greedy Documentation

Release 1.0.1

Paul Yushkevich

May 18, 2023

Contents:

Greedy is a tool for fast medical image registration. It was developed by Paul Yushkevich at the *Penn Image Computing and Science Lab* at the University of Pennsylvania. The motivation for developing greedy was to have a really fast CPU-based deformable image registration tool that could be used in applications where many images have to be registered in parallel - like multi-atlas image segmentation.

CHAPTER 1

What is Greedy?

Greedy is a tool for fast medical image registration. It was developed by Paul Yushkevich at the *Penn Image Computing and Science Lab* at the University of Pennsylvania. The motivation for developing greedy was to have a really fast CPU-based deformable image registration tool that could be used in applications where many images have to be registered in parallel - like multi-atlas image segmentation.

Greedy shares multiple concepts implementation strategies with *Symmetric Normalization (SyN) in ANTS*. But greedy is non-symmetric, which makes it faster (in applications like multi-atlas segmentation, symmetric property is not required). Greedy also uses highly optimized code for image metric computation that adds extra speed. This work is funded by the NIH/NIBIB under grants R01 EB-017255 and R01 EB-014146

CHAPTER 2

Installation

2.1 Compiling from Source Code

Building from source code is the best way to obtain the latest version of Greedy and it works on all platforms. The instructions below are for Linux and MacOS.

2.1.1 Prerequisites

1. ITK 5.1.2 or later
2. CMake 3.16 or later (and basic familiarity with this tool)
3. Git (and basic familiarity with this tool)

2.1.2 Building Greedy

Set up the directory tree and clone greedy repository:

```
cd my_directory
git clone https://github.com/pyushkevich/greedy greedy
mkdir build
```

At this point, `my_directory/greedy` contains the source code and `my_directory/build` is where we will compile greedy.

Run `ccmake` from the build directory:

```
cd build
ccmake ../greedy
```

When running `ccmake`, set the following variables:

Variable	Value
ITK_DIR	Point to the directory where you compiled ITK
CMAKE_BUILD_TYPE	Release (unless you want to debug)
USE_FFTW	OFF (you can set it to on, but it's not going to affect greedy)

After you configure and generate makefiles in `ccmake`, run:

```
make
```

You should now find the executable `greedy` created in the `build` directory. You can test that Greedy was compiled by running:

```
./greedy -v
```

2.2 Using Pre-compiled Binaries

- Nightly builds of greedy are regularly updated at https://sourceforge.net/projects/greedy-reg/files/
- As of late 2018, greedy is part of ITK-SNAP software. Simply download ITK-SNAP 3.8 or later for your platform and run `Help->Install Command-Line Tools` from the ITK-SNAP main menu.

CHAPTER 3

Quick Start

In this example, we will register brain MRI scans from two different individuals. One MRI will be designated as "fixed" (`fixed.nii.gz`) and the other as "moving" (`moving.nii.gz`). We also assume that we have a multi-label segmentation of the moving image, called `moving_seg.nii.gz`.

To run the code in this section, you must make sure that the executable `greedy` is in your system path. On Linux and MacOS, run:

```
> export PATH=/path/to/directory/containing/greedy:$PATH
```

On Windows, run:

```
> PATH=c:\path\to\directory\containing\greedy;%PATH%
```

3.1 Affine Registration

The two brains are in different physical locations. We first perform affine registration to correct for differences in brain position, rotation, and size:

```
> greedy -d 3 -a \
    -m WNCC 2x2x2 \
    -i fixed.nii.gz moving.nii.gz \
    -o affine.mat \
    -ia-image-centers -n 100x50x10
```

This call to greedy uses some of the most common options:

- `-d 3` specifies the dimensionality of the problem (3D registration)
- `-a` specifies that we are performing affine registration
- `-i` specifies the fixed/moving image pair
- `-m WNCC 2x2x2` specifies the image dissimilarity metric. Greedy will use the **weigthed normalized cross-correlation** metric with a 2x2x2 patch radius (patch size 5x5x5). The weighted metric better accounts for

non-overlapping regions between the fixed and moving images during registration and is recommended over the NCC metric, especially for affine registration.

