Bowtie Documentations

bowtie

Personal Robotic Lab
Entertainment Technology Center

Contents

1 Introduction

Welcome to the HERB bowtie documentations. In this manual, we are trying to walk you through environment setup, provide you basic knowledge about HERB, and help you trouble-shooting some common errors. The content is summarization of collective effort on PRL's wiki and Bowtie's experience of trouble shootings, it's created by all, and hopefully will benefit all.

It will cover programmers' guide on setting up python programming environment(Chapter 2) as well as an artists' guide to install the HERB Blender add-on(Chapter 3). After which you will be walked through a basic tutorial of programming HERB using two methods in Chapter4. It also shows how to set up and use the Puredata GUI developed by Garth Zeglin in PRL lab, in Chapeter 5.

First, let's take a brief look of frequently used terminologies in HERB, as well as in robotics at large.

1.1 HERB Terminologies

1.1.1 ROS

ROS Robot Operating System is the open-source infrastructure on which all HERB software runs, comprising an interprocess communication system and a large library of robot code, including planning, vision, and simulation. ROS nodes communicate over a local network via the ROS master, issuing synchronous remote procedure calls via 'services' and broadcasting state information on 'topics'.

1.1.2 openRave

An environment for testing, developing, and deploying motion planning algorithms in real-world robotics applications. The main focus is on simulation and analysis of kinematic and geometric information related to motion planning. OpenRAVE's stand-alone nature allows is to be easily integrated into existing robotics systems. It provides many command line tools to work with robots and planners, and the run-time core is small enough to be used inside controllers and bigger frameworks. An important target application is industrial robotics automation.

1.1.3 herbpy

A simplified Python API for scripting the robot, including basic motion commands, state queries, named poses, and planning collision-free motions to positions. The library uses either a ROS interface to the real robot or an OpenRAVE simulator.

1.1.4 Motion Planning

The process of breaking down a desired movement task into discrete motions that satisfy movement constraints and possibly optimize some aspect of the movement.

1.1.5 YAML file

A human-readable data serialization format that takes concepts from programming languages such as C, Perl, and Python, and ideas from XML and the data format of electronic mail. YAML is a recursive acronym for "YAML Ain't Markup Language". Early in its development, YAML was said to mean "Yet Another Markup Language", but it was then reinterpreted to distinguish its purpose as data-oriented, rather than document markup.

1.1.6 CHOMP

The “Covariant Hamiltonian Optimization for Motion Planning” planner, developed by Personal Robotic Lab which uses a gradient descent technique.

1.1.7 FCurve

A cubic spline joint-space robot trajectory describing joint positions versus time via a set of knot points at keyframes. The name comes from Blender, the open-source 3D animation package.

2 Guide to Programmers on Setting up Python programming Environment

2.1 If you are using ubuntu Virtual Machine

It’s preferable to use a native Linux ubuntu for better performance. But in cases when you don’t have an ubuntu machines, you can also set up a virtual machine.

Following are notes for setting up a virtual machine; this section can be skipped if using a native Linux installation. These notes have been tested using a fresh virtual machine created using VirtualBox on a 64-bit Windows 7 host with 4G physical RAM.

Note that VirtualBox has relatively poor OpenGL performance, so this may not be satisfactory for using blender or the OpenRAVE visualizer. On a Mac, Parallels seems to do better.

Minimum settings when creating the new machine:

```
64-bit Ubuntu Linux
```

```
1.5G RAM allocated (more if feasible)

minimum 10G virtual disk allocated (20G suggested)
```

The full installation consumes about 8.9G when freshly installed, of which 3.3G is Ubuntu, 3.1G is ROS, and the remaining 2.5G PRL packages and dependencies.

The first post-installation launch of herbpy triggers the generation and compilation of the ikfast inverse kinematics library. This requires a lot of memory; anecdotally 1.5G RAM + 1.0G swap failed, 1.5G + 2.0G swap succeeded. If it fails, add additional swap space manually with mkswap and swapon and try again.

Other VirtualBox-specific adjustments: installing the guest additions will allow the screen to resize correctly; adding a second host-only network interface will allow logging in using ssh from the host computer (make sure openssh-server is installed on the virtual machine); removing the avahi-daemon package will eliminate annoying notifications. The default configuration does not enable 3D Acceleration; OpenRAVE will work but be slow.

