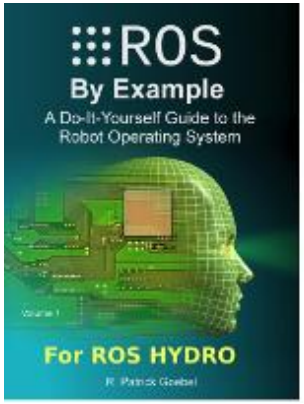
MANIPULATION IN ROS USING BAXTER

DR. JUAN ROJAS
www.JuanRojas.net
Guangdong University of Technology
Biomimetics and Robotics Lab (BIRL)

ROS TRAINING DAY

June 16, 2016

ROS By Example



+ 中山大學 SUN YAT-SEN UNIVERSITY + GaiTech =

- 由Peter Goebel著写的ROS实例系列书籍，在所有ROS文献中享有最多的版本数量。自从ROS发布了Electirc版本以来，Patrick就开始为其编写该书籍，直到后来的Fuerte，Groovy，Hydro，Indigo和即将到来的Jade版本。随着ROS版本的更新，该书也不断更新。没有任何一本其他书籍拥有如此之多的修正，也没有任何一本其他书籍拥有如此大的奉献。
- ROS实例有两部书：这是目前市场上ROS类书籍中，唯一的一套涵盖内容如此全面的系列书籍。
- 本书拥有最好的源代码基础和源代码支持。《ROS入门实例》和《ROS进阶实例》都在github上分享高质量源代码。这些源代码被全世界的ROS爱好者广泛测试并改进，并且涵盖了从最简单的到相当高阶的ROS运行实例。
- 本书拥有最强的社区支持。数以百计的使用者在本书的谷歌论坛中活跃着。（https://groups.google.com/forum/#!forum/ros-by-example）
- 本书是在Juan Rojas博士的指导下，由拥有机器人专业知识的学生工作组翻译而成。Juan Rojas博士在机器人领域研究长达14年之久，是ROS专家，也是机器人工程师专业团队的一员。毫无疑问你将获得质量上乘的译本

GaiTech

ROS by Example Vol. 2 Indigo – coming out in the Fall of 2016!!

SUPPORT PROGRAMS

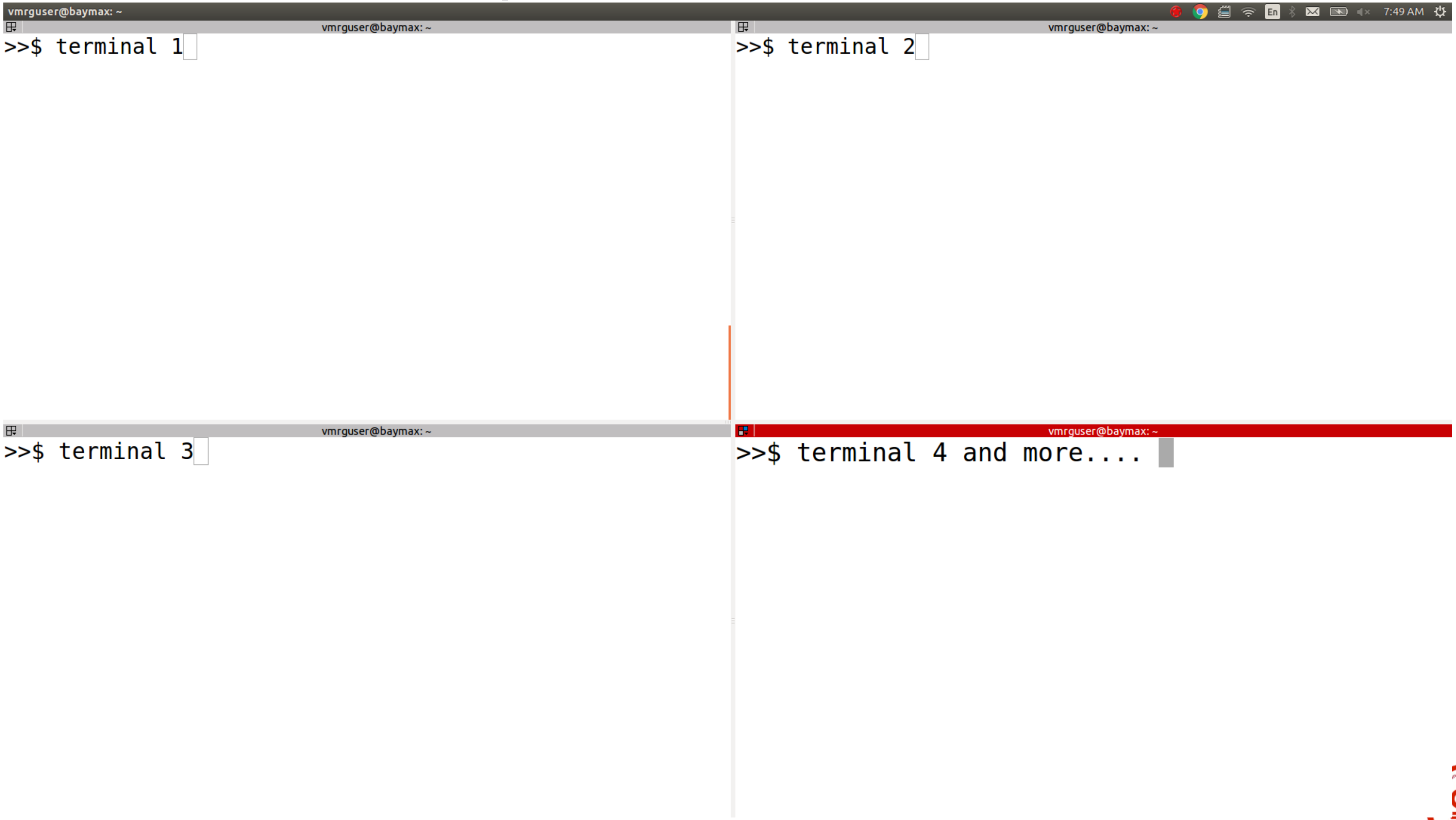
APT-GET

- Used to download programs in linux

```
sudo apt-get update
sudo apt-get upgrade
```

TERMINATOR

Great to run multiple terminals in the same window.