- `-o` specifies the file where the output affine transformation (a matrix) will be written
- `-ia-image-centers` specifies that the affine transform is initialized by matching the centers of the images. This is useful when your images do not occupy the same physical space.
- `-n 100x50x10` instructs greedy to run for 100 iterations at the lowest resolution level, 50 at intermediate resolution and 10 at full resolution.

The result of the registration is the file `affine.mat`. We can view the contents of this file:

```
> cat affine.mat
```

So far, we have not actually applied the registration to the moving image; this requires a separate call to greedy, as discussed below.

3.2 Deformable Registration

We can now perform deformable registration between the fixed and moving images:

```
> greedy -d 3 \
    -m WNCC 2x2x2 \
    -i fixed.nii.gz moving.nii.gz \
    -it affine.mat \
    -o warp.nii.gz \
    -oinv inverse_warp.nii.gz \
    -sv -n 100x50x10
```

This is pretty similar to the affine command. Notice the differences:

- We are no longer supplying the `-a` flag. This activates the default deformable registration mode.
- `-it` is used to specify the initial transformation of the moving image. We pass the affine registration output as the initial transformation.
- `-o` is used to specify a filename of the image where the resulting dense deformable transformation (warp) will be stored.
- Optional command `-oinv` is used to tell greedy to approximate the inverse warp. Inverse warps are rarely needed in practical applications.
- Optional command `-sv` enables the *log-Demons* deformable registration algorithm (Vercauteren et al., 2008), which results in better diffeomorphic properties of the output warp and inverse warp. There is a very small additional computational cost associated with this algorithm that is almost always worth it.

After you run the command above, the `warp.nii.gz` file will be generated. You can view it in ITK-SNAP or other viewer. It is a 3D image volume where each voxel stores the *x,y,z* components of a displacement vector. *For every voxel in the fixed image space, the warp image specifies the displacement that maps that voxel's center to the corresponding location in the moving image.*

3.3 Applying warps to images and segmentations

So far we have just computed the affine and deformable transformation between the fixed image and the moving image. We now need to apply these transformations to the moving image. We also have a segmentation of the moving image,

`moving_seg.nii.gz`, which we would also like to warp into the fixed image space. We do this using the greedy's **reslice mode** (`-r` commands):

```
> greedy -d 3 \
  -rf fixed.nii.gz \
  -rm moving.nii.gz resliced.nii.gz \
  -ri LABEL 0.2vox \
  -rm moving_seg.nii.gz resliced_seg.nii.gz \
  -r warp.nii.gz affine.mat
```

Let's deconstruct this command:

- `-rf` specifies the reference/fixed image space. Images will be re-sliced into this space.
- `-rm` specifies an input/output pair. The input is in the moving image space, the output will contain the image resliced (warped) into the fixed image space. Multiple `-rm` commands can be provided as above.
- `-ri` specifies the interpolation mode for subsequent `-rm` commands. Default interpolation is `LINEAR` interpolation. When warping multi-label segmentations, greedy provides a special `LABEL` interpolation mode. This mode applies a little bit of smoothing to each label in your segmentation (including the background), warps this smoothed segmentation, and then performs voting among warped smoothed binary segmentations to assign each voxel in reference space a label. This works better than nearest neighbor interpolation (less aliasing). The `0.2vox` part of the command specifies the amount of smoothing.
- `-r` is used to specify a sequence of transformations, from last to first.

3.4 Applying the Inverse Warp

Optionally, you can warp the fixed image back into the space of the moving. This requires using the (`-oinv`) option when calling greedy for deformable registration. You can also invert a warp computed previously using the `-iw` command (see documentation).

The following command warps the fixed image to the moving image space:

```
> greedy -d 3 \
    -rf moving.nii.gz \
    -rm fixed.nii.gz reslice_fix_into_mov.nii.gz \
    -r affine.mat,-1 inverse_warp.nii.gz
```

The call to greedy is very similar to above, except that:

- the roles of fixed and moving is switched (obviously)
- the affine and deformable transformations are provided in reverse order
- the affine transformation is inverted (the -1 after `affine.mat`)

That's pretty much it for learning to use greedy. See detailed documentation for other options and more details. Enjoy!

