How to Use Niftyreg For Image Registration

Installation

1. Create a folder named 'niftyreg_source'

2. Download Niftyreg from https://sourceforge.net/projects/niftyreg/

    -Go to 'Files' tab, and then 'Download Latest Version'. Extract all files to the 'niftyreg_source' folder.

3. On Linux terminal command line type: mkdir niftyreg_build niftyreg_install

4. Change directory to niftyreg_build folder: cd niftyreg_build

5. Run CMake on the source files: ccmake ../niftyreg_source

6. This should generate a GUI. Press "c" to configure project, set cmake_install_prefix to the directory where 'niftyreg_source' folder is stored. Press "g" to generate the Makefiles.

7. On command line: make

8. On command line: make install

    -The niftyreg_build and niftyreg_install folders should now be populated

Additional comments: Read the .pdf file "install_NiftyReg_macLinux.pdf" for other installation options

    -Follow the instructions **very** carefully, each step is important.

    -Mainly, have niftyreg_build, niftyreg_install, and niftyreg_source folders in the same directory.

    -The following last step in the .pdf was not necessary for me, but you may choose to use it for your own purposes:

In order to use NiftyReg in any terminal, you will need to edit your .bashrc or .profile file by adding the following lines:
NIFTYREG_INSTALL=<path_to_your_niftyreg_install> (/Users/mmodat/niftyreg_install in the current example)
export PATH=${PATH}:/${NIFTYREG_INSTALL}/bin
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${NIFTYREG_INSTALL}/lib

3. You can call Niftyreg in your terminal window with the functions in the niftyreg_install/bin folder. Reg_f3d, reg_resample, and reg_average are most likely high priority functions. Reg_f3d deforms a 'floating image' to match your 'reference image' (type reg_f3d -h in command line for input help). Reg_resample is used to take a set of control points generated from reg_f3d to register images. Reg_average generates an averaged image from multiple images.

Example usage:

Note: -ref reference image -flo floating image -res deformed image result -cpp control points used for deformation

a) reg_f3d -ref ref.nii -flo flo.nii -cpp cpp.nii -res result.nii

b) reg_resample -ref ref.nii -flo flo.nii -res result.nii -cpp cpp.nii

Usage

For additional details: http://cmictig.cs.ucl.ac.uk/wiki/index.php/NiftyReg_documentation

To obtain good registration between two images, generally 2 different steps are required:

1. Affine registration
   -preserves points, straight lines, and planes
   -translation, scaling, rotation, and shear
2. Nonrigid registration
   -the default optimization scheme in niftyreg uses normalized mutual information (NMI), which allows for inter-modal image registration (images of different contrasts)
   -essentially minimizes the entropy cost function between two images, which in practice, is the joint histogram
   -the full objective function is a bit lengthy and can be found here along with details of the registration method: *Modat, M., McClelland, J., Ourselin, S. Lung Registration using the NiftyReg Package. Medical Image Analysis for the Clinic: A Grand Challenge, Workshop Proc. from MICCAI 2010.* I know it says 'lung registration' but there are several papers that use this tool for brains as well. This paper just happens to have details of the registration method.

Important niftyreg functions (as used in MATLAB script) and details:
**All direct inputs to niftyreg functions must be in NIFTI (.nii) data format

reg_aladin

Usage: reg_aladin -rmask rmask.nii -lp 2 -ref ref.nii -flo flo.nii -aff outAff.txt -res block.nii
Details: reg_aladin is a block-matching algorithm that aligns subvolumes of two images together. This essentially performs the affine registration step. 'ref.nii' is the reference image and 'flo.nii' is the floating image that will be registered **to** the reference image. A reference mask 'rmask' improves speed of registration by limiting calculations to the masked region. Also prevents errors from artifacts being registered outside the region of interest. '-lp 2' runs only the first two levels of the pyramid registration scheme (once down-sampled, no down-sample). By default, niftyreg uses three levels of registration (twice down-sampled, once down-sampled, no down-sample) to achieve a coarse-to-fine resolution of registration. For low resolution images, I suggest not using too many levels of down-sampling otherwise registration errors start to occur.
Help: http://cmictig.cs.ucl.ac.uk/wiki/index.php/Reg_aladin or command line 'reg_aladin -h'

reg_f3d

Usage: reg_f3d -aff outAff.txt -ref ref.nii -flo flo.nii -cpp cpp.nii -res result.nii
Details: reg_f3d performs nonrigid registration. '-aff' option allows usage of a previously defined affine transformation matrix. We can use the output affine matrix from reg_aladin as input to

reg_f3d. Note here that we still use the same original ref.nii and flo.nii and **not** the resultant image from reg_aladin 'block.nii'. cpp.nii contains the control points for the nonrigid registration and includes the affine transformation. 'result.nii' is the registered image.

reg_resample

Usage: reg_resample -ref ref.nii -flo flo.nii -cpp cpp.nii -res result.nii

Details: reg_resample calculates a transformation matrix based on control points in cpp.nii to register floating image flo.nii to reference image ref.nii. 'result.nii' is the registered image.

