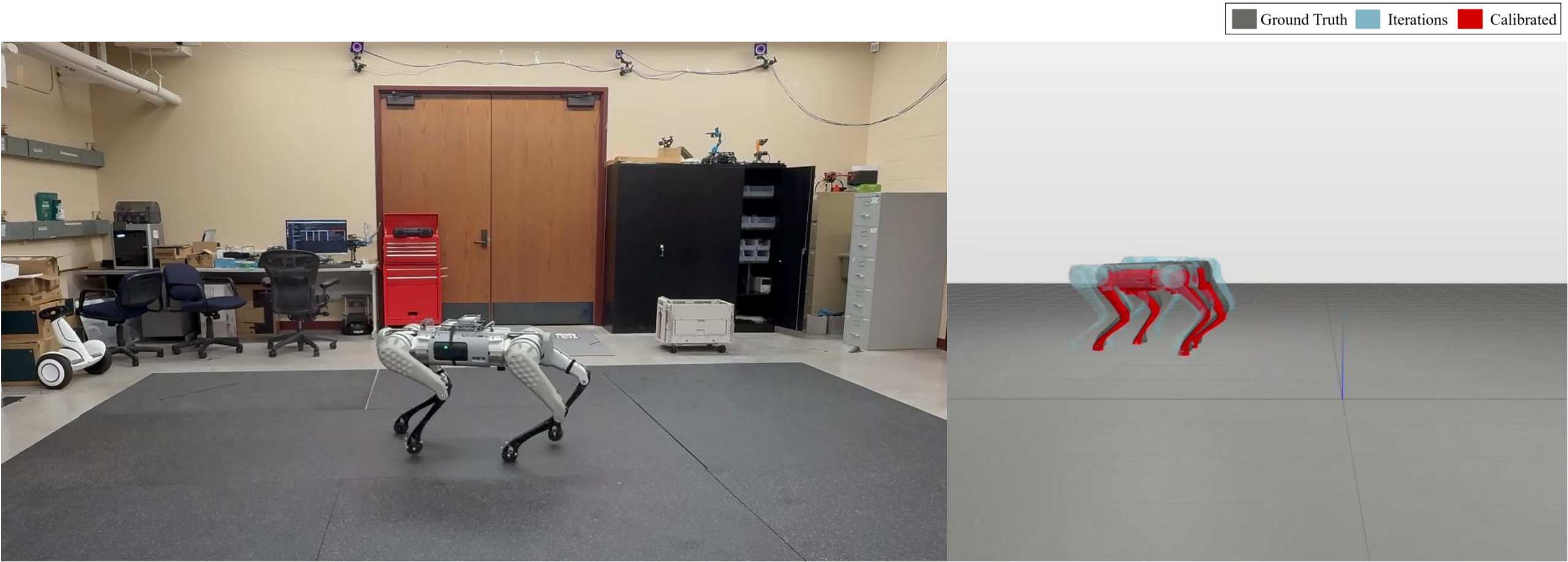
Legged AI Lab

ICRA IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION 2026 VIENNA

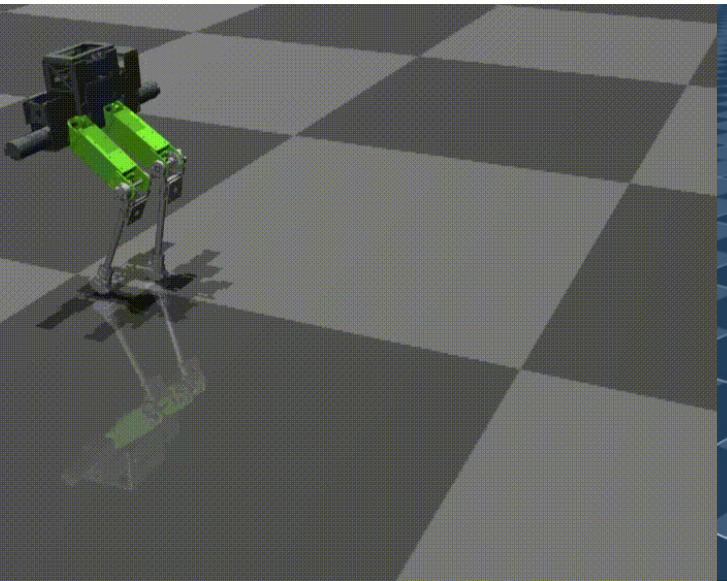
Simultaneous Calibration of Noise Covariance and Kinematics for State Estimation of Legged Robots via Bi-level Optimization

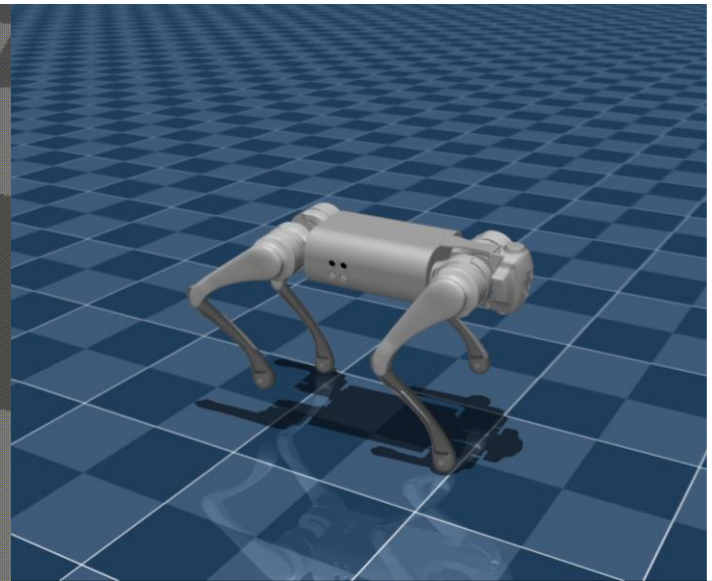
Denglin Cheng*, Jiarong Kang*, Xiaobin Xiong

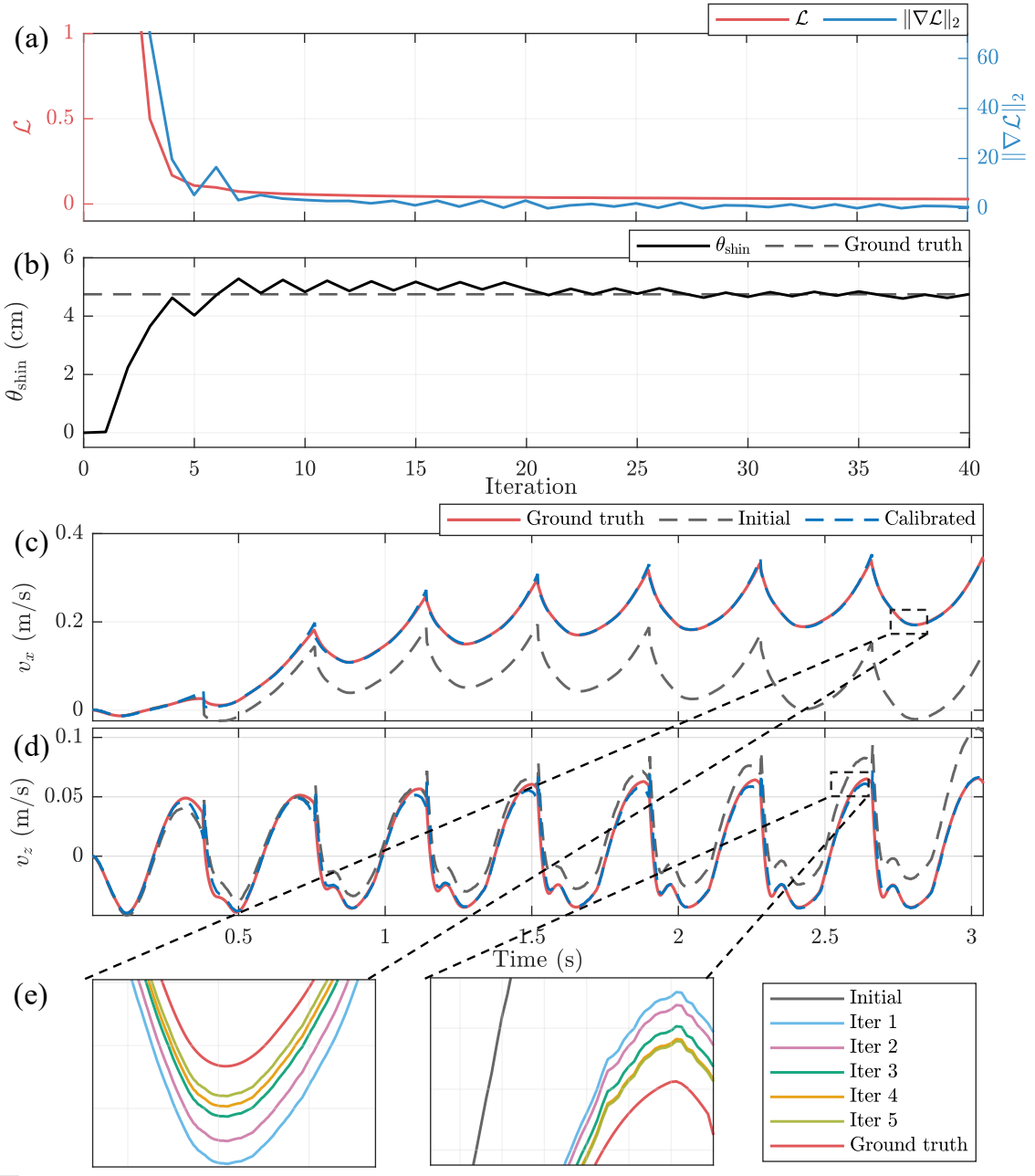


Motivation: make legged state estimation self-calibrating

- Legged autonomy relies heavily on accurate state estimation (esp. base **velocity & orientation**).
- In practice, **Q/R are unknown → manual tuning is** costly and unreliable; can be **sub-optimal / inconsistent**.
- **Key idea:** differentiate through an estimator and use bi-level optimization, **jointly calibrate noise covariances + kinematics** via bi-level optimization.

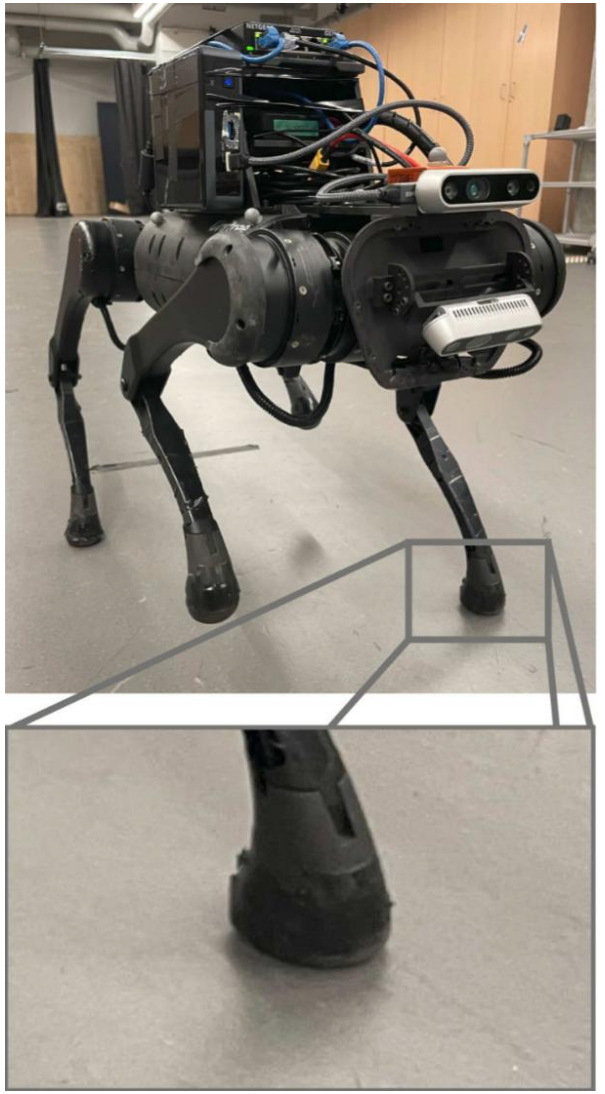






Related Work

- Many legged estimators still rely on **fixed / hand-tuned covariances**, limiting accuracy especially under kinematic uncertainty.
- Noise-statistics identification exists (e.g., EM / learning), but often **sensor-specific / restrictive assumptions / careful init**, and not tailored for complex legged platforms.
- SysID often treats **noise, model parameters, estimator design separately**; lacks a unified calibration loop.



