Dynamic Modeling and Simulation-based Optimizations

Dr. Yuhao Jiang

Reconfigurable Robotics Laboratory

EPFL, Switzerland

Topics

- Introduction to Dynamic Modeling
  - What is dynamic system
  - What is dynamic modeling
  - Why we need dynamic model in Mechanical Engineering
  - Dynamic modeling in automations and controls

- Introduction to Simulation and Optimization
  - Static and dynamic simulations
  - General methods for system simulations
  - Design and control optimization

- Example

What is Dynamic System?

General defination from Webster Distionary:

- **Dynamic**: A branch of mechanics that deals with forces and their relation primarily to the motion but sometimes also to the equilibrium of bodies;

- **System**: A regularly interacting or interdependent group of items forming a unified whole

Our scope as of robotics and mechanical engineering:

- **Dynamic:** Change of internal physical varaiables over time: force, velocity, pressure, fluid flow rate, current, voltage, temperature, etc.

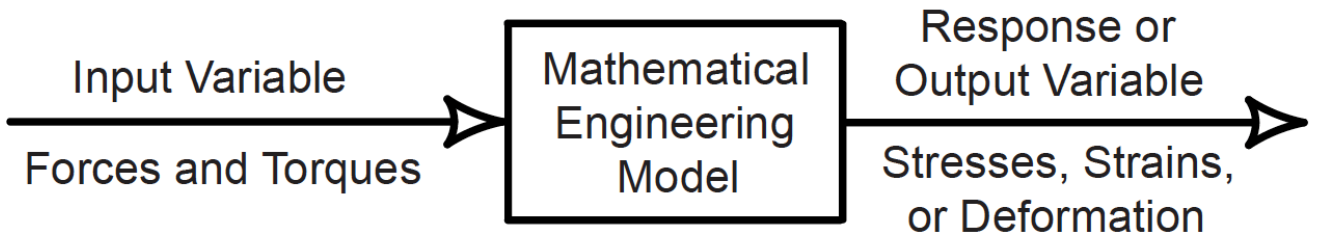
- **System:** robotic systems and the interacting environments (air, water, ground, human bodies, granular, gravity, magnetic, etc.)

Why we need dynamic modeling?

Starship prototype re-entry and landing

ANYmal on Wheels

What is Dynamic Modeling?



What is Dynamic Modeling?

System Identification: the process of building mathematical models of the dynamic system directly from measured data.

- **Data-driven:** relies on actual data;
- **Tool:** General fitting tools, machine learning tools;
- **Example:** Model a DC motor by apply a range of voltages and measure speed response to fit the transfer function.

Dynamic Modeling: the process of building mathematical models of the dynamic system from first principles. (Newton's laws, Kirchhoff's laws, etc.)

- **Physics-based:** equations derived from the physics and engineering principles;
- **Tool:** Transfer functions, state-space models, differential equations, etc.;
- **Example:** Model a DC motor by write down electrical and mechanical equations (voltage, torque, friction, inertia, etc.) and derive transfer function.

How Dynamic Modeling can Help?

Design

- Simulate the designate motion, analyze the workspace, load distribution, verify your design;
- Optimize the design for better performance at lower cost

Control

- Understand the responds from the system;
- Simulate and optimize the control law

Machine Learning

- Simulation, iterate to train the system

General Steps for Dynamic Modeling of Robotic Systems

1. **Define the system**
   Analyze the system's degrees of freedom, types of joints, locations, mass and inertias, end-effector's functions, etc.

2. **Develop kinematic equations: relation between joints and the end-effector**
   Forward kinematic: $\boldsymbol{\chi}_e = \boldsymbol{\chi}_e(\mathbf{q})$.
   Inverse kinematic: $\mathbf{q} = \mathbf{q}(\boldsymbol{\chi}_e^*)$

3. **Develop the equations of motion:** relationship between forces/torques and motion

$$\mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} + \mathbf{J}_c(\mathbf{q})^T\,\mathbf{F}_c$$

