Simulation and Graphics

CS225A

Robert Sun, Toki Migimatsu
Spring 2017

Cool simulation video first

Why Simulate?

Design

Test Model Scenarios

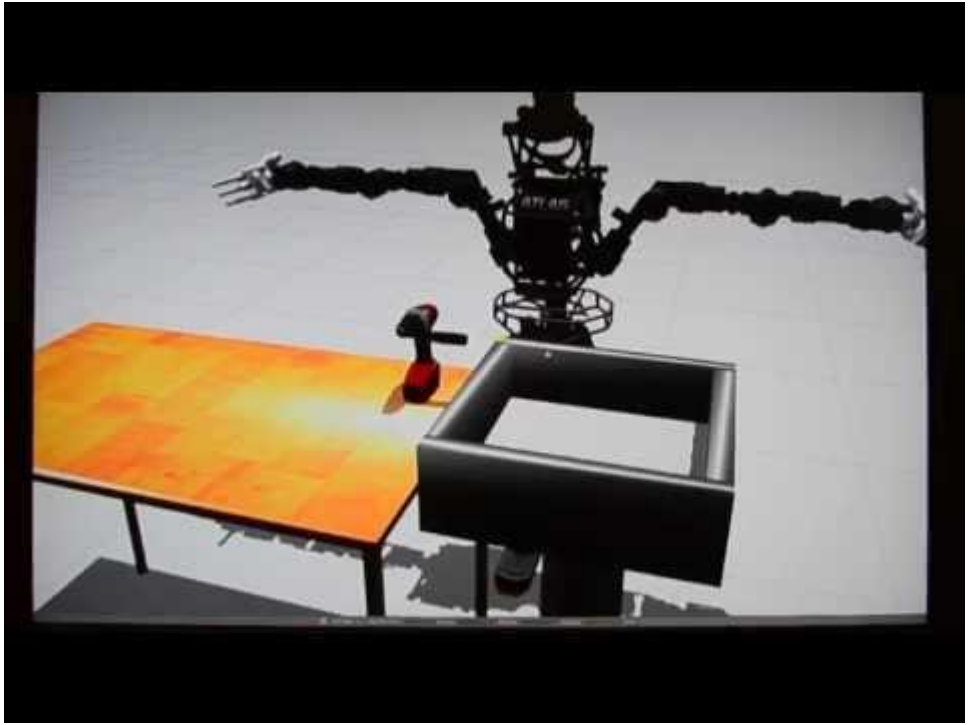
Test Robustness

Faster than real life, cheaper

Motion Planning, online estimation

As an educational tool, adds additional introspection, cases/visualization not possible in real life