CHAPTER 4

Reference

4.1 Greedy Usage

```
greedy: Paul's greedy diffeomorphic registration implementation
Usage:
  greedy [options]
Required options:
  -d DIM                  : Number of image dimensions
  -i fix.nii mov.nii      : Image pair (may be repeated)
  -o <file>               : Output file (matrix in affine mode; image in deformable␣
↪mode,
                            metric computation mode; ignored in reslicing mode)
Mode specification:
  -a                      : Perform affine registration and save to output (-o)
  -brute radius           : Perform a brute force search around each voxel
  -moments <1|2>          : Perform moments of inertia rigid alignment of given order.
                                order 1 matches center of mass only
                                order 2 matches second-order moments of inertia tensors
  -r [tran_spec]          : Reslice images instead of doing registration
                                tran_spec is a series of warps, affine matrices
  -iw inwarp outwarp      : Invert previously computed warp
  -root inwarp outwarp N  : Convert 2^N-th root of a warp
  -jac inwarp outjac      : Compute the Jacobian determinant of the warp
  -metric                 : Compute metric between images
Options in deformable / affine mode:
  -w weight               : weight of the next -i pair
  -m metric               : metric for the entire registration
                                SSD:          sum of square differences (default)
                                MI:           mutual information
                                NMI:          normalized mutual information
                                NCC <radius>: normalized cross-correlation
                                MAHAL:        Mahalanobis distance to target warp
  -e epsilon              : step size (default = 1.0),
                                may also be specified per level (e.g. 0.3x0.1)
```

(continues on next page)

(continued from previous page)

```
  -n NxNxN                : number of iterations per level of multi-res (100x100)
  -threads N              : set the number of allowed concurrent threads
  -gm mask.nii            : fixed image mask (metric gradients computed only over the␣
↪mask)
  -gm-trim <radius>       : generate the fixed image mask by trimming the extent
                            of the fixed image by given radius. This is useful during␣
↪affine
                            registration with the NCC metric when the background of␣
↪your images
                            is non-zero. The radius should match that of the NCC␣
↪metric.
  -mm mask.nii            : moving image mask (pixels outside are excluded from metric␣
↪computation)
Defining a reference space for registration (primarily in deformable mode):
  -ref <image>            : Use supplied image, rather than fixed image to define the␣
↪reference space
  -ref-pad <radius>       : Define the reference space by padding the fixed image by␣
↪radius. Useful when
                            the stuff you want to register is at the border of the␣
↪fixed image.
  -bg <float|NaN>         : When mapping fixed and moving images to reference space,␣
↪fill missing values
                            with specified value (default: 0). Passing NaN creates a␣
↪mask that excludes
                            missing values from the registration.
  -it filenames           : Specify transforms (matrices, warps) that map moving image␣
↪to reference space.
                            Typically used to supply an affine transform when running␣
↪deformable registration.
                            Different from -ia, which specifies the initial transform␣
↪for affine registration.
Specific to deformable mode:
  -tscale MODE            : time step behavior mode: CONST, SCALE [def], SCALEDOWN
  -s sigma1 sigma2        : smoothing for the greedy update step. Must specify units,
                            either `vox` or `mm`. Default: 1.732vox, 0.7071vox
  -oinv image.nii         : compute and write the inverse of the warp field into image.
↪nii
  -oroot image.nii        : compute and write the (2^N-th) root of the warp field into␣
↪image.nii, where
                            N is the value of the -exp option. In stational velocity␣
↪mode, it is advised
                            to output the root warp, since it is used internally to␣
↪represent the deformation
  -wp VALUE               : Saved warp precision (in voxels; def=0.1; 0 for no␣
↪compression).
  -noise VALUE            : Standard deviation of white noise added to moving/fixed␣
↪images when
                            using NCC metric. Relative to intensity range. Def=0.001
  -exp N                  : The exponent used for warp inversion, root computation,␣
↪and in stationary
                            velocity field (Diff Demons) mode. N is a positive integer␣
↪(default = 6)
  -sv                     : Performs registration using the stationary velocity model,␣
↪similar to diffeomoprhic
                            Demons (Vercauteren 2008 MICCAI). Internally, the␣
↪deformation field is
                            represented as 2^N self-compositions of a small␣
↪deformation and
```