4. **Solve the equations in time domain for analytical simulation**

Introduction to Simulation: Static V.S. Dynamic

Goal of Static Simulation

- Structure analysis;
- Stability analysis;
- Design validation

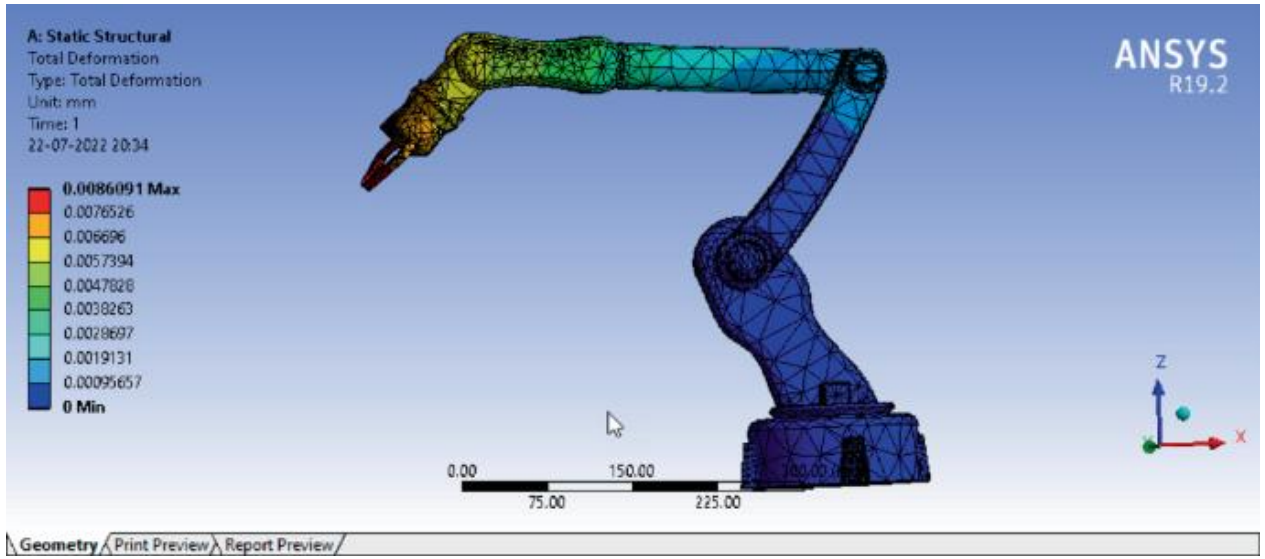
Goal of Dynamic Simulation

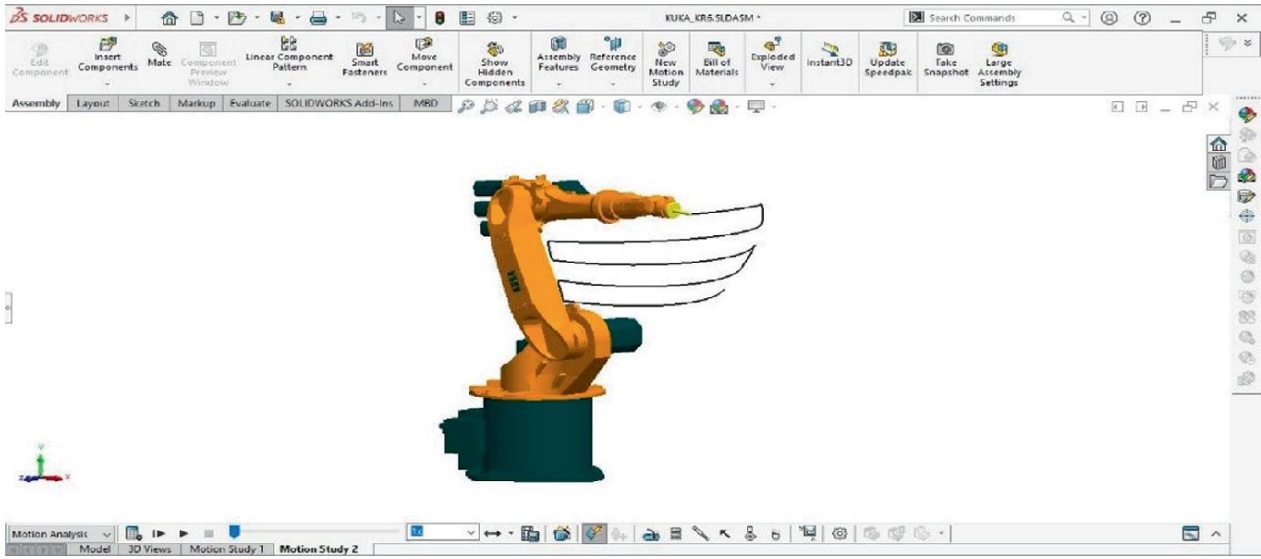
- Motion analysis;
- Control system design and validation;
- Trajectory and workspace planning;
- Optimization;
- Bridge to real-world task