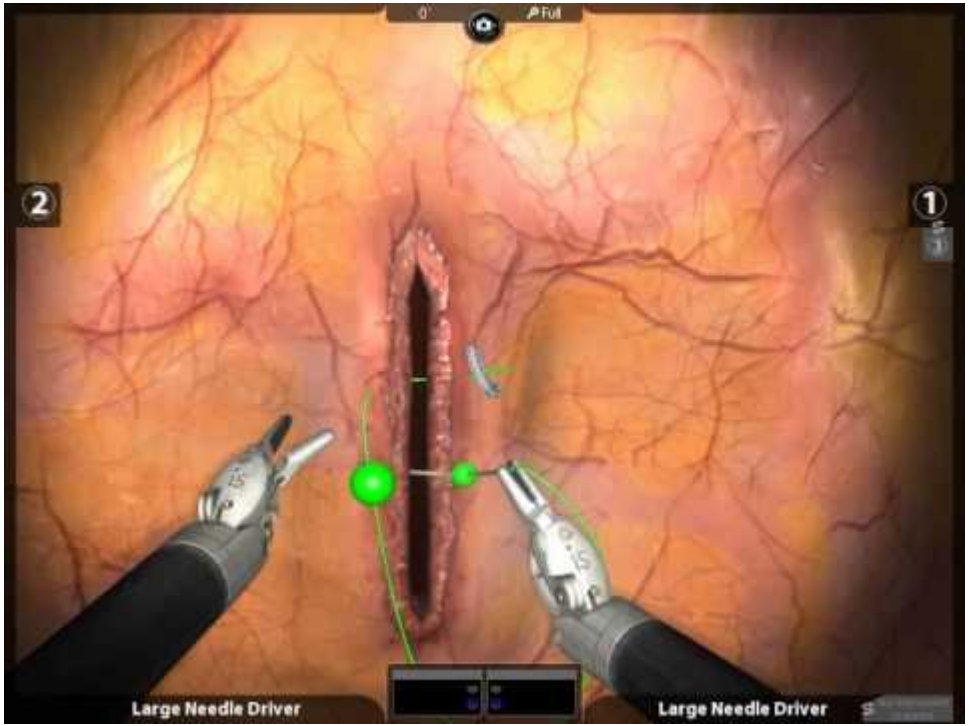
Gazebo simulator in DARPA challenge



Used for testing algorithms, teleop strategies

1st round of DARPA qualifiers in simulation only

Intuitive Training