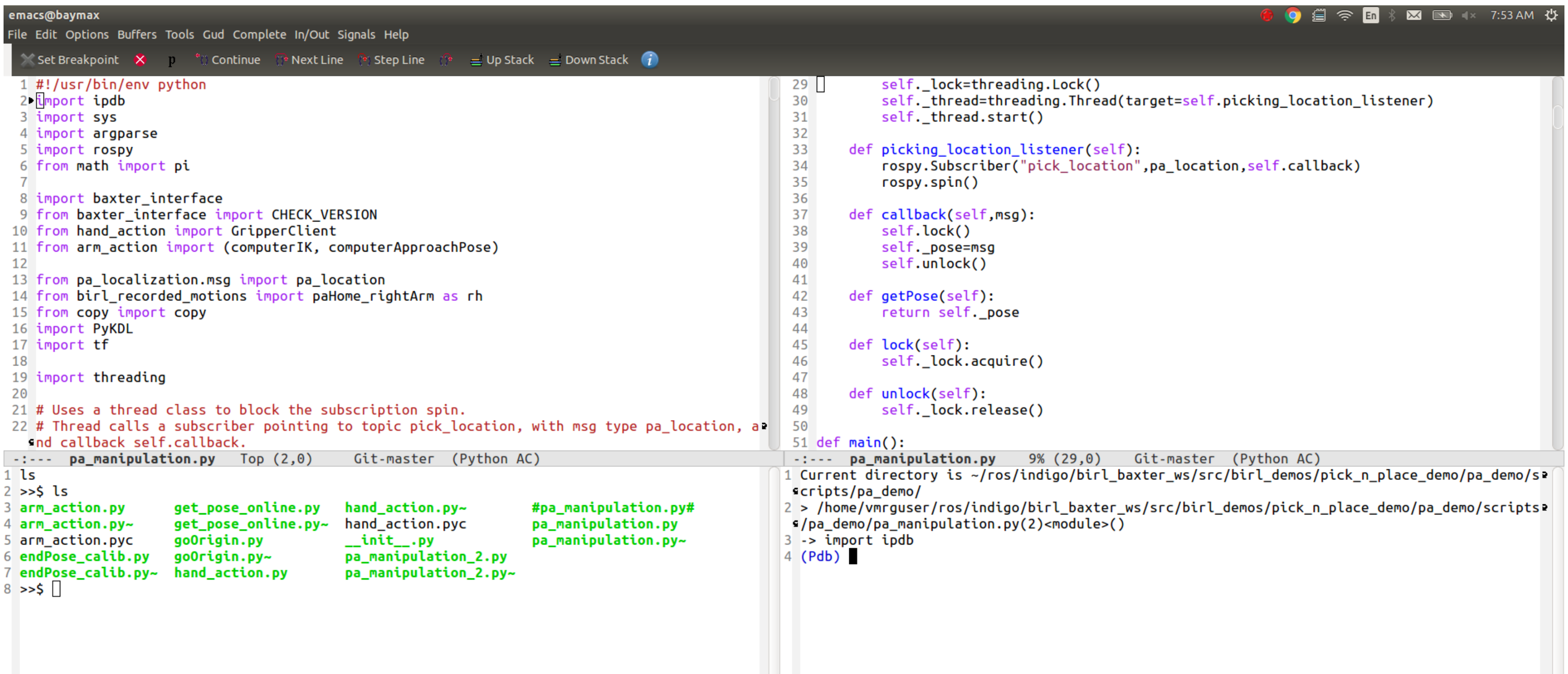


EMACS OR VIM

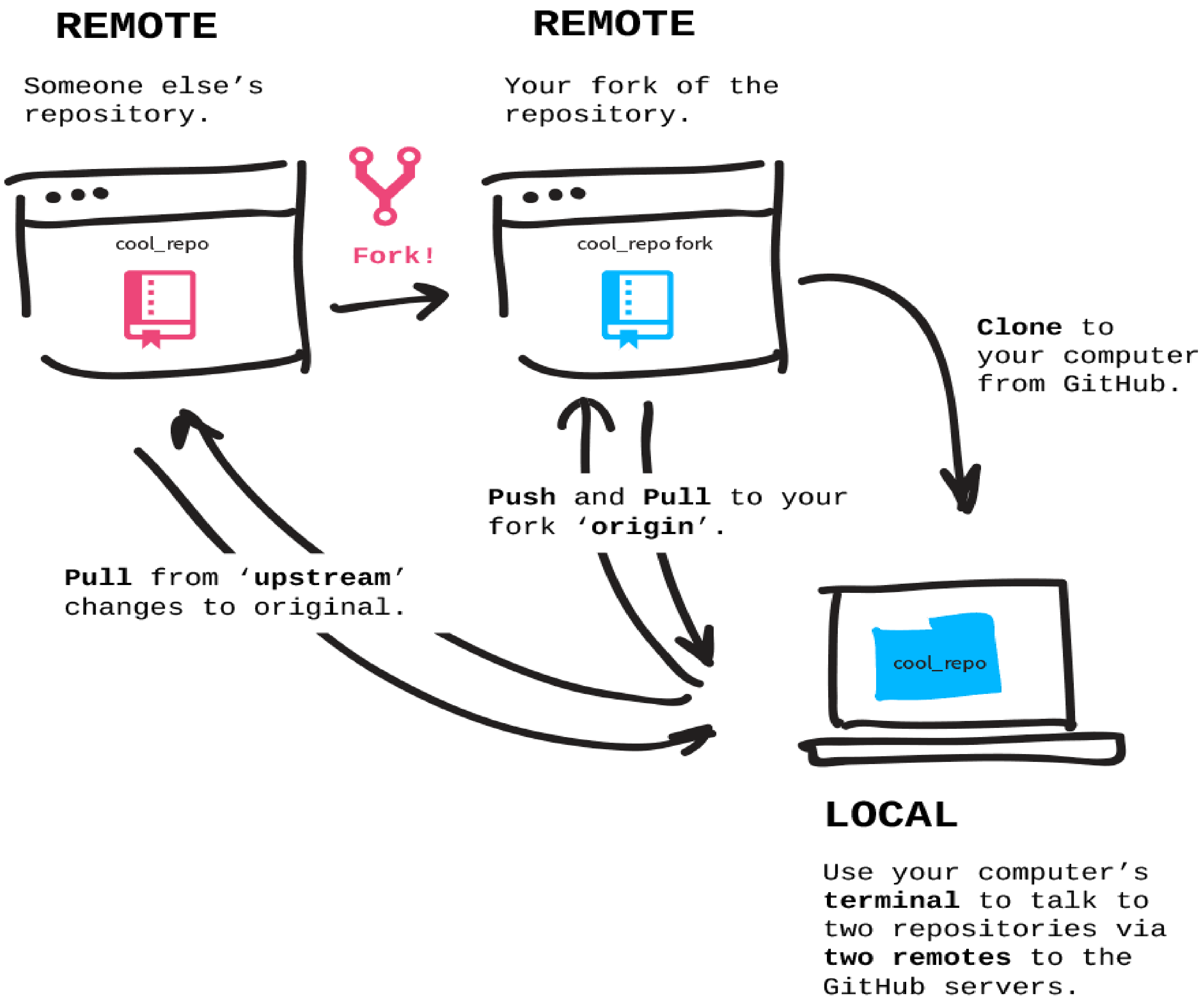
- Extremely powerful editor and more.
  - Powerful editor
  - Strong integration with GDB/PDB