(continues on next page)

(continued from previous page)

```
                          greedy updates are applied to this deformation. N is␣
↪specified with the -exp
                          option (6 is a good number). This mode results in better␣
↪behaved
                          deformation fields and Jacobians than the pure greedy␣
↪approach.
  -svlb                 : Same as -sv but uses the more accurate but also more␣
↪expensive
                          update of v, v <- v + u + [v,u]. Experimental feature
  -sv-incompr           : Incompressibility mode, implements Mansi et al. 2011␣
↪iLogDemons
  -id image.nii         : Specifies the initial warp to start iteration from. In␣
↪stationary mode, this
                          is the initial stationary velocity field (output by -oroot␣
↪option)
Initial transform specification (for affine mode):
  -ia filename          : initial affine matrix for optimization (not the same as -
↪it)
  -ia-identity          : initialize affine matrix based on NIFTI headers
  -ia-image-centers     : initialize affine matrix based on matching image centers
  -ia-image-side CODE   : initialize affine matrix based on matching center of one␣
↪image side
  -ia-moments <1|2>     : initialize affine matrix based on matching moments of␣
↪inertia
Specific to affine mode (-a):
  -dof N                : Degrees of freedom for affine reg. 6=rigid, 12=affine
  -jitter sigma         : Jitter (in voxel units) applied to sample points (def: 0.5)
  -search N <rot> <tran> : Random search over rigid transforms (N iter) before␣
↪starting optimization
                          'rot' may be the standard deviation of the random rotation␣
↪angle (degrees) or
                          keyword 'any' (any rotation) or 'flip' (any rotation or␣
↪flip).
                          'tran' is the standard deviation of the random offset, in␣
↪physical units.
Specific to moments of inertia mode (-moments 2):
  -det <-1|1>           : Force the determinant of transform to be either 1 (no␣
↪flip) or -1 (flip)
  -cov-id               : Assume identity covariance (match centers and do flips␣
↪only, no rotation)
Specific to reslice mode (-r):
  -rf fixed.nii         : fixed image for reslicing
  -rm mov.nii out.nii   : moving/output image pair (may be repeated)
  -rs mov.vtk out.vtk   : moving/output surface pair (vertices are warped from fixed␣
↪space to moving)
  -ri interp_mode       : interpolation for the next pair (NN, LINEAR*, LABEL sigma)
  -rb value             : background (i.e. outside) intensity for the next pair␣
↪(default 0)
  -rc outwarp           : write composed transforms to outwarp
  -rj outjacobian       : write Jacobian determinant image to outjacobian
Specific to metric computation mode (-metric):
  -og out.nii           : write the gradient of the metric to file
For developers:
  -debug-deriv          : enable periodic checks of derivatives (debug)
  -debug-deriv-eps      : epsilon for derivative debugging
  -debug-aff-obj        : plot affine objective in neighborhood of -ia matrix
  -dump-pyramid         : dump the image pyramid at the start of the registration
```

(continues on next page)

(continued from previous page)

```
  -dump-moving           : dump moving image at each iter
  -dump-freq N           : dump frequency
  -dump-prefix <string>  : prefix for dump files (may be a path)
  -powell                : use Powell's method instead of LGBFS
  -float                 : use single precision floating point (off by default)
  -version               : print version info
  -V <level>             : set verbosity level (0: none, 1: default, 2: verbose)
Environment variables:
  GREEDY_DATA_ROOT       : if set, filenames can be specified relative to this path
```

4.2 Greedy modes

Greedy offers a number of **modes**. Each mode, except the default deformable registration mode, is entered by specifying the corresponding switch somewhere on the command line.

The most commonly used modes are * Deformable registration (default) * Affine/rigid registration (`-a`) * Reslicing and transform composition mode (`-r`) * Metric computation mode (`-metric`) * Matching by moments mode (`-moments`)

Some additional, less frequently used modes are * Warp inversion mode (`-iw`, deprecated in favor of ``-sv` and `-oinv` commands) * Warp root approximation mode (`-root`, deprecated in favor of `-sv` and `-oroot` commands) * Warp jacobian approximation mode (`-root`, deprecated in favor of `-sv` and `-rj` commands) * Brute force deformation search mode (`-brute`, abandoned feature)

The five more common modes are described in greter detail below.