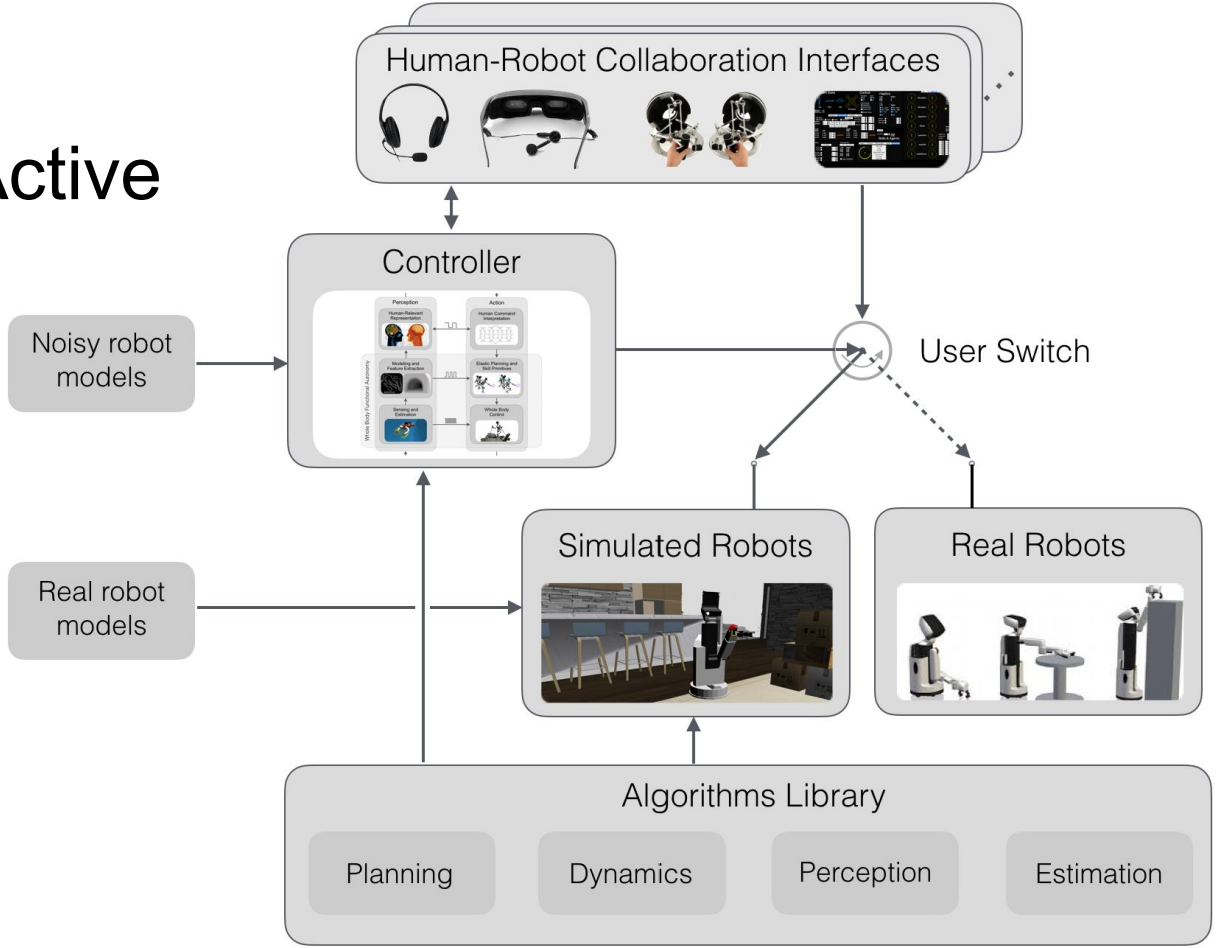


Training for user operation of robot

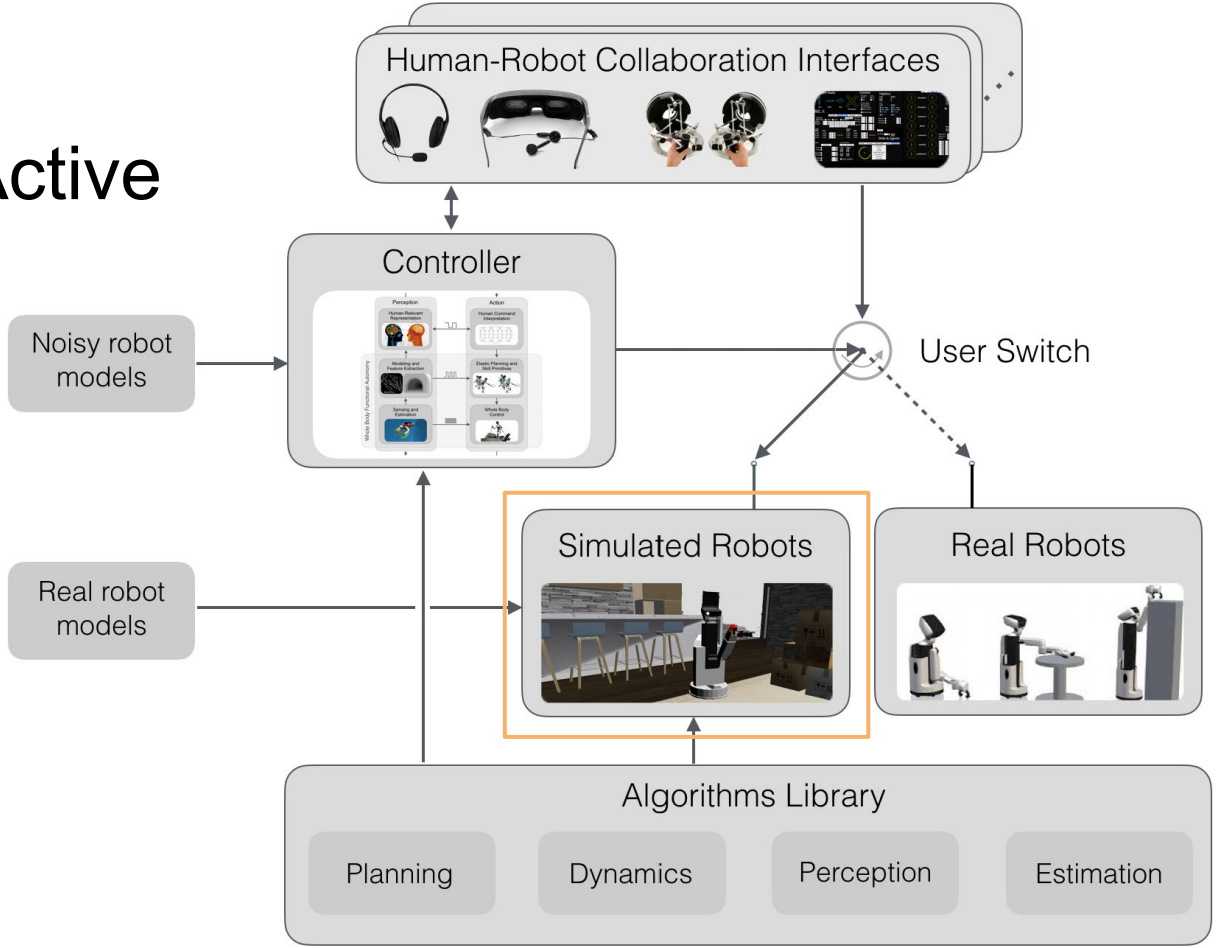
Can allow for haptic input

Can run indefinitely, no permanent damage...