Live terminals

Easily expandable



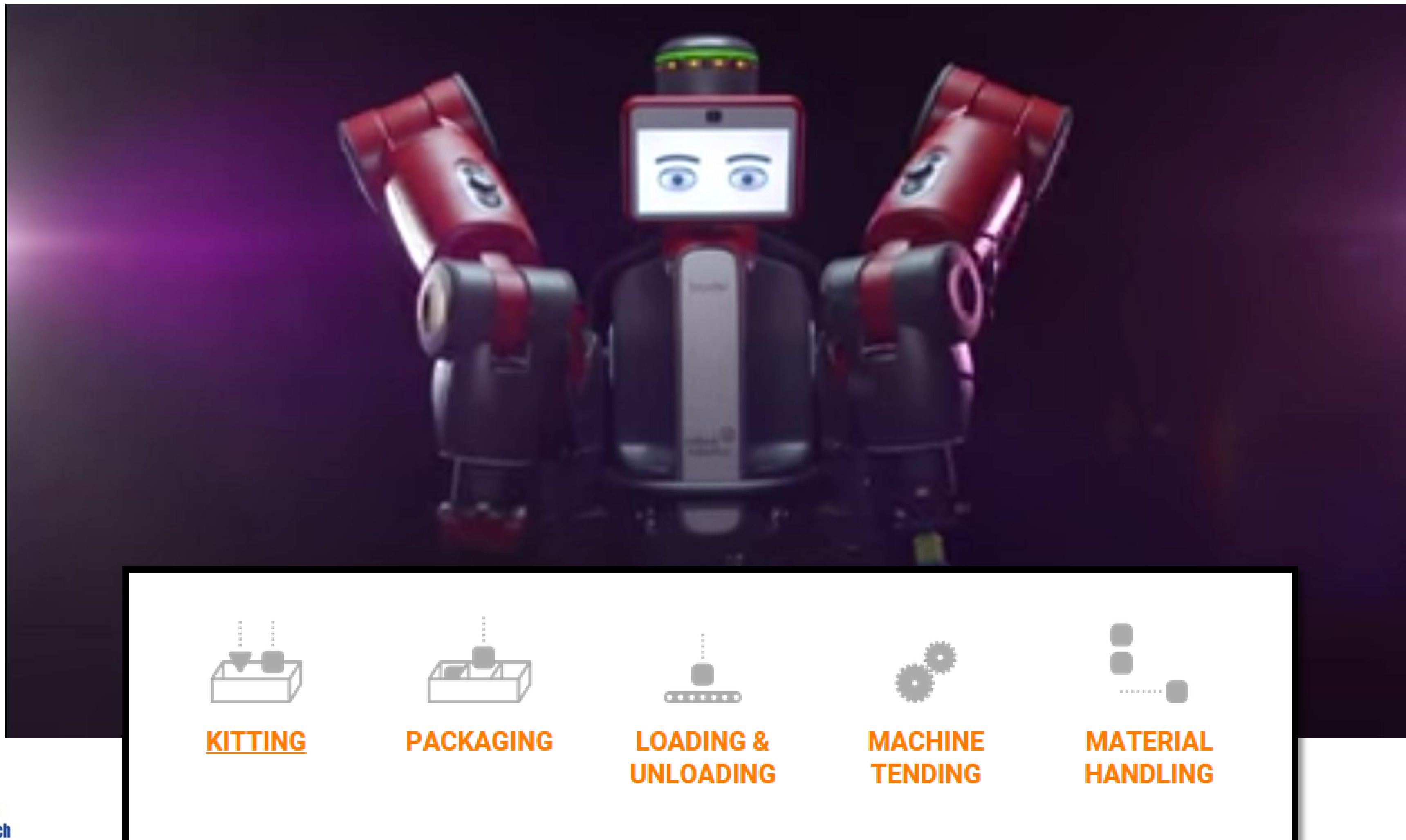
GIT



REMOTE
Someone else's repository.
REMOTE
Your fork of the repository.
cool_repo
Fork!
cool_repo fork
Clone to your computer from GitHub.
Push and Pull to your fork 'origin'.
Pull from 'upstream' changes to original.
cool_repo
LOCAL
Use your computer's terminal to talk to two repositories via two remotes to the GitHub servers.

GETTING TO KNOW BAXTER

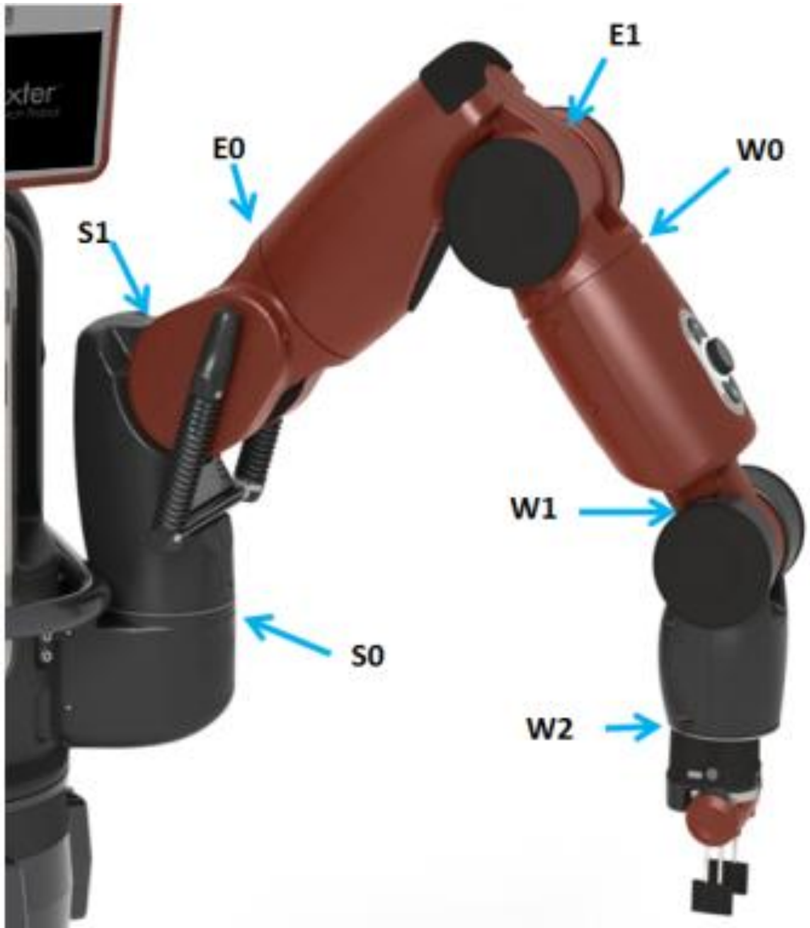
So, what can this robot do?



Baxter’s Arms

7 Degrees of Freedom (DoF)

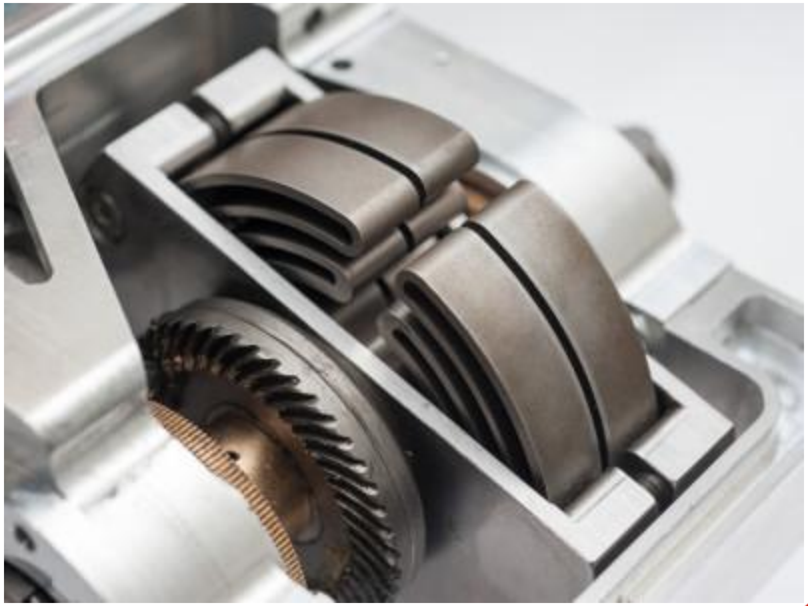
7vs6 DoF = wider mobility.



