Day 1: Qualcomm MuJoCo Tutorial

Ananth Rachakonda
Spandan Roy

Robotics Research Center
International Institute of Information Technology, Hyderabad

27th September, 2023

What is a Robot Simulation Environment?

Robotics Simulation Summarized

Virtual Robot + Virtual Environments to test the Robot Software

What is a Robot Simulation Environment?

Robotics Simulation Summarized

Virtual Robot + Virtual Environments to test the **Robot Software**??

Robot Software

Perception → SLAM → Planning & Navigation → Control

Perception & SLAM

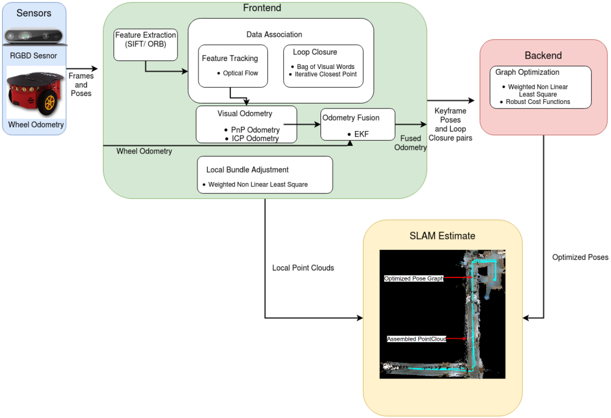


Figure: Block Diagram of the Perception and SLAM stack, (MR Course, RRC, IIITH)

Planning

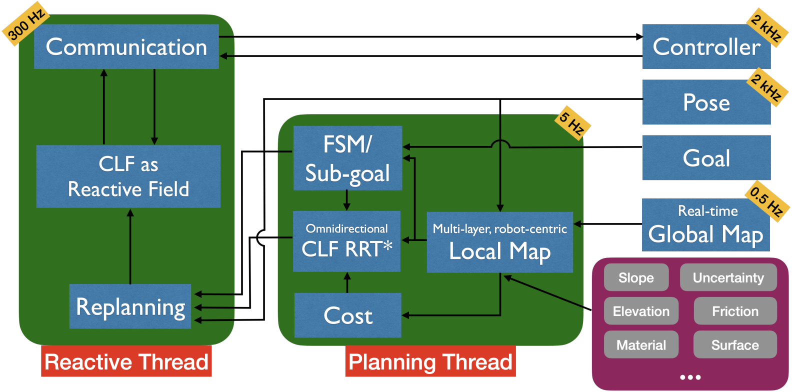


Figure: Block Diagram of CLF Reactive Planning Stack(University of Michigan, Biped Lab)

Control

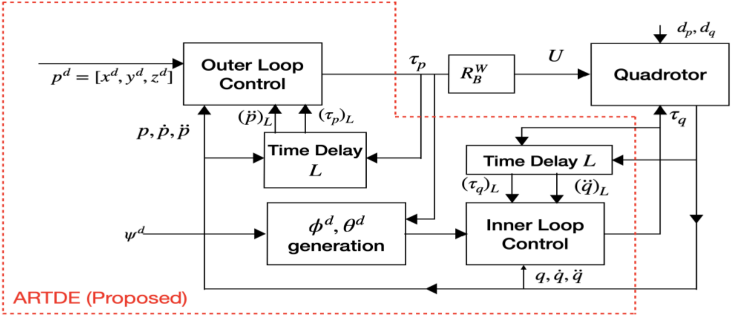


Figure: ARTDE Control for Drones, Block Diagram(Swati Dantu et al. RRC, IIITH)

MuJoCo Robot Software Validation Goal

Perception → SLAM → **Planning & Navigation** → **Control**

Robot Simulation Environment

Robotics Simulation Summarized

1. **Virtual World**
   The simulation space or environment where the virtual robot operates
2. **Virtual Robot Model**
   The digital twin of the robot within the simulation, including its geometry, kinematics, dynamics, and other properties.
3. **Physics Engine**
   Simulates physical laws for realistic interaction between the robot and its environment
4. **Sensors and Actuators**
   Provides feedback of robot state given by virtual environment.
5. **Control Suite**
   Includes the algorithms to command and control the robot within the simulation

