1 Introduction

For this project, a Linux environment with a GPU is highly recommended for rendering the simulation environment. To solve this issue, we will be using an Amazon Web Services (AWS) command line interface (CLI) and then visualizing results using a remote viewer tool.

AWS Installation:

- YouTube Tutorial: https://www.youtube.com/watch?v=CjKhQoYeR4Q

NICE DCV for Visualizing:

- Guide: https://jeremypedersen.com/posts/2023-03-06-nice-dcv-ubuntu-22/
- Note: Use Ubuntu 20.04 instead of 22.04, as the latter is not compatible with ROS Noetic.

Installing LocoMan:

- Follow instructions on GitHub: https://github.com/linchangyi1/locoman
- Common issue with new installations of Linux (sudo access not present): https://www.baeldung.com/linux/username-not-in-sudoers-file

In the field of robotics, legged robots offer distinct advantages over wheeled locomotion strategies: the ability to traverse sloped environments, robustness to vertically large or entangling obstacles that impede or obscure robots, and navigation between constrained spaces. However, dexterous manipulation presents a challenge to legged robots, including quadrupeds. One possible solution to this has been displayed in the paper "LocoMan: Advancing Versatile Quadrupedal Dexterity with Lightweight Loco-Manipulators", in which 3-DOF manipulators are attached to the robot's front legs to enhance manipulation tasks. Thus, the robot can both navigate complex terrain and, upon reaching a goal location, grasp objects by switching to a manipulation-specific operation mode using a Finite State Machine (FSM).

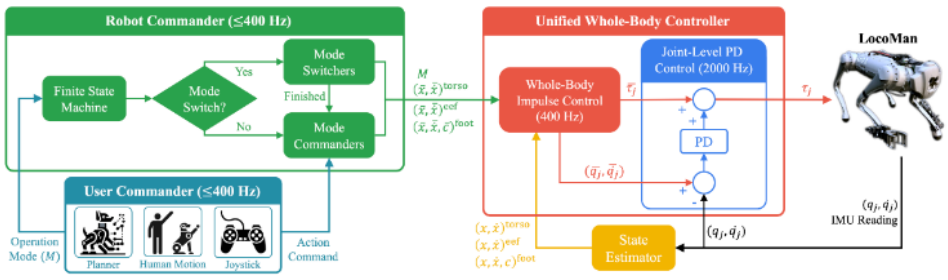

Figure 1: Block Diagram for LocoMan. A state machine (left, green) dictates the torso, end effector (eef) and foot desired positions and a whole body controller (right, red) computes torques for the simulated or hardware version of the robot. A state estimator feeds sensor readings back into the whole body controller for feedback control.

Arguably one of the most complicated part of quadruped robots is the control strategy behind legged motion. One assumption used in many control strategies is that of massless legs, which simplify the problem of control dramatically to focus on body orientation and point feet simplifications rather than controlling lightweight legs that can be moved quickly anyways. Using this simplification, torques are then computed that dictate the trajectory of the feet, stabilize the center of mass of the robot, and complete other tasks.

2 Model

For the purpose of this assignment, the state estimation process and finite state machine will not be altered; development will focus on designing the whole body controller and implementing the design to meet objectives.

For each operation mode M, LocoMan has several sub-objectives to complete tasks effectively. Each operation mode uses a null-space projection, as seen later in the assignment, and includes the following desired states:

$$(\bar{x}, \bar{\dot{x}}) \quad \text{torso desired state} \tag{1}$$

$$(\bar{x}, \bar{\dot{x}}) \quad \text{end effector desired state} \tag{2}$$

$$(\bar{x}, \bar{\dot{x}}, \bar{c}) \quad \text{foot desired state} \tag{3}$$

Table 1: Hierarchical Tracking Objectives in Whole-Body Control (WBC).

