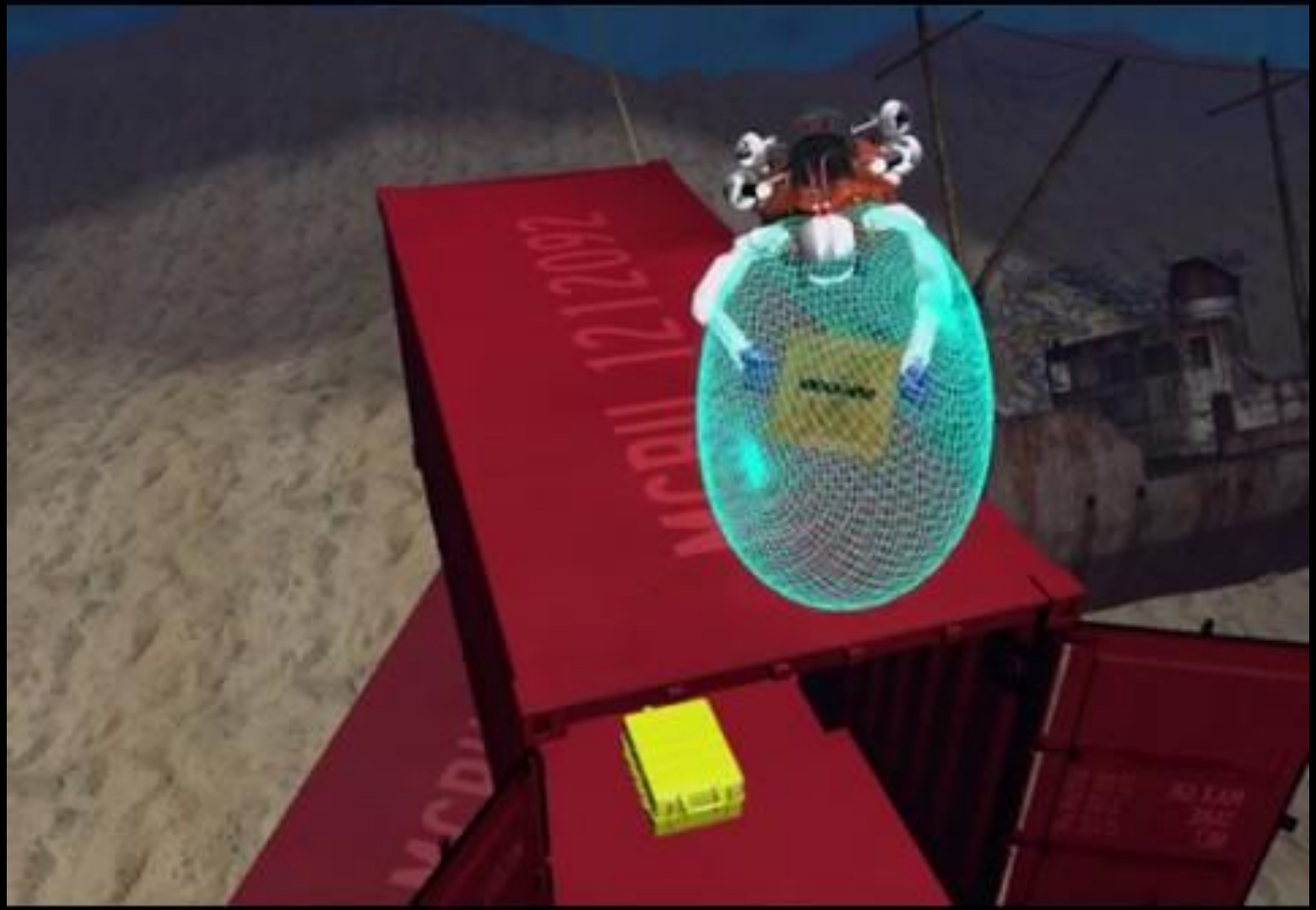
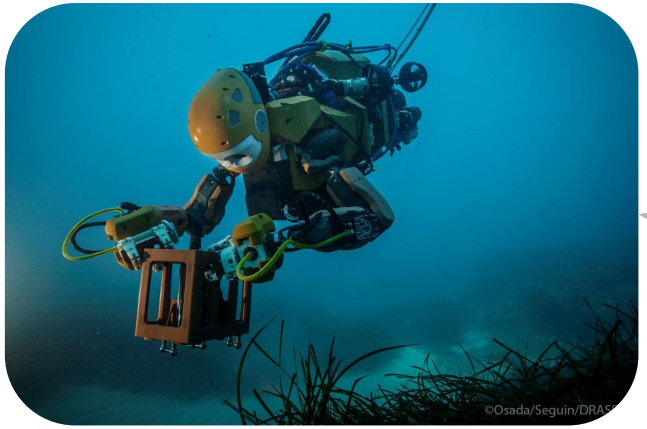
SAI 2.0 Environment (Simulation and Graphics)