General methods for system simulations

Static Simulation

- **Mathematical Simulation**: solving mathematical models in Python, Matlab, etc.
- **CAD softwares:** Solidworks, Fusion 360, etc.
- **Finite Element Analysis(FEA) Softwares**: ANSYS, COMSOL, Abaqus, PyChrono, etc.





General methods for system simulations

Dynamic Simulation

- **Mathematical Simulation**: solving equations of motion overtime in Python, Matlab Simulink, etc.
  - Pros: Fast, easily applied for simple systems;
  - Cons: Hard to apply on complex, non-linear systems;
- **Finite Element Analysis(FEA), Computational Fluid Dynamic(CFD), Fluid Structure Interaction(FSI) tools:** ANSYS, COMSOL, Abaqus, PyChrono, etc.
  - Pros: Commercial software, reliable, precise, friendly GUI, good for complex systems;
  - Cons: Commercial software, expensive, slow, hard to integrate to other functions.
- **Physics simulating libraries:** MuJoCo, PyBullet, Pynamics, etc.
  - Pros: Fast, acceptably precise, easy to integrate to other code/functions;
  - Cons: Steep learning curve

Simulation-based Optimization

Why optimization?

- **Better performance**
  - Reliable
  - Faster, higher force, more stable, etc.
  - Achieve more complex tasks
- **Save resources**
  - Prototyping time
  - Materiel cost
  - Labor cost
  - Sustainable production
- **Meet constraints**
  - Obey limits: size, cost, safety, environments, etc.
  - Feasibility: manufacturing, material, etc.

Simulation-based Optimization

How to conduct optimization?

- Iterate through the parameters (design, control, etc.) to be optimized and find the best set that yield the best desired result.
- Optimization requires repeatedly modifying and evaluating the system to achieve the best results.
- When using a prototyping-based approach, this process can be less time-efficient, less cost-effective, and less sustainable.

In complex systems, prototyping-based optimization is impossible. (think about rocket optimizations)

Simulation-based Optimization

Why Use Simulation-Based Optimization?

- **Complex Systems:** Simulation allows us to evaluate systems with nonlinear behavior, dynamic effects, and intricate interactions—situations where analytical models are unavailable or impractical.

- **Lower Cost:** By enabling virtual prototyping, simulation reduces the need for expensive and time-consuming physical prototypes and experiments.

- **Broad Applicability:** Simulations can model a wide range of scenarios and operating conditions, making it possible to search for solutions that are robust and optimal across diverse situations.

- **Efficiency:** Simulation-based optimization automates the exploration of design alternatives, systematically searching for the best solutions with minimal manual effort.

Simulation-based Optimization

Cons?

- **Sim to real gap:** Simulations cannot fully capture all aspects of real-world systems, which can lead to discrepancies between simulated and actual performance.
  - **Solution:** Calibrate simulation models using experimental or prototype data.

- **Problem complexity and computational demand:** Some systems are difficult to model accurately or require complex simulations that are computationally intensive. Optimization in such cases may be slow to converge or even impractical.
  - For especially challenging or high-fidelity problems, prototype-based (hardware-in-the-loop) optimization may be more effective.