2.2 General Installation Instructions

Install any new packages from the Ubuntu repositories and reboot in case there are kernel updates:
(This is only really necessary if this is a fresh install.)

```
sudo apt-get -y update && sudo apt-get -y dist-upgrade && sudo reboot
```

Add the ROS repository to your sources.list and install ROS Fuerte:
(If you already have ROS installed, you can skip this step.)

```
wget -qO - http://packages.ros.org/ros.key | sudo apt-key add -

sudo add-apt-repository 'deb http://packages.ros.org/ros/ubuntu precise main'

sudo apt-get update

sudo apt-get install ros-fuerte-desktop-full
```

Add the OpenRAVE PPA to your sources.list:

```
sudo add-apt-repository ppa:openrave/release
```

Add the Personal Robotics apt server to your sources.list:

```
wget -qO - http://packages.personalrobotics.ri.cmu.edu/pr.key | sudo apt-key add -

sudo add-apt-repository 'deb http://packages.personalrobotics.ri.cmu.edu/ precise main'
```

Update apt to use the new sources we just added, and then install pr-herbpy. *This will autoselect the latest version of herbpy and install the dependency packages from PR.(Observed: several versions of herbpy were presented.) Sometimes, the server will drop the connection midway through downloading all of these packages. If so, just rerun the second command.*

```
sudo apt-get update

sudo apt-get install pr-herbpy
```

Note: pr-herbpy is a virtual package, so you will be presented with several candidates of the form pr-herbpy-r[0-9]+, where the suffix is a subversion revision identifier. You will need to explicitly choose one (perhaps the most recent revision). At the moment, due to packaging bugs, there are some packages that need to be manually added as well:

```
sudo apt-get install libnewmat10-dev
```

2.3 Testing the installation

Add ROS and the Personal Robotics packages to your environment:

```
source /opt/ros/fuerte/setup.bash

export ROS_PACKAGE_PATH=$HOME/ros:/opt/pr/:$ROS_PACKAGE_PATH

rospack profile
```

You should now be able to run herbpy on your machine:

```
rosrun herbpy console.py --sim
```

At this point, any public PR package can be installed using apt-get.

2.4 launch IDE in .bashrc environment

If you are to debug in your IDE, you need to let your IDE ‘know’ the environment that’s set up in .bashrc file in previous chapter 2.2. Otherwise you will get import errors.

.bashrc is the configuration file being run everytime you open a new terminal. Here we set up the ROS_PACKAGE_PATH and PYTHONPATH.If you run python script in your Eclipse or Eric, these ROS_PACKAGE_PATH and PYTHONPATH are not being setup, so you may get errors on importing certain libraries.

A good way to solve this is to launch your IDE in terminal, which already has the environment set up.

For example if you are using Eclipse, you can launch Eclipse by

1. Open a new terminal and cd to the eclipse directory

2. Key in:

```
bash -i -c eclipse
```

In this way, eclipse is using the environment that’s set up in .bashrc file.

2.5 Final .bashrc file

The final .bashrc file should be something like this, make sure you have these lines in your .bashrc file in your home folder, to setup up the environment paths. Situations may vary if there are package changes.

```
source /opt/ros/fuerte/setup.bash

export ROS_PACKAGE_PATH=$HOME/ros:/opt/pr/:$ROS_PACKAGE_PATH

################for blender addon####################

export PYTHONPATH=$PYTHONPATH:~/ros/local/pr-ros-pkg/trunk/drama/pose_grabber/src

export PYTHONPATH=$PYTHONPATH:~/ros/local/pr-ros-pkg/trunk/drama/rosblender/src

export PYTHONPATH=$PYTHONPATH:/usr/lib/pymodules/python2.7

################for blender addon####################

export PYTHONPATH=$PYTHONPATH:/opt/pr/herbpy/src
export ROS_PACKAGE_PATH=/home/ruobingl/Desktop/etc_gui:$ROS_PACKAGE_PATH
```

2.6 Installation of PyQt IDE

First, install synaptic software manager for a convenient installation.

```
  sudo apt-get synaptic
```

Launch synaptic software manager, check these three packages:

```
  Python-qt4

  Pyqt4-dev-tools

  Pyqt-tools
```

Then apply the changes, synaptic will auto detect the dependencies and install these packages for you.