Series Elastic Actuators

Spring between motor/gear:

1. Stable, low-noise Force Control.
2. Compliant.
3. Measure Torque at each joint.



GaiTech

Programming Layers

API

- Python interface for Baxter.
- Interface interacts with ROS.
- Goal to facilitate programming.

SDK

- Defines ROS: messages, topics, services, action libs.
- Also provides command line tools.

GaiTech

Getting the Baxter Code

- Open source @ sdk.rethinkrobotics.com/wiki/Workstation_Setup

Contents [hide]

Baxter's SDK

- As part of the SDK, Rethink has defined:
  - Topics: /robot/limb/….
    /robot/head/…
  - Message Types:
    baxter_core_msgs/
  - Parameters:
    /baxter_emulator/left_gripper_type
  - Services:
    /ExternalTools/PositionKinematicsNode/IKService
  - Action Libs:
    /robot/limb/<limb>/follow_joint_trajectory/feedback
    /robot/limb/<limb>/follow_joint_trajectory/result
    /robot/limb/<limb>/follow_joint_trajectory/status
  - User Tools
    rosrun baxter_tools ….

Getting Baxter Started

- Setting the Baxter environment:

```
>> roscd          (ROS_WORKSPACE=/your_fav_ws_path)
>> ./baxter.sh (sim for simulator)
```

- Starting the Simulator:

```
>> roslaunch baxter_gazebo baxter_world.launch
```

- For real Baxter, you can check for automatic connection:

```
>> roslaunch baxter_gazebo baxter_world.launch
```

```
[baxter - http://011405P0002.local:11311] >>$ rostopic list
```

Baxter's Arm and Head Joints

- The 7 DoF arms and Head pan consists of joints states, including:
  - Position – joint angles (radians)
  - Velocities – joint velocities (rad/s)
  - Effort – torque exerted at each joint (Nm)
- Topic

```
/robot/joint_states
```

- Message Type:

```
sensor_msgs/JointState
```

Baxter's Arms: Control Modes

- Arms can be controlled in 4 different modes. Top 3:
  - Position Control – controller moves to target joint angles
  - Velocity Control – controller moves to target joint velocities
  - Torque Control – controller moves to target joint torques

- Switch modes by pub commands (pos,vel,effort) @ > 5Hz

```
/robot/limb/<side>/joint_command        (baxter_core_msgs/JointCommand.msg)
```

- Message Type: baxter_core_msgs/JointCommand

```
int32 POSITION_MODE=1, int32 VELOCITY_MODE=2,
int32 TORQUE_MODE=3, int32 RAW_POSITION_MODE=4
int32 mode,
float64[] command
string[] names
```

Move Arm Manually…

```
rostopic pub -r 1000
/robot/limb/right/joint_command
baxter_core_msgs/JointCommand
'{mode: 1, command: [0.1744], names: ['right_s0']}'
```

Publish to joint_command

- Manually test right position/velocity control.

- Simple Position Control Command

```
rostopic pub -r 10   /robot/limb/right/joint_command baxter_core_msgs/JointCommand   '{mode: 1, command: [-1.0], names: ['right_s0']}'
```

- Simple Velocity Control Command

```
rostopic pub -r 10   /robot/limb/right/joint_command baxter_core_msgs/JointCommand   '{mode: 2, command: [-0.01], names: ['right_s0']}'
```

EndPointState

- Provides the following at the end-effector:
  - Pose (m)
    (position, orientation)
  - Twist (m/s)
    (lin vel, angular vel)
  - Wrench (N/m)
    (forces, torques)

/robot/limb/<side>/endpoint_state (baxter_core_msgs-EndpointState)

BAXTER API

API

What is the API?

A new layer of code (based on python) is built on top of ROS.

- Instead of having to:
  - Publish or subscribe
  - Call services
- Call one of the API methods and
  - read/write data through function arguments.

- API is organized according to:
  - Modules
    - Sub-modules.

The Baxter Interface – Python Module

- baxter_interface
  - This module consists of sub-modules to help interact with different parts of the robot.
  - Each sub-module consists of a class of the same name. baxter_interface::limb::Limb
  - The class is a wrapper around ROS communications.
- Sub-Modules (Interfaces)

Robot Enable	Limb	Head	Camera
Gripper	Navigator	Digital IO	Analog IO

For more see: http://sdk.rethinkrobotics.com/wiki/Baxter_Interface

Limb

- Limb is the *class* within the limb sub-module.
  - Queries the joint state
  - Switches between control modes
  - Sends Joint Commands (pos, vel, torque)

```
from baxter_interface import Limb
```

```
right_arm = Limb('right')
left_arm = Limb('left')
```

- Topics

/robot/joint_states
/robot/limb/<side>/joint_command

Limb Class Overview

- The methods below consider position only but…
- The same routines exist for velocity and effort.

[str]	`joint_names(self)` Return the names of the joints for the spec
float	`joint_angle(self, joint)` Return the requested joint angle.
dict({str:float})	`joint_angles(self)` Return all joint angles.
dict({str:Limb.Point,str:Limb.Quaternion})	`endpoint_pose(self)` Return Cartesian endpoint pose {position, orientation}.
	`set_joint_positions(self, positions, raw=False)` Commands the joints of this limb to the specified positions.
	`move_to_neutral(self, timeout=15.0)` Command the joints to the center of their joint ranges

For more see: http://api.rethinkrobotics.com/baxter_interface/html/baxter_interface.limb-pysrc.html#Limb

BAXTER REPO

https://github.com/birlrobotics/

Create ROS Workspace

```
$ mkdir -p ~/ros_ws/src
# ros_ws (short for ROS Workspace)
```

Source ROS Setup

```
$ source /opt/ros/indigo/setup.bash
```

Build and Install

```
$ cd ~/ros_ws
$ catkin_make
$ catkin_make install
```

Install SDK Dependencies

```
$ sudo apt-get update
$ sudo apt-get install git-core python-argparse python-wstool python-vcstools python-rosdep ros-indigo-control-msgs ros-indigo-joystick-drivers
```

Install Baxter SDK

Using the wstool workspace tool, we will checkout all required Baxter Github Repositories into your ROS workspace source directory.

```
$ cd ~/ros_ws/src
$ wstool init .
$ wstool merge https://raw.githubusercontent.com/RethinkRobotics/baxter/master/baxter_sdk.rosinstall
$ wstool update
```