Introduction to MuJoCo

MuJoCo: Multi-**Jo**int dynamics with **Co**ntact, a physics simulator developed by Google Deep mind

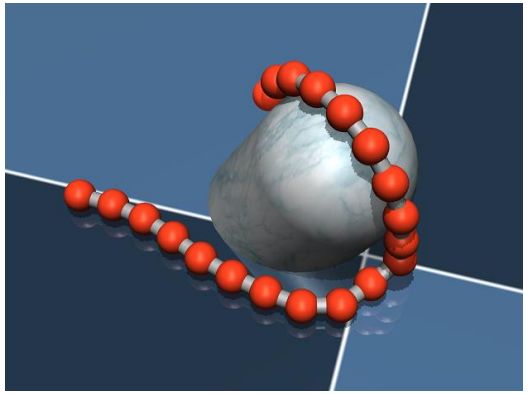
- C/C++ library with a C API;
- Python bindings;
- Unity plug-in
- GPU computation

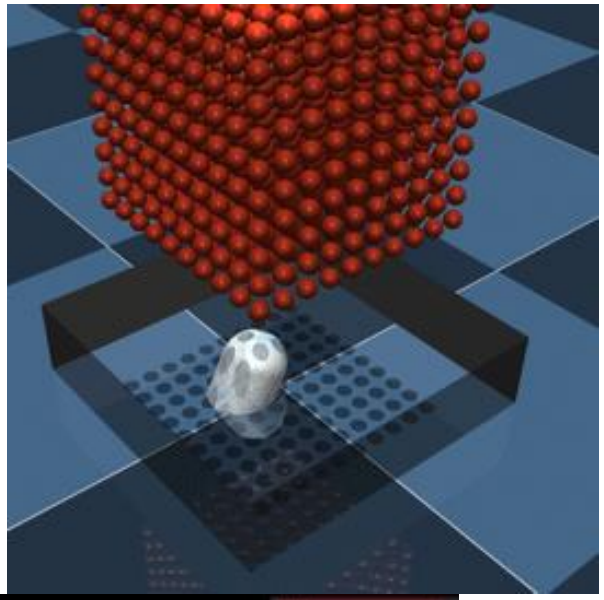
Application:

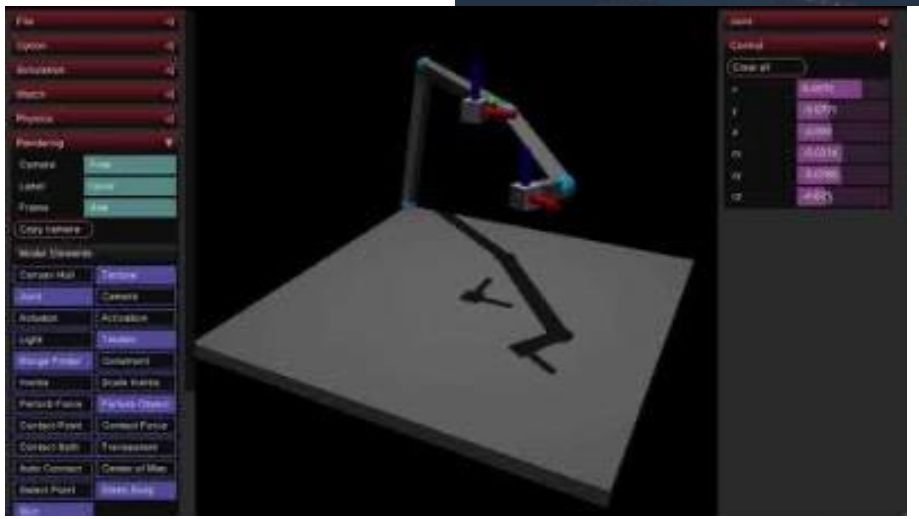
- Model-based computations such as control synthesis, state estimation, system identification, mechanism design, data analysis through inverse dynamics, and parallel sampling for machine learning applications.
- Traditional simulator, including for gaming and interactive virtual environments.