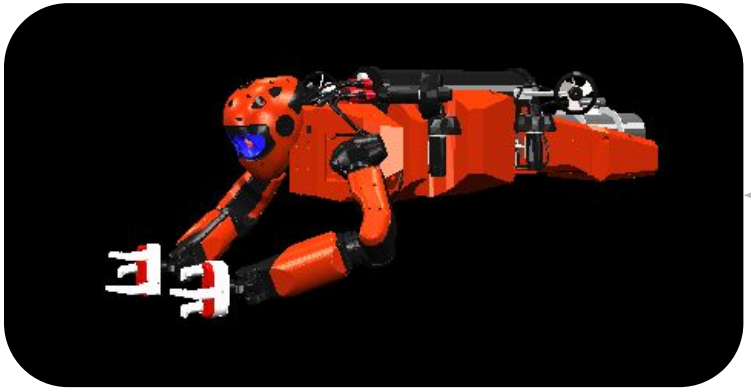
William Chong
Adrian Piedra

CS225A, Spring 2021



Introduction

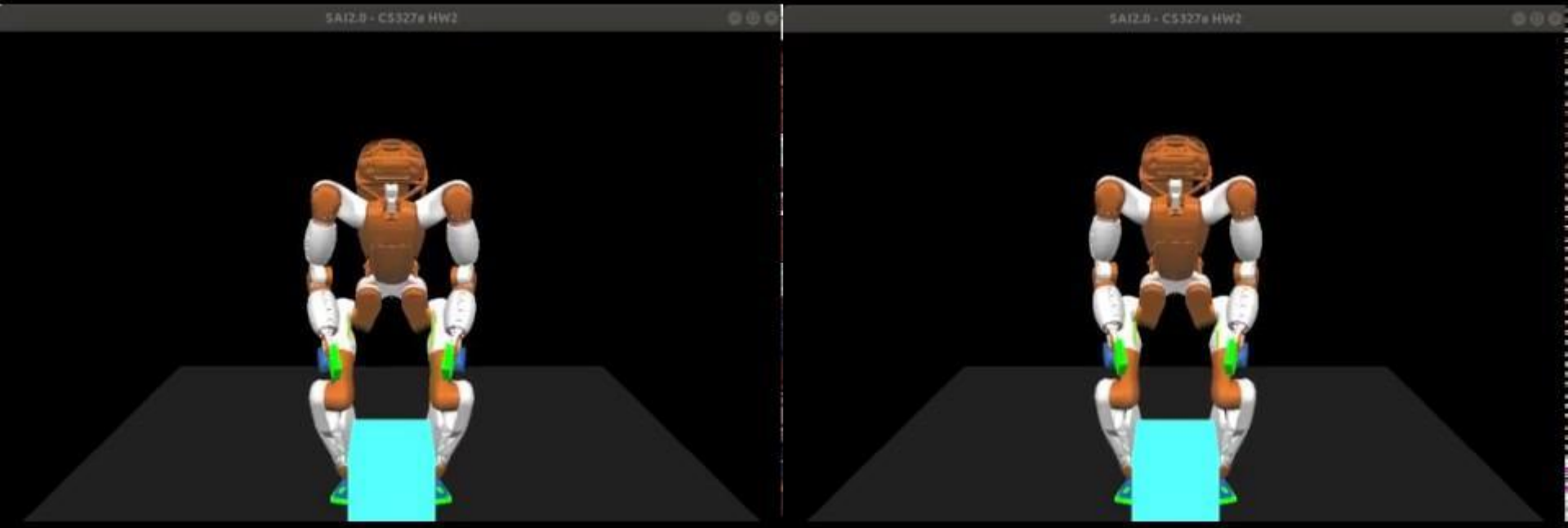




Benefits of Simulation

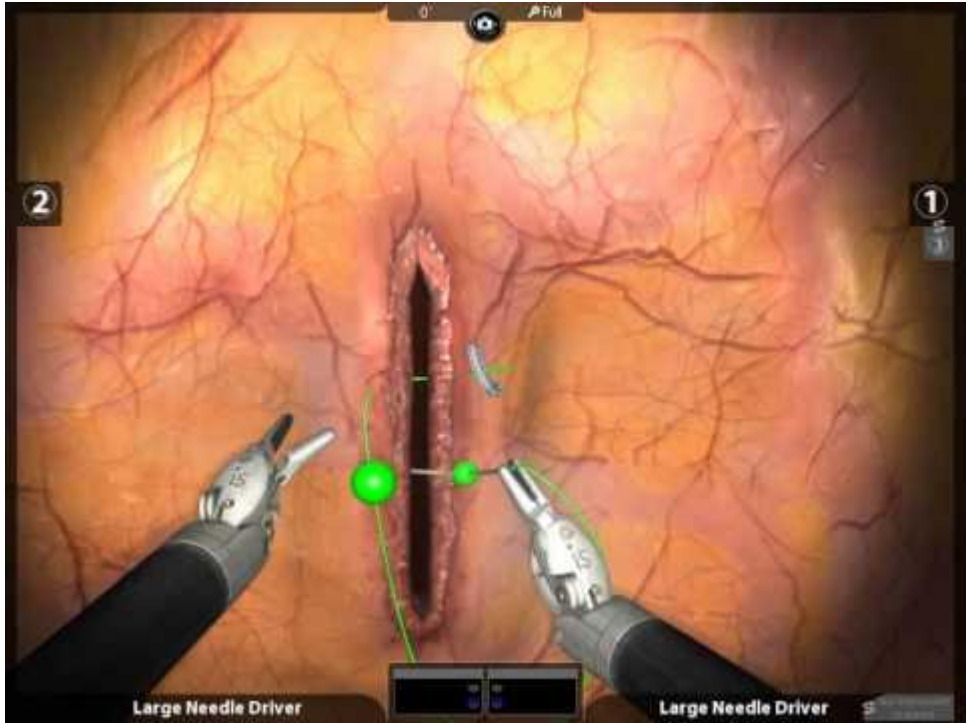
- Cheap and fast testing of control algorithm (robustness to noise, model errors)
- System analysis, design
- Motion planning, online estimation
- Training (human or AI)