4.2.1 Deformable registration mode

The deformable registration mode is the default mode in greedy. If you don't specify any mode setting switches, this mode will be used. Here are some general notes on deformable registration:

- In deformable registration mode, the fixed and moving images, specified with the `-i` command, are matched by computing a spatial transformation (warp) that deforms the moving image into the fixed image. This warp is defined in the space of the fixed image. Specifically, the warp describes the displacement at each position in the fixed image at which the corresponding position in the moving image is found. The warp is saved as a multi-component image with the `-o` command, and you can also save the inverse warp (`-owarp`).
- In deformable registration mode, the fixed and moving images are assumped to occupy the same space. In fact, the moving image is automatically resampled to the fixed image space. You can specify a spatial transformation between the fixed image space and the moving image space with the `-it` command. Sometimes it is useful to perform registration is a space that is different from the fixed image space (e.g., a larger space). This can be done by supplying the reference space `-ref` or defining the reference space by padding the fixed image `-ref-pad`.
-

4.3 General Options

4.3.1 Image dimensionality (`-d`)

- Format: `-d <2|3>`
- Required

- Available in all modes

Specifies whether registration or other operations should be performed in two or three dimensions.

4.3.2 Number of parallel threads (`-threads`)

- Format: `-threads <number>`
- Available in all modes

By default, greedy will run in multithreaded mode, using all of your available CPU cores. You can restrict the number of cores used to any given number, or set to zero to use the default behavior. On many clusters, the `NSLOTS` environment variable is defined and can be used to set the number of threads correctly:

```
greedy -d 3 -threads ${NSLOTS-0} ...
```

4.3.3 Floating point precision (`-float`)

- Format: `-float`
- Available in all modes

By default, greedy uses double precision floating point to represent images and transformations in memory. This option uses single-precision instead. This is faster and uses less memory, but at some small loss of precision (especially during NCC metric computation). *We recommend not using this option, as double precision floating point has been tested far more extensively*.

4.3.4 Verbosity (`-V`)

- Format: `-V <0|1|2>`
- Available in all modes
- Default: `1`

Sets the verbosity of the program's output (0: quiet, 1: default, 2: extra verbose).

4.3.5 Command-line help (`-h`)

- Format: `-h`

Use this command to list all the commands and options for greedy. Some commands are esoteric or developer-oriented and are not discussed here.

4.4 Mode Specification Commands

Greedy has several operating modes. The most common modes are deformable registration (default mode), affine registration (`-a` command) and reslicing mode (`-rXXX` commands).

4.4.1 Deformable Registration Mode

The deformable registration mode is the default mode in greedy. If you don't specify and mode setting parameters, this mode will be used.

4.4.2 Affine Registration Mode (-a)

Used to perform affine and rigid registration. In this mode, the

4.5 Common Commands in Deformable Registration Mode

4.5.1 Input Image Pair and Weight Specification (-i, -w)

-i <fixed_image> <moving_image>

-w <weight>

This command specifies the fixed/moving image pair. Multiple such commands can be provided, in which case there will be multiple fixed and multiple moving images. However, all the fixed images must be in the same physical space, as must be all the moving images. You can use the **-w** command to assign different weights to different fixed/moving pairs. Note that the **-w** command applies to all subsequent **-i** commands.

> greedy -d 3 \

-w 0.25 -i fixed_t1.nii moving_t1.nii \

-w 0.75 -i fixed_t2.nii moving_t2.nii \

. . .

The fixed and moving images may also be multi-component images (e.g, images of vectors or tensors).

4.5.2 Output Warp Specification: (-o)

-o <warp_image>

Specifies where the warp image will be stored. The warp image will be in the same space as the fixed image and will have three components per pixel. The warp image is specified as follows. Suppose that A is a voxel coordinate in the fixed image and B is a voxel coordinate in the moving image, and that registration matched A to B. Then

ras(A) + warp[A] = ras(B)

where ras(A) is the physical coordinate of voxel coordinate A in the RAS coordinate space (space used by NIFTI).