Build and Install

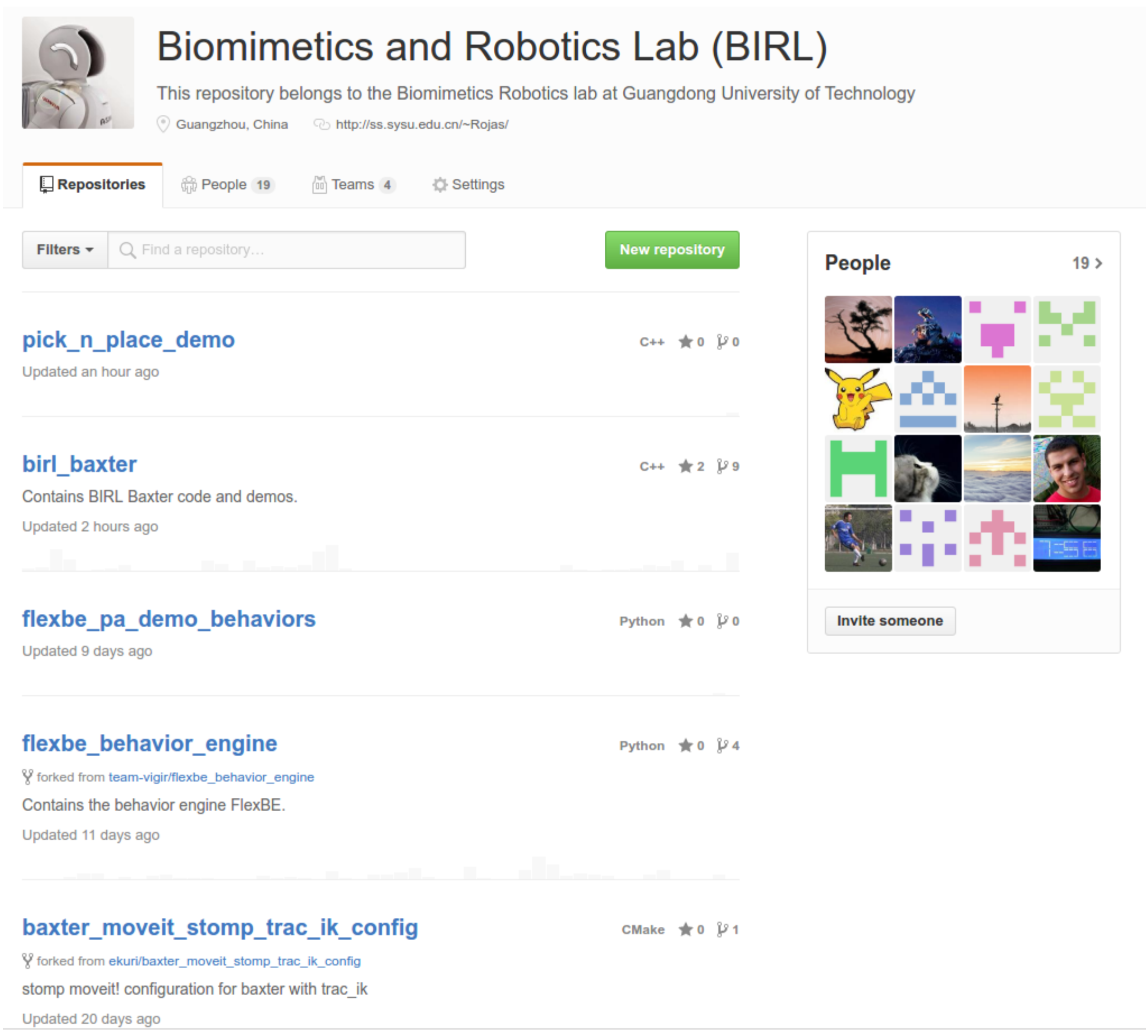
```
$ cd ~/ros_ws
$ catkin_make
$ catkin_make install
```

BIRLROBOTICS REPO

https://github.com/birlrobotics/

BIRL Robotics GitHub Repo

https://github.com/birlrobotics



BAXTER EXAMPLES

Baxter Examples

http://sdk.rethinkrobotics.com/wiki/Examples

SDK Examples

Fundamentals

Enable Robot Example *(Start Here)* - This tool is responsible for enabling (powering and state monitoring) Baxter. Enabling the rob

Movement

Joint Position Waypoints Example - The basic example for joint position moves. Hand-over-hand teach and recording a number of

Joint Position Keyboard Example - This example demonstrates numerous joint position control.

Joint Position Example - Joystick, keyboard and file record/playback examples using joint position control of Baxter's arms.

Joint Torque Springs Example - Joint torque control example applying virtual spring torques.

Joint Velocity Wobbler Example - Simple demo that moves the arm with sinusoidal joint velocities.

Joint Velocity Puppet Example - Simple demo which mirrors moves of one arm on the other in Zero-G.

Inverse Kinematics Service Example - Basic use of Inverse Kinematics solver service.

Simple Joint Trajectory Example - Simple demo using the joint trajectory interface.

Joint Trajectory Playback Example - Trajectory playback using the joint trajectory interface.

Head Movement Example - Simple demo moving and nodding the head.

Head Action Client Example - A demo to showcase the functionality of the head trajectory action server.

Gripper Example - Joystick and Keyboard control for the grippers.

Gripper Cuff Control Example - Simple cuff-interaction control with Zero-G mode.