Debug Controller for Free

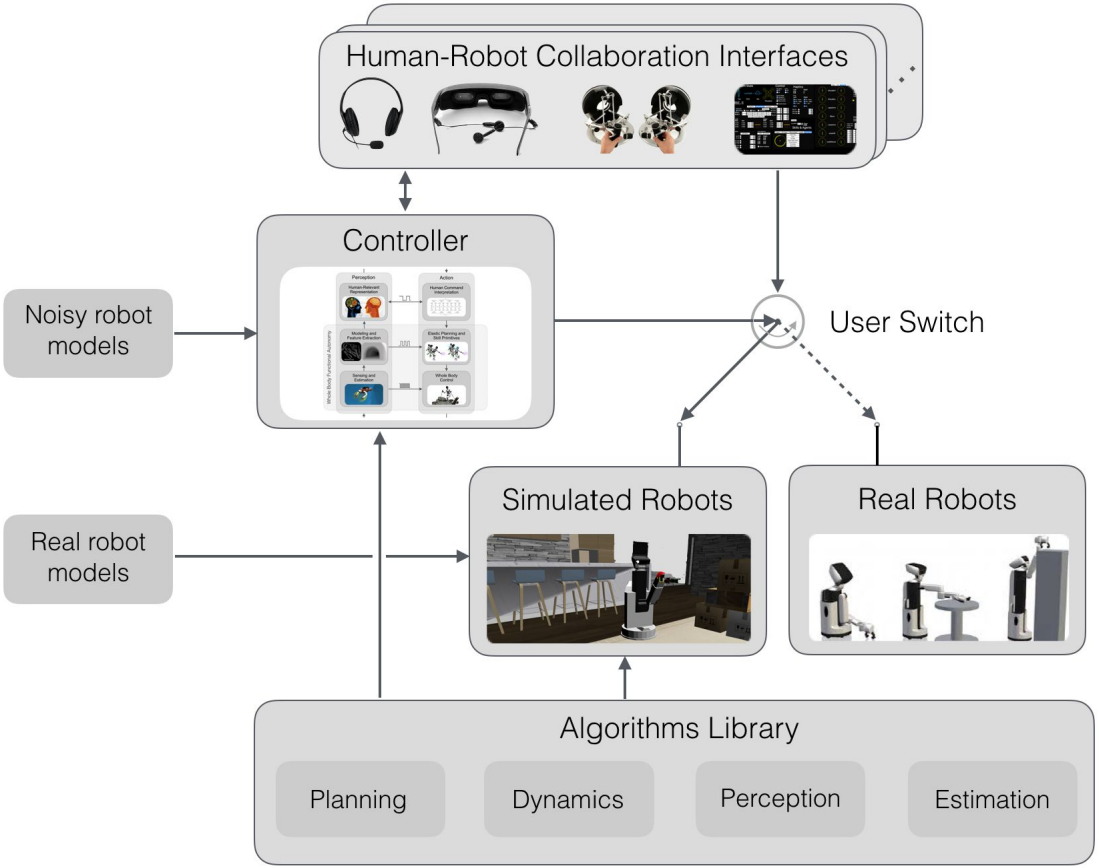


Training

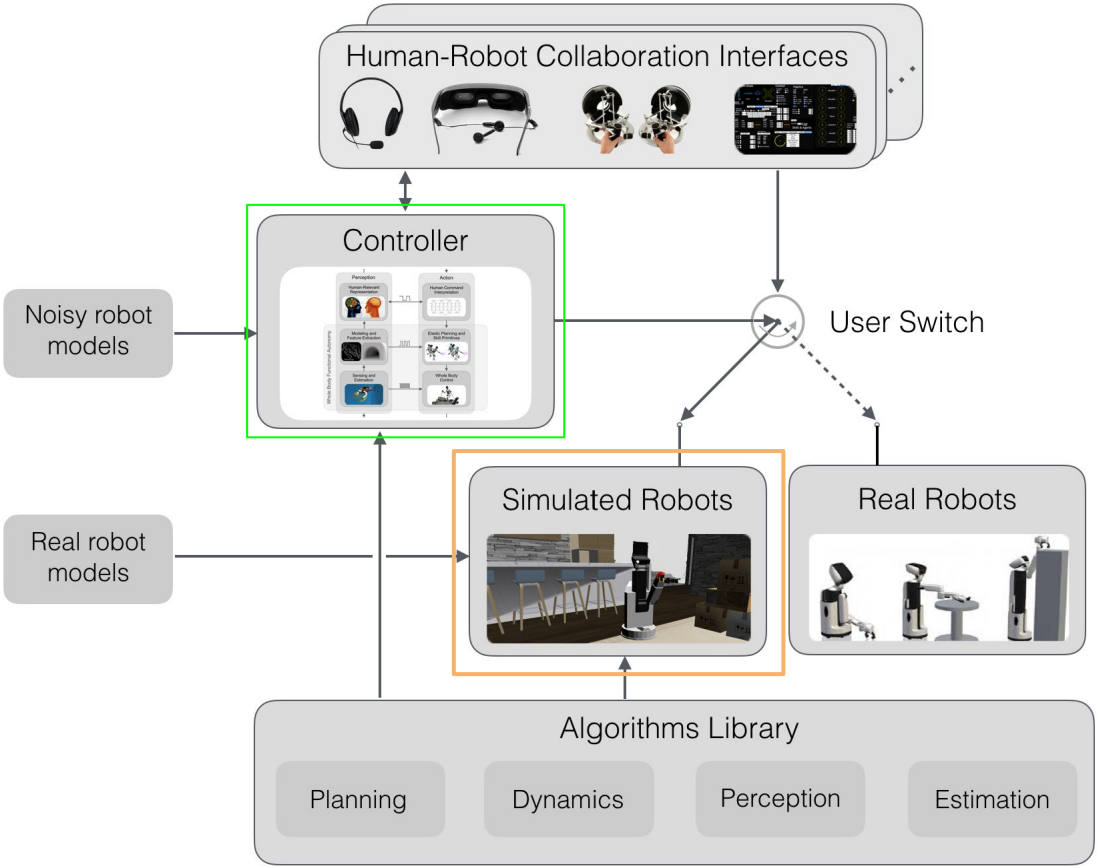
- Training for user operation of robot
- Can allow for haptic input
- Can run indefinitely, no permanent damage