2.7 Convert Qt designer Generated GUI to Python Files

Qt Designer is Qt's tool for designing and building graphical user interfaces (GUIs) from Qt components. You can compose and customize your widgets or dialogs in a what-you-see-is-what-you-get (WYSIWYG) manner, and test them using different styles and resolutions.

Widgets and forms created with Qt Designer integrated seamlessly with programmed code, using Qt's signals and slots mechanism, that lets you easily assign behavior to graphical elements. All properties set in Qt Designer can be changed dynamically within the code. Furthermore, features like widget promotion and custom plugins allow you to use your own components with Qt Designer.

You can save the GUI design of Qt Designer into file format of .ui. Then in terminal, you can convert the .ui into python files. So you can write implementations of complicated functions in your python files. This is how you convert .ui file to .py files:

Open your terminal, type in:

```
pyuic4 -x <your .ui file name> -o <output file name>
```

Then you will have successfully converted the .ui file into python file, where you can put complex implementations of functions behind the GUI.

3 Guide to Artists on Installing the HERB Blender plugin

3.1 Installing latest Blender on ubuntu

To install the latest blender on 12.04:

```
  sudo add-apt-repository ppa:irie/blender

  sudo apt-get update

  sudo apt-get install blender
```

Summary of dependencies for using Blender for HERB animation:

Ubuntu packages:

```
 blender (see above)
```

PRL debian pckages:

```
pr-herbpy-r27552   (or newer)
```

svn checkouts from pr-ros-pkg/trunk:

```
drama/rosblender

drama/pose_grabber

fcurves

or_plugins/or_fcurve_trajectory
```

Note that all packages need to built before use, e.g. 'roscd rosblender;make', etc.

3.2 Setting up the HERB Blender Plugin

In order to run the Blender Herb tools, you must first load the Blender addon “pr-addon.py”.The directory you need for the addon doesn't exist until you save your preferences once with blender. So, open blender, go to File→User Preferences and click the “Save User Settings” button at the bottom. Then close blender.You may also need to create the following directories. Please **verify** your Blender version and change 2.69 as appropriate to match.

```
mkdir ~/.config/blender/2.69/scripts

mkdir ~/.config/blender/2.69/scripts/addons
```

Now copy the “pr-addon.py” to your blender addons directory. If ROS is correctly configured, this will find your installation of rosblender and copy the file:

```
roscd rosblender

cp src/pr-addon.py ~/.config/blender/2.69/scripts/addons/
```

Then make sure that the src directory of the rosblender and pose_grabber packages are in your python path. Please **verify** that the following represents the correct package path on your account; there are different conventions for checking out packages within $HOME/ros.

```
export PYTHONPATH=$PYTHONPATH:$HOME/ros/local/drama/pose_grabber/src

export PYTHONPATH=$PYTHONPATH:$HOME/ros/local/drama/rosblender/src
```

Also, we have to make sure the python version in Blender (3.3) is able to see the ros python packages in 2.x

Add the following to your .bashrc:

```
export PYTHONPATH=$PYTHONPATH:/usr/lib/pymodules/python2.7
```

In order to run blender connected with the physical HERB robot in the lab, run this in a fresh terminal:

```
rosmasterherb

blender
```

Otherwise, just run blender by itself:

```
blender
```

Once blender is open, load the addon:

```
File->User Preferences
```

Under the “Addons” tab at the top, select “User” under “Categories” on the left. Click the box next to “System: Personal Robotics Drama”.