Operation Mode	**Objective 1**	**Objective 2**	**Objective 3**	**Objective 4**	**State Estimator**
Single Foot Manipulation	Torso Position	Torso Orientation	Foot_{m} Position	-	Kinematics-Based
Single Gripper Manipulation	Torso Position	Torso Orientation	$\text{Gripper}_{\text{m}}$ Position	$\text{Gripper}_{\text{m}}$ Orientation	Kinematics-Based
Bimanual Manipulation	$\text{Gripper}_{\text{l,r}}$ Positions	$\text{Gripper}_{\text{l,r}}$ Orientations	-	-	Kinematics-Based
Locomotion	Torso Velocity	Torso Orientation	Foot_{s} Positions	-	Kalman Filter-Based
Loco-Manipulation	Torso Velocity	Torso Orientation	Foot_{s} Positions	$\text{Gripper}_{\text{l,r}}$ Orientations	Kalman Filter-Based

Deriving the whole body controller with the hierarchical objective structure has multiple parts, which are outlined as:

1. Compute Jacobians and their pseudoinverses for solving inverse kinematics problems for joint angles of the robot

2. Project non-primary objectives into the null space of the problem to be solved simultaneously as the control problem

3. Formulate a Quadratic Programming (QP) optimization problem to solve for the optimal ground forces.

4. Compute the optimal torques while damping the response with a low-level feedforward PD controller.

2.1 Jacobian Calculation

Since the objective hierarchy for LocoMan changes depending on the control mode, the structure of the dynamics and control problem must be properly accounted for.

Using Table 1, one may note there are n tracking items, each of which are at at ordinal position i . For each tracking item, the desired position, velocity, and acceleration for the body and joints of the robot are computed using

$$\overline{q}_i = \overline{q}_{i-1} + J^{\dagger}_{i|pre}(\overline{x}_i - x_i - J_i(\overline{q}_{i-1} - q)) \tag{4}$$

$$\overline{\dot{q}}_i = \overline{\dot{q}_{i-1}} + J^{\dagger}_{i|pre}(\overline{x}_i - J_i\overline{q}_{i-1}) \tag{5}$$

$$\overline{\ddot{q}_i} = J^{dyn\ddagger}_{i,pre}(\ddot{x}^{cmd}_i - \dot{J}_i\dot{q} - J_i\overline{\ddot{q}}_{i-1}), \tag{6}$$

where the acceleration command $\dot{x}^{\text{cmd}}_i$, is given as

$$\ddot{x}^{\text{cmd}}_i = \overline{\ddot{x}_i} + K^{\text{acc}}_p(\bar{x}_i - x_i) + K^{\text{acc}}_d(\bar{\dot{x}}_i - \dot{x}_i), \tag{7}$$

where K^{acc}_p and K^{acc}_d are position and velocity feedback gains.

Two pseudo-inverse Jacobian matrices are used for these calculations: $J^{\dagger}$, which is based on a singular value decomposition calculation, and $J^{\ddagger}$, which is a dynamically-consistent pseudo-inverse. The Jacobian for this problem has to have a pseudo-inverse to compute the desired position and velocity from the tracking objective. Having a dynamically-consistent pseudo-inverse is essential for tracking the acceleration constraints. Both of these pseudo-inverses are calculated as follows:

$$J = U\Sigma V^T, \quad J\dagger = V\Sigma^{\dagger}U^T \tag{8}$$

$$J^{\ddagger} = A^{-1}J^T(JA^{-1}J^T)^{-1} \tag{9}$$

2.2 Null Space Projection

Table 1 displays different control objectives for Locoman, but how should all these control objectives be met simultaneously? One strategy is to use the null space projection, which projects non-primary control objectives into the null space of the controller. Recall the null space can be constructed as:

$$J\dot{q} = 0$$

with the Jacobian matrix J representing the original Jacobian matrix. To project a control action into the null space, a dynamically consistent pseudoinverse is needed to ensure the control actions for the non-primary and primary control actions are dynamically feasible.

For LocoMan, the null space projection is calculated via the following equations:

$$
\begin{aligned}
N_{i-1} &= N_0 N_{1|0} \cdots N_{i-1|i-2} \\
J_{i|\text{pre}} &= J_{i-1} N_{i-1} \\
J_{i|i-1} &= J_i \left(I - J_i^{\dagger} J_{i-1} \right) \\
N_{i|i-1} &= I - J_{i|i-1}^{\dagger} J_{i|i-1} \\
N_{i-1}^{\text{dyn}} &= N_0^{\text{dyn}} N_{1|0}^{\text{dyn}} \cdots N_{i-1|i-2}^{\text{dyn}} \\
J_{i|pre}^{\text{dyn}} &= J_i N_{i-1}^{\text{dyn}} \\
J_{i|i-1}^{\text{dyn}} &= J_i \left(I - J_{i-1}^{\dagger} J_{i-1} \right) \\
N_{i|i-1}^{\text{dyn}} &= I - J_{i|i-1}^{\text{dyn}\dagger} J_{i|i-1}^{\text{dyn}}
\end{aligned}
$$

where

$$N_{i-1}, J_{i|i-1}, and N_{i|i-1}$$

are from the null space projection for the last tasks and the Jacobian matrix

$$J_i$$

is the projection from the previous task.