SAI 2.0 (Simulation and Active Interfaces)



SAI 2.0 (Simulation and Active Interfaces)



SAI 2.0 Core Modules

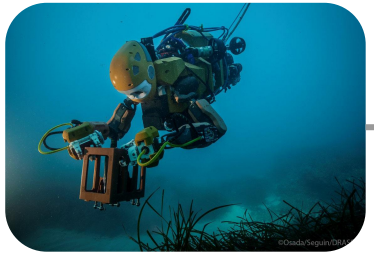
- sai2-urdfreader: Robot + world model specification
- sai2-model: Articulated Rigid Body Modeling - Kinematics + Dynamics
- sai2-simulation: Physics engine
- sai2-graphics: Scene rendering, visualization
- sai2-common: helper functions (filters, force sensors, Redis modules)
- sai2-primitives: high-level interface for low-level control specification

Your application will use these modules

Demo

- Let us simulate a pendulum (rbot) and control it to oscillate around an off centered position
- Code will be provided after class
- To run the demo, compile the code and go to bin/lecture2-demo and run the programs
  - ./visualization
  - ./simviz
  - ./controller

SAI 2.0 Framework



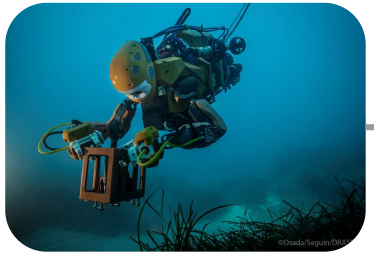
Parser

Kinematics & dynamics

Graphics

Interfacing

Robot control



Parser

Kinematics & dynamics

Graphics

Interfacing

Robot control

Parser

- Syntax: http://wiki.ros.org/urdf/XML
- Interprets dynamic model of robot + environment
- Reads robot kinematic data
  - Assumes chain/tree structure (i.e., no cycles)
  - Link lengths
  - Joint types
- Reads robot inertia parameters
  - Mass
  - Location of center of mass
  - Rotational inertia parameters

Parser: Links

```
<link name="Body">
    <inertial>
      <origin xyz="-0.034 -0.001 0.134" rpy="0.0 0.0 0.0" />
      <mass value="150.8" />
      <inertia ixx="0.291" iyy="0.458" izz="0.369" ixy="0.0" ixz="0.0" iyz="0.0"
/>
    </inertial>
    <visual>
      <origin xyz="0.095 0.0 -0.03" rpy="0.0 0.0 0.0" />
      <geometry>
      <mesh filename="../../../model/ocean1/ocean1_graphics/O2_body.obj"/>
      </geometry>
    </visual>
</link>
```

