Robot Control Equations

Table 1: MuJoCo Controller Key Bindings

Category	**Key Binding and Description**
General	
ESC	Exit simulation
ENTER	Reset simulation
Gripper Control	
Z	Close gripper (jaw)
X	Open gripper (jaw)
C	Side gripper outward
V	Side gripper inward
B	Tip gripper open
N	Tip gripper close
Arm Movement	
W / S	Forward / Backward
A / D	Left / Right
Q / E	Up / Down
Base Movement (relative to orientation)	
$\uparrow$ / $\downarrow$	Forward / Backward
$\rightarrow$ / $\leftarrow$	Right / Left
T / R	Rotate Counterclockwise / Clockwise
Camera Control (Free Mode)	
Mouse Left Drag	Rotate camera
Mouse Right Drag	Pan camera
Mouse Middle Drag	Zoom
Mouse Scroll	Zoom

1 Control for Mobile Base (`control_base`)

This method drives a mobile robot base to a target position $(x_{\text{target}}, y_{\text{target}})$ and orientation θ_{target} using PID control. Errors are transformed into the robot's local frame, and PID gains compute linear ($v_{x_{\text{local}}}, v_{y_{\text{local}}}$) and angular ($\omega$) velocities.

These are mapped to wheel velocities for a four-wheeled differential drive, with small commands clipped to zero to avoid unnecessary actuation. The key bindings (e.g., ↑/↓ for forward/backward, T/R for rotation) allow manual control of the base movement, complementing the automated PID control.

1.1 Position and Orientation Errors

The errors in position and yaw angle are calculated as:

$$\Delta x = x_{\text{target}} - x_{\text{current}} \tag{1}$$
$$\Delta y = y_{\text{target}} - y_{\text{current}} \tag{2}$$
$$\Delta\theta = \arctan 2\left(\sin(\theta_{\text{target}} - \theta_{\text{current}}), \cos(\theta_{\text{target}} - \theta_{\text{current}})\right) \tag{3}$$

where θ_{target} and θ_{current} are derived from quaternions using:

$$\theta = \arctan 2\left(2(wz + xy), 1 - 2(y^2 + z^2)\right) \tag{4}$$

for quaternion $[w, x, y, z]$.

1.2 Local Frame Transformation

Errors are transformed into the robot's local frame:

$$\Delta x_{\text{local}} = \cos(\theta_{\text{current}})\Delta x + \sin(\theta_{\text{current}})\Delta y \tag{5}$$
$$\Delta y_{\text{local}} = -\sin(\theta_{\text{current}})\Delta x + \cos(\theta_{\text{current}})\Delta y \tag{6}$$

1.3 Integral Terms

The integral terms accumulate errors over time:

$$I_x = I_x + \Delta x_{\text{local}} \cdot \Delta t \tag{7}$$
$$I_y = I_y + \Delta y_{\text{local}} \cdot \Delta t \tag{8}$$
$$I_\theta = I_\theta + \Delta\theta \cdot \Delta t \tag{9}$$

where Δt is the simulation timestep.

1.4 Derivative Terms with Smoothing

Derivative terms are computed and smoothed with a low-pass filter ($\alpha = 0.7$):

$$D_{x_{\text{local}}} = \frac{\Delta x_{\text{local}} - \Delta x_{\text{prev}}}{\Delta t} \tag{10}$$
$$D_{y_{\text{local}}} = \frac{\Delta y_{\text{local}} - \Delta y_{\text{prev}}}{\Delta t} \tag{11}$$
$$D_\theta = \frac{\Delta\theta - \Delta\theta_{\text{prev}}}{\Delta t} \tag{12}$$
$$D_x = \alpha D_x + (1 - \alpha)D_{x_{\text{local}}} \tag{13}$$
$$D_y = \alpha D_y + (1 - \alpha)D_{y_{\text{local}}} \tag{14}$$
$$D_\theta = \alpha D_\theta + (1 - \alpha)D_\theta \tag{15}$$

1.5 Control Velocities

The control velocities are computed using PID gains ($k_p = 3.0$, $k_i = 0.02$, $k_d = 0.3$ for position; $k_{p_\theta} = 3.0$, $k_{i_\theta} = 0.03$, $k_{d_\theta} = 0.3$ for orientation):

$$v_{x_{\text{local}}} = k_p \Delta x_{\text{local}} + k_i I_x + k_d D_x \tag{16}$$

$$v_{y_{\text{local}}} = k_p \Delta y_{\text{local}} + k_i I_y + k_d D_y \tag{17}$$

$$\omega = k_{p_\theta} \Delta\theta + k_{i_\theta} I_\theta + k_{d_\theta} D_\theta \tag{18}$$

1.6 Wheel Velocities

The velocities are mapped to four wheels of a differential drive robot, where D is the distance from the robot's center to the wheels and r is the wheel radius:

$$v_{\text{target}}[0] = \frac{v_{x_{\text{local}}} - v_{y_{\text{local}}} - \omega D}{r} \tag{19}$$

$$v_{\text{target}}[1] = \frac{v_{x_{\text{local}}} + v_{y_{\text{local}}} + \omega D}{r} \tag{20}$$

$$v_{\text{target}}[2] = \frac{v_{x_{\text{local}}} + v_{y_{\text{local}}} - \omega D}{r} \tag{21}$$

$$v_{\text{target}}[3] = \frac{v_{x_{\text{local}}} - v_{y_{\text{local}}} + \omega D}{r} \tag{22}$$

2 Inverse Kinematics Solver (`ik_solution`)

The `ik_solution` method solves an inverse kinematics problem to position a robotic arm's end-effector by optimizing joint parameters h_1, h_2, and θ.