2.3 Quadratic Programming

Quadratic programming (QP) solves an optimization by structuring the problem in quadratic form. In this assignment, a QP structure is designed to compute the optimal ground forces using desired acceleration values. Normally, QP problems can be arranged and computed using one of several solvers to handle different structures of problems, most of which are outside of the scope of this assignment. Each QP problem minimizes a cost function subject to different constraints that are critical to the problem. One example commonly used in control theory is torque or velocity limits for motors; any optimal solutions must consider the hardware limitations of the system to ensure that any trajectories can be followed. The QP problem setup for LocoMan is as follows:

$$
\begin{aligned}
&\min f Q_1 f_r + \delta Q_2 \dot{\delta} \\
&S_f(A\ddot{q}^{cmd} + b + g) = S_f J_c^T f_r \quad \text{(floating torso dyn.)} \\
&\qquad\qquad \ddot{q}^{cmd} = \bar{\ddot{q}} + \begin{bmatrix} \delta_t \\ 0_j \end{bmatrix} \quad \text{(floating torso acceleration)} \\
&\qquad\qquad W f_r \geq 0 \quad \text{(contact force constraints)}
\end{aligned}
$$

where S is the selection matrix for the dynamics of the torso, A is the mass matrix of the system, b is the Coriolis force in the quadruped, g is the gravitational force in the bodies and W is the constraint force matrix for foot contact. For this assignment, the `quadprog` Python library will be used.

2.4 Torque Control Calculation

A PD controller is added to the problem formulation to accommodate faster actions than the whole body controller and to demonstrate robustness to foot placements outside of the expected final time. The calculations for defining the torques are as follows:

$$\begin{bmatrix} \tau_t \\ \tau_j \end{bmatrix} = Aq^{cmd} + b + g - J^T f_r \quad \text{(Feedforward PD Torque Control)}$$

The final joint torque under PD control is as follows:

$$\tau_j = \tau_j + K_p^{tau}(q_j - q_j) + K_d^{tau}(\dot{q}_j - \dot{q}_j)$$
$$\text{where } \tau_j = \text{joint torques}$$

3 Problems

Exercise 1. To complete these exercises, a few key functions need to be implemented in `whole_body_controller.py`:

- `_hierarchical_tracking`
- `qp_optimization`
- `compute_dynamically_consistent_pseudo_inverse`

For the hierarchical tracking function:

1. Create the PD controller for Objectives 1-3
2. Compute the null space projection based on the formulas outlined in this assignment.

For the QP solving function:

1. Define the system matrices (A, b, g, W, etc.) to calculate the QP
2. Formulate the equality constraint CE and the inequality constraint CI

For the pseudo-inverse:

1. Compute the pseudo-inverse for the dynamically-consistent Jacobian $J^{\ddagger}$

Exercise 2. With this implementation of LocoMan working, the robot should be able to change operating mode. Investigate both `joystick.py` and `keyboard.py` to determine how to change the operating modes in simulation. For Exercises 2 and 3, you are welcome to use the keyboard controller directly, write a bash script to automate the keyboard commands, or devise a script to use the commands automatically.

1. Test that the functions are correctly implemented by starting the robot in the Single Gripper Manipulation mode. Move one of the front quadruped legs from the initial configuration to any configuration. This will simulate "grabbing" an object because grasping an object in a dependable fashion can be challenging to say the least.

Note: if you are encountering an issue related to the parsing of the URDF file for the simulation, ensure that the commands are being run in the `LocoMan` directory.

Exercise 3. Now that LocoMan can "grab" objects, the demonstration of the robot under the crib at https://linchangyi1.github.io/LocoMan/ can be recreated in Isaac Gym.

Switch from the Single Gripper Manipulation mode, reach out to "grab" an object, change to Locomotion mode, then back to Single Gripper Manipulation mode, and drop the object. This exercise fully encapsulates some of the core functionality of LocoMan.

4 Reference

1. Lin, C., Liu, X., Yang, Y., Niu, Y., Yu, W., Zhang, T., Tan, J., Boots, B., & Zhao, D. (2024). LocoMan: Advancing Versatile Quadrupedal Dexterity with Lightweight Loco-Manipulators. arXiv preprint arXiv:2403.18197.