lfd Documentation

Release 0.0.1

Alex Lee, Dylan Hadfield-Menell

January 23, 2015

Contents

lfd is a software framework for generalizing robot trajectories from demonstrations into new situations, thus enabling robots to learn to perform tasks from demonstrations.

This software follows the line of work of these papers: [ISRR2013], [IROS2013], [IROS2014].

Source code is available on github.

CHAPTER 1

Contents

1.1 Installation

This code has been tested on Ubuntu 12.04.

1.1.1 Dependencies

- bulletsim
- trajopt
  - OpenRAVE >= 0.9
  - PCL 1.7
- Python 2.7
- NumPy >= 1.8.1
- SciPy >= 0.9
- HDF5
- h5py
- joblib

1.1.2 Instructions

- Install bulletsim from source. Use the `lite` branch and follow its README instructions.
- Install trajopt from source. Follow the installation instructions but with the following modifications:
  - Use this fork and the `trajopt-jointopt` branch instead.
  - Install OpenRAVE 0.9 or later from the OpenRAVE testing PPA.

```
sudo add-apt-repository ppa:openrave/testing
sudo apt-get update
sudo apt-get install openrave
```

  - Install PCL 1.7.

```
sudo apt-get install libpcl-1.7-all
```

  - Run the `cmake` command with the option `BUILD_CLOUDPROC=ON`, that is:

```
cmake /path/to/trajopt -DBUILD_CLOUDPROC=ON
```

- Install NumPy, SciPy and HDF5.

```
sudo apt-get install python-numpy python-scipy libhdf5-serial-dev
```

- Install h5py and joblib with pip.

```
sudo pip install h5py joblib
```

Add the following path to your `PYTHONPATH`:

```
/path/to/lfd
```

Now you should be able to run the scripts in the `examples` directory.

1.1.3 Running the Test Suite

You can run the test suite using this command:

```
python -m unittest discover -s /path/to/lfd/test/
```

1.2 Additional Installation to Use the GPU

1.2.1 Dependencies

- gfortran
- cmake
- boost-python
- CUDA >= 6.0
- PyCUDA >= 2013.1.1
- CUDA SciKit >= 0.5.0
- Mako
- CULA >= R12 (optional)

1.2.2 Instructions

- CUDA:
  - Get the CUDA installers from the CUDA download site and install it.

```
sudo dpkg -i cuda-repo-ubuntu1204_6.5-14_amd64.deb
sudo apt-get update
```

  - Then you can install the CUDA Toolkit using apt-get.

```
sudo apt-get install cuda
```

  - You should reboot the system afterwards and verify the driver installation with the nvidia-settings utility.
  - Set the environment variable `CUDA_HOME` to point to the CUDA home directory. Also, add the CUDA binary and library directory to your `PATH` and `LD_LIBRARY_PATH`.

```
export CUDA_HOME=/usr/local/cuda
export PATH=${CUDA_HOME}/bin:${PATH}
export LD_LIBRARY_PATH=${CUDA_HOME}/lib64:$LD_LIBRARY_PATH
```

- Install PyCUDA with pip. Make sure that `PATH` is defined as root.

```
sudo PATH=$PATH pip install pycuda
```

- Install CUDA SciKit with pip.

```
sudo pip install pycuda scikits.cuda>=0.5.0a1 Mako
```

- CULA (optional):
  - Linear systems can optionally be solved on the GPU using the CULA Dense toolkit.
  - Download and install the full edition of CULA. The full edition is required since the free edition only has single precision functions. The full edition is free for academic use, but requires registration.
  - As recommended by the installation, set the environment variables `CULA_ROOT` and `CULA_INC_PATH` to point to the CULA root and include directories. Also, add the CULA library directory to your `LD_LIBRARY_PATH`.

```
export CULA_ROOT=/usr/local/cula
export CULA_INC_PATH=$CULA_ROOT/include
export LD_LIBRARY_PATH=${CULA_ROOT}/lib64:$LD_LIBRARY_PATH
```

- Build the lfd sources with cmake as you would normally do.

```
mkdir build_lfd
cd build_lfd
cmake /path/to/lfd
make -j
```

To use the compiled libraries from python, add the following path to your `PYTHONPATH`:

```
/path/to/build_lfd/lib
```

For more information, check out the README from the tpsopt module.

1.3 Documenting

1.3.1 Dependencies

- Sphinx >= 1.3b.
- sphinx_rtd_theme
- mock

1.3.2 Instructions

Install Sphinx, sphinx_rtd_theme and mock with pip.

```
sudo pip install sphinx>=1.3b1 sphinx_rtd_theme mock
```

The documentation is generated from ReStructured Text using Sphinx.

The documentation sources are in the `doc/` directory. To locally build the documentation, go to the `doc/` directory and run:

```
make html
```

The built documentation will be in the `_build/html/` directory.

The online documentation can be found at rll.berkeley.edu/lfd. Whenever new commits are pushed to the `master` branch, the docs are rebuilt from this branch (assuming the build doesn't fail).

Use Google style docstrings for documenting code. These Sections can be used inside the docstrings. For docstring examples, see Example Google Style Python Docstrings or the module `lfd.registration`.

1.4 Miscellaneous

1.4.1 Settings files

The `lfd` package and its subpackages have a `settings.py` file with setting variables. You can easily override these variables by creating a package `lfd_settings` with the same structure as `lfd`. The variable that should be overriden can be defined in the corresponding `settings.py` file of the `lfd_settings` package.

This is best illustrated with an example. Suppose you want to override the `EM_ITER` variable from the `lfd.registration.settings` module to be `5` instead. First, you can generate the `lfd_settings` package with the provided script:

```
cd /path/to/lfd
python scripts/make_lfd_settings_package.py ../lfd_settings/lfd_settings
```

Remember to add the `lfd_settings` package to your `PYTHONPATH`:

```
export PYTHONPATH=/path/to/lfd_settings:$PYTHONPATH
```

Then, in the file `/path/to/lfd_settings/lfd_settings/registration/settings.py`, add python code that overrides the `EM_ITER` variable:

```
EM_ITER = 5
```

1.4.2 Downloading test data

First navigate to the `bigdata` directory, and then run the `download.py` script.

1.4.3 Cache files

By default, some functions cache results in the default cache directory `/path/to/lfd/.cache/`. If you are running out of space, consider deleting this directory.

CHAPTER 2

Index

- *genindex*
- *modindex*
- *search*

CHAPTER 3

References

Bibliography

[ISRR2013] John Schulman, Jonathan Ho, Cameron Lee, Pieter Abbeel, "Learning from Demonstrations through the Use of Non-Rigid Registration," *in Proceedings of the 16th International Symposium on Robotics Research (ISRR), 2013*.

[IROS2013] John Schulman, Ankush Gupta, Sibi Venkatesan, Mallory Tayson-Frederick, Pieter Abbeel, "A Case Study of Trajectory Transfer through Non-Rigid Registration for a Simplified Suturing Scenario," *in Proceedings of the 26th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013*.

[IROS2014] Alex X. Lee, Sandy H. Huang, Dylan Hadfield-Menell, Eric Tzeng, Pieter Abbeel, "Unifying Scene Registration and Trajectory Optimization for Learning from Demonstrations with Application to Manipulation of Deformable Objects," *in Proceedings of the 27th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2014*.