4.5.3 Metric Specification (-m)

-m <SSD | NMI | NCC <radius> >

Specifies the image similarity metric used for the registration. Greedy does not allow mixing multiple metrics in the same registration (weighting multiple metrics in non-trivial anyway). So the position of the command on the command line does not matter.

Three metrics are supported:

- **Sum of squared differences (SSD) - fastest but only suitable for** same-modality registration where intensity ranges of the fixed and moving images are the same. For example, two CT scans. This metric just tries to match the intensity of the fixed and moving images at every voxel.
- **Normalized cross-correlation (NCC) - relatively fast too, but more** robust to noise and intensity differences. Tries to maximize the correlation coefficient between the neighborhood of each voxel in the fixed image and the corresponding neighborhood in the moving image. The size of the neighborhood is specified

by **<radius>**. For example **NCC 2x2x2** specifies a 5x5x5 neighborhood. Note that there is almost to performance cost for using larger radii due to efficient implementation.

- **Normalized mutual information (NMI) - should be used when intensity** spaces of the moving and fixed images are very different, e.g., registering T1-MRI to T2-MRI. Does not work very well for deformable registration, better for affine/rigid.

> greedy -d 3 \

-m NCC 4x4x4 -i fixed_t1.nii moving_t1.nii \

. . .

4.5.4 Initial Transformations (-it)

-it <transform> [transform] . . .

Provides a chain of transformations (affine matrices, warps) to apply to the moving image before registration. This is equivalent to first reslicing the moving image into the fixed image space using the same chain of transformations (**-r** command). The most common scenario is to provide the output of affine/rigid registration to the **-it** command.

> greedy -d 3 \

-it affine.mat -i fixed_t1.nii moving_t1.nii \

. . .

4.5.5 Fixed Image Mask (-gm)

**-gm <mask_image> **

Specifies a mask that restricts registration to a region of the fixed image. This can make registration faster and more robust and is highly recommended, particularly when there is a lot of intensity variation along the boundaries of the fixed image. The mask image is typically a binary image, but a soft mask can also be provided.

4.5.6 Multi-resolution schedule (-n)

**-n <iteration_spec> **

Specify how many iterations of registration to do at each iteration level. For example **-n 100x40x20** does three levels of super-resolution (4x, 2x and 1x) and does 100 iterations at 4x (coarsest level), 40 iterations at 2x (intermediate) and 20 iterations at 1x (full resolution).

4.5.7 Inverse warp output (-oinv, -invexp)

-oinv <warp_image>

**-invexp <exponent> **

Unlike symmetric normalization (SyN), greedy does not compute the inverse of the deformation field at each iteration of image registration. However, you can still generate an inverse warp post-hoc. This uses the fixed point method of warp inversion [reference!]. This adds some extra time at the end of the registration.

To improve the performance of the inverse algorithm, the forward warp is first taken to a power -2, -4, -8, etc. In other words, we find a warp psi, such that psi(psi(. . . psi(psi(x)))) = warp(x). The **exponent** parameter to **-invexp** is used to specify the power, with power = 2^-exponent. Default value is 2. If you get bad (self-intersecting) inverse warps, try a larger value.

> greedy -d 3 \

. . . -invexp 4 -oinv inverse_warp.nii.gz

4.6 Deformable Registration Parameters

4.6.1 Smoothing Kernels (-s)

-s <gradient_sigma> <warp_sigma>

Probably the most crucial parameter for deformable registration. This specifies the amount of regularization applied to the deformation field during registration. Just like in SyN (and in Demons registration before that), there are two types of regularization applied:

- **Metric gradient regularization: this is applied to smooth the** gradient of the image match metric at each iteration. The smoothed gradient is used to update the current estimate of the warp via composition. Larger values of smoothing (**gradient_sigma**) result is smoother deformation fields.
  - **The default value of gradient_sigma is 1.732vox (square root** of 3). This default matches the default in SyN.
- **Warp regularization: the entire warp field is smoothed after each** iteration. This dampens the overall deformation. Larger values of **warp_sigma** give smaller deformations.
  - **The default value of gradient_sigma is 0.707vox (square root** of 0.5). This default matches the default in SyN.