Robot Simulation Environments

Open Source or Free

- Gazebo: ROS based, popular.
- V-REP (CoppeliaSim): Multi-robot tools.
- Stage: 2D, ROS compatible.
- Webots (Open Source): Complete dev environment.
- Microsoft AirSim: Drones, cars, Unreal Engine.
- Morse: Academic use, realistic.
- ARGoS: Large swarms of robots.
- URSim: For Universal Robots.
- pyBullet: Python based, versatile.
- MuJoCo: High Fidelity, accurate.

Commercial

- Webots (Commercial): Professional dev environment.
- Vortex Studio: Land and sea vehicles.
- RobotStudio by ABB: For ABB robots.
- RoboDK: Industrial robots.
- MATLAB Simscape: MATLAB and Simulink based.
- ANYbotics ANYmal Research: Quadrupedal robots.

Specialized or Domain-Specific

- Drake: Planning and control by MIT CSAIL.
- FlightGear: For aerial robots.
- CARLA: For autonomous vehicles.

Robot Simulation Environments

Open Source or Free

- Gazebo: ROS based, popular.
- V-REP (CoppeliaSim): Multi-robot tools.
- Stage: 2D, ROS compatible.
- Webots (Open Source): Complete dev environment.
- Microsoft AirSim: Drones, cars, Unreal Engine.
- Morse: Academic use, realistic.
- ARGoS: Large swarms of robots.
- URSim: For Universal Robots.
- pyBullet: Python based, versatile.
- MuJoCo: High Fidelity, accurate.

Commercial

- Webots (Commercial): Professional dev environment.
- Vortex Studio: Land and sea vehicles.
- RobotStudio by ABB: For ABB robots.
- RoboDK: Industrial robots.
- MATLAB Simscape: MATLAB and Simulink based.
- ANYbotics ANYmal Research: Quadrupedal robots.

Specialized or Domain-Specific

- Drake: Planning and control by MIT CSAIL.
- FlightGear: For aerial robots.
- CARLA: For autonomous vehicles.

Why MuJoCo?

1. **Efficiency and Speed:**
   - *Highlight:* Exceptionally fast and efficient in simulating high-dimensional continuous control tasks.
   - Crucial for real-time applications and optimization.
   - Code-base in C++.
2. **Continuous Time Integration:**
   - *Highlight:* Employs continuous time integration.
   - Suited for stiff systems; allows accurate simulation with larger time steps.
3. **Gradient-Based Optimization:**
   - *Highlight:* Optimized for gradient-based optimization.
   - Supports computation of derivatives, essential for ML, control, and optimization applications.
4. **Soft Constraints:**
   - *Highlight:* Efficiently handles soft constraints.
   - Accurate simulation of contact dynamics and friction, even under fast motion and high force.
5. **Usability and Modeling:**
   - *Highlight:* Offers a well-documented and user-friendly interface.
   - Constraint definitions in native XML allow for modeling closed chains.

Core Ingredients for Bipedal Simulation in MuJoCo

- **Robot Sketch+Drawing:**
  - Generally created using CAD software like Fusion360, Onshape etc.
- **MuJoCo Precompiled Binaries, Library, and Bindings**
  - Setup the MuJoCo environment, ensuring all simulation dependencies are installed.
- **Robot Description File:**
  - Define the robot's structure, joints, and other characteristics using XML.
- **Python Simulation File:**
  - Implement the simulation control, dynamics, and interaction in Python, interfacing with MuJoCo.

Introduction to Kinematic Chains

- Kinematic chains are assemblies of links and joints.
- They can be classified based on their mobility.
- Have a wide range of applications in machinery, robotics, biomechanics, etc.