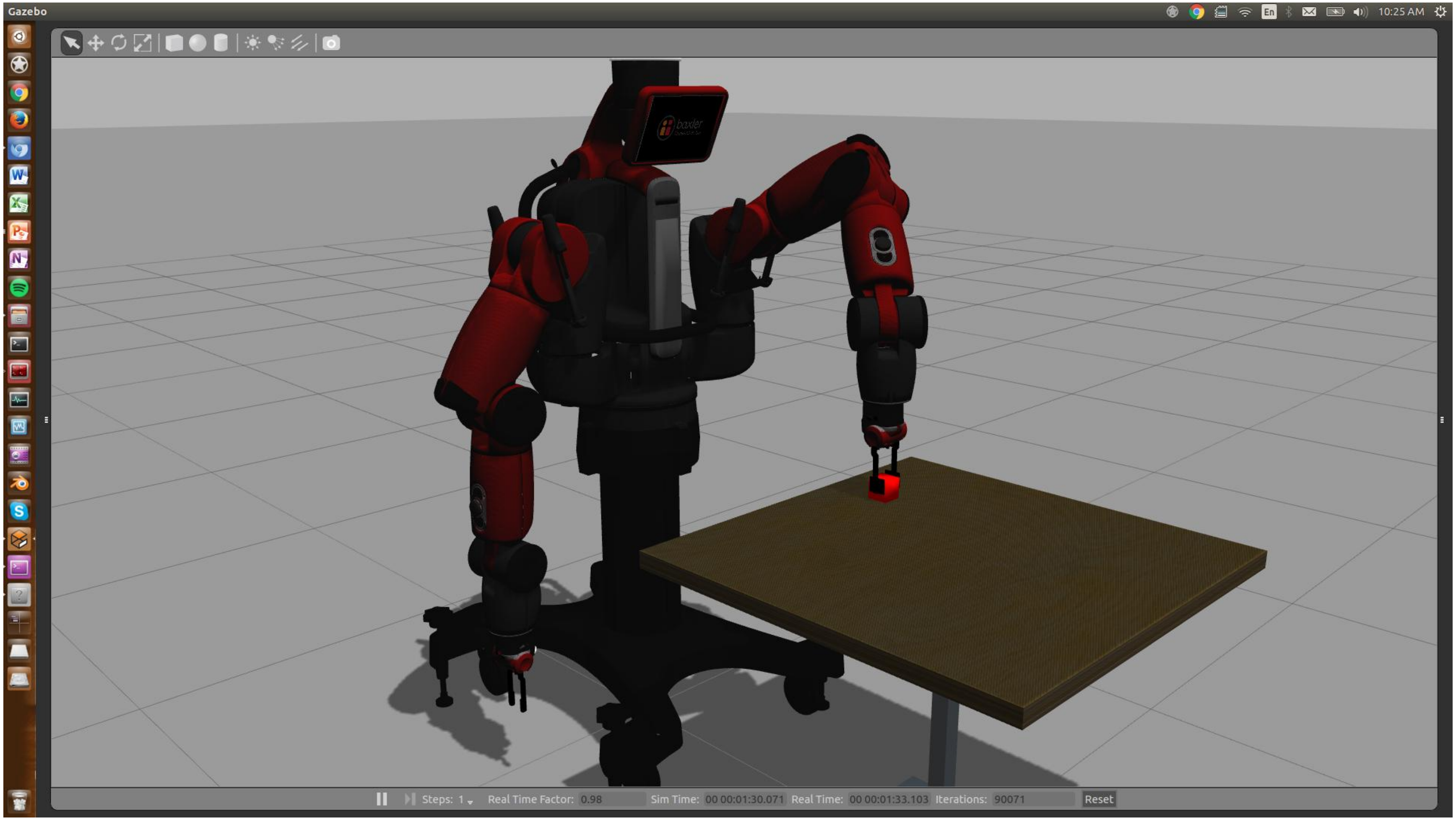
Robot Configuration

URDF Configuration Example - A simple ROS node that shows how to add segment and joint subtrees to the robot's model.

Simulator

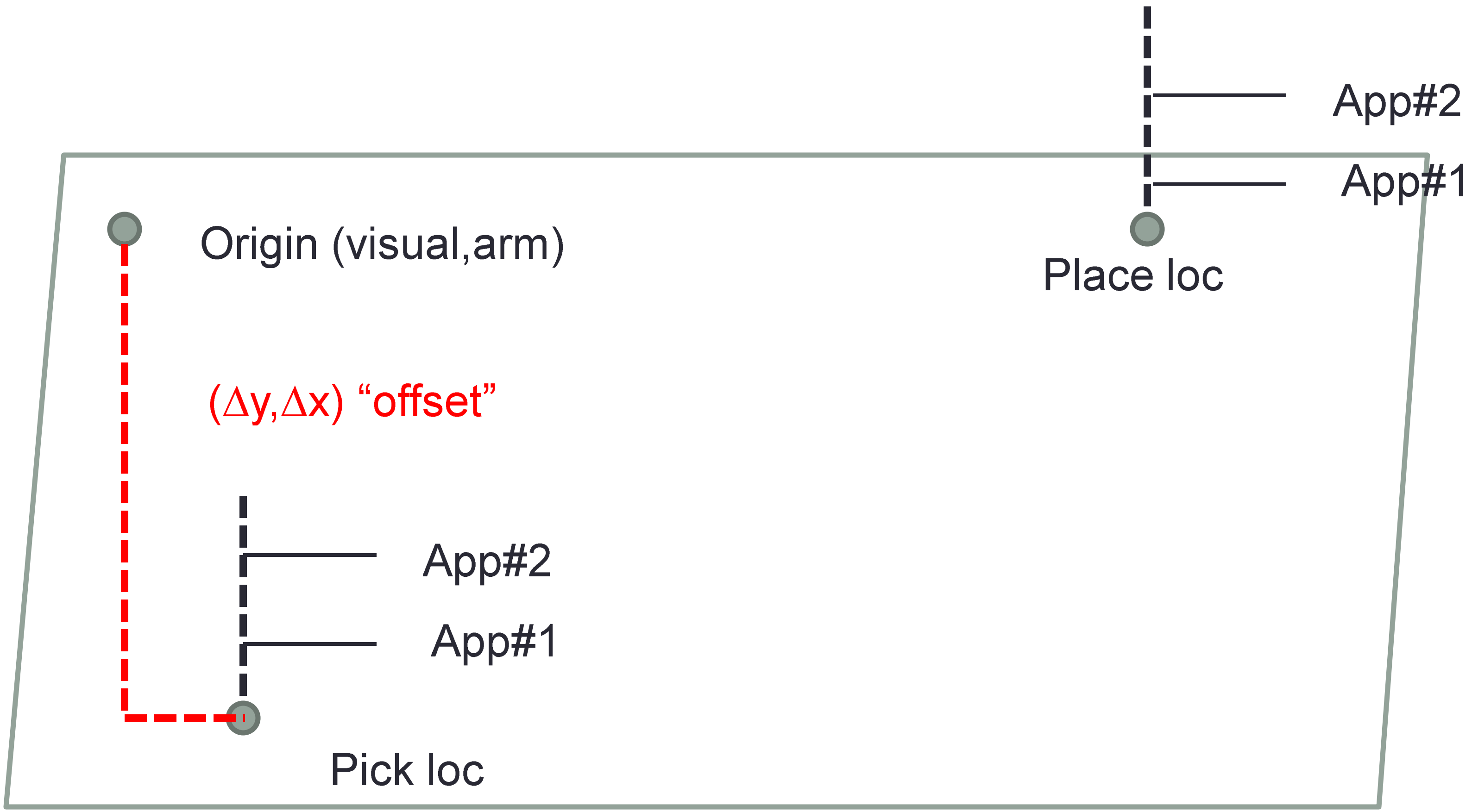
IK Pick and Place Demo - An intermediate example for combining Inverse Kinematics Service calls with Arm movement, gripper ac

Baxter_Sim_Examples



THE MANIPULATION TASK

Experiment Set-Up



VISUAL LOCALIZATION

CALIBRATION

Visual Calibration

- All cameras need to be calibrated.
- Two main types of calibration:
  - Intrinsic Calibration: defines internal parameters of the camera
  - Extrinsic Calibration: defines transforms between cameras/camera-robot

