CS225A : Experimental Robotics
Lecture 2 : Simulation & Graphics
Getting started with robot programming
Samir Menon
Apr 02, 2015
Robots in Simulation...
How are the robots controlled?

- Puma: Point control (hand) + posture
  - Trajectory: Circle in the x-y plane
How are the robots controlled?

• Puma : Point control (hand) + posture
  • Trajectory : Circle in the x-y plane

• WAM : Point control (hand) + posture
  • Trajectory : Circle in the y-z plane
How are the robots controlled?

- **Puma**: Point control (hand) + posture
  - Trajectory: Circle in the x-y plane

- **WAM**: Point control (hand) + posture
  - Trajectory: Circle in the y-z plane

- **Pr2**: Point control (2 hands) + posture
  - Trajectory: Circle in the x-y and y-z plane
How are the robots controlled?

- **Puma**: Point control (hand) + posture
  - Trajectory: Circle in the x-y plane

- **WAM**: Point control (hand) + posture
  - Trajectory: Circle in the y-z plane

- **Pr2**: Point control (2 hands) + posture
  - Trajectory: Circle in the x-y and y-z plane

- Controller & Task code is identical!

- Different trajectories, control points and gains...
A general theoretical framework can enable efficient robot-agnostic control specifications
Lecture goals

- Introduction to dynamic simulation and control
- Control software architecture
- Creating controllers that work for multiple robots
- Robots: PUMA, WAM, KUKA LBR iiwa
- Project discussion & Homework 0
Lecture goals

- Introduction to dynamic simulation and control
- Control software architecture
- Creating controllers that work for multiple robots
- Robots: PUMA, WAM, KUKA LBR iiwa
- Project discussion & Homework 0
Theory: Dynamics formulation

\[ A(q) \ddot{q} + b(q, \dot{q}) + g(q) = \Gamma, \]
Theory: Task space formulation

\[ A(q) \ddot{q} + b(q, \dot{q}) + g(q) = \Gamma, \]
\[ \Lambda(q) F^* + \mu(q, \dot{q}) + p(q) = F_{task}, \]
Theory: Task space control

\[ A(q) \ddot{q} + b(q, \dot{q}) + g(q) = \Gamma, \]
\[ \Lambda(q) F^* + \mu(q, \dot{q}) + p(q) = F_{\text{task}}, \]
\[ \Gamma_{\text{task}} = J_t^T F_{\text{task}}, \]
Theory → Simulation

\[ A(q) \ddot{q} + b(q, \dot{q}) + g(q) = \Gamma, \]

\[ \Lambda(q) F^* + \mu(q, \dot{q}) + p(q) = F_{task}, \]

\[ \Gamma_{task} = J_t^T F_{task}, \]
Theory → Simulation → Physical Robot

\[ A(q) \ddot{q} + b(q, \dot{q}) + g(q) = \Gamma, \]
\[ \Lambda(q) F^* + \mu(q, \dot{q}) + p(q) = F_{task}, \]
\[ F_{task} = J_t^T F_{task}, \]

Simulation : Test Controller

Robot : Run Controller
Lecture goals

• Introduction to dynamic simulation and control

• Control software architecture

• Creating controllers that work for multiple robots

• Robots : PUMA, WAM, KUKA LBR iiwa

• Project discussion & Homework 0
Connecting theory to software....

\[
A(q) \ddot{q} + b(q, \dot{q}) + g(q) = \Gamma,
\]
\[
\Lambda(q) F^* + \mu(q, \dot{q}) + p(q) = F_{task},
\]
\[
\Gamma_{task} = J_t^T F_{task},
\]
Reading the simulated robot's sensors

Robot Servo

$\Gamma_{\text{robot}} \leftarrow \Sigma$

CTaskServo

Joint Space

$\Gamma_1 = J_t^T(q(x)) [A(q) F^*(q,x') + b(q, q') + g(q)]$

Task Space

$\Gamma_2 = J_t^T(q(x)) [A(q,x) F^*(x,x') + \mu(q, q', x, x') + p(q, x) + F_{\text{contact}}]$

Null Space

$\Gamma_t = \text{End null space torques}$

Robot Sensors: Read $(q, q', x, x', F_{\text{contact}})$

Robot Model

CTaskModel\(i\)

$J_t^T(q(x)), J_s^T A(q,x), \mu(q, q', x, x'), p(q, x)$

CJSpaceModel

$A(q), b(q, q'), g(q)$

\[ A(q \ddot{q}) + b(q, \dot{q}) + g(q) = \Gamma, \]

\[ A(q) F^* + \mu(q, \dot{q}) + p(q) = F_{\text{task}}, \]

\[ \Gamma_{\text{task}} = J_t^T F_{\text{task}}, \]
Sending joint torques to the simulated robot

\[ A(q) \ddot{q} + b(q, \dot{q}) + g(q) = \Gamma, \]

\[ \Lambda(q) F^* + \mu(q, \dot{q}) + p(q) = F_{task}, \]

\[ \Gamma_{task} = J_t^T F_{task}, \]
Combining multiple control task commands

\[ A(q) \ddot{q} + b(q, \dot{q}) + g(q) = \Gamma, \]

\[ \Lambda(q) F^* + \mu(q, \dot{q}) + p(q) = F_{task}, \]

\[ \Gamma_{task} = J_t^T F_{task}, \]
Computing a single task's command

\[ A(q) \ddot{q} + b(q, \dot{q}) + g(q) = \Gamma, \]
\[ \Lambda(q) F^* + \mu(q, \dot{q}) + p(q) = F_{task}, \]
\[ \Gamma_{task} = J_t^T F_{task}, \]
Computing various dynamic quantities

\[ \begin{align*}
A(q) \ddot{q} + b(q, \dot{q}) + g(q) &= \Gamma, \\
\Lambda(q) F^* + \mu(q, \dot{q}) + p(q) &= F_{task}, \\
F_{task} &= J_t^T F_{task},
\end{align*} \]
Software simply implements the equations...