Types of Joints

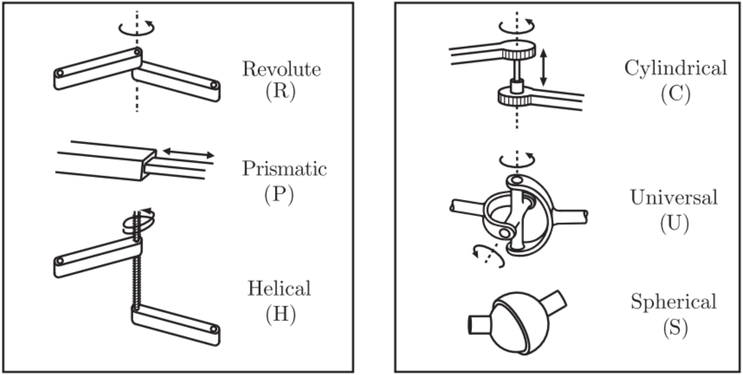


Figure: Types of Robot Joints)

Open and Closed Chains

- Open Chain: Has a free end
- Closed Chain: Forms a loop

Example of Open Chains

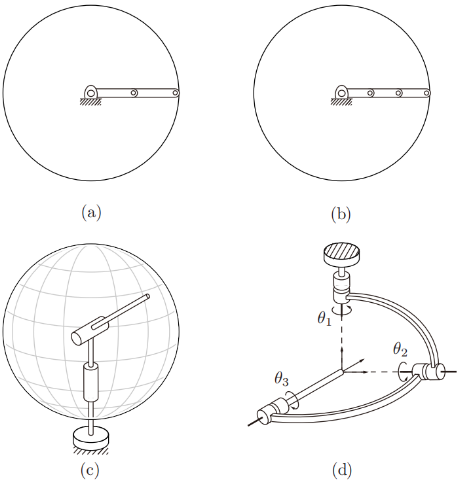


Figure: Open Chain Examples

Example of Closed Chains

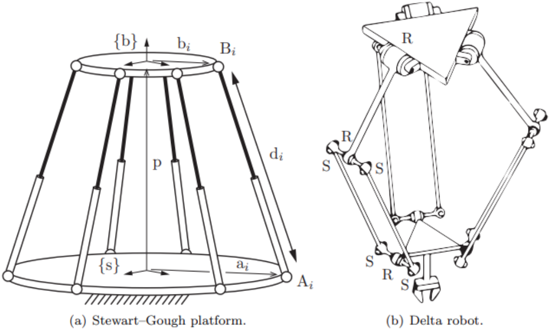


Figure: Closed Chain Examples

Biped as Closed/Open Chain

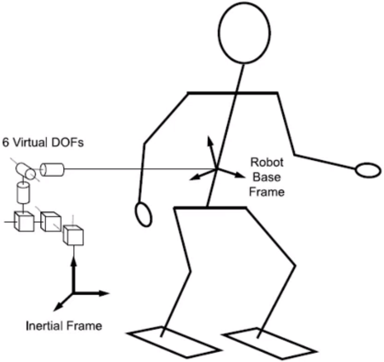


Figure: biped: Open /Closed Chain Example

Aside: Gruebler's Equation

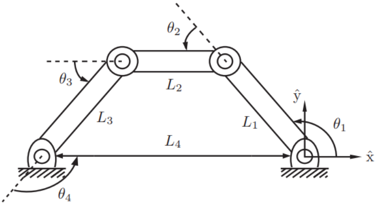
Gruebler's equation is used to determine the degree of freedom (mobility) of a mechanism:

$$M = 3(n - 1) - 2j - h$$

where:

- M: Degree of freedom.
- n: Number of links.
- j: Number of lower pairs (1 degree of freedom).
- h: Number of higher pairs (2 degrees of freedom).

Aside: Loop Closure Equations Example



$$
\begin{aligned}
L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) + \cdots + L_4 \cos(\theta_1 + \cdots + \theta_4) &= 0, \\
L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2) + \cdots + L_4 \sin(\theta_1 + \cdots + \theta_4) &= 0, \\
\theta_1 + \theta_2 + \theta_3 + \theta_4 - 2\pi &= 0.
\end{aligned}
$$

Figure: loop closure constraints

Introduction to URDF

URDF (Unified Robot Description Format):

- XML format for representing a robot model.
- Used in ROS (Robot Operating System) for defining robots, their structures, properties, and how they are connected.
- **Main Elements:**
  - `<robot>`: Root element containing robot metadata.
  - `<link>`: Defines the rigid bodies.
  - `<joint>`: Describes the relationships, movements, and restrictions between links.
- Provides a structured and standardized way to describe the physical configuration of robots.

