Part-based Multisensory Shape Representations Software Notes

Göker Erdoğan

June 19, 2015

1 Required Software

Code is developed on Ubuntu 12.04. However, rational rules model and shape grammar with vision forward model are also tested on OS X 10.7.5. Unfortunately, GraspIt only runs on Ubuntu.

- python 2.7.3
- python packages
  - libblas-dev, liblapack-dev, gcc, g++, gfortran, python-dev. These are required for numppy and scipy. Install using `apt-get install`
  - numpy 1.7.0, install from source, binary package is for an earlier version of numpy. For all python packages, download source, unpack it and run `sudo python setup.py install`
  - scipy 0.12.0, install from source
  - matplotlib 1.2.1, install from source. First install all the dependencies with
  - zss (zhang-shasha algorithm for tree edit distance), install from https://github.com/timtadh/zhang-shasha

    `sudo apt-get build-dep python-matplotlib`

    You can install matplotlib directly from the binary package python-matplotlib too.
  - treelib, implements tree data structure that we use for representing parse trees. Get the source from https://pypi.python.org/pypi/treelib. However, `expand_tree` and `show` methods of `tree` class have a small bug where the leaves of a node are not traversed in the order they are added to the node. So we need to change a few lines in `tree.py` file. Fixed treelib is in `./treelib-fix/` folder. Install that instead of the one online. Note that we are using an older version of treelib; new versions won't work!
- VTK 5.10.1, note that using the latest version VTK 6.0 may require some changes in the code. Again, only option is to compile from source. Make sure that you install pyton wrappers for VTK which we need for using VTK from python. Readme file provided with VTK contains detailed installation instructions. A summary is given below.
  - Install libraries `cmake` and `cmake-curses-gui`
  - Extract VTK source and VTKData archives to two folders, `VTK` and `VTKData` respectively. These folders should be in the same directory.

- Create another folder named `build` in the same folder with source and data folders.
- Open up a terminal, `cd` into `build` folder and run `ccmake ../VTK`
  * Configuration interface will start. Press `c` to run initial configuration.
  * Set `BUILD_EXAMPLES` (not necessary, but nice for checking if VTK works fine), `BUILD_SHARED_LIBS` and `VTK_WRAP_PYTHON` to `ON`.
  * Press `c` as many times as necessary until you get the option to generate and exit.
  * Press `g` to generate and exit
- Run `make`. This takes some time.
- Run `sudo make install` to install VTK.
- Set environment variables `LD_LIBRARY_PATH` to VTK library folder (default is `build/bin`) and `PYTHONPATH` to library folder and VTK python wrapping folder (default `build/Wrapping/Python`).
  * We can set session-wide environment variables in Ubuntu 12.04 as follows.
  * Create a file named `.pam_environment` under home folder.
  * Add the following lines

    ```
    LD_LIBRARY_PATH=vtk_folder/build/bin
    PYTHONPATH=vtk_folder/build/Wrapping/Python:vtk_folder/build/bin
    ```
  * There is a bug in Ubuntu 12.04 that clears `LD_LIBRARY_PATH`, to fix it change `use-ssh-agent` to `no-use-ssh-agent` in `/etc/X11/Xsession.options` as mentioned here `https://bugs.launchpad.net/ubuntu/+source/xorg/+bug/366728`
  * Logoff and login again for the changes to take effect.

- GraspIt, get it from http://www.cs.columbia.edu/~cmatei/graspit/. In order to get joint angles of a 3D object we create in python, we communicate with GraspIt over TCP to issue commands and get its responses. However, GraspIt's current functionality available over TCP is very limited. For this reason, we have added some new functions to GraspIt's TCP server interface. To get these, replace `graspitServer.cpp` and `graspitServer.h` files with the ones we have under `./Graspit-update` folder. Also, TCP server is disabled by default, open `scr/main.cpp` and uncomment line

  ```
  GraspItServer server(4765);
  ```

  Follow the instructions in the manual for installing GraspIt which are summarized below.
  - Install packages `libqt4-dev`, `libqt4-opengl-dev`, `libqt4-sql-psql`, `libcoin60-dev`, `libsoqt4-dev`, `libblas-dev`, `liblapack-dev` (these last two actually should be already installed), `libqhull-dev`
  - Set the environment variable GRASPIT to the directory where you unzipped GraspIt. You can do this by adding the following line to `~./pam_environment` file.

    ```
    GRASPIT=graspit_folder
    ```
  - Go into GraspIt folder and run `qmake graspit.pro`
  - Type `make` to compile GraspIt. Make sure you remove any left over object or moc files before you compile, just remove the folders named `.obj` and `.moc` (maybe hidden)
  - Run GraspIt with `./bin/graspit`

GraspIt Installation on Mac OS X seems to fail, but I'm providing the steps required to compile GraspIt source as someone may be able to solve the runtime problem.

- Setup Qt
- Build Coin3d and SoQt from source
- Setup QHull
- Lapack and Blas are already installed with Apple's Accelerate Framework
- Build GraspIt
  - Set environment variable `GRASPIT` by adding `export GRASPIT=/Users/gerdogan/Documents/GraspIt` to `.bash_profile` file. You need to start a new terminal for the changes to take effect.
  - qmake graspit.pro
  - Open xcodeproj file and replace every instance of

    ```
    isa = PBXFrameworkReference;
    ```

    with the following:

    ```
    lastKnownFileType = wrapper.framework;
    isa = PBXFileReference;
    ```

  - Open Inventor/SbBasic.h and add the following header file

    ```
    #include <Inventor/C/errors/debugerror.h>
    ```

  - Replace `-lGl` with `-framework OpenGL` in linker parameters in project settings window
  - Add following folders to library search paths in XCode project settings

    ```
    /Library/Frameworks/Inventor.framework/Versions/C/Libraries
    /Library/Frameworks/SoQt.framework/Versions/A/Libraries
    ```

These steps make it possible to compile and build GraspIt XCode project. However, despite all these efforts, GraspIt still does not run on MacOsX. There is a NULL pointer error in the following function. Data in pointer.dat file is read to `pointers` object with `SoDB::readAll` function, however this function cannot read the file and returns NULL.

```
void
IVmgr::setupPointers()
{
  SoInput in;
  in.setBuffer((void *) pointersData, (size_t) sizeof(pointersData));
  pointers = SoDB::readAll(&in);   //ERROR: Returns NULL
  //assert(pointers != NULL);
  //pointers = new SoSeparator;
  pointers->ref();
}
```

2 Talking with GraspIt from Python

- Run graspit by going into the folder Graspit and running `./bin/graspit`.
- Open the world file `aomr.xml` in folder Graspit/worlds.
- Start another terminal and run `nc 127.0.0.1 4765`. This command creates a TCP socket for communicating with GraspIt.
- Now you can use this socket to send commands to GraspIt.
- For example, write `loadObject` and press enter. This command will load the object `obj.wrl` from Graspit/models folder.
- The other available commands are: `autoGrasp`, `autoOpen`, `loadObject`, `removeObject`, `rotateObject`, `aomrGetJointAngles`.