S. Yang, H. Choset, and Z. Manchester, “Online kinematic calibration for legged robots,” IEEE Robotics and Automation Letters, vol. 7, no. 3, pp. 8178–8185, 2022.

Contribution

- **hand-tuned covariances.**
- **sensor-specific / restrictive assumptions / careful initial guess.**
- **separate noise / model parameters / estimator SysID.**

$\Rightarrow$

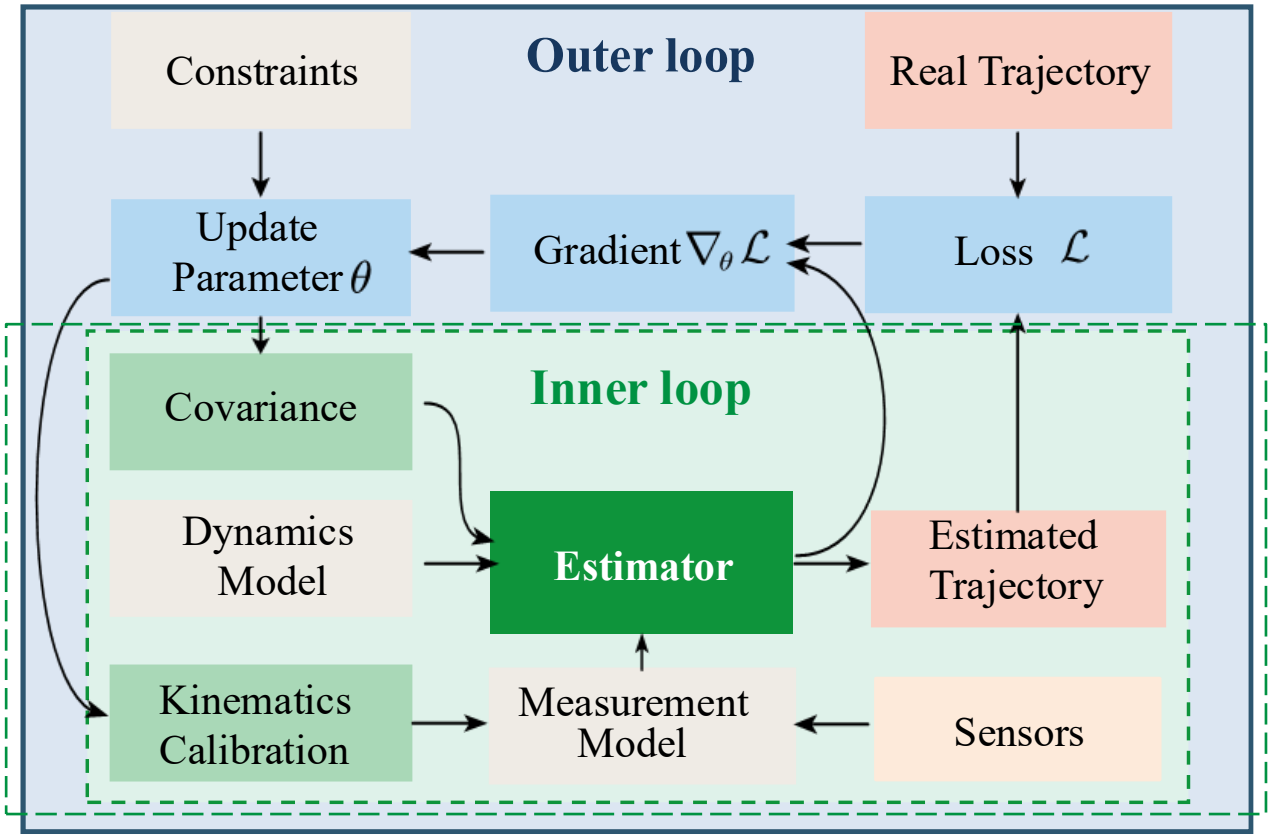
- **Bi-level, estimator-in-the-loop calibration**
- **Joint calibration of covariance + kinematics**
- **Validated on multiple platforms**

Problem: learn Q/R + kinematics

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^m} \quad \mathcal{L}(\hat{\mathbf{x}}(\boldsymbol{\theta}), \mathbf{x}_{\mathrm{GT}}, \boldsymbol{\theta}),$$

$$\text{s.t.} \quad \boldsymbol{\theta} \in \mathcal{C},$$

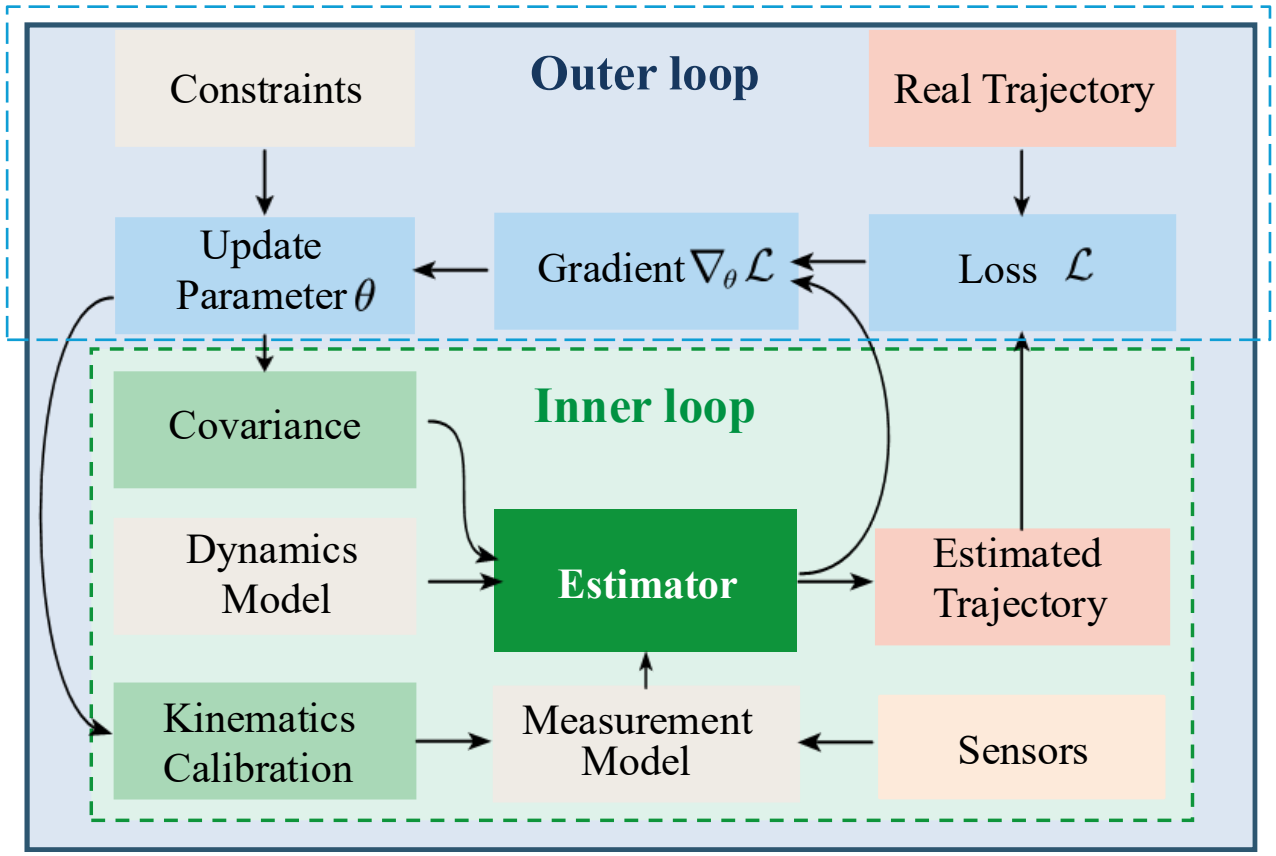
$$\hat{\mathbf{x}}(\boldsymbol{\theta}) = \underset{\mathbf{x} \in \mathbb{R}^{n \times (T+1)}}{\arg\min} \ \mathcal{J}(\mathbf{x}, \boldsymbol{\theta}).$$



- Input: proprioceptive sensors (**IMU / encoders / contact**) + a segment of mocap **ground truth** (pose & velocity).
- Goal: calibrate **noise covariances (Q,R)** and **uncertain kinematics** so estimates match ground truth.
- Target: a tuning-free method that improves accuracy/consistency across tasks and estimators (EKF/MHE).

Problem: learn Q/R + kinematics

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^m} \quad \mathcal{L}(\hat{\mathbf{x}}(\boldsymbol{\theta}), \mathbf{x}_{\mathrm{GT}}, \boldsymbol{\theta}),$$
$$\text{s.t.} \quad \boldsymbol{\theta} \in \mathcal{C},$$
$$\hat{\mathbf{x}}(\boldsymbol{\theta}) = \underset{\mathbf{x} \in \mathbb{R}^{n \times (T+1)}}{\arg\min} \ \mathcal{J}(\mathbf{x}, \boldsymbol{\theta}).$$



- Input: proprioceptive sensors (**IMU / encoders / contact**) + a segment of mocap **ground truth** (pose & velocity).
- Goal: calibrate **noise covariances (Q,R)** and **uncertain kinematics** so estimates match ground truth.
- Target: a tuning-free method that improves accuracy/consistency across tasks and estimators (EKF/MHE).