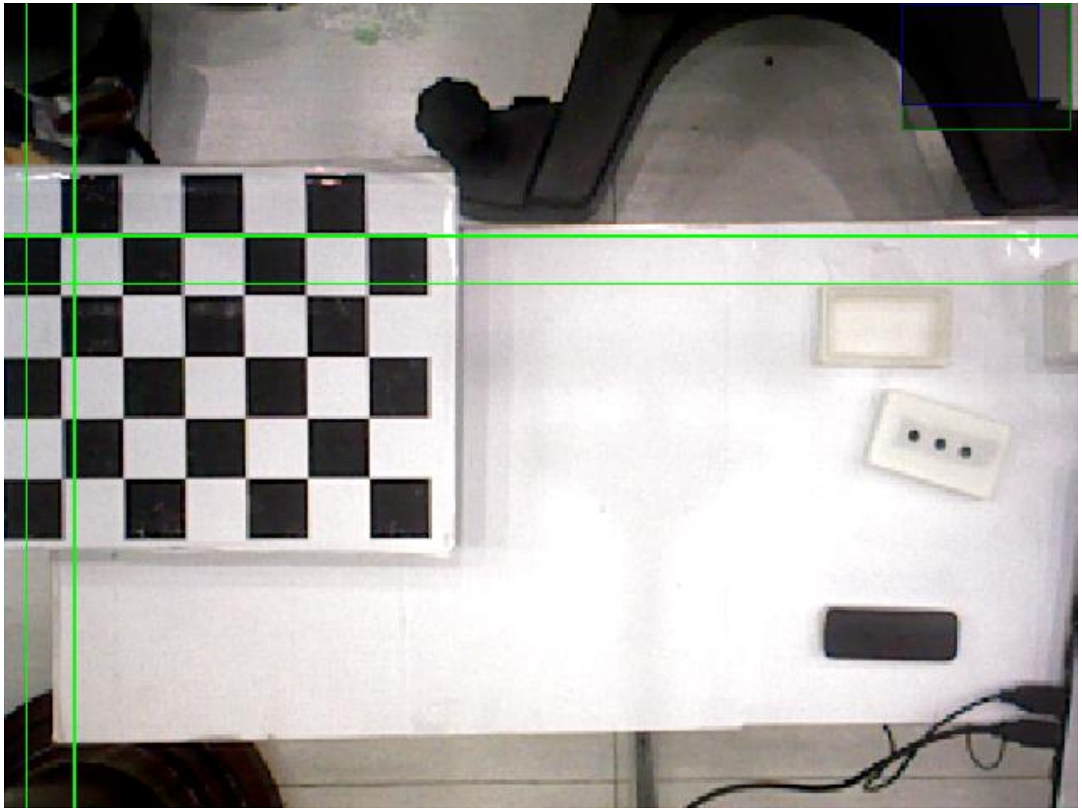
- Packages
  - Openni Calib:
    http://wiki.ros.org/openni_launch/Tutorials/IntrinsicCalibration
    http://wiki.ros.org/openni_launch/Tutorials/ExtrinsicCalibration
  - ROS Camera Calib
    http://wiki.ros.org/camera_calibration

Visual Calibration: Origin

- Run the calibration algorithm

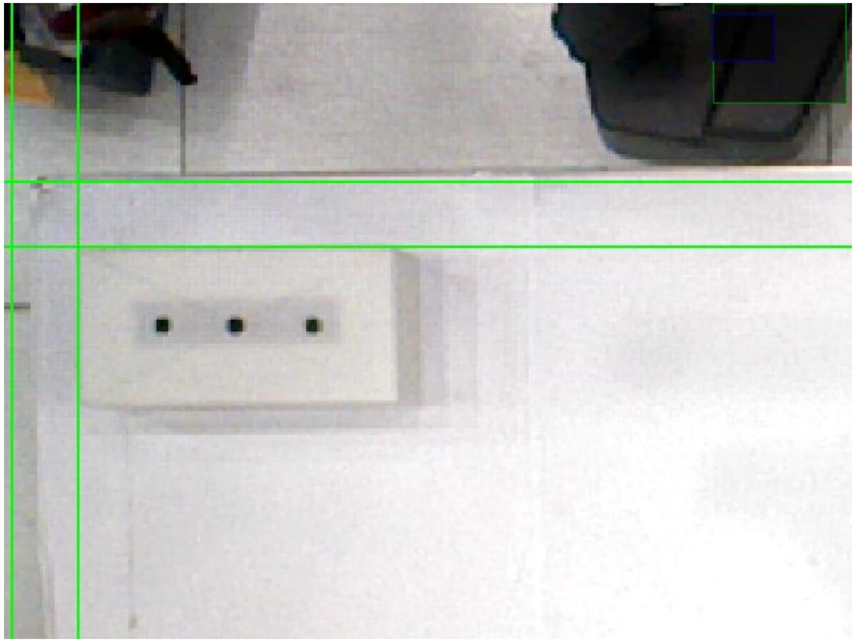
```
roslaunch openni2_launch oppeni2.launch
rosrun pa_localiztion table_pos_calibration.py
```

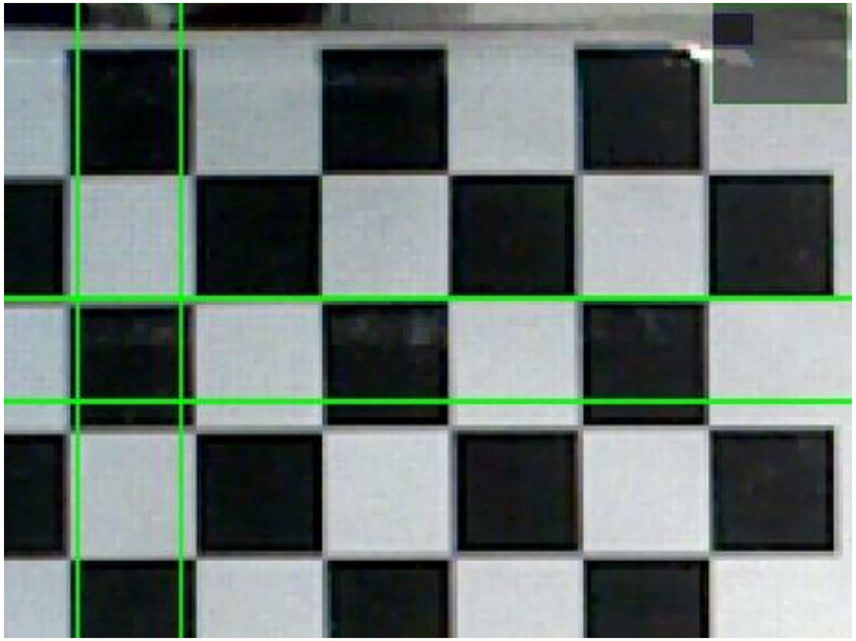
- Define the origin as the crossing of two green lines



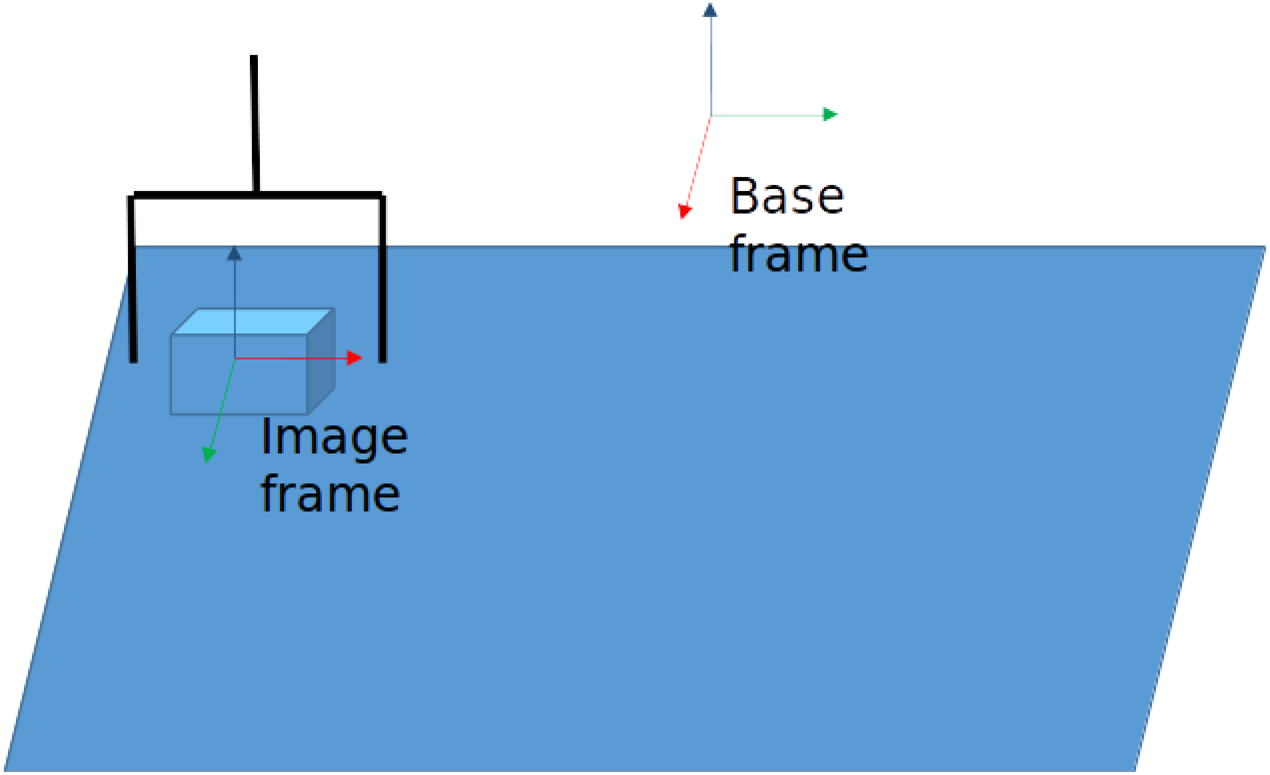
Visual Calibration: Pixel Scale for Objects