Both sigmas can be provided in voxel units (suffix **vox**) or physical units (suffix **mm**).

> greedy -d 3 -s 2mm 0.7mm . . .

> greedy -d 3 -s 1.5x1.8x2.0vox 0.2vox . . .

4.6.2 Step Size (-e) and time step scaling (-tscale)

-e <step_spec>

-tscale <SCALE | SCALEDOWN | CONST>

Command **-e** specifies the "time step" size used to update the warp at each iteration. Larger values can speed up registration but can also cause deformation to become non-diffeomorphic. The default value is 1.0, and typical values are in the 0.25 to 0.5 range.

> greedy -d 3 -e 0.5 . . .

> greedy -d 3 -n 100x40x20 -e 1.0x0.5x0.2 . . .

The second form of the command specifies different step size for each multi-resolution level. This has not proven useful in my experience.

By default, the time step is applied after scaling the smoothed metric gradient so that the norm of the largest gradient across the whole image is 1 voxel. This behavior can be modified with the **-tscale** command, but this is not recommended. Other options are **SCALEDOWN** (where the gradient is only scaled down to have maximum norm 1 but never up) and **CONST** (the gradient is never scaled, so you have to set your time step extremely carefully).

4.6.3 Warp field precision (-wp)

-e <real_value>

Warp fields have great potential to take over disk space. By default, greedy stores warp fields only to the precision of 1/10 of voxel size. In most applications, there is no real difference to warping an image by 2.2 voxels or 2.24 voxels. By lowering precision, you can achieve much better compression when storing warp files in **.nii.gz** and other compressed formats. You can change the precision from the default 0.1 (1/10 voxel) to full precision (0) or any other value between 0 and 1.

> greedy -d 3 -wp 0.01 ...

4.7 Affine and Rigid Registration

4.7.1 Affine mode (-a, -dof)

-a

-o <affine_matrix>

-dof <6|7|12>

Calling greedy with **-a** command switches the tool to affine/rigid mode. Affine/rigid mode can not be combined with deformable mode in the same command.

By default, full affine registration is performed (12 degrees of freedom in 3D). To use rigid registration, pass in **-dof 6**. To use rigid + uniform scaling, use **-dof 7**.

In affine mode, many of the same options as in deformable mode are used, with some minor differences.

- **-o command will write out a matrix encoding the affine transform.** This is a N+1 x N+1 matrix that maps voxels in fixed image space to voxels in moving image space. Specifically, if voxel coordinate A in the fixed image corresponds to voxel coordinate B in the moving image, then

[ras(B); 1] = Matrix * [ras(B); 1]

- **If you wish to convert the matrix file to a different format or** perform various operations on matrix files, use the **c3d_affine_tool** in *Convert3D*.
- **-i, -w, -m, -n, -gm behave the same way as in defor** mable mode.
- **-ia or -ia-identity should be used to initialize affine** registration (instead of **-it** in deformable mode)
- **-s** and **-e** have no effect.
- **-oinv is not supported. If you want to invert the affine** transformation, use the **c3d_affine_tool** in *Convert3D*.

Typical example of rigid registration:

> greedy -d 3 **-a** \

-i fixed.nii.gz moving.nii.gz \

-gm fixed_mask.nii.gz \

-ia-identity \

-dof 6 -o rigid.mat \

-n 100x50x0 -m NCC 4x4x4

4.7.2 Initial transform specification for affine/rigid mode

-ia <affine_matrix>

-ia-identity

-ia-image-centers

You can initialize rigid/affine registration with a given matrix or with the identity matrix. Using the identity matrix will initialize the image alignment based on image headers (i.e., assume that ras(A) = ras(B)). Command **-ia-image-centers** matches image centers (by translation).

If **neither** of these three options is given, images are initialized based on voxel coordinates, rather than on image headers. This can result in registration failures for many images.