Parametrization and Constraint Set

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^m} \quad \mathcal{L}(\hat{\mathbf{x}}(\boldsymbol{\theta}), \mathbf{x}_{\mathrm{GT}}, \boldsymbol{\theta}),$$

$$\text{s.t.} \quad \boldsymbol{\theta} \in \mathcal{C},$$

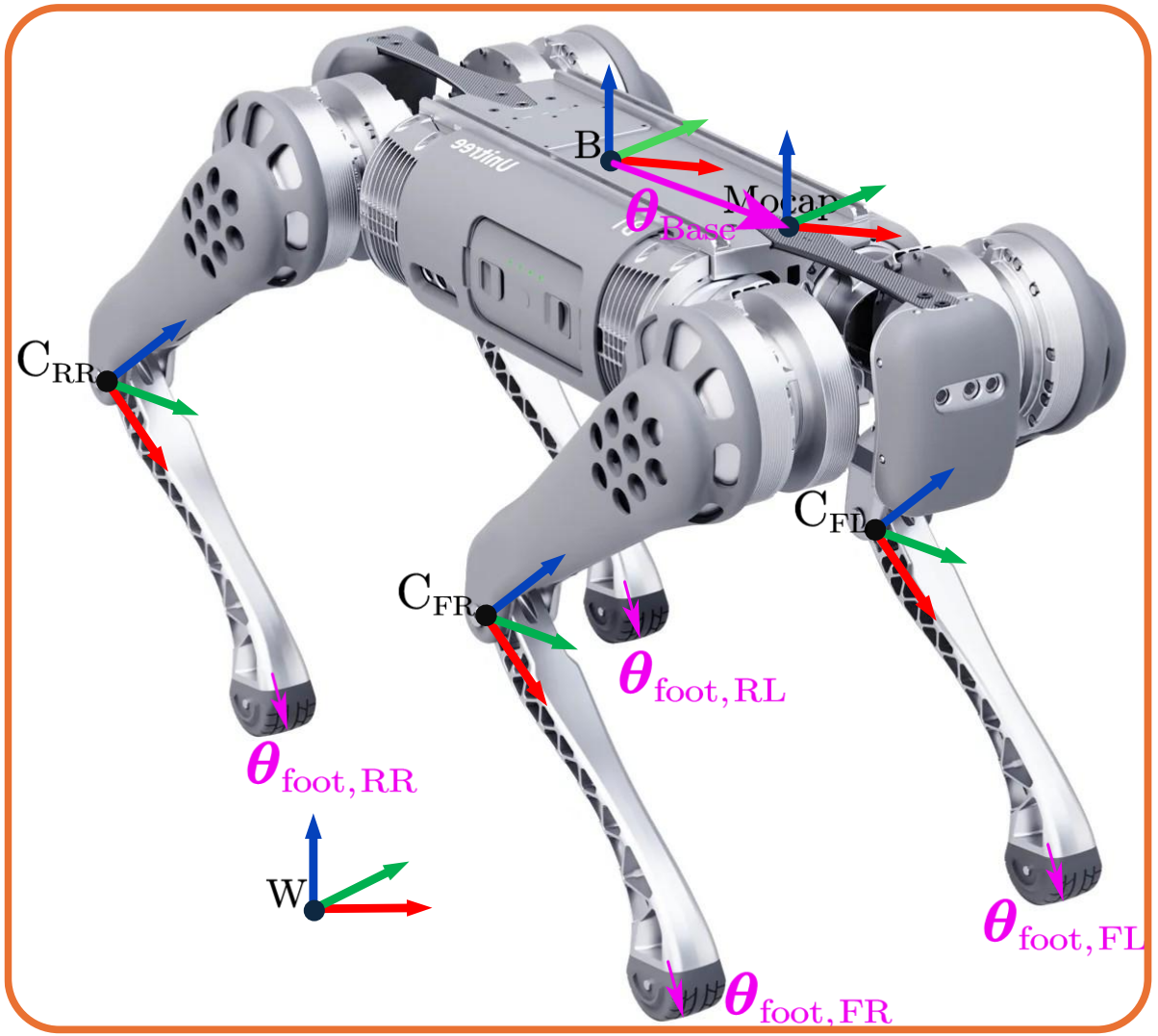
$$\hat{\mathbf{x}}(\boldsymbol{\theta}) = \underset{\mathbf{x}\in\mathbb{R}^{n\times(T+1)}}{\arg\min}\ \mathcal{J}(\mathbf{x}, \boldsymbol{\theta}).$$

$$\boldsymbol{\theta} = (\boldsymbol{\theta}_{\mathrm{cov}}, \boldsymbol{\theta}_{\mathrm{foot}}, \boldsymbol{\theta}_{\mathrm{base}})$$

$$\boldsymbol{\theta}_{\mathrm{cov}} = \mathrm{vec}(\Sigma_i) \in \mathbb{R}^{d_i(d_i+1)/2},\ \Sigma_i \in \mathbb{S}^{d_i}_{++},\ \forall i$$

- **Feasible set:** Need covariance matrices to be **positive definite (SPD)** + relaxed **box constraint**
- ⇒ prevents degenerate covariances / unphysical solutions

➢ Kinematics uncertainty:



$$p^{\mathrm{W}}_{\mathrm{real},j} = p^{\mathrm{W}}_{\mathrm{foot},j} + \mathbf{R}_{\mathrm{WC_j}}\,\boldsymbol{\theta}_{\mathrm{foot},j}$$

Lower Level: Estimation (Single Leg)

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^m} \quad \mathcal{L}(\hat{\mathbf{x}}(\boldsymbol{\theta}), \mathbf{x}_{\mathrm{GT}}, \boldsymbol{\theta}),$$
$$\text{s.t.} \quad \boldsymbol{\theta}\in\mathcal{C},$$
$$\hat{\mathbf{x}}(\boldsymbol{\theta}) = \arg\min_{\mathbf{x}\in\mathbb{R}^{n\times(T+1)}} \mathcal{J}(\mathbf{x}, \boldsymbol{\theta}).$$

$$\mathbf{x}_k = [\, p^\top \quad v^\top \quad q^\top \quad p_{\mathrm{foot},k}^\top \quad b_a^\top \quad b_\omega^\top \,]^\top$$

- **Process Model**

$$p^+ = p + v\,\Delta t + \tfrac{1}{2}(\mathbf{R}_{\mathrm{WB}}(\tilde{a} - b_a) + g)\Delta t^2 \; + \; \delta_p,$$
$$v^+ = v + (\mathbf{R}_{\mathrm{WB}}(\tilde{a} - b_a) + \mathbf{g})\Delta t \; + \; \delta_v,$$
$$q^+ = \zeta((\tilde{\omega} - b_\omega)\,\Delta t) \otimes q \; + \; \delta_q,$$
$$p_{\mathrm{foot}}^+ = p_{\mathrm{foot}} + \delta_{\mathrm{foot}}, \quad b_a^+ = b_a + \delta_{b_a}, \quad b_\omega^+ = b_\omega + \delta_{b_\omega},$$

- **Measurement Model**

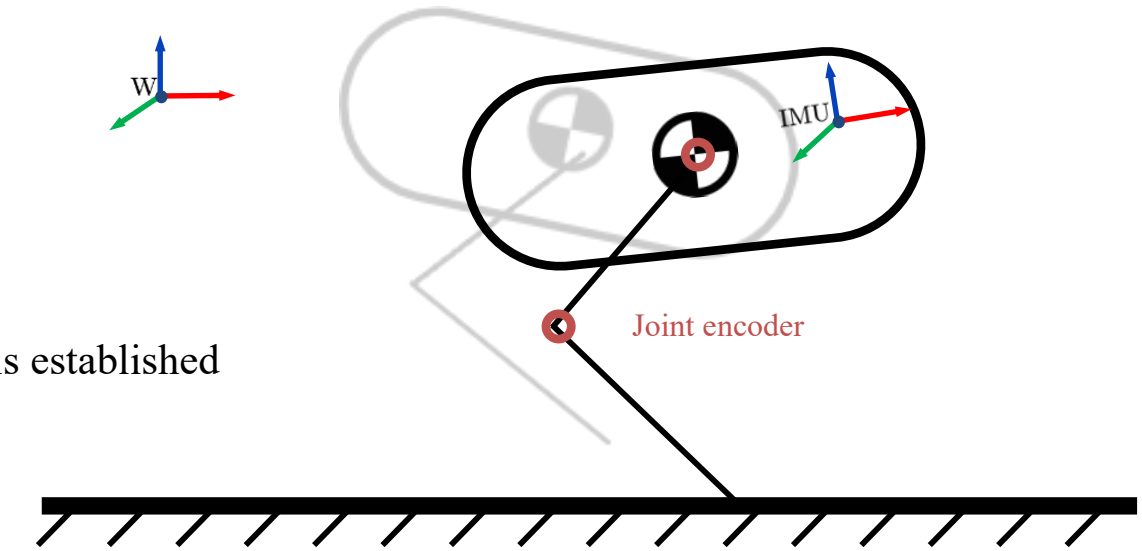
$$y_p = p - p_{\mathrm{foot}} + \delta_{y_p},$$
$$\tilde{y}_p = -\mathbf{R}_{\mathrm{WB}}\boldsymbol{g}(\tilde{\alpha}),$$
$$y_v = v + \delta_{y_v},$$
$$\tilde{y}_v = -\mathbf{R}_{\mathrm{WB}}J(\tilde{\alpha})\dot{\tilde{\alpha}} - \mathbf{R}_{\mathrm{WB}}[\tilde{\omega} - b_\omega]_\times \boldsymbol{g}(\tilde{\alpha}),$$

Once contact is established

- Moving Horizon Estimator

$$\min_{\{\mathbf{x},\mathbf{w},\mathbf{n}\}_{[T-N,T]}} \quad \Gamma(\mathbf{x}_{T-N}) + \sum_{k=T-N}^{T-1} \mathbf{w}_k^\top Q_k^{-1}\mathbf{w}_k \; + \sum_{k=T-N}^{T} \mathbf{n}_k^\top R_k^{-1}\mathbf{n}_k,$$
$$\text{s.t.} \quad \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k, \;\; \forall k = T-N, \ldots, T-1,$$
$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{n}_k, \;\; \forall k = T-N, \ldots, T,$$
$$g(\mathbf{x}_k) \le 0, \;\; \forall k = T-N, \ldots, T.$$
$$\Gamma(\mathbf{x}_{T-N}) = (\mathbf{x}_{T-N} - \mathbf{x}_{\mathrm{prior}})^\top P_{T-N}^{-1}(\mathbf{x}_{T-N} - \mathbf{x}_{\mathrm{prior}})$$

- All sensors are proprioceptive :



Lower Level: SRB MAP Estimation (Single Leg)

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^m} \quad \mathcal{L}(\hat{\mathbf{x}}(\boldsymbol{\theta}), \mathbf{x}_{\mathrm{GT}}, \boldsymbol{\theta}),$$
$$\text{s.t.} \quad \boldsymbol{\theta}\in\mathcal{C},$$
$$\hat{\mathbf{x}}(\boldsymbol{\theta}) = \underset{\mathbf{x}\in\mathbb{R}^{n\times(T+1)}}{\arg\min}\ \mathcal{J}(\mathbf{x}, \boldsymbol{\theta}).$$

$$\mathbf{x}_k = [\, p^\top \quad v^\top \quad q^\top \quad p_{\mathrm{foot},k}^\top \quad b_a^\top \quad b_\omega^\top \,]^\top$$

- **Process Model**

$$p^+ = p + v\,\Delta t + \tfrac{1}{2}(\mathbf{R}_{\mathrm{WB}}(\tilde{a} - b_a) + g)\Delta t^2 \ + \ \delta_p,$$
$$v^+ = v + (\mathbf{R}_{\mathrm{WB}}(\tilde{a} - b_a) + \mathbf{g})\Delta t \ + \ \delta_v,$$
$$q^+ = \zeta((\tilde{\omega} - b_\omega)\,\Delta t) \otimes q \ + \ \delta_q,$$
$$p_{\mathrm{foot}}^+ = p_{\mathrm{foot}} + \delta_{\mathrm{foot}}, \quad b_a^+ = b_a + \delta_{b_a}, \quad b_\omega^+ = b_\omega + \delta_{b_\omega},$$