Parser: Joints

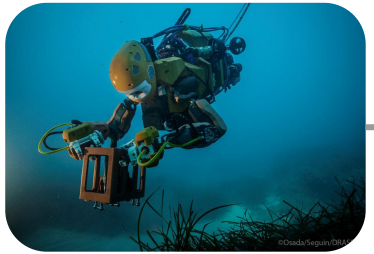
```
<joint name="shoulderT_right_Joint" type="revolute">
    <origin rpy="3.141 0.0 -0.785" xyz="0.577 -0.195 -0.189"/>
    <parent link="Body" />
    <child link="shoulderT_right" />
    <axis xyz="1.0 0.0 0.0" />
    <calibration falling="-22.452"/>
    <limit effort="85.0" velocity="3.0"/>
</joint>
```

Parser: World

```
<world name="demo_world" gravity="0.0 0.0 -9.81">
    <robot name="RBot">
        <model dir="./resources" path="rbot.urdf" name="rbot" />
    </robot>
    <light name="light1" type="directional">
        <position xyz="2.0 2.0 2.0" />
        <lookat xyz="0.0 0.0 0.0" />
    </light>
    <camera name="camera_fixed">
        <position xyz="2.0 0.0 -0.5" />
        <vertical xyz="0.0 0.0 1.0" />
        <lookat xyz="0.0 0.0 -0.3" />
    </light>
</world>
```

SAI 2.0 Robot Models

- Framework is flexible and allows for any robots and any worlds
  - Specified through URDF (XML) file
- Robot kinematics and dynamics information provided in sai2-model
  - Encourage you to read header files and study demo
  - Most of the functions you need for your controllers are in sai2-model/src/Sai2Model.h



Parser

Kinematics & dynamics

Graphics

Interfacing

Robot control

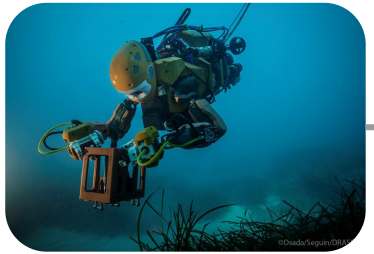
Kinematics and Dynamics

$$\ddot{q} = A^{-1}(q)\left(\Gamma - b(q,\dot{q}) - g(q)\right)$$

- Rigid Body Dynamics Library for kinematics & dynamics
- Closed-source software for multi-articulated body collision detection & contact resolution

Sai2Model (Useful Functions)

- `auto robot = new Sai2Model::Sai2Model (...);`
- `robot->updateModel();`
- `robot->gravityVector(...);`
- `robot->J_0(...);`
- `robot->Jv(...);`
- `robot->Jw(...);`
- `robot->position(...);`
- `robot->rotation(...);`



