Some Logistics

- **Website**: https://cs.stanford.edu/groups/manips/teaching/cs225a/
- **Piazza**: http://piazza.com/stanford/spring2019/cs225a

**Homeworks**
- Released Thurs evening (HW0 today)
- Due next Thurs at beginning of class (3:00 pm)
- No late days
- Submission via gradescope or AFS (TBD)

**Office Hours**
- Adrian: Tues 1:00 - 3:00 pm
- Mikael: Mon, Wed 2:00 - 3:00 pm
- Rooms TBD

**Session to help you install sai2 on Friday (tomorrow)**
Projects

- 8 groups of 4 students
- Projects groups and outline due on April 25th
- Second part of quarter will be focused on projects
- Several updates to present in class

START EARLY
Sai2 Environment

CS225A

Mikael Jorda
Spring 2019
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 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)
Sai2 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)

Your application (cs225a) will use these modules
Demo

Let us simulate a pendulum (rbot) and control it to oscillate around an off centered position.

Code is part of cs225a git repo.

To run the demo, compile the code and go to bin/lecture2-demo and run the programs: `./visualization ./simulation` and `./controller`.

To see the source code, go to lecture2-demo folder.
URDF

XML specifying robot parameters, simulation parameters, graphics parameters

SAI2 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 Sai2Model

- Encourage you to read header files and study demo + hw0 code well
- Will definitely help you for the final project
- Most of the functions you need for your controllers are in sai2-model/src/Sai2Model.h
Sai2 Model - useful functions

auto robot = new Sai2Model::Sai2Model (const std::string path_to_model_file, bool verbose);
robot->updateModel();
robot->gravityVector(Eigen::VectorXd& g, const Eigen::Vector3d& gravity);
robot->J_0(Eigen::MatrixXd& J, const std::string& link_name, const Eigen::Vector3d& pos_in_link);
robot->Jv(Eigen::MatrixXd& J, const std::string& link_name, const Eigen::Vector3d& pos_in_link);
robot->Jw(Eigen::MatrixXd& J, const std::string& link_name);
robot->position(Eigen::Vector3d& pos, const std::string& link_name, const Eigen::Vector3d& pos_in_link);
robot->rotation(Eigen::Matrix3d& rot, const std::string& link_name);
CS225A Architecture

2 Separate Applications

- Controller, Simviz
- Operating independently
- Can run on separate computers

Robot state is shared through Redis
Simviz

2 threads (combination of visualization and simulation from demo):

- 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, 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

\[
\Gamma_{\text{command}} = \hat{M}(q)(-K_p(q - q_d) - K_v(\dot{q} - \dot{q}_d)) + \hat{V}(q, \dot{q}) + \hat{G}(q)
\]
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: `delkey`
  - Delete all keys: `flushall`
  - Can interact over the network: `redis-cli -h <ip address> -p <port>`
Pre-requisites for Sai2

- 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. We recommend sublime-text
  - In sublime, the package control is very useful and contains a lot of interesting 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 cs225a repository. You will work on this repository
- Your applications will have 2 programs: simviz (provided) and controller (that you will need to write in most cases)
- Code in C++, compile using cmake, and communicate between programs using Redis