- **Measurement Model**

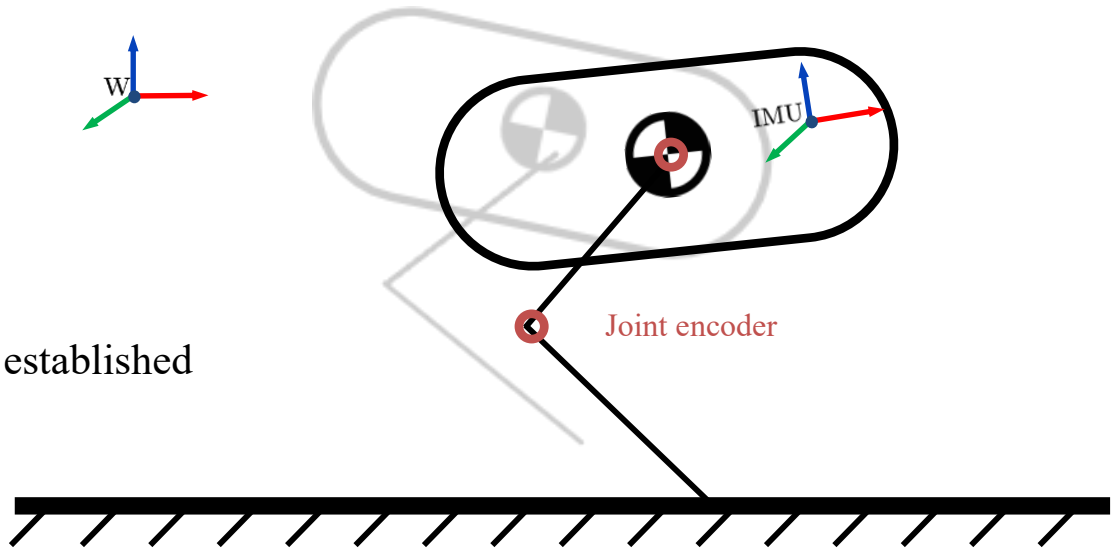
$$y_p = p - p_{\mathrm{foot}} + \delta_{y_p},$$
$$\tilde{y}_p = -\mathbf{R}_{\mathrm{WB}}\boldsymbol{g}(\tilde{\alpha}),$$
$$y_v = v + \delta_{y_v},$$
$$\tilde{y}_v = -\mathbf{R}_{\mathrm{WB}} J(\tilde{\alpha})\dot{\tilde{\alpha}} - \mathbf{R}_{\mathrm{WB}}[\tilde{\omega} - b_\omega]_\times \boldsymbol{g}(\tilde{\alpha}),$$

Once contact is established

- Full information estimator

$$\min_{\{\mathbf{x},\mathbf{w},\mathbf{n}\}_{[0,T]}} \quad \Gamma(\mathbf{x}_0) + \sum_{k=0}^{T-1} \mathbf{w}_k^\top Q_k^{-1} \mathbf{w}_k \ + \ \sum_{k=0}^{T} \mathbf{n}_k^\top R_k^{-1} \mathbf{n}_k,$$
$$\text{s.t.} \quad \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k, \ \forall k = 0, \ldots, T-1,$$
$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{n}_k, \ \forall k = 0, \ldots, T,$$
$$g(\mathbf{x}_k) \le 0, \ \forall k = 0, \ldots, T.$$
$$\Gamma(\mathbf{x}_0) = (\mathbf{x}_0 - \mathbf{x}_{\mathrm{prior}})^\top P_0^{-1} (\mathbf{x}_0 - \mathbf{x}_{\mathrm{prior}})$$

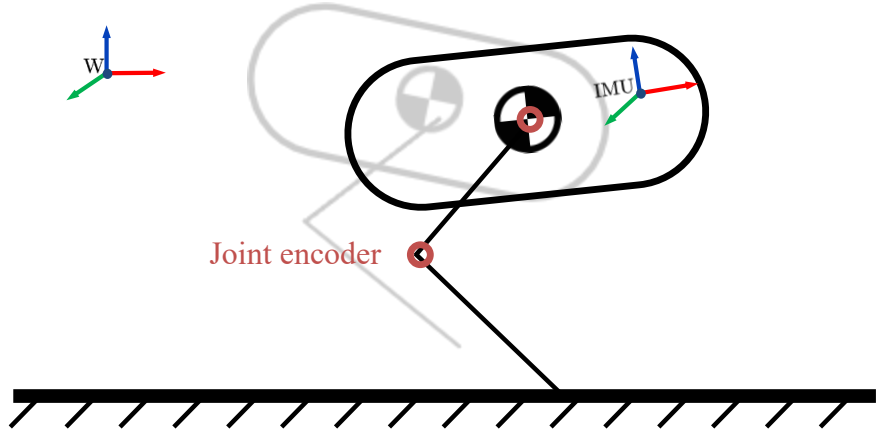
- All sensors are proprioceptive :



Lower Level: Estimation Uncertainty

$$
\begin{aligned}
\min_{\{\mathbf{x},\mathbf{w},\mathbf{n}\}_{[0,T]}} \quad & \Gamma(\mathbf{x}_0) + \sum_{k=0}^{T-1} \mathbf{w}_k^\top Q_k^{-1} \mathbf{w}_k + \sum_{k=0}^{T} \mathbf{n}_k^\top R_k^{-1} \mathbf{n}_k, \\
\text{s.t.} \quad & \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k, \ \forall k = 0, \ldots, T-1, \\
& \mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{n}_k, \ \forall k = 0, \ldots, T, \\
& g(\mathbf{x}_k) \le 0, \ \forall k = 0, \ldots, T. \\
& \Gamma(\mathbf{x}_0) = (\mathbf{x}_0 - \mathbf{x}_{\text{prior}})^\top P_0^{-1} (\mathbf{x}_0 - \mathbf{x}_{\text{prior}})
\end{aligned}
$$

- All proprioceptive sensors:



- **Process Model**

$$\mathbf{w} = [\delta_p^\top, \delta_v^\top, \delta_q^\top, \delta_{\text{foot}}^\top, \delta_{b_a}^\top, \delta_{b_\omega}^\top]^\top \sim \mathcal{N}(0, Q_{\mathbf{w}})$$

$$Q_{\mathbf{w}} = \text{blkdiag}(Q_p, Q_a, Q_\omega, Q_{\text{foot}}, Q_{b_a}, Q_{b_\omega})$$

$$\mathbf{w} = \begin{bmatrix} \hat{R}_{\text{WB}} \delta_p + \frac{1}{2} \hat{R}_{\text{WB}} \Delta t^2 \delta_a \\ \hat{R}_{\text{WB}} \Delta t \delta_a \\ \hat{R}_{\text{WB}} \delta_a \\ \hat{R}_{\text{WB}} \delta_{\text{foot}} \\ \delta_{b_a} \\ \delta_{b_\omega} \end{bmatrix}$$

- **Measurement Model**

$$\delta z = [\delta\alpha^\top \ \delta\dot{\alpha}^\top \ \delta\omega^\top]^\top \sim \mathcal{N}(0, R_z),$$

$$R_z = \text{blkdiag}(R_\alpha, R_{\dot{\alpha}}, Q_\omega) \in \mathbb{S}_{++}^9,$$

$$\delta y_{\text{pf}} = G_p \, \delta z, \ \delta y_{\text{vf}} = G_v \delta z$$

$$\begin{bmatrix} G_p \\ G_v \end{bmatrix} = \begin{bmatrix} \frac{\partial y_{\text{p}}}{\partial z} \\ \frac{\partial y_{\text{v}}}{\partial z} \end{bmatrix} \in \mathbb{R}^{6 \times 9}.$$

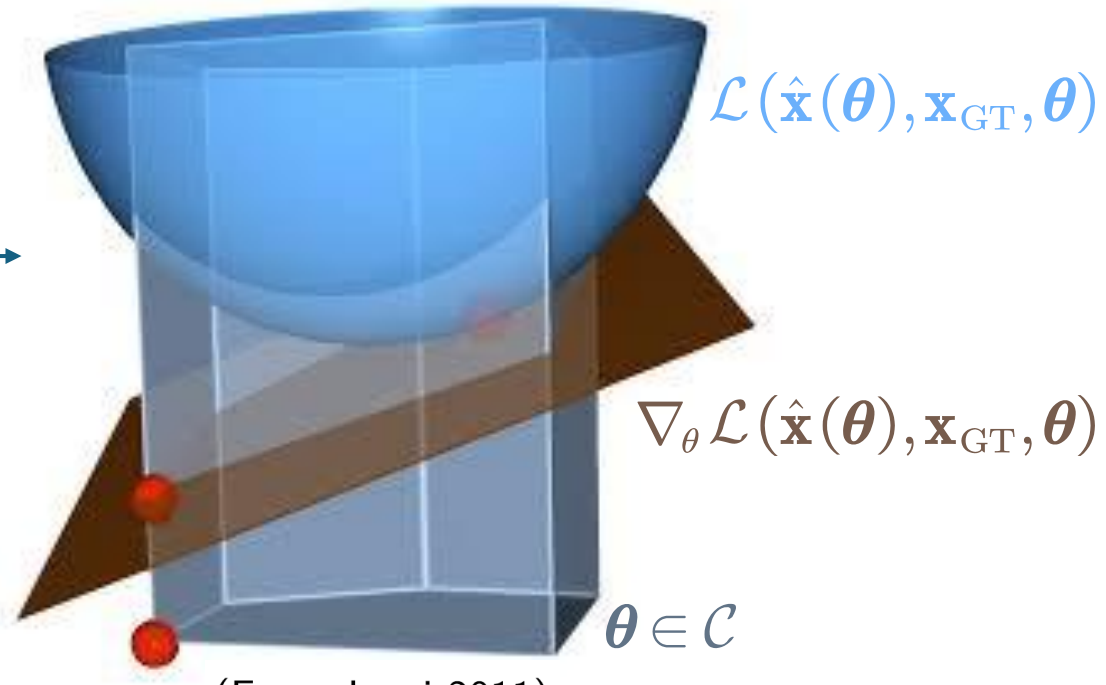
$$R = \begin{bmatrix} R_p \\ R_v \end{bmatrix} = \begin{bmatrix} G_p \, R_z \, G_p^\top \\ G_v \, R_z \, G_v^\top \end{bmatrix}$$

Upper Level: Frank Wolfe Linearization

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^m} \quad \mathcal{L}(\hat{\mathbf{x}}(\boldsymbol{\theta}),\mathbf{x}_{\mathrm{GT}},\boldsymbol{\theta}) = \frac{1}{2}\sum_{k=0}^{T} \| \hat{\mathbf{x}}_k \boxminus \mathbf{x}_{\mathrm{GT},k} \|_2^2,$$

$$\text{s.t.} \quad \boldsymbol{\theta}\in\mathcal{C},$$

$$\hat{\mathbf{x}}(\boldsymbol{\theta}) = \underset{\mathbf{x}\in\mathbb{R}^{n\times(T+1)}}{\arg\min}\ \mathcal{J}(\mathbf{x},\boldsymbol{\theta}).$$



(From Jaggi 2011)

Algorithm 1 Frank–Wolfe Algorithm

1: **Input:** $\mathcal{L}$, $\boldsymbol{\theta}^t$, $\mathcal{C}_t$, t

2: **Gradient compute:** $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^t)$ ▷ introduced later

3: **Direction finding (LMO):**

$$\mathbf{s}^\star = \operatorname{argmin}\ \mathbf{s}^\top \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^t)$$

$$\text{subject to}\quad \mathbf{s}\in\mathcal{C}_t\subset\mathbb{R}^m \quad (l_\infty \text{ball as trust region})$$

4: **Step-size:** $\gamma_t\in(0,1] \leftarrow \textsc{LineSearch}(\mathcal{L},\nabla\mathcal{L},\boldsymbol{\theta}^t,\mathbf{s}^\star)$