Parser

Kinematics & dynamics

Graphics

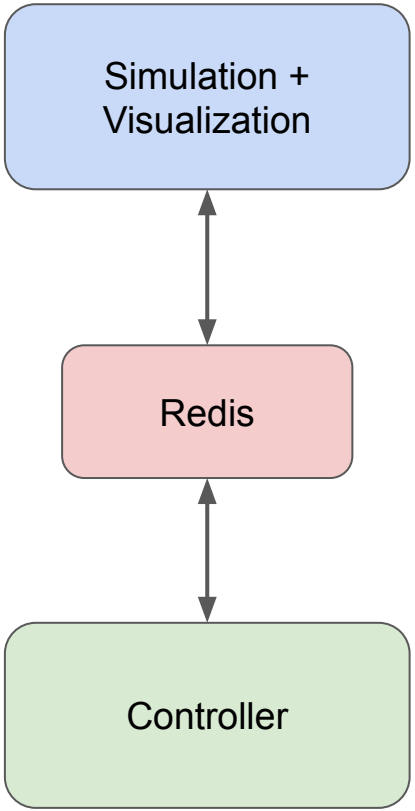
Interfacing

Robot control

Graphics, Control, and Interfacing

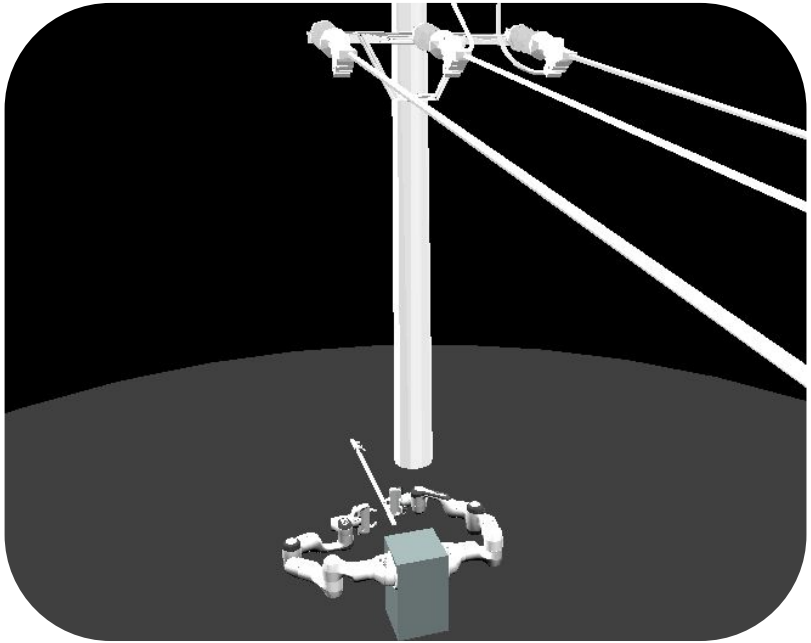
Executable Architecture

- Two separate applications
  - simviz, controller
  - Operating independently
  - Can run on separate computers
