $n \times n$ identity matrix. $[\cdot]_\times$ converts a 3d vector into a skew-symmetric matrix, so that for $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$, $\boldsymbol{a} \times \boldsymbol{b} = [\boldsymbol{a}]_\times \boldsymbol{b}$.

A.2 Details of the Stance Leg Controller

CoM PD Controller Given the desired CoM velocity in the sagittal plane $\left[v_x^{\text{ref}}, v_z^{\text{ref}}, \omega_y^{\text{ref}}\right]$, we first find the reference pose $\boldsymbol{q}^{\text{ref}}$ and velocity $\dot{\boldsymbol{q}}^{\text{ref}}$ of the robot base. We set $\boldsymbol{q}^{\text{ref}} = [p_x, p_y, p_z, 0, \theta, \psi]$ to be the current pose of the robot with the roll angle set to 0, and $\dot{\boldsymbol{q}}^{\text{ref}} = \left[v_x^{\text{ref}}, 0, v_z^{\text{ref}}, 0, \omega_y^{\text{ref}}, 0\right]$ to follow the policy command in the sagittal plane and keep the remaining dimensions to 0. We then find the CoM acceleration using a PD controller:

$$\ddot{\boldsymbol{q}}^{\text{ref}} = \boldsymbol{k_p}(\boldsymbol{q}^{\text{ref}} - \boldsymbol{q}) + \boldsymbol{k_d}(\dot{\boldsymbol{q}}^{\text{ref}} - \dot{\boldsymbol{q}}) \tag{9}$$

where we set $\boldsymbol{k_p} = [0, 0, 0, 50, 0, 0]$ to only track the reference roll angle, and $\boldsymbol{k_d} = [10, 10, 10, 10, 10, 10]$ to track reference velocity in all dimensions.

Centroidal Dynamics Model Our centroidal dynamics model is based on [8] with a few modifications. We assume massless legs, and simplify the robot base to a rigid body with mass m and inertia $\mathbf{I}_{\text{base}}$ (in the body frame). The rigid body dynamics in local coordinates are given by:

$$\mathbf{I}_{\text{base}}\dot{\boldsymbol{\omega}} = \sum_{i=1}^{4} \boldsymbol{r}_i \times \boldsymbol{f}_i \tag{10}$$

$$m\ddot{\boldsymbol{p}} = \sum_{i=1}^{4} \boldsymbol{f}_i + \boldsymbol{g} \tag{11}$$

where $\boldsymbol{g}$ is the gravity vector transformed to the base frame.

With the above simplifications, we get the linear, time-varying dynamics model:

$$\underbrace{\begin{bmatrix} \dot{\boldsymbol{\omega}} \\ \ddot{\boldsymbol{p}} \end{bmatrix}}_{\ddot{\boldsymbol{q}}} = \underbrace{\begin{bmatrix} \mathbf{I}_{\text{base}}^{-1}[\boldsymbol{r}_1]_\times & \mathbf{I}_{\text{base}}^{-1}[\boldsymbol{r}_2]_\times & \mathbf{I}_{\text{base}}^{-1}[\boldsymbol{r}_3]_\times & \mathbf{I}_{\text{base}}^{-1}[\boldsymbol{r}_4]_\times \\ \mathbf{I}_3/m & \mathbf{I}_3/m & \mathbf{I}_3/m & \mathbf{I}_3/m \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \\ \boldsymbol{f}_3 \\ \boldsymbol{f}_4 \end{bmatrix}}_{\boldsymbol{f}} + \underbrace{\begin{bmatrix} \mathbf{0} \\ \boldsymbol{g} \end{bmatrix}}_{\boldsymbol{g}} \tag{12}$$

as seen in Eq. (3).

A.3 Reference Trajectory for Swing Legs

For swing legs, we design the reference trajectory to always keep the feet tangential to the ground, and use residuals from the centroidal policy to generate vertical movements. To find the reference trajectory, we interpolate between three key frames ($\boldsymbol{p}_{\text{lift-off}}, \boldsymbol{p}_{\text{air}}, \boldsymbol{p}_{\text{land}}$) based on the gait timing. The lift-off position $\boldsymbol{p}_{\text{lift-off}}$ is the foot location at the beginning of the swing phase. The mid-air position $\boldsymbol{p}_{\text{air}}$ is the position of the robot's hip projected onto the ground plane. We use the Raibert Heuristic [40] to estimate the desired foot landing position:

$$\boldsymbol{p}_{\text{land}} = \boldsymbol{p}_{\text{ref}} + \boldsymbol{v}_{\text{CoM}} T_{\text{stance}}/2 \tag{13}$$

where $\boldsymbol{v}_{\text{CoM}}$ is the projected robot's CoM velocity onto the $x-y$ plane, and T_{stance} is the expected duration of the next stance phase, which is estimated using the stepping frequency from the centroidal policy. Raibert's heuristic ensures that the stance leg will have equal forward and backward movement in the next stance phase, and is commonly used in locomotion controllers [**?** 8].