In the Matlab script that calls these niftyreg functions, a prefix may be required in front of the niftyreg commands:

```
cmd1=sprintf('"%s %s %s %s %s %s %s %s %s %s %s"',...
        'reg_aladin','-rmask',[regdir 'rmask.nii'],'-lp 2','-ref',[regdir 'ref.nii'],...
        '-flo',[regdir 'flo.nii'],'-aff',[regdir 'outAff.txt'],'-res',...
        [regdir 'block.nii']);
```

i.e.

For Mac: [status,result]=system(['source ~/.profile;',cmd1]);

For Windows (if using Cygwin):

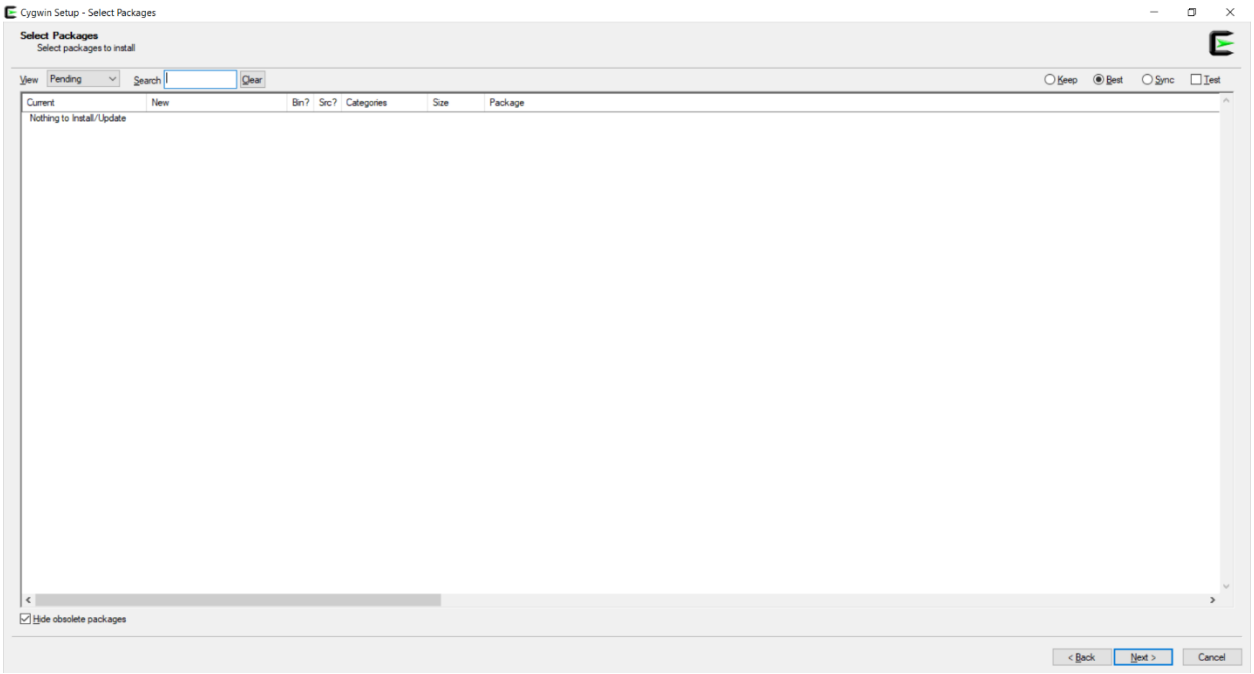
[status,result]=system(['C:/cygwin64/bin/bash --login -c ',cmd1]);