It optimizes joint parameters h_1, h_2, and θ to minimize the angle between vectors from two vertical links points to the target, subject to length, angle, and z-height constraints. The solution ensures the arm's configuration is feasible and avoids singularities or ground penetration.

2.1 Base to End-Effector Transformation

The end-effector's position is transformed into the base frame:

$$\mathbf{R}_{\text{base}} = \text{quaternion\_to\_matrix}(\mathbf{q}_{\text{base}}) \tag{23}$$

$$\mathbf{ee}_{\text{base}} = \mathbf{R}_{\text{base}}^T(\mathbf{ee}_{\text{world}} - \mathbf{p}_{\text{arm\_base}}) \tag{24}$$

$$\mathbf{target} = \mathbf{ee}_{\text{base}} + \mathbf{err}_{\text{base}} \tag{25}$$

2.2 Base Points Calculation

Two base points are defined for the arm segments:

$$\mathbf{b}_1 = \begin{bmatrix} -\frac{D_2}{2}\cos(\theta) \\ -\frac{D_2}{2}\sin(\theta) \\ h_1 \end{bmatrix}, \quad \mathbf{b}_2 = \begin{bmatrix} \frac{D_2}{2}\cos(\theta) \\ \frac{D_2}{2}\sin(\theta) \\ h_2 \end{bmatrix} \tag{26}$$

where $D_2 = 0.2$.

2.3 Objective Function

The objective is to minimize the negative angle between vectors $\mathbf{v}_1 = \mathbf{target} - \mathbf{b}_1$ and $\mathbf{v}_2 = \mathbf{target} - \mathbf{b}_2$:

$$l_1 = \|\mathbf{v}_1\|, \quad l_2 = \|\mathbf{v}_2\| \tag{27}$$

$$\cos(\theta_{\text{angle}}) = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\max(l_1 l_2, 10^{-12})} \tag{28}$$

$$\text{obj}(\mathbf{x}) = -\arccos(\cos(\theta_{\text{angle}})) \tag{29}$$

2.4 Constraints

The optimization is subject to the following constraints:

- Length constraints:

$$l_{1,\text{min}}^2 \leq \|\mathbf{target} - \mathbf{b}_1\|^2 \leq l_{1,\text{max}}^2 \tag{30}$$

$$l_{2,\text{min}}^2 \leq \|\mathbf{target} - \mathbf{b}_2\|^2 \leq l_{2,\text{max}}^2 \tag{31}$$

  where $l_{1,\text{min}} = 0.4$, $l_{1,\text{max}} = 1.0$, $l_{2,\text{min}} = 0$, $l_{2,\text{max}} = 0.6$.

- Angle constraint:

$$\theta_{\text{angle}} \geq 20^\circ \tag{32}$$

- Projection constraints:

$$\text{tol} \geq \text{target}[0](-\sin(\theta)) + \text{target}[1]\cos(\theta) \tag{33}$$

$$\text{tol} \leq \text{target}[0](-\sin(\theta)) + \text{target}[1]\cos(\theta) \tag{34}$$

  where $\text{tol} = 10^{-3}$.

- Z-height constraints:

$$\mathbf{l}_{1\text{end}}[2] \geq -z_{\text{threshold}} \tag{35}$$

$$\mathbf{l}_{2\text{end}}[2] \geq -z_{\text{threshold}} \tag{36}$$

  where:

$$\mathbf{l}_{1\text{end}} = \mathbf{b}_1 - \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|}(\text{arm\_length} - l_1) \tag{37}$$

$$\mathbf{l}_{2\text{end}} = \mathbf{b}_2 - \frac{\mathbf{v}_2}{\|\mathbf{v}_2\|}(\text{arm\_length} - l_2) \tag{38}$$

  with $\text{arm\_length} = 1.15$, $z_{\text{threshold}} = 0.05$.

2.5 Optimization

The optimization problem is:

$$\min_{\mathbf{x}} \text{obj}(\mathbf{x}), \quad \mathbf{x} = [h_1, h_2, \theta] \tag{39}$$

subject to the constraints, with bounds:

$$0.0 \leq h_1 \leq 1.0 \tag{40}$$

$$0.0 \leq h_2 \leq 1.0 \tag{41}$$

$$-3.14 \leq \theta \leq 3.14 \tag{42}$$

The SLSQP algorithm is used with a tolerance of 10^{-5} and a maximum of 15 iterations.