Given these three key points, $\boldsymbol{p}_{\text{lift-off}}, \boldsymbol{p}_{\text{air}},$ and $\boldsymbol{p}_{\text{land}}$, we fit a quadratic polynomial, and computes the foot's desired position in the curve based on its progress in the current swing phase. Given the desired foot position, we then compute the desired motor position using inverse kinematics, and track it using a PD controller. We re-compute the desired foot position of the feet at every step (500Hz) based on the latest velocity estimation.

B Experiment Details

B.1 Reward Function

Our reward function consists of 9 terms. We provide the detail about each term and its corresponding weight below:

1. **Upright (0.02)** is the projection of a unit vector in the z-axis of the robot frame onto the z-axis of the world frame, and rewards the robot for keeping an upright pose.
2. **Base Height (0.01)** is the height of the robot's CoM in meters, and rewards the robot for jumping higher.
3. **Contact Consistency (0.008)** is the sum of 4 indicator variables: $\sum_{i=1}^{4} \mathbb{1}(c_i = \hat{c}_i)$, where c_i is the actual contact state of leg i, and $\hat{c}_i$ is the desired contact state of leg i specified by the gait generator. It rewards the robot for following the desired contact schedule.
4. **Foot Slipping (0.032)** is the sum of the world-frame velocity for contact-legs: $\sum_{i=1}^{4} \hat{c}_i \sqrt{v_{i,x}^2 + v_{i,y}^2}$, where $\hat{c}_i \in \{0, 1\}$ is the desired contact state of leg i, and $v_{i,x}, v_{i,y}$ is the *world-frame* velocity of leg i. This term rewards the robot for keeping contact legs static on the ground.
5. **Foot Clearance (0.008)** is the sum of foot height (clipped at 2cm) for non-contact legs. This term rewards the robot to keep non-contact legs high on the ground.
6. **Knee Contact (0.064)** is the sum of knee contact variables $\sum_{i=1}^{4} kc_i$, where $kc_i \in \{0, 1\}$ is the indicator variable for knee contact of the ith leg.
7. **Stepping Frequency (0.008)** is a constant plus the negated frequency $1.5 - \text{clip}(f, 1.5, 4)$, which encourages the robot to jump at large steps using a low stepping frequency.
8. **Distance to goal (0.016)** is the Cartesian distance from the robot's current location to the desired landing position, and encouarges the robot to jump close to the goal.
9. **Out-of-bound-action (0.01)** is the normalized amount of excess when the policy computes an action that is outside the action space. We design this term so that PPO would not excessively explore out-of-bound actions.

B.2 PPO details

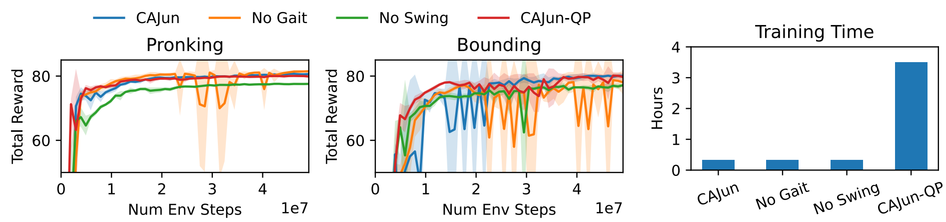
As listed in Table. 3, we use the same set of hyperparameters for all PPO training, including the CAJun policies and baseline policies.

B.3 Setup for End-to-end RL Baseline

We use a similar MDP setup as CAJun (section. 5) for the end-to-end RL baseline. More specifically, we use the same gait generator as CAJun to generate reference foot contacts, and include stepping frequency as part of the action space so that the policy can modify the gait schedule. However, unlike CAJun, this reference gait is only used for reward computation, and does not directly affect leg

Parameter	Value
Learning rate	0.001, adaptive
# env steps per update	98,304
Batch size	24,576
# epochs per update	5
Discount factor	0.99
GAE λ	0.95
Clip range	0.2

Table 3: Hyperparameters used for PPO.



(a) Reward Curves

(b) Training Time

Figure 7: Reward curve and training time of CAJun compared to the ablated methods.

controllers. For reward, we keep the same reward terms and weights (Appendix. B.1). However, since the initial exploration phase of end-to-end RL can lead to a lot of robot failures with negative rewards, we add an additional alive bonus of 0.02 to ensure that the reward stays positive.

B.4 Additional Result for Ablation Study

For each baseline, we report its total reward over 6 consecutive jumps with a desired distance of 1m per jump (Fig. 7a). We train each baseline using 5 random seeds and report the average and standard deviations. We also report the wall-clock training time in Fig. 7b.