Finally, open the Herb Rig, found in the rosblender package:

```
~/ros/local/drama/rosblender/rigs/herb.blend
```

4 Basic Python tutorial to HERB

4.1 Method One: Run Herbpy Console

To start up a herbpy console in simulation mode type the following from a command line:

```
$ rosrun herbpy console sim viewer
```

There are several methods for planning motions for arms. The first is to plan to a pre-defined conguration. For example the following plans the arms to the configuration of relaxed_home:

```
robot.PlanToNamedConfiguration(‘relaxed_home’)
```

Here are several frequently used pre-defined configurations:

- *home: arms up, elbows back, hands forward*
- *relaxed_home: arms down, elbows back, hands in*
- *vertical: arms stright up, elbows straight (pointed out)*
- *relaxed: same as relaxed_home (hands right-side up)*

- *carry_pose: arms up, hands down (elbows closed)*

For moving the segway of the robot, to drive the robot forward 0.5 meters(negative input will drive the robot backwards):

```
robot.DriveSegway(0.5)
```

Return the 24 DOF values for the robot

```
robot.GetDOFValues()
```

Before using CHOMP, the distance field must be initialized via the following command:

```
 robot.chomp_planner.ComputeDistanceField(robot)
```

4.2 Method Two: Use Herbpy In your Python Scripts

Each of the python commands shown above can be used inside a python script. The following shows an example script that initializes a model of the robot and the environment:

The robot and env objects are now the same as those in the herbpy console.

```
import herbpy

env,robot = herbpy.initialize(sim=False,  attach_Viewer=True)
```

5 Using RI Lab's Puredata GUI

5.1 Compiling and running the Pd GUI system

The puredata GUI system is a graph based GUI controller developed by Garth Zeglin in PRL. Here's how to install it.

Required Ubuntu packages:

```
puredata

puredata-dev

festival
```

PRL debian pckages:

```
pr-herbpy-r27552   (or newer)
```

svn checkouts from pr-ros-pkg/trunk:

```
puredata

fcurves

or_rviz

or_plugins/or_fcurve_trajectory

drama/drama_data

drama/drama_pd

drama/bowtie_assets

drama/festival-voices
```

To run the simulation interface:

```
roscore   (if not already running)

rosrun drama_pd run-zero-order-performance-simulation
```

There are other scripts in drama_pd/scripts to start the real robot, the ETC panel, etc.

5.2 Using FCurve Trajectories

To convert an animation from blender to a openrave fcurve trajectory, you need to get three nodes up:

- blenderPub (from the blender interface)
- blenderSub (an intermediate node, we might want to get rid of this)
- fcurveListener (in or_fcurve_trajectory)

Follow the procedure and you should be able to get an fcurve trajectory running:

1. svn checkout/update rosblender and make rosblender
2. svn checkout/update https://svn.personalrobotics.ri.cmu.edu/pr-ros-pkg/trunk/or_plugins/or_fcurve_trajectory/ and make or_fcurve_trajectory
3. make sure openrave knows the new plugin, put this in bashrc

```
export OPENRAVE_PLUGINS=$(rospack find or_fcurve_trajectory)/lib:$OPENRAVE_PLUGINS
```

4. launch roscore or rosmasterherb
5. rosrun rosblender blenderSub.py
6. In or_fcurve_trajectory/src, run python test.py for simulation or saving curve, or python execute_on_robot.py.
7. launch blender
8. In blender, load the animation
9. In scripting screen, load the python file rosblender/src/blenderPub.py and hit ``run script``

6 Reference and Acknowledgement

The content in the documentation aims to help Bowtie and potential future teams to start on programming HERB. The content is summarization of collective effort on wiki and Bowtie's experience of trouble-shootings, it's not authored by anyone. Like the open source code of the Bowtie project, this document is created by all, and hopefully will benefit all. It will be subject to modifications and updates.

Many thanks to contributors (no orders):

Aaron Walsman	aaronwalsman@gmail.com
Garth Zeglin	garthz@cmu.edu
Pras Velagapudi	pkv@cs.cmu.edu
Yuyang Guo	yuyangg@andrew.cmu.edu
Don Zheng	zhaodongz2013062@gmail.com
Mac	djmaculate@gmail.com
Juan	jjramire@cmu.edu

Jon	jonlewkf@cmu.edu
Simon	jiahaox@andrew.cmu.edu
Robin	ruobingl@andrew.cmu.edu
Xiaoyi	xiaoyiz@andrew.cmu.edu
Rachina	rachinaa@andrew.cmu.edu
Mike Christel	christel@cs.cmu.edu
Shirley Saldamarco	shirley@andrew.cmu.edu

Reference:

wikipedia

wiki.personalrobotics.ri.cmu.edu