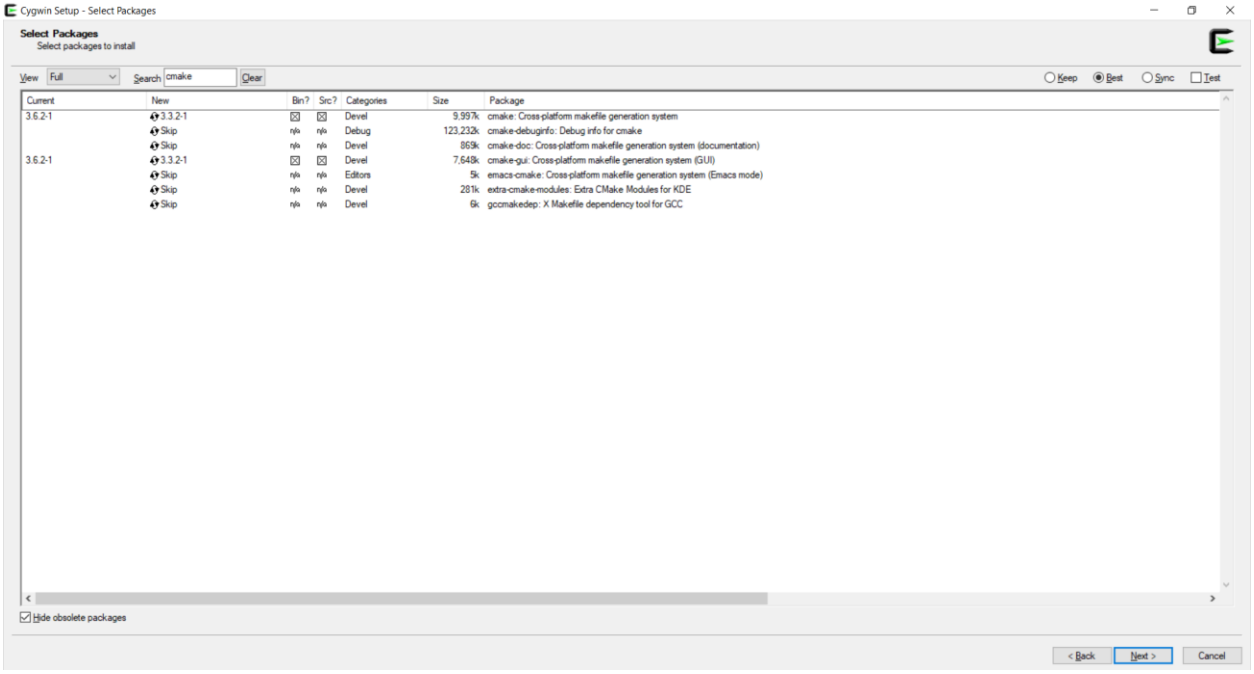
For Linux

[status,result]=system(cmd1); % remove double quotes in cmd1 such that

```
cmd1=sprintf('%s %s %s %s %s %s %s %s %s %s %s',...
        'reg_aladin','-rmask',[regdir 'rmask.nii'],'-lp 2','-ref',[regdir 'ref.nii'],...
        '-flo',[regdir 'flo.nii'],'-aff',[regdir 'outAff.txt'],'-res',...
        [regdir 'block.nii']);
```

**Troubleshooting Notes:
-If you have Windows system, you will need to create a Linux environment on your computer to execute the commands written in the .pdf. I recommend Cygwin (https://www.cygwin.com/). Cygwin only comes with basic packages, so you may need to install certain packages to execute the commands in the .pdf. You can install Cygwin packages by running the Cygwin setup again (search for setup-x86_64.exe with the Cygwin logo on the Windows search bar). Hit 'Next' until you get to the following screen. Switch the View to 'Full' and in the search bar, you will need to add the following: cmake, cmake-gui, make, xinit, and xorg-server (https://x.cygwin.com/docs/ug/setup.html for info on X forwarding in Cygwin).




Cygwin Setup - Select Packages
Select Packages
Select packages to install
View Full
Search cmake
Clear
Keep
Best
Sync
Test
Current New Bin? Src? Categories Size Package
3.6.2-1 3.3.2-1 Devel 9,997k cmake: Cross-platform makefile generation system
Skip n/a n/a Debug 123,232k cmake-debuginfo: Debug info for cmake
Skip n/a n/a Devel 869k cmake-doc: Cross-platform makefile generation system (documentation)
3.6.2-1 3.3.2-1 Devel 7,648k cmake-gui: Cross-platform makefile generation system (GUI)
Skip n/a n/a Editors 5k emacs-cmake: Cross-platform makefile generation system (Emacs mode)
Skip n/a n/a Devel 281k extra-cmake-modules: Extra CMake Modules for KDE
Skip n/a n/a Devel 6k gccmakedep: X Makefile dependency tool for GCC
Hide obsolete packages
< Back
Next >
Cancel