- Place object corner on origin.
- Place checkerboard on top.
- With a ruler measure distance of block.
- Then with opencv image viewer count the number of pixels across 1/2/3 blocks and compute average





- Place object @ origin of image frame
- Teleoperate robot arm to grasp object
  - Record current end point position as reference point.

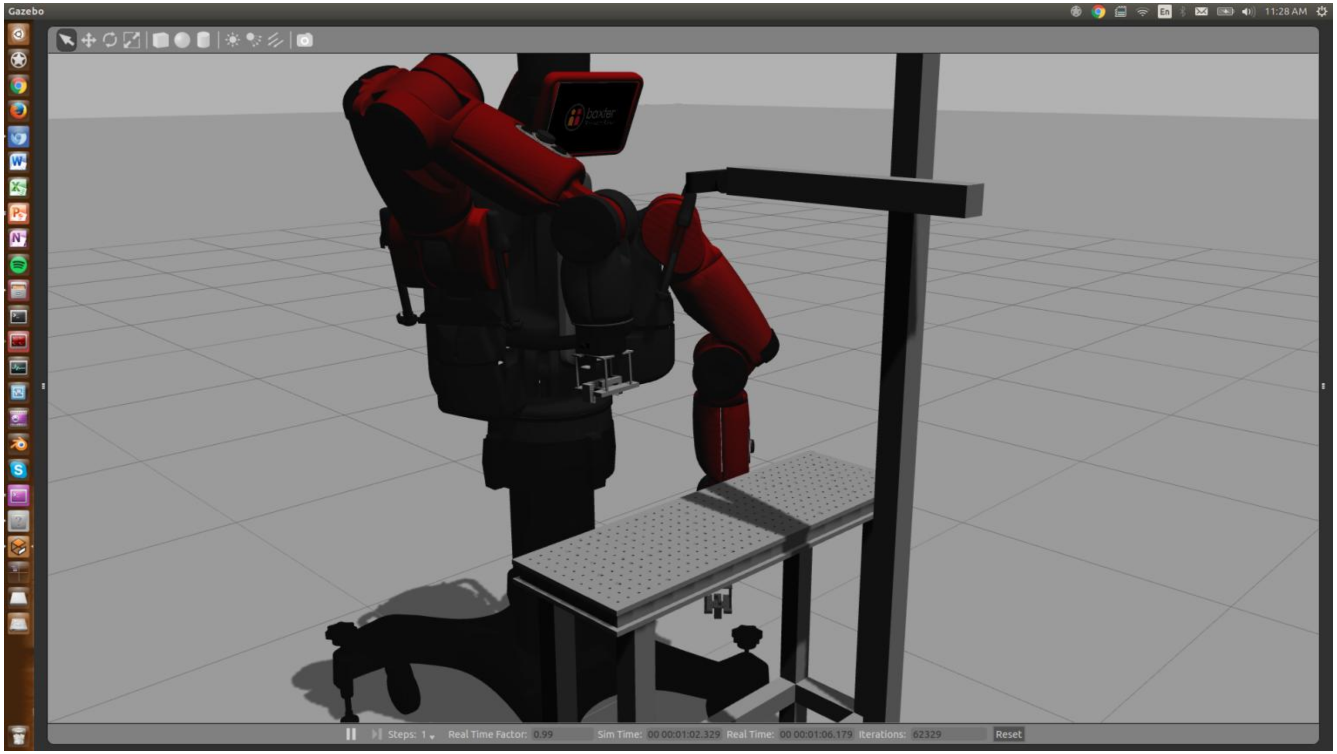


MANIPULATION

Live Demonstration

- Follow the code @ this [link]()

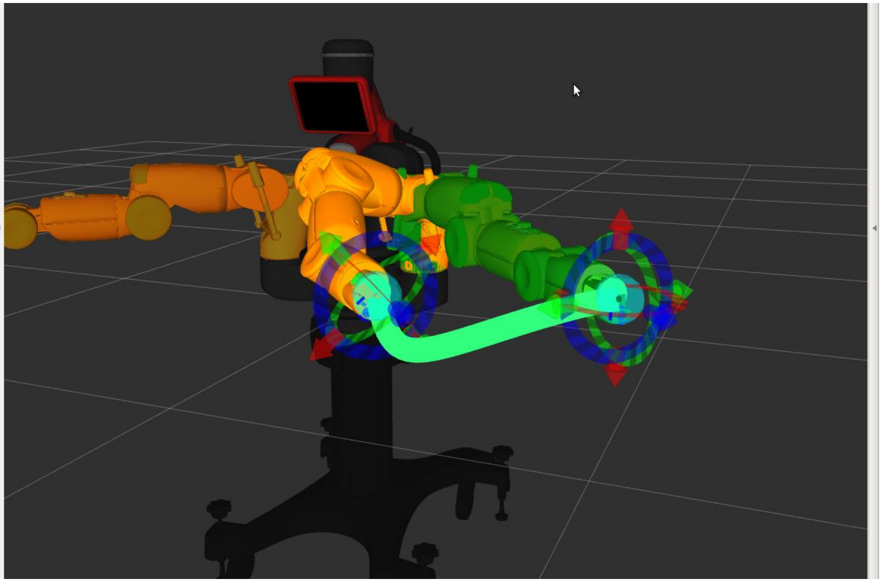
- We will step through it, using ipdb.



OTHER POSSIBILITIES

Different Ways of Moving

- Instead of moving point-to-point…
  - Use the trajectory_action_server
    - Helps you keep track of trajectory
  - Use motion_planning: try our stomp-tracIK here.



Different Kinematic Solvers

- This one is using Baxter PyKDL
  - Based on Orocos KDL

- Other Solutions
  - IK_Fast from openrave
  - Track_IK

  https://github.com/birlrobotics/birl_baxter/tree/master/birl_manipulation/birl_kinematics

QUESTIONS