5: **Update:** $\boldsymbol{\theta}^{t+1} \leftarrow (1-\gamma_t)\,\boldsymbol{\theta}^t + \gamma_t\,\mathbf{s}^t$

- Every LMO is solved by SDP (or Cholesky decomposition $\boldsymbol{\theta}_{\mathrm{cov}} = \mathrm{vec}(\Sigma_i),\ \Sigma_i\in\mathbb{S}_{++}^{d_i},\ \forall i$)

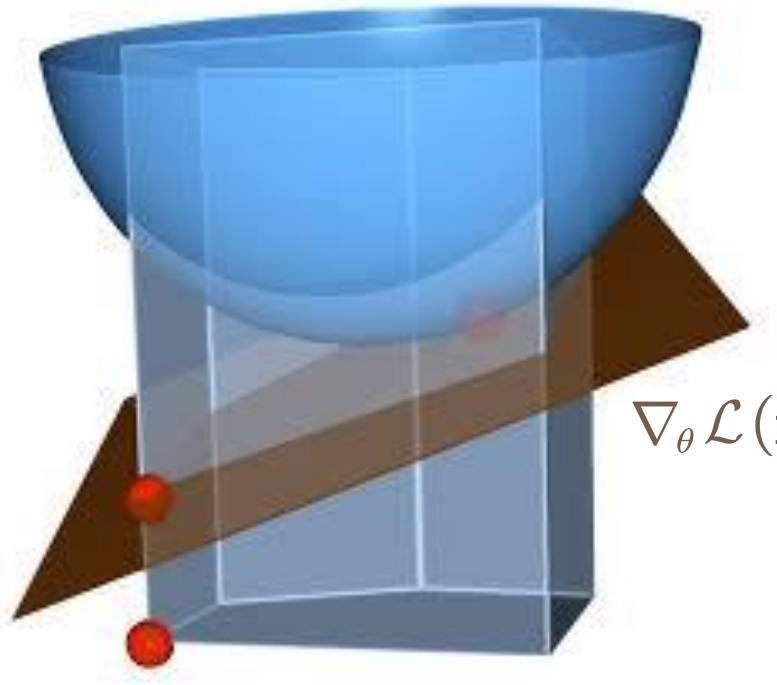
$$\min_{\mathbf{s}} \quad \mathbf{s}^\top \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^t),$$

$$\text{s.t.} \quad \boldsymbol{l}\le\mathbf{s}\le\boldsymbol{u},\ \sum_i s_i F_i \succeq \epsilon I.$$

Upper Level: Objective and Gradient

$$\mathcal{L}(\hat{\mathbf{x}}(\boldsymbol{\theta}), \mathbf{x}_{\mathrm{GT}}, \boldsymbol{\theta}) = \frac{1}{2}\sum_{k=0}^{T} \left\| \hat{\mathbf{x}}_k \boxminus \mathbf{x}_{\mathrm{GT},k} \right\|_2^2,$$

$$= \frac{1}{2}\sum_{k=0}^{T} \left\| \left(\log\left(\mathbf{T}_k\, \mathbf{T}_{\mathrm{GT},k}^{-1}\right)\right)^{\vee} \right\|_2^2 + \left\| \mathbf{e}_k - \mathbf{e}_{\mathrm{GT},k} \right\|$$



$$\nabla_\theta \mathcal{L}(\hat{\mathbf{x}}(\boldsymbol{\theta}), \mathbf{x}_{\mathrm{GT}}, \boldsymbol{\theta})$$

$$\hat{\mathbf{x}}_k = (\mathbf{T}_k, \mathbf{e}_k) \in \mathrm{SE}(3) \times \mathbb{R}^{n_{\mathrm{eucl}}}$$

- Manifold gradient

$$\frac{{}^{e}D\mathcal{L}(\hat{\mathbf{x}})}{D\hat{\mathbf{x}}} = \lim_{\epsilon \to 0} \frac{\mathcal{L}(\hat{\mathbf{x}} \boxplus (\epsilon\tau)) - \mathcal{L}(\hat{\mathbf{x}})}{\epsilon}, \quad \tau \in T_e(\mathcal{G})$$

$$\frac{{}^{e}D\hat{\mathbf{x}}(\boldsymbol{\theta})}{D\boldsymbol{\theta}} = \lim_{\epsilon \to 0} \frac{\hat{\mathbf{x}}(\boldsymbol{\theta} + \epsilon\nu) \boxminus \hat{\mathbf{x}}(\boldsymbol{\theta})}{\epsilon}, \quad \nu \in \mathbb{R}^m$$

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \left(\frac{D\mathcal{L}(\hat{\mathbf{x}}(\boldsymbol{\theta}))}{D\hat{\mathbf{x}}}\right)\left(\frac{D\hat{\mathbf{x}}(\boldsymbol{\theta})}{D\boldsymbol{\theta}}\right) + \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}},$$

- Finite Difference through **CPU parallel computation**

$$\left[\frac{{}^{e}D\mathcal{L}(\mathbf{T})}{D\mathbf{T}}\right]_i = \lim_{\epsilon \to 0} \frac{\mathcal{L}\left(\exp(\epsilon\boldsymbol{\xi}_i) \cdot \mathbf{T}\right) - \mathcal{L}(\mathbf{T})}{\epsilon}, \quad \forall i.$$

$$\left[\frac{{}^{e}D\mathbf{T}}{D\boldsymbol{\theta}}\right]_j = \lim_{\epsilon \to 0} \frac{\left(\log(\mathbf{T} \cdot \tilde{\mathbf{T}}_j^{-1})\right)^{\vee}}{\epsilon}, \quad \forall j.$$

Upper Level: Analytical Gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^m} \quad \mathcal{L}(\hat{\mathbf{x}}(\boldsymbol{\theta}), \mathbf{x}_{\mathrm{GT}}, \boldsymbol{\theta}),$$

$$\text{s.t.} \quad \boldsymbol{\theta} \in \mathcal{C},$$

$$\hat{\mathbf{x}}(\boldsymbol{\theta}) = \underset{\mathbf{x} \in \mathbb{R}^{n \times (T+1)}}{\arg\min}\ \mathcal{J}(\mathbf{x}, \boldsymbol{\theta}).$$

- The optimal solution is implicitly defined by the first-order optimality conditions.

- Optimizer (lower level) terminates when the **KKT residual** falls below tolerance:

$$\mathcal{F}(\hat{\mathbf{x}}^\star, \boldsymbol{\theta},\ \mathbf{y}(\boldsymbol{\theta}), \mathbf{G}(\boldsymbol{\theta})) = \epsilon \approx 0$$

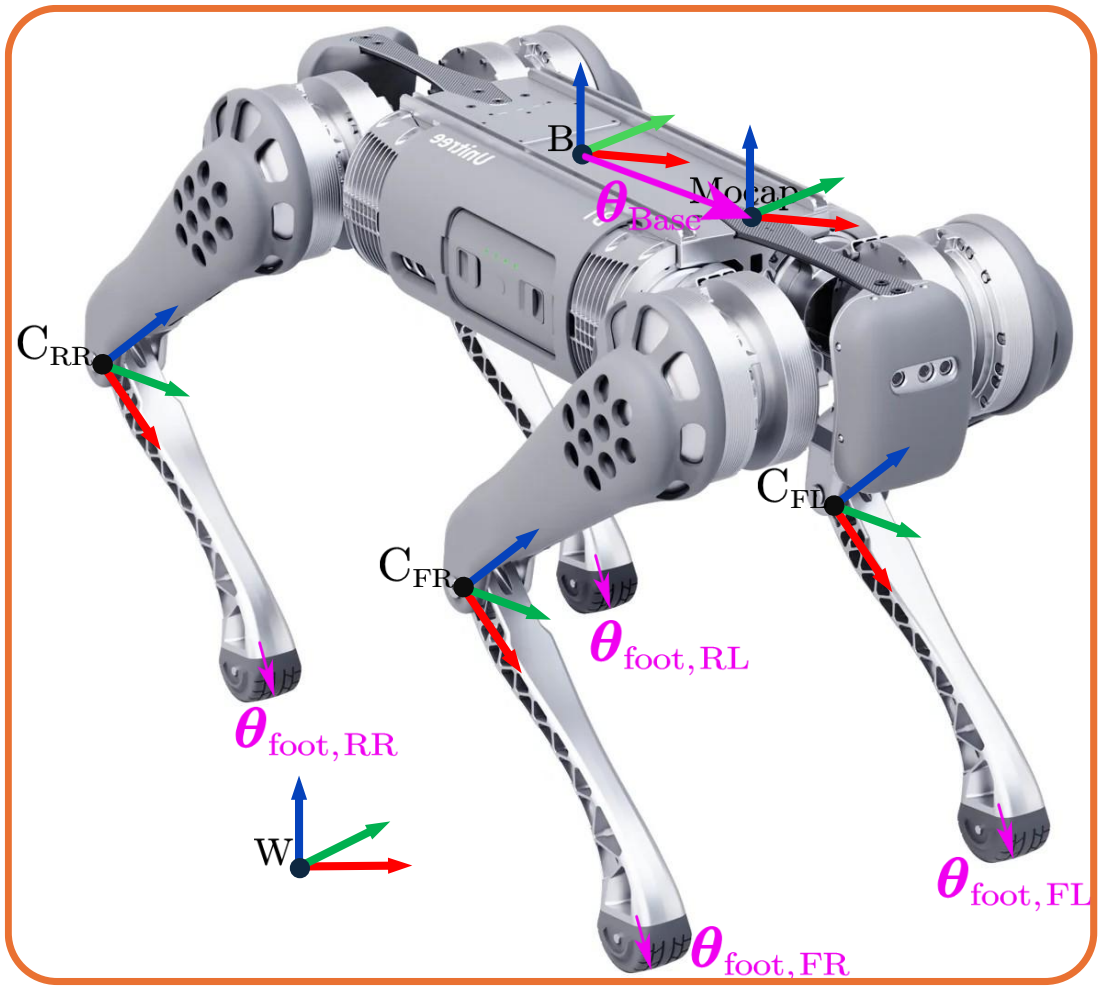
$$\frac{\partial \hat{\mathbf{x}}^\star}{\partial \boldsymbol{\theta}} = -\left(\frac{\partial \mathcal{F}}{\partial \hat{\mathbf{x}}^\star}\right)^{-1}\left(\frac{\partial \mathcal{F}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathcal{F}}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathcal{F}}{\partial \mathbf{G}}\frac{\partial \mathbf{G}}{\partial \boldsymbol{\theta}}\right)$$

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \left(\frac{D\mathcal{L}(\hat{\mathbf{x}}(\boldsymbol{\theta}))}{D\hat{\mathbf{x}}}\right)\left(\frac{D\hat{\mathbf{x}}(\boldsymbol{\theta})}{D\boldsymbol{\theta}}\right) + \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}},$$