- Robot state can be shared through Redis



Simulation + Visualization (simviz)

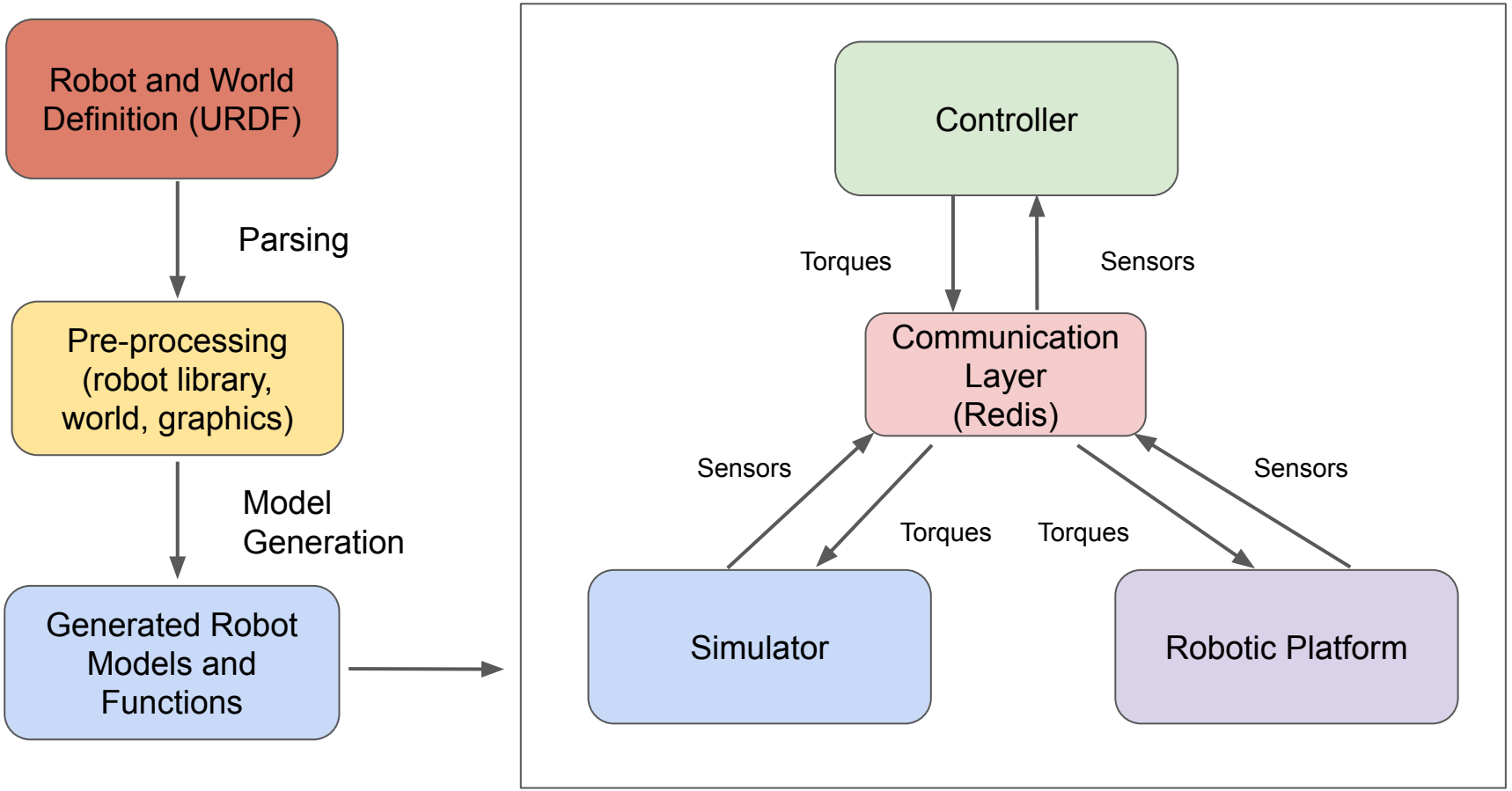
- Two threads
  - Simulation: Discrete physics integration, resolves contacts
  - Visualization: Displays the simulated world from a virtual camera point of view
- Can be replaced by real world robot + driver