URDF Tags Overview

1. **`<robot>` Tag:**
   - Root element.
   - Contains metadata, links, joints, etc.
2. **`<link>` Tag:**
   - Represents rigid bodies.
   - Contains `<inertial>`, `<visual>`, and `<collision>` elements.
3. **`<joint>` Tag:**
   - Describes the kinematic and dynamic properties of the connection between links.
   - Types: revolute, continuous, prismatic, fixed, floating, planar.

URDF Tags Overview Continued..

4. `<transmission>` **Tag:**

- Describes the relationship between an actuator and a joint.
- Contains `<type>`, `<joint>`, and `<actuator>` elements.

5. `<sensor>` **Tag:**

- Represents sensors attached to the links.
- Contains `<camera>`, `<ray>`, `<contact>` etc.

Sub-Elements:

- `<inertial>`: Inertial properties.
- `<visual>`: Visual properties.
- `<collision>`: Collision properties.

Sample URDF File

```
<robot name="sample_robot">
  <link name="base_link">
    <inertial>
      <mass value="1.0"/>
      <origin xyz="0\␣0\␣0"/>
    </inertial>
  </link>

  <joint name="base_joint" type="fixed">
    <parent link="base_link"/>
    <child link="link1"/>
    <origin xyz="0␣0␣0.1"/>
  </joint>

  <link name="link1">
    <inertial>
      <mass value="1.0"/>
      <origin xyz="0␣0␣0"/>
    </inertial>
  </link>
</robot>
```

MJCF: An Overview and Its Edge over URDF in MuJoCo

MJCF (MuJoCo XML format):

- Specialized XML format for MuJoCo.
- Focused on efficiency, accuracy, and advanced simulation features.

MJCF vs. URDF in MuJoCo:

- **Native Format:** Tailored for MuJoCo's advanced features and capabilities.
- **Enhanced:** Supports extended functionalities and complex modeling not available in URDF.
- **User-Friendly:** Simplified representation for defining various model components and highly readable.

MJCF Tags Overview

1. <mujoco>
   - Root tag of the MJCF file.
   - Holds the model and can have compiler settings and defaults.
2. <worldbody>
   - Contains the entire physical structure of the model.
   - Includes objects, lights, and cameras.
3. <body>
   - Represents a single rigid or articulated body.
   - Holds joints, sites, and other bodies.
4. <joint>
   - Represents joints in the model, defining their type, axis, and limits.
5. <geom>
   - Defines shapes, sizes, materials, and positions.

Sample MJCF File

```xml
<mujoco model="sample_model">
  <worldbody>
    <body name="body1" pos="0␣0␣0">
      <joint name="joint1" type="hinge" axis="1␣0␣0"/>
      <geom name="geom1" type="box" size="0.1␣0.1␣0.1"/>
    </body>
    <body name="body2" pos="0␣0␣-0.2">
      <joint name="joint2" type="hinge" axis="1␣0␣0"/>
      <geom name="geom2" type="sphere" size="0.1"/>
    </body>
  </worldbody>
</mujoco>
```

MuJoCo: User Responsibility and Units

No Specific Set of Units:

- MuJoCo does not come with a predefined set of units.
- It is agnostic to units like meters, kilograms, seconds, etc.

User's Duty:

- It is crucial for users to ensure the consistency of units throughout the simulation.
- Inconsistent use of units can lead to incorrect simulation results and behaviors.

MuJoCo: Coordinate Frames and the World Frame

Right-Hand Rule:

- Coordinate frames in MuJoCo adhere to the right-hand rule.
- It is essential for defining directions and rotations in 3D space.

The World Frame:

- The World Frame is a crucial reference frame in MuJoCo.
- It serves as the absolute reference for positioning and orienting all other objects in the simulation environment.