Key Features of MuJoCo

- **General actuation model**
  - Motors
  - Pneumatic and hydraulic cylinders,
  - PD controllers
  - Biological muscles
- **Soft, convex and analytically-invertible contact dynamics**
  - Interactions with various environments: ground, water, granular, etc.
- **Tendon geometry**
  - minimum-path-length strings obeying wrapping and via-point constraints
- **Reconfigurable computation pipeline**
  - Reconfigure your simulation on the fly using build-in flags
- **Interactive simulation and visualization**
  - 3D visualizer, easy for debugging and modeling







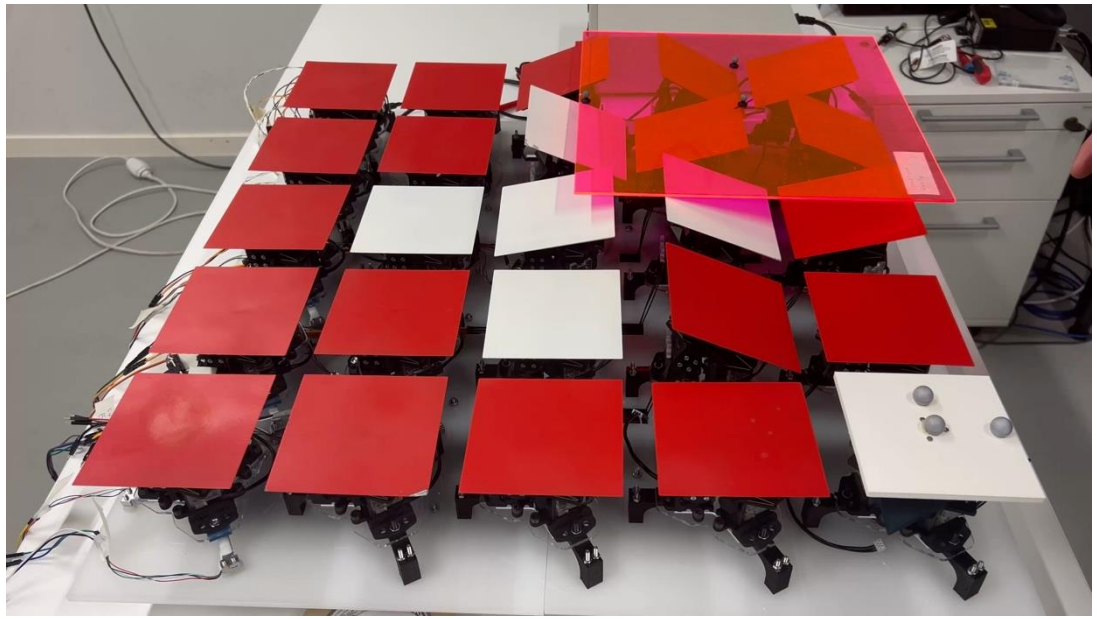
Example: Control parameters optimization

Problem

- Complex control and motion planning for multi-module origami robot surface (Ori-pixel, 5X5 modules with 75 servos) due to high DoF and actuator coordination.

Solution

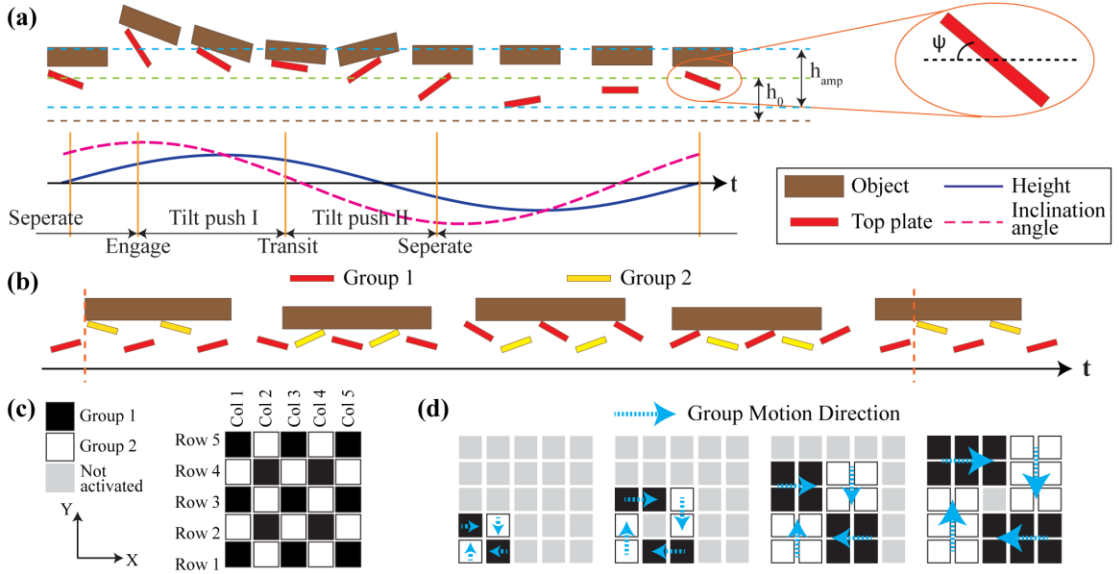
- Central Pattern Generator (CPG)-based motion generator
  - Reduces motion parameters.
  - Adapts to different system configurations.
- A simulation-based optimization framework tunes the motion parameters for specific tasks.