Controller

- Reads in robot sensor values (q, dq) and publishes output torques
- Needs to know tasks/jacobians, positional information, mass and coupling information for feedback linearization (unit mass decoupling)
- Needs to know how to control robot
  - Joint space, operational space, null space control
- Most of your code will be in here

$$\Gamma_{command} = \hat{M}(q)(-K_p(q - q_d) - K_v(\dot{q} - \dot{q}_d)) + \hat{V}(q, \dot{q}) + \hat{G}(q)$$


Robot and World Definition (URDF)
Parsing
Pre-processing (robot library, world, graphics)
Model Generation
Generated Robot Models and Functions
Controller
Torques
Sensors
Communication Layer (Redis)
Sensors
Sensors
Torques
Torques
Simulator
Robotic Platform

Controller Servo Loop

1. Read goal/task
2. Read sensors
3. Compute forward kinematics, jacobians, velocities, forces
    a. Interested in controlling a point/frame on robot
4. Compute control law
5. Check safety and validity of computed torques
6. Check goal/task completion
7. Write torques

Redis (Key-Value Database)

- redis-server
  - Once launched, will run in the background
  - Can choose port to run on (6379 by default): redis-server --port 6379
- redis-cli
  - List all keys: `keys *`
  - Set key value: `set <key> <val>`
  - Get key value: `get <key>`
  - Monitor transactions: `monitor`
  - Delete key: `del <key>`
  - Delete all keys: `flushall`
  - Can interact over the network: `redis-cli -h <ip address> -p <port>`

Pre-requisites for SAI 2.0

- Ubuntu or MAC
- Know the basics of UNIX command line
  - Commands like "mkdir, cd, cp, mv, rm" will be used
  - Remember there is auto-completion with Tab key
- Have a text editor to write your code (e.g. sublime-text)
  - In sublime, the package control is useful and contains helpful packages
- Know the basics of C++
- Familiarize yourself with the Eigen library (documentation online, a lot of answered questions on stack overflow)

Summary

- Some core modules for sai2 are provided (put them in core folder)
- In an application folder you will put your course repository. You will work on this repository
- Your applications will have 2 programs
  - simviz (provided)
  - controller (that you will need to write in most cases)
- Code in C++, compile using cmake, and communicate between programs using Redis

SAI 2.0 Installation

- We will host an installation session tomorrow (Zoom OH Room, 1-4 pm)
- Windows users, please look into adding Ubuntu to your system with the following options:
  - Dual boot (install Ubuntu on a partition of your hard drive)
  - Virtual machine (one example): https://www.lifewire.com/install-ubuntu-linux-windows-10-steps-2202108