\[ \begin{align*}
A(q) \ddot{q} + b(q, \dot{q}) + g(q) &= \Gamma, \\
\Lambda(q) F^* + \mu(q, \dot{q}) + p(q) &= F_{task}, \\
\Gamma_{task} &= J_t^T F_{task},
\end{align*} \]
Lecture goals

• Introduction to dynamic simulation and control

• Control software architecture

• Creating controllers that work for multiple robots

• Robots: PUMA, WAM, KUKA LBR iiwa

• Project discussion & Homework 0
Demonstration with PUMA, WAM, KUKA
Lecture goals

• Introduction to dynamic simulation and control

• Control software architecture

• Creating controllers that work for multiple robots

• Robots: PUMA, WAM, KUKA LBR iiwa

• Project discussion & Homework 0
Deciding your project: Your primary goal!
Project details

• Recommended team size: 3-4 students
Project details

• Recommended team size: 3-4 students

• At least 1-2 should be good C++ programmers
Project details

- Recommended team size: 3-4 students
- At least 1-2 should be good C++ programmers
- Ideal to form {CS / EE / ME / Aero} partnerships
Project details

- Recommended team size: 3-4 students
- At least 1-2 should be good C++ programmers
- Ideal to form {CS / EE / ME / Aero} partnerships
- Discuss project feasibility with CAs & Instructors
Project details

• Recommended team size: 3-4 students

• At least 1-2 should be good C++ programmers

• Ideal to form {CS / EE / ME / Aero} partnerships

• Discuss project feasibility with CAs & Instructors

• Robots and equipment will be assigned by us based on the project's requirements
Supported Project: Robot Control

\[ A(q) \ddot{q} + b(q, \dot{q}) + g(q) = \Gamma, \]
\[ \Lambda(q) F^* + \mu(q, \dot{q}) + p(q) = F_{task}, \]
\[ \Gamma_{task} = J_t^T F_{task}, \]

Simulation: Test Controller

Robot: Run Controller
Supported Project: Haptics & Simulation

Control & Teleoperation

Haptic interfaces
Supported Project: Human Motion Modeling

Motion/Trajectory Mapping

Whole body control
Homework 0: Please attend Friday's section!

CS225A: Homework 0
Due: October 2, 2014

This homework is meant to review basic kinematics taught in Introduction to Robotics (CS223A), and to help students set up the control and simulation environment. Please direct questions to Piazza.

NOTE: If you do not have experience with programming, or have not taken Introduction to Robotics, please make sure to attend the associated Section on Friday.

Step 0
Install Ubuntu 14.04 on your computer.

Step 1
Make an account on bitbucket.org.

Step 2
Obtain and install the control and simulation system, SCL.

If you do not have Linux installed. Or if you are not familiar with programming in C++, please feel free to follow instructions on the Wiki.
Available resources....

Standard Control Library
web.stanford.edu/~smenon/scl.html
Sep 3, 2014 - Whole-Body Multi-Task Control. Whole-Body Control: SCL controls three very different robots using the same operational space controller.

Standard instrument control library - Dictionary.com
dictionary.reference.com/.../standard+instrument+control... - Dictionary.com
Standard instrument control library definition at Dictionary.com, a free online dictionary with pronunciation, synonyms and translation. Look it up now!

Instrument control - Wikipedia, the free encyclopedia
The standard was updated in 1987 and again in 1992. This bus is known by three different... These drivers require a VISA library to be installed on the PC. History - Software - See also - References

Windows Controls (Windows) - MSDN - Microsoft
msdn.microsoft.com/...library/...77317 - Microsoft Developer Network

STANDARD CONTROL LIBRARY
A Control And Simulation Framework For Whole-Body Multi-Task Control

Whole-Body Control: SCL controls three very different robots using the same operational space controller.

SCL CODE REPOSITORY & WIKI  SCL CODE TUTORIALS  CONTROL MATH TUTORIALS : 3-DOF AND 6-DOF  SCL CLASS DESCRIPTIONS

OUR METHODS
Standard Control Library (SCL) is a control and simulation framework that aims to provide a generic implementation of robot agnostic control algorithms for use in simulation or on real robots. It achieves robot agnostic control by specifying closed loop control tasks in operational space, which can then be projected on to the articulated body dynamics for arbitrary robots. Multiple control tasks are readily accommodated using a prioritized control subspaces, which disallow competing control tasks from interfering and destabilizing the controller.
Feel free to edit it! Some links might be stale. Also feel free to notify the TAs...
The S<Name> pages correspond to data structures
Do take a look! (no code; only data)
The Bug Tracker

If you find a bug. Report it!!
Valid and non-trivial bug reports will earn you extra credit!
Library Architecture

A lot of (almost) independent modules
Library Architecture

A lot of (almost) independent modules
(Tied together by shared data structures)
A glance at the code....
Building advanced controllers

CcontrollerMultiTask :: Specify your own tasks

https://bitbucket.org/samirmenon/scl-manips-v2/wiki/docu/additional_controllers
Project Discussion / Questions...