Example: Control gait optimization

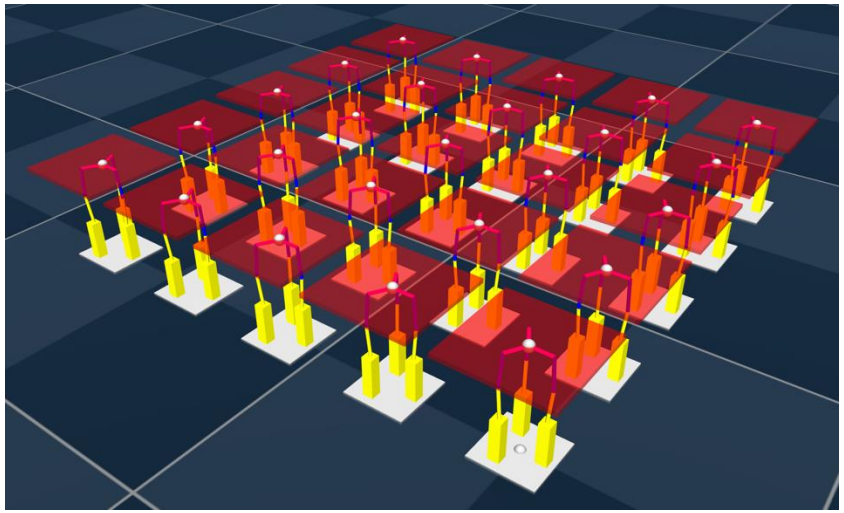
Objective:
Find optimal parameter set in formulas below to achieve fastest object moving speed

$$H(t) = h_{\text{amp}} \sin(2\pi f \cdot t + \phi) + h_0$$
$$\psi(t) = \psi_{\text{amp}} \sin(2\pi f \cdot t + \phi + \sigma) + \psi_0$$



Search Space:

Parameter	Symbol	Search Space	Unit
Height amplitude	h_{amp}	$[0.005, 0.04]$	m
Inclination angle amplitude	ψ_{amp}	$[0.35, 0.79]$	radian
Frequency	f	$[0.1, 0.8]$	Hz
Resting height	h_0	$[0.02, 0.04]$	m
Resting inclination angle	ψ_0	$[-0.26, 0.26]$	radian
Height-inclination phase shift	ϕ	$[0, \pi]$ or $[\pi, 2\pi]$	radian
Inter-group phase shift	δ	$[0, 2\pi]$	radian
Tile contact threshold	ϵ	$[0.1, 0.5]$	-



Example: Control gait optimization

Work flow:

1. Optimizer suggests parameters
2. Run Mujoco simulation with suggested parameters
3. Evaluate speed of the object
4. Send result (speed) back to optimizer
5. Converge?
(If not, repeat from step 1)