SAI2 (Simulation and Active Interfaces)



SAI2 (Simulation and Active Interfaces)



Simulation Components

Articulated Rigid Body Modeling - Kinematics

Geometric Processing - Represent of objects and their relationships with point clouds, polygons, etc.

Dynamics - physics integration, collision resolution

Graphics - visualization, introspection

Demo sai2-common Pbot

Go over cpp code

URDF

XML specifying robot parameters, simulation parameters, graphics parameters

Syntax spec: http://wiki.ros.org/urdf/XML

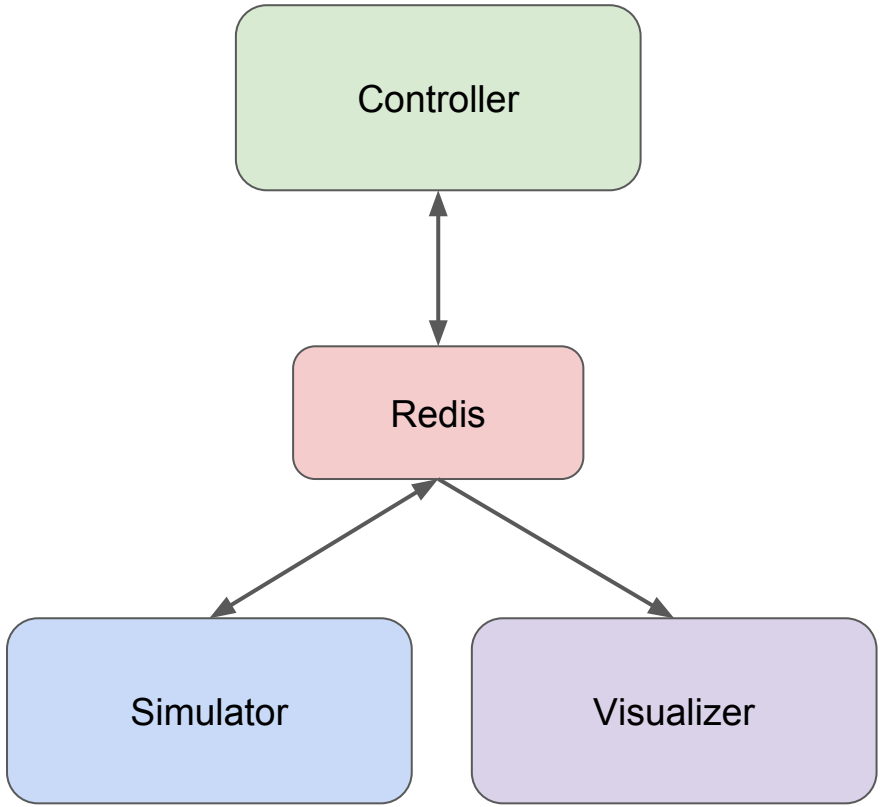
Show pbot example in sai2-common

CS225A Architecture

3 Separate Applications

- Controller, Simulator, Visualizer
- Operating independently
- Can run on separate computers

Robot state is shared through Redis



Demo KUKA Position Controller

SAI2 Robots

Framework is flexible and allows for any robots and any worlds

- Specified through URDF (XML) file

Robot kinematics information provided in `Model::ModelInterface`

- Encourage you to read header files and study hw0 code well
- Will definitely help you for the final project
- Most of the functions you need for your controllers are in `sai2-common/src/model/ModelInterface.h`

Go over ModelInterface headerfile

Controller

Reads in robot sensor values (q, dq), and publishes output torques

- Needs to know tasks/jacobians, positional information, outside data, mass and coupling information for feedback linearization (unit mass decoupling).
  - Small aside on feedback linearization and “b” component?
- Needs to know how to *control* robot
- Joint space, op space, null space control
- Most of your code will be in here

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

Simulator

Takes current state (q, dq) and commanded torques, and performs a physics integration to next time step. Outputs next (q, dq)

- Does so on a very steady rate, can introduce simulation errors
- Might add or remove energy from system if numerical integration is poor