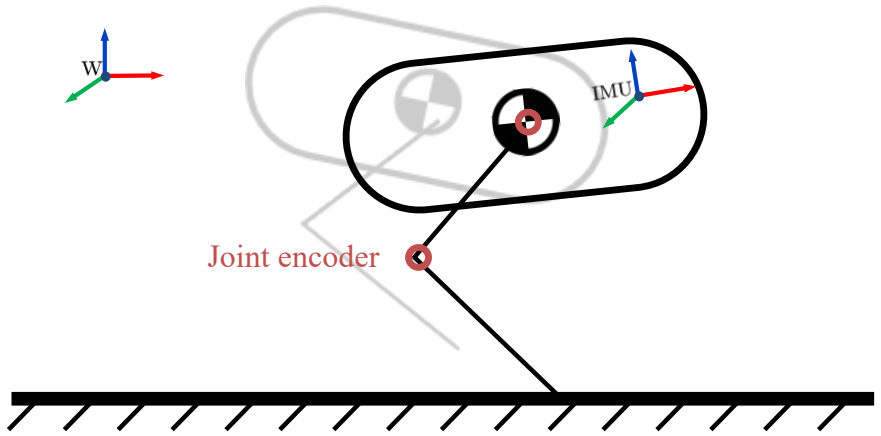
Upper Level: Analytical Gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$

$$\frac{\partial \hat{\mathbf{x}}^{\star}}{\partial \boldsymbol{\theta}} = -\left(\frac{\partial \mathcal{F}}{\partial \hat{\mathbf{x}}^{\star}}\right)^{-1}\left(\frac{\partial \mathcal{F}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathcal{F}}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathcal{F}}{\partial \mathbf{G}}\frac{\partial \mathbf{G}}{\partial \boldsymbol{\theta}}\right)$$

- Kinematics uncertainty:



- Lower Level Estimation



- $\boldsymbol{\theta}$ implicitly affects measurement model $\mathbf{y}$ (forward kinematics vector) and uncertainty jacobian $\mathbf{G}$

$$\begin{bmatrix} G_p \\ G_v \end{bmatrix} = \begin{bmatrix} J(\alpha) & 0_{3\times 3} & 0_{3\times 3} \\ \dot{J}(\alpha,\dot{\alpha}) + \omega^{\times} J(\alpha) & J(\alpha) & -[fk(\alpha)]_{\times} \end{bmatrix} \in \mathbb{R}^{6\times 9}.$$

Upper Level: Analytical Gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$

$$
\begin{aligned}
\min_{\boldsymbol{\theta} \in \mathbb{R}^m} \quad & \mathcal{L}(\hat{\mathbf{x}}(\boldsymbol{\theta}), \mathbf{x}_{\mathrm{GT}}, \boldsymbol{\theta}), \\
\text{s.t.} \quad & \boldsymbol{\theta} \in \mathcal{C}, \\
& \hat{\mathbf{x}}(\boldsymbol{\theta}) = \underset{\mathbf{x} \in \mathbb{R}^{n \times (T+1)}}{\arg\min}\ \mathcal{J}(\mathbf{x}, \boldsymbol{\theta}).
\end{aligned}
$$

- Optimizer (lower level) terminates when the **KKT residual** falls below tolerance:

$$\mathcal{F}(\hat{\mathbf{x}}^\star, \boldsymbol{\theta},\ \mathbf{y}(\boldsymbol{\theta}), \mathbf{G}(\boldsymbol{\theta})) = \epsilon \approx 0$$

Inverse of this Hessian matrix is computationally very expensive

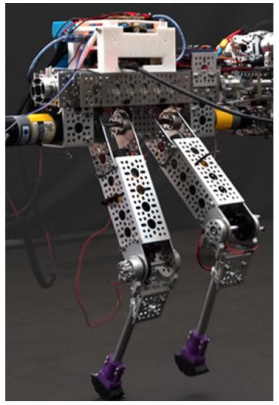
$$\frac{\partial \hat{\mathbf{x}}^\star}{\partial \boldsymbol{\theta}} = -\left(\frac{\partial \mathcal{F}}{\partial \hat{\mathbf{x}}^\star}\right)^{-1}\left(\frac{\partial \mathcal{F}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathcal{F}}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathcal{F}}{\partial \mathbf{G}}\frac{\partial \mathbf{G}}{\partial \boldsymbol{\theta}}\right)$$

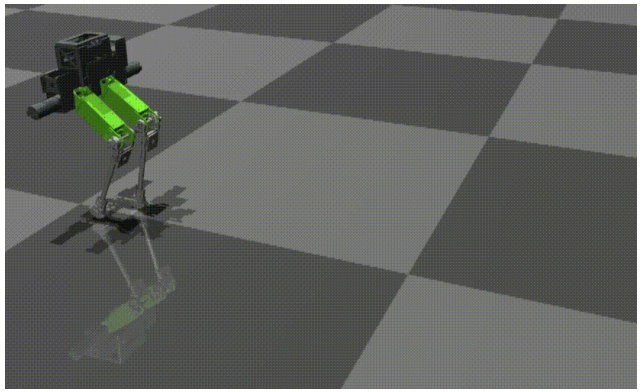
- High-performance sparse linear solvers (PyPardiso) uses banded/block-tridiagonal structure solve the inverse efficiently

$$\left(\frac{\partial \mathcal{F}}{\partial \hat{\mathbf{x}}^\star}\right) Z = -\left(\frac{\partial \mathcal{F}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathcal{F}}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathcal{F}}{\partial \mathbf{G}}\frac{\partial \mathbf{G}}{\partial \boldsymbol{\theta}}\right), \quad Z := (\partial \hat{\mathbf{x}}^\star)/(\partial \boldsymbol{\theta})$$

Result

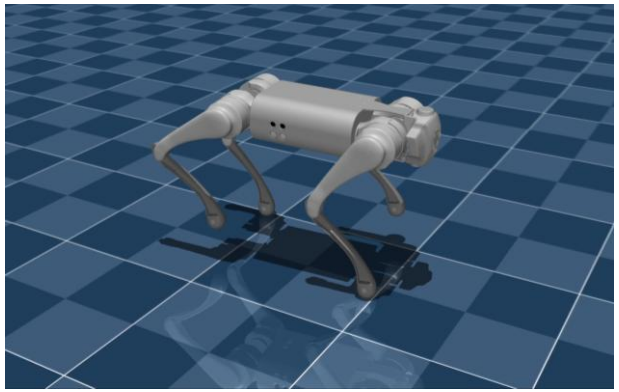
(a)





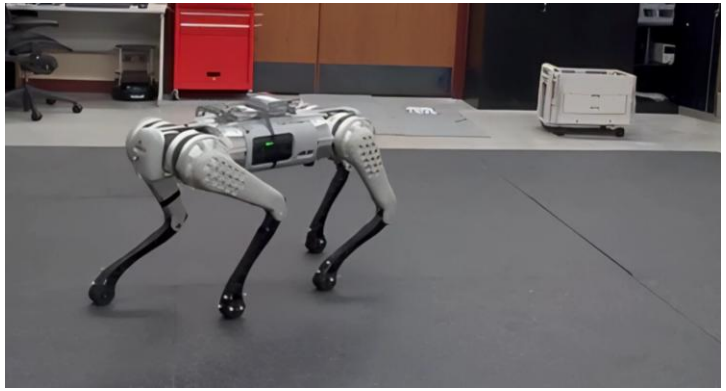
- 5-linkage walker (Matlab sim): **sensor noise + calf length error injected (deterministic dynamics)**

(b)



- Go1 (MuJoCo): **process + sensor noise + mocap offset**

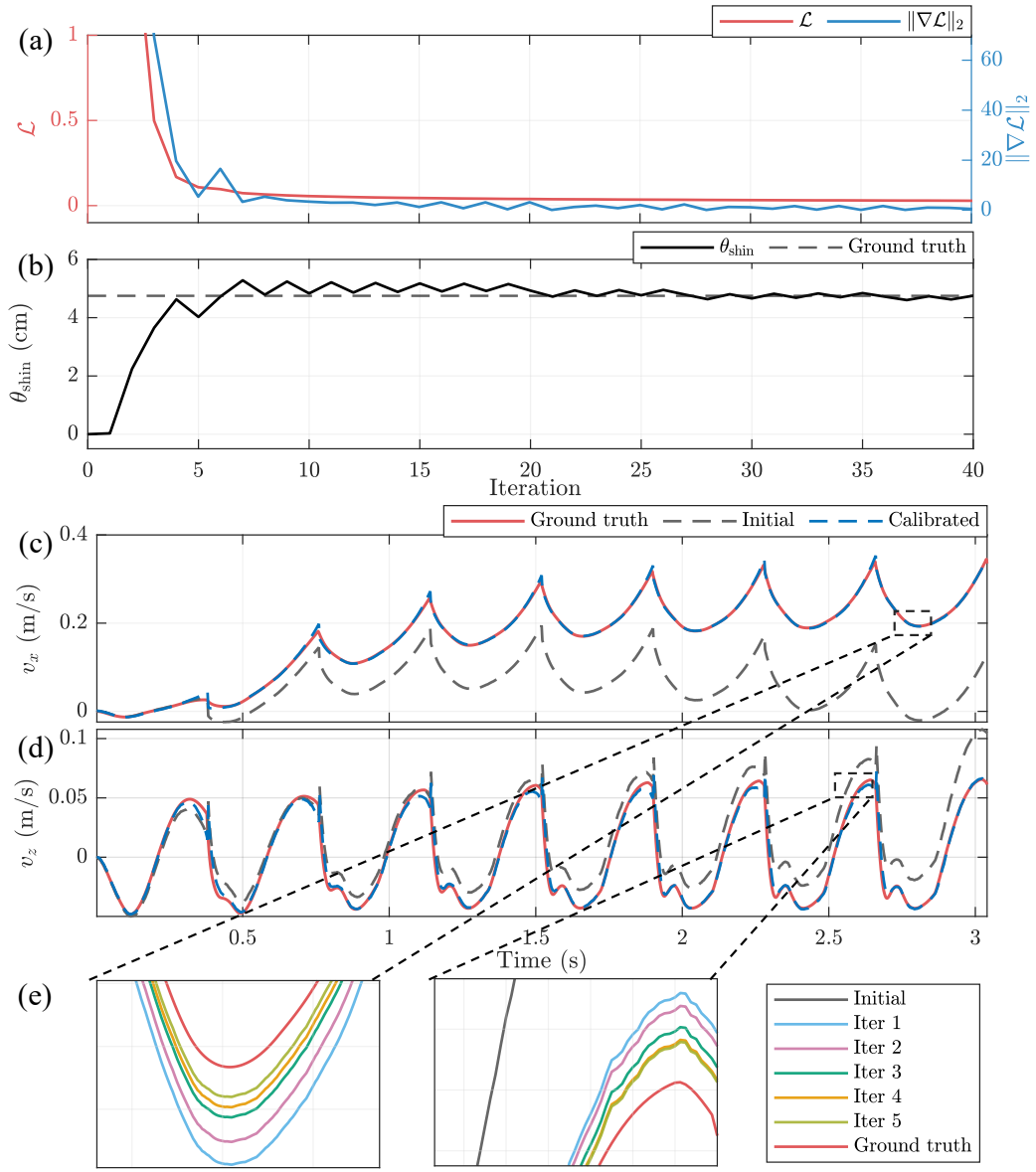
(c)



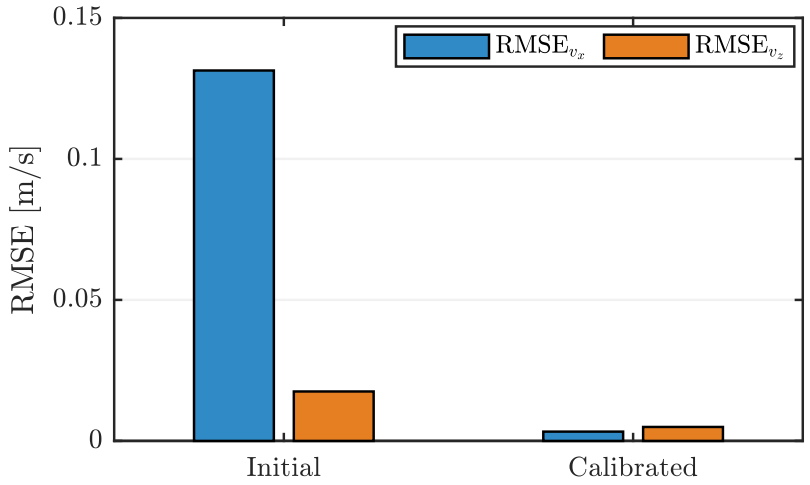
- B1 (Hardware mocap): **noise + kinematics + base offset, plus new-segment evaluation**