Example: Control parameters optimization

Manipulation in different modes

A. Fast omni-directional translational manipulations

Object: 300x300 mm Acrylic Plate

Manipulation of different objects

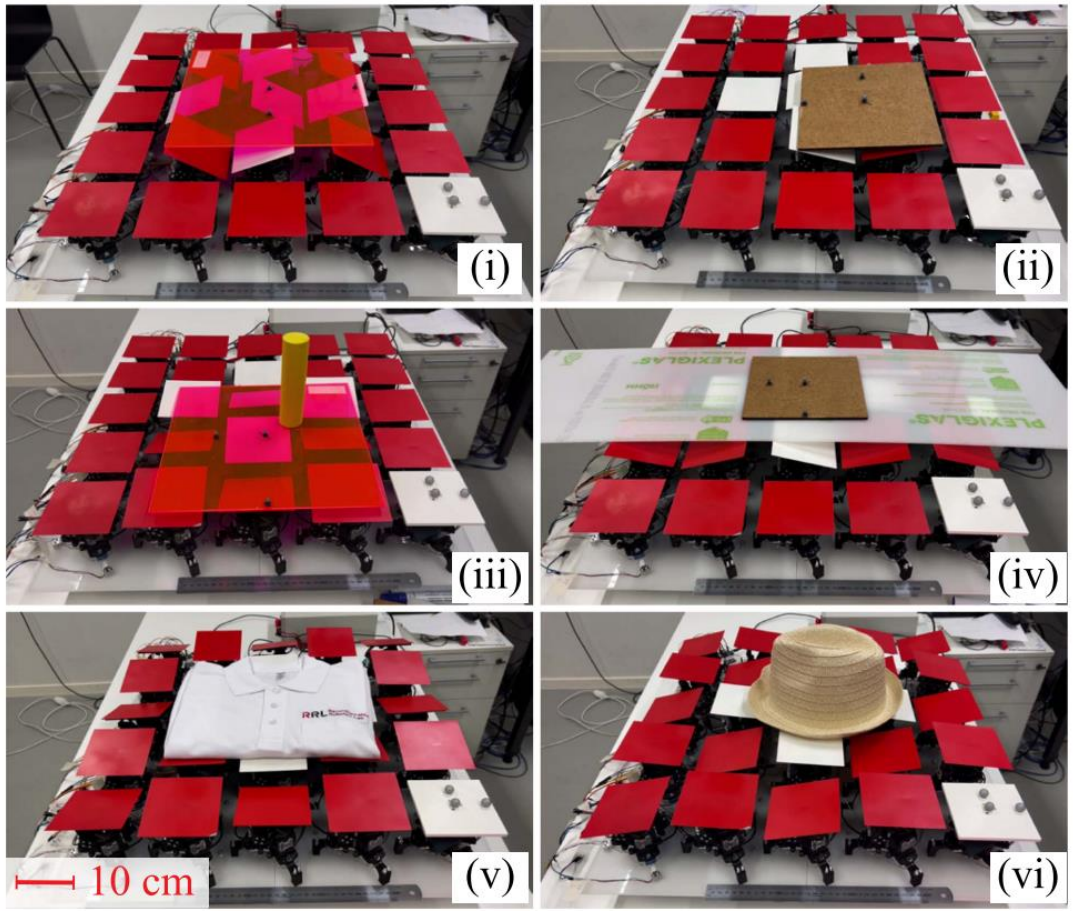
A. Large Object manipulation I

Object: 800x300 mm Foam Plate

Example Resources

Code Repo: https://github.com/Reconfigurable-Robotics-Lab/Simulation_CPG_Manipulation_Oripixel

Paper: https://ieeexplore.ieee.org/document/10943123



Learning Resources

- MuJoCo official document: https://mujoco.readthedocs.io/en/latest/overview.html
- Online course: https://pab47.github.io/mujoco.html
- Robot dynamics and simulation Lecture Notes, Allison Okamura, Stanford University: https://web.stanford.edu/class/me328/lectures/lecture5-dynamics.pdf
- Robot Dynamics Lecture Notes, Robotic Systems Lab, ETH Zurich: https://ethz.ch/content/dam/ethz/special-interest/mavt/robotics-n-intelligent-systems/rsl-dam/documents/RobotDynamics2017/RD_HS2017script.pdf
- Handbook of Robotics: https://link.springer.com/book/10.1007/978-3-540-30301-5
- Robotics Modelling, Planning and Control: https://link.springer.com/book/10.1007/978-1-84628-642-1

Thank you!