Can resolve contact forces or external forces

Physical robot + nature replaces simulator in real life

- Robot drivers take in torque commands and output sensor (q, dq) values

$$M(q)\ddot{q} + V(q, \dot{q}) + G(q) = \Gamma_{command}$$

Visualizer

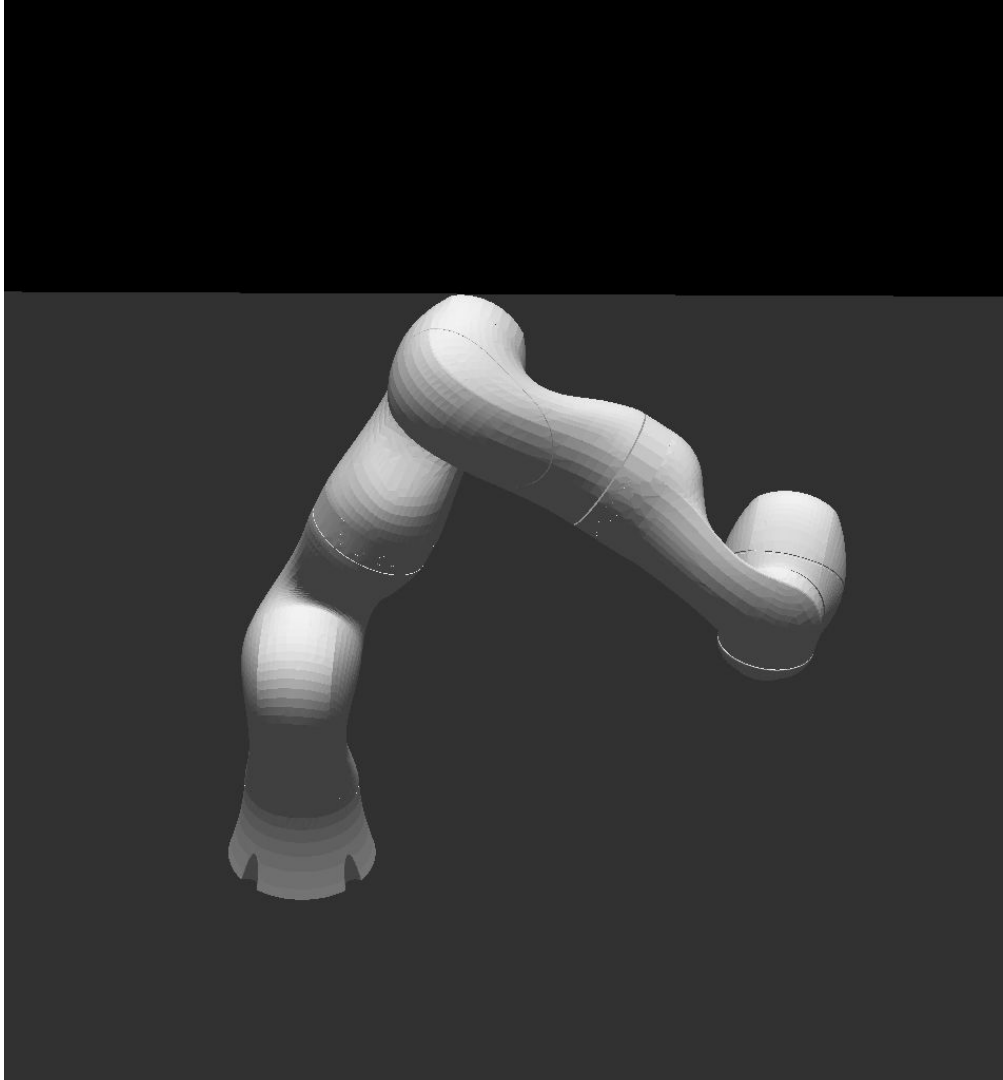
Displays robot in the environment

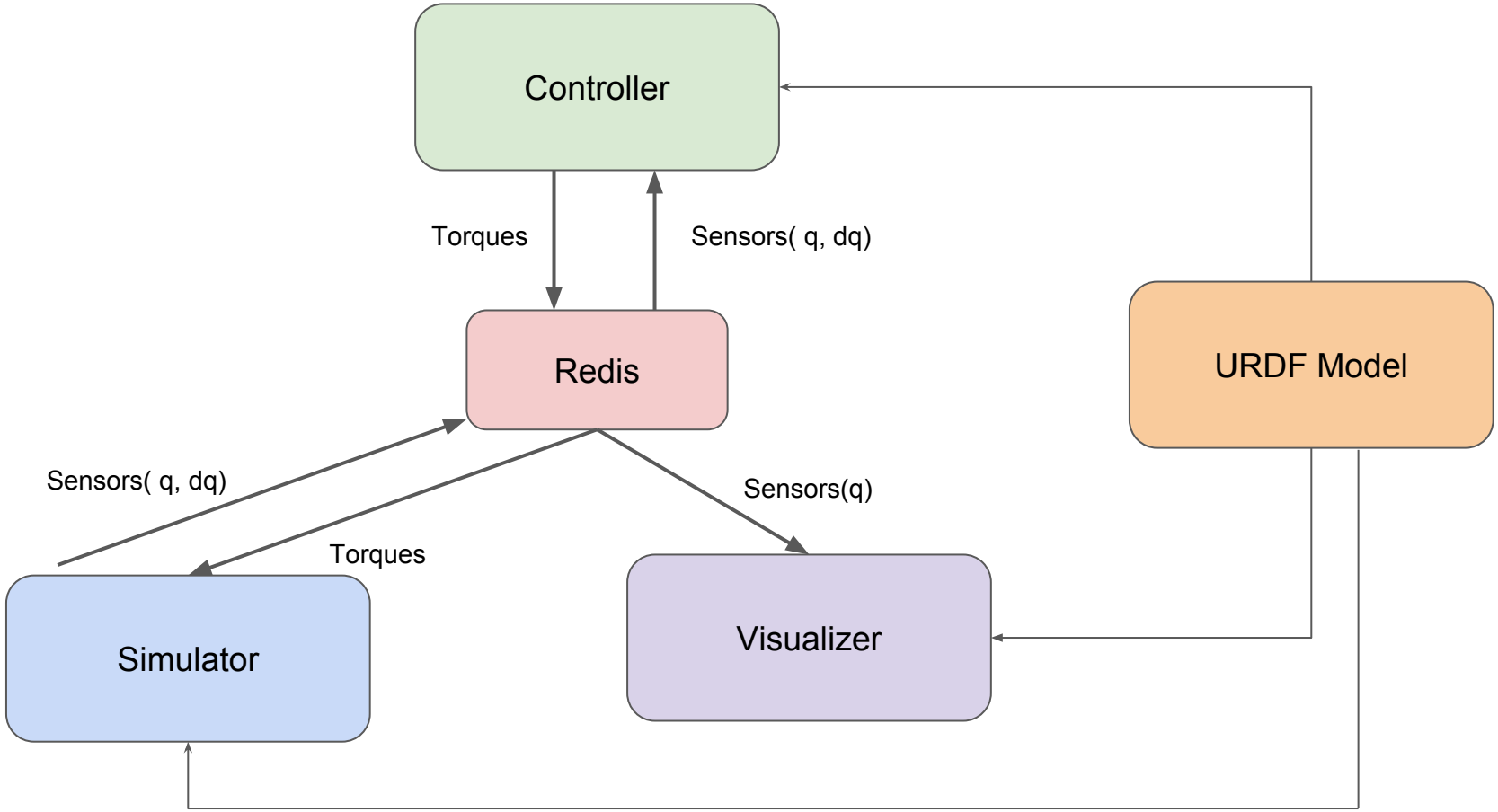
- Reads joint positions (q)
- Updates with specified rate

Lights, cameras, meshes

- In world or robot urdf file

Can apply external forces through interaction




Controller
Torques
Sensors(q, dq)
Redis
URDF Model
Sensors(q, dq)
Sensors(q)
Torques
Simulator
Visualizer

Redis

Key-Value Database:

- redis-server
  - Runs as daemon
  - 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>`

Redis Demo

Show redis cli, keys and values

Show redis usage in visualizer/simulation

- Show cli usage (open right robot)

Show hw0 redis keys

Workshop: HW0 setup