4.7.3 Affine initialization via random search (-search)

-search <N_iter> <sigma_angle> <sigma_offset>

This command will randomly sample **N_iter** starting positions for affine registration and start optimization from the best position found. Random sampling generates rigid transformations of the moving image. The **sigma** parameters specify the range of the angles of rotation (in degrees) and range of the offset (in voxels).

> greedy -d 3 -a \

-i fixed.nii.gz moving.nii.gz \

-gm fixed_mask.nii.gz \

-ia-identity \

-dof 6 -o rigid.mat \

-n 100x50x0 -m NCC 4x4x4 \

-search 1000 10 20

4.7.4 Random jitter (-jitter)

-jitter <real_value>

Affine registration tends to converge better when the sample locations where metric is calculated are randomly displaced from voxel centers (this avoids spurious local minima). By default a random jitter with range [-0.5 0.5] is applied to the voxel coordinates where images are sampled. For faster initialization, set jitter to 0.0.

4.8 Image Reslicing Mode

The image reslicing mode is used to apply warps and affine matrices to images. It can also be used to compose multiple transforms into a single transform, and to apply warps to meshes. Reslicing mode is activated when the **-r** command is used. Reslicing mode cannot be combined with registration in the same command line.

- See examples under *Quick Start*

4.8.1 Reference (fixed) space specification (-rf)

-rf <reference_image>

Specify the reference image for the reslicing. All images will be resliced into the space of the reference image.

4.8.2 Input/output pair specification (-rm)

-rm <input_image> <output_image>

Specify an image to be resliced and the corresponding output image. You can have any number of **-rm** commands in the same command line. The input images provided to **-rm** commands do not have to be in the same physical space. They can be scalar or multi-component images.

4.8.3 Interpolation mode specification (-ri)

-ri <NN | LINEAR | LABEL <sigma_spec> >

Specify the interpolation mode to use for reslicing. This command only affects the subsequent **-rm** commands on the command line (so should precede the **-rm** command you want it to affect). The following modes are available;

- **Nearest neighbor (NN): rarely recommended, results in the most** aliasing
- **Bilinear/trilinear interpolation (LINEAR): default interpolation** mode, fast and less aliasing
- **Label interpolation (LABEL): this special mode is used for** warping/reslicing multi-label segmentations. This mode applies a little bit of smoothing to each label in your segmentation (including the background), warps this smoothed segmentation, and then performs voting among warped smoothed binary segmentations to assign each voxel in reference space a label. This works better than nearest neighbor interpolation (less aliasing).
  - **The <sigma_spec> parameter to the -ri LABEL command** specifies the standard deviation of the Gaussian kernel used to smooth the labels. It can be provided in voxel units (e.g., 0.2vox) or millimeter units (e.g., 0.2mm). Value of 0.2vox works well in most situations.

> greedy -d 3 \

-rf reference.nii \

-ri LINEAR \

-rm t1mri.nii.gz warped_t1mri.nii.gz \

-ri LABEL 0.2vox \

-rm segmentation.nii.gz warped_seg.nii.gz \

-r . . .

4.8.4 Transform chain specification (-r)

-r <transform_spec> [transform_spec] . . .

Specify the chain of transforms to be applied to the moving image. The last transform is applied first. In most cases, you would do affine registration followed by deformable registration. To reslice your original moving image into the space of the fixed image you would use the command

> greedy -d 3 \

-rf fixed.nii.gz \

-rm moving.nii.gz resliced.nii.gz \

-r warp.nii.gz affine.mat

So the moving image will first be transformed by the affine transform, and then by the warp. Or in other words, if A is a voxel coordinate in fixed image space, then the corresponding voxel coordinate B in the moving image is found according to

ras[B] = warp(affine(ras[A])

- It is a common error to provide transforms in the wrong order.
- **You can provide as many transforms as you wish - it is possible to** chain a dozen transforms.
- **To specify that the affine transform should be inverted, use affine.mat,-1** syntax.
  - **For example, to reslice the fixed image into the space of the** moving image in the above example, use

> greedy -d 3 \

-rf moving.nii.gz \

-rm fixed.nii.gz resliced_backwards.nii.gz \

-r affine.mat,-1 inverse_warp.nii.gz

- Note that the order of transforms has switched. This is because

ras[A] = inverse_affine(inverse_warp((