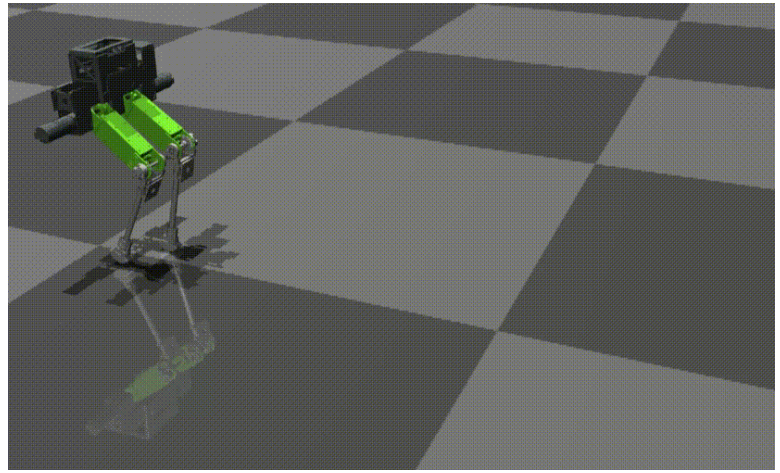
Result: Five Linkage Walker

- 5-linkage walker (Matlab sim): **sensor noise + claf length error injected (deterministic dynamics)**



➢ RMSE comparison:

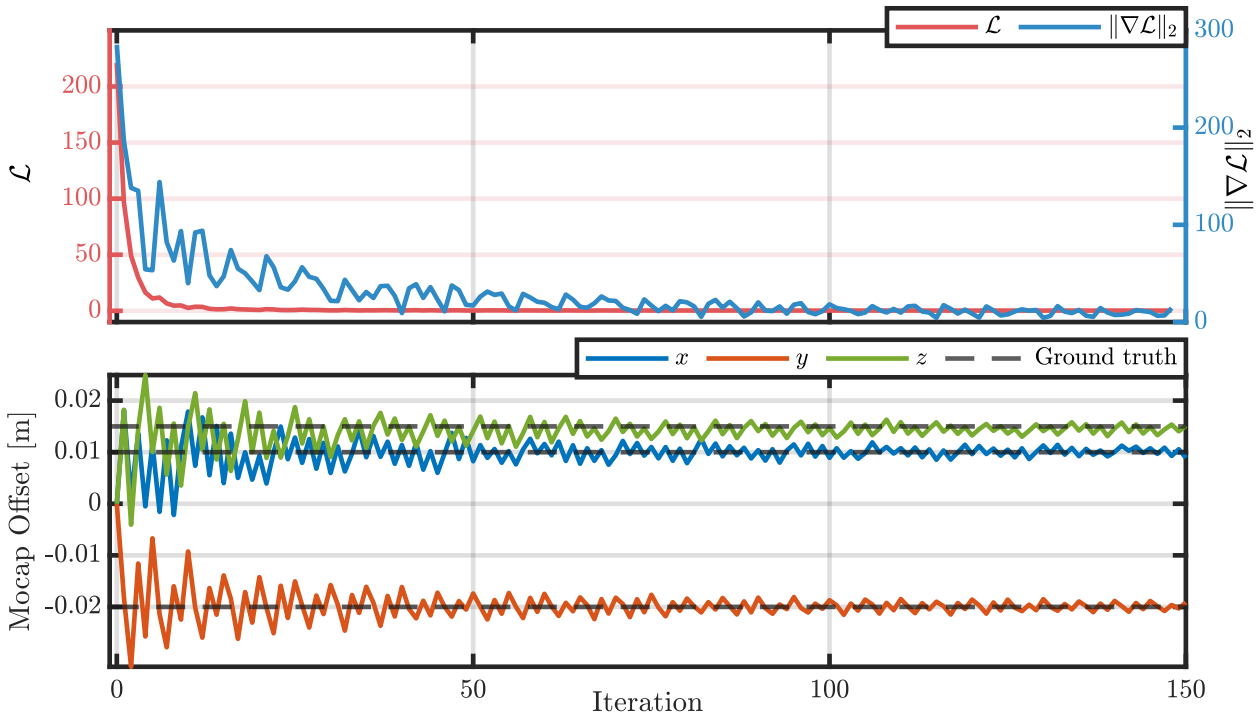




- Estimated trajectory converged to the ground truth with kinematics offsets calibrated.

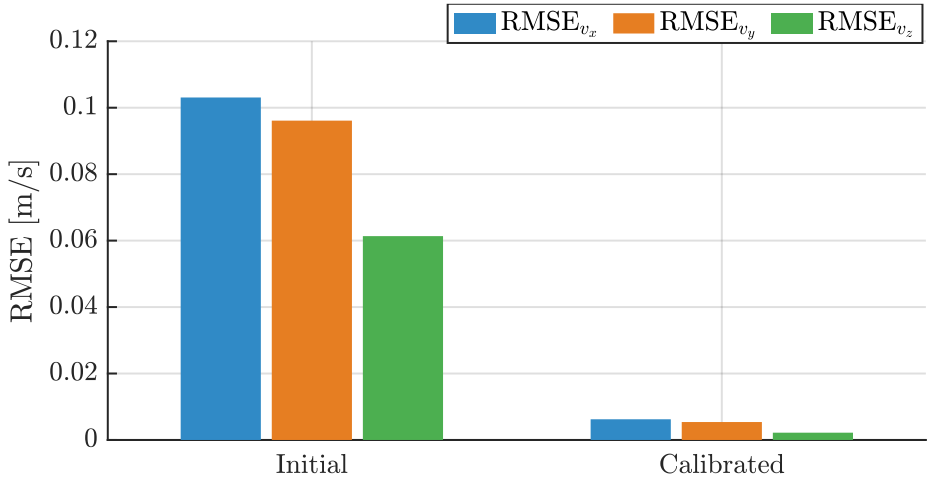
Result: Quadrupedal Robot in MuJoCo.

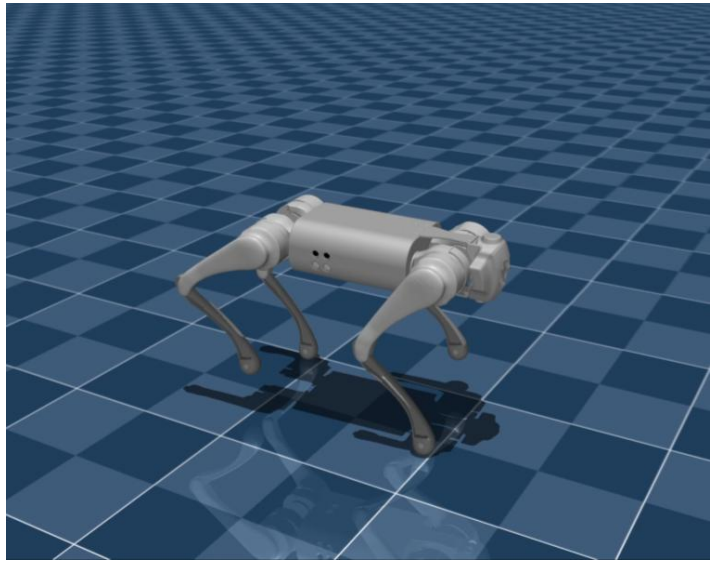
- Unitree Go1 (MuJoCo): **process + sensor noise + offset between mocap and Pinocchio URDF center**



- With ground-truth data measured in a different artificial mocap frame, the offsets are also calibrated.

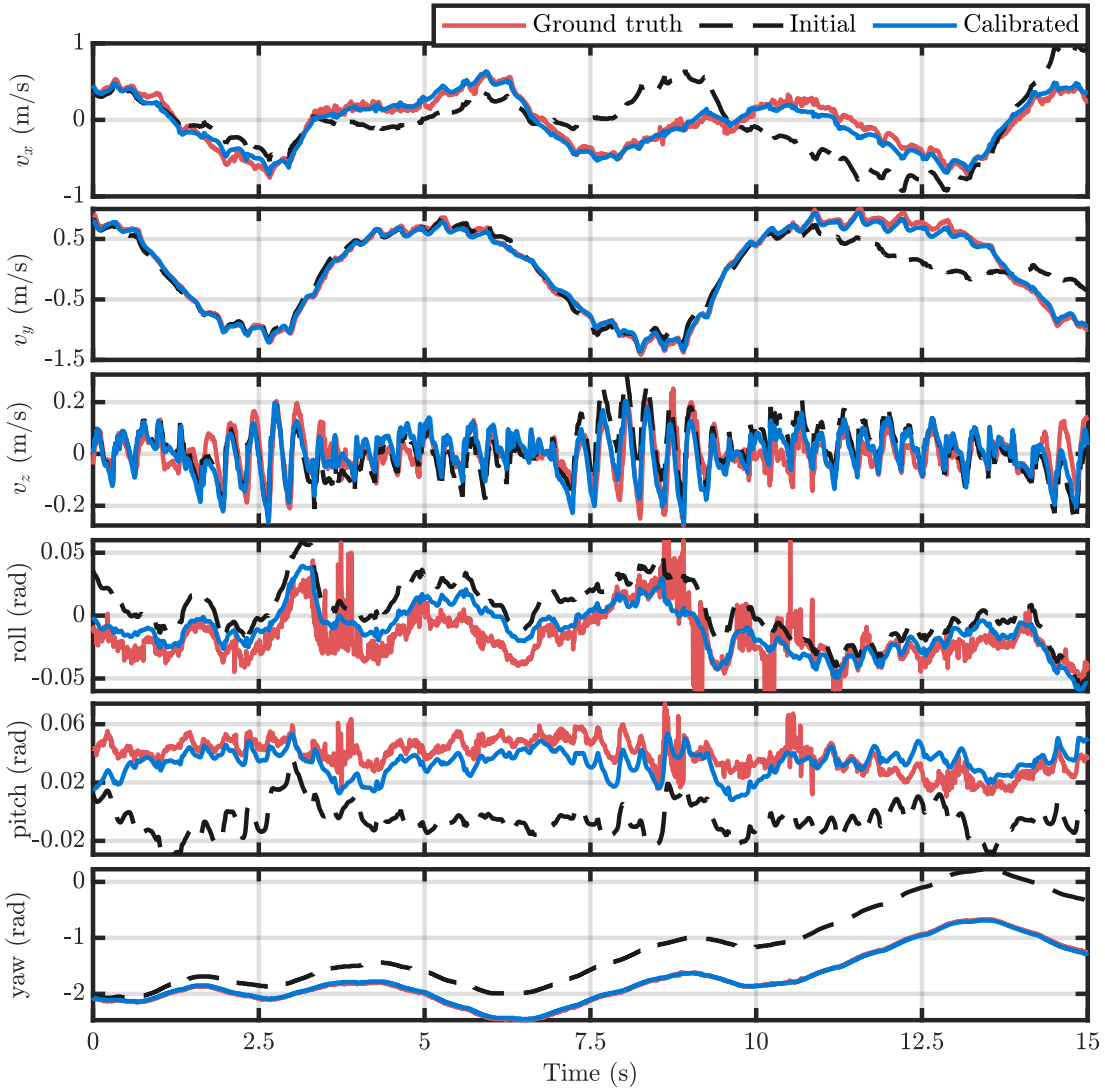
➢ RMSE comparison:



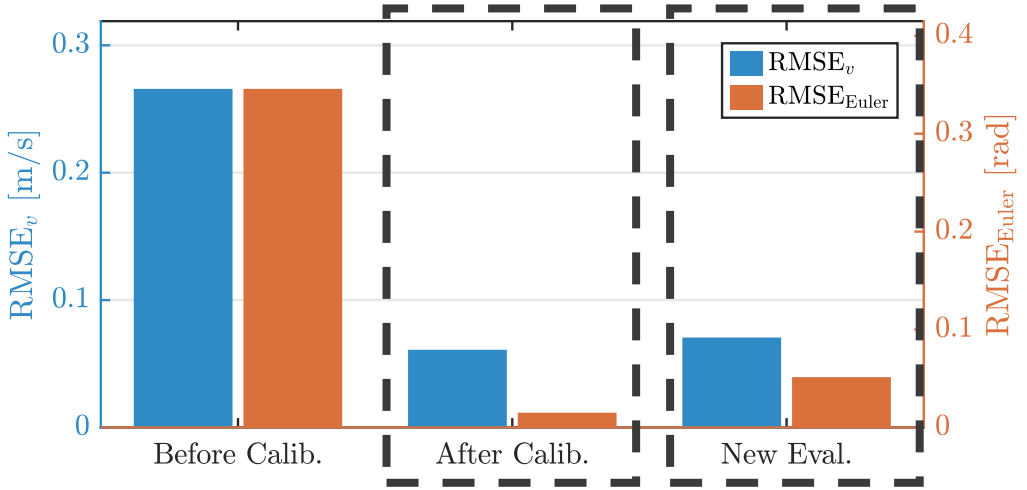


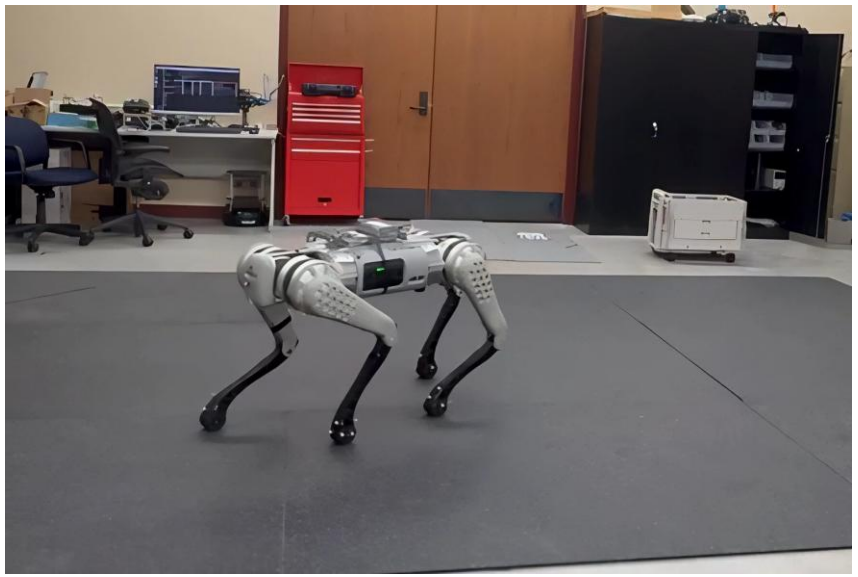
Result: Unitree B1 hardware

- Calibration result:



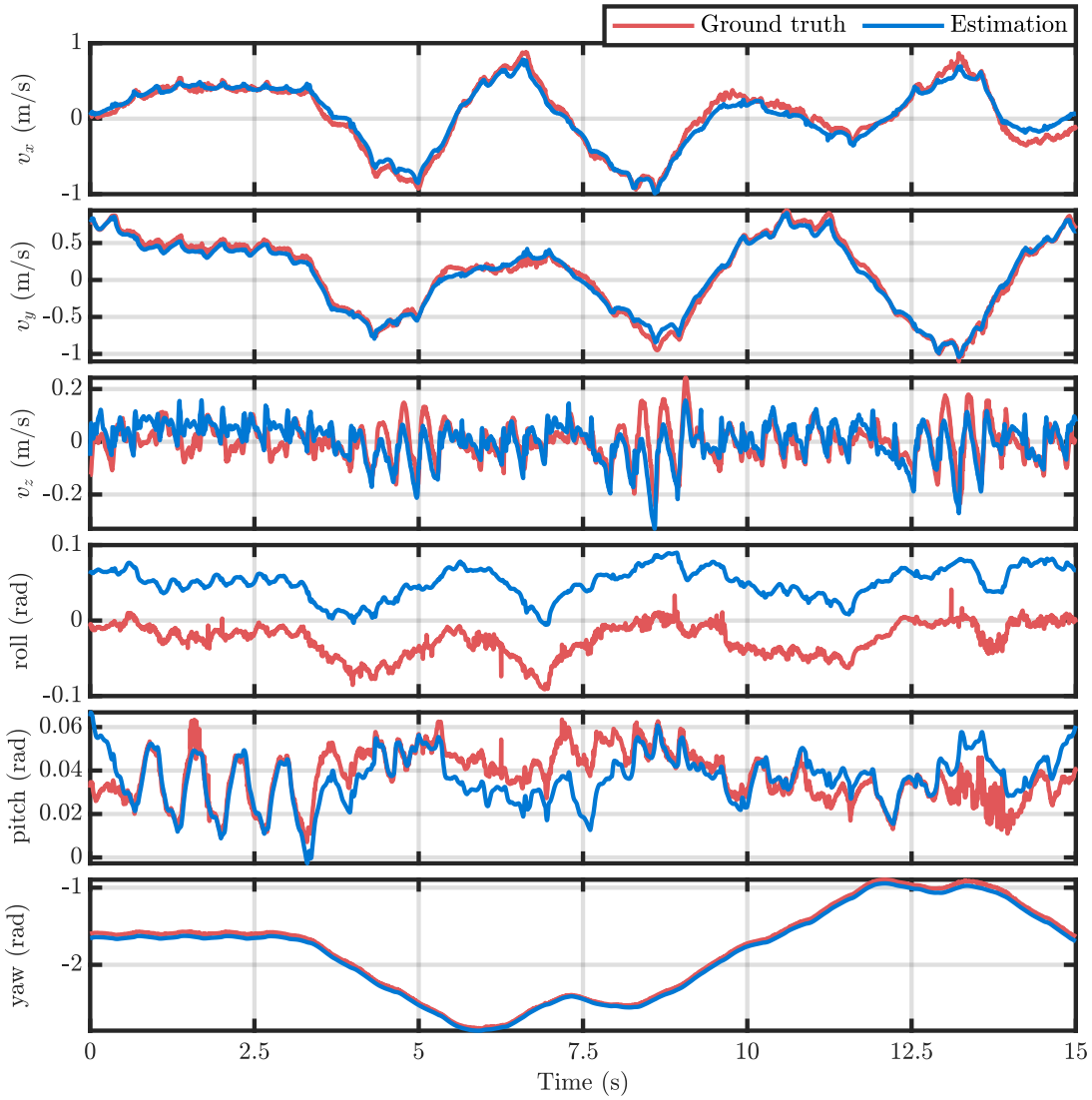
- RMSE comparison:



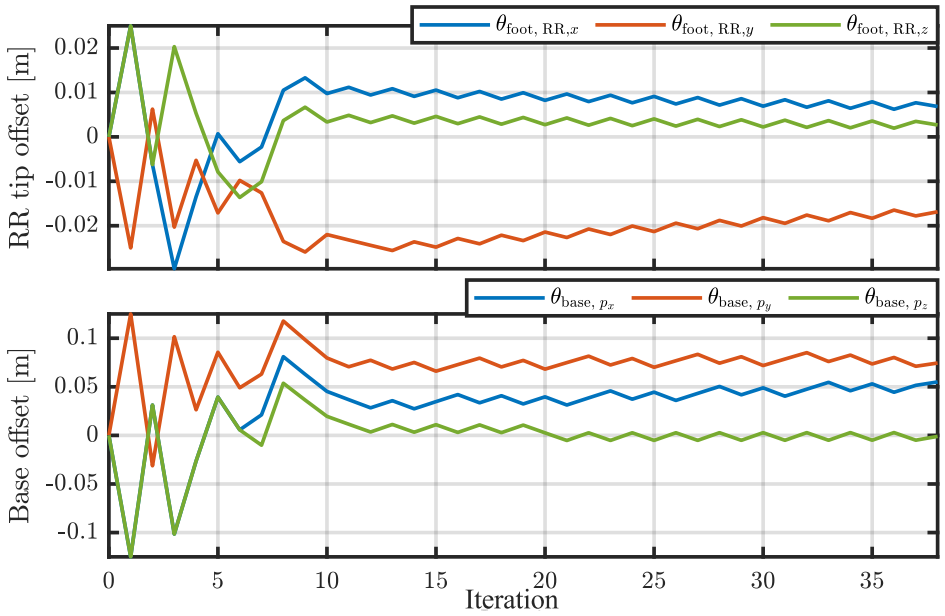


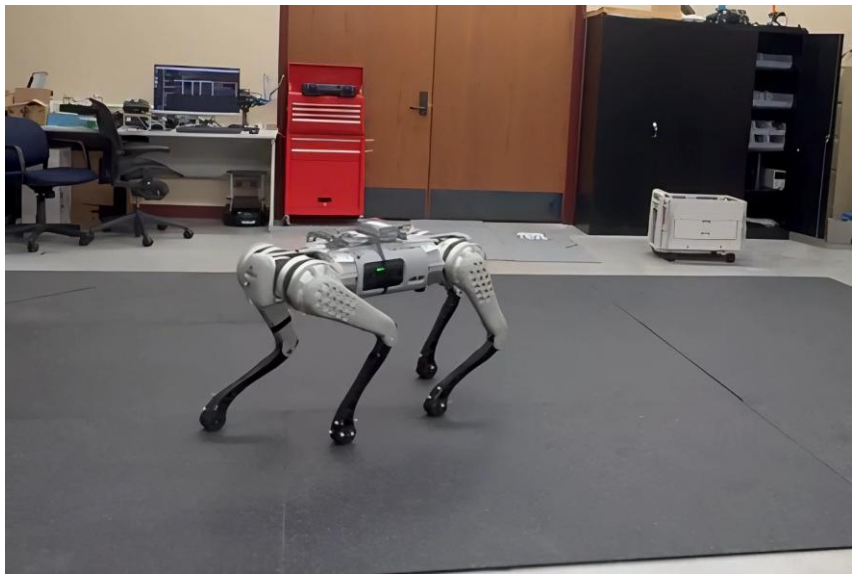
Result: Unitree B1 hardware

- Evaluation result after calibration:



- Kinematic parameter:





Future work

- Formalize **observability** of covariances and kinematic offsets within the bi-level calibration

- Extend the framework to **general robot** state estimation across legged, aerial, and wheeled platforms, incorporating **exteroceptive sensing**.

- Replace generic LMO-SDP solves with a custom primal–dual IPM using the log-det barrier for the SPD cone, and a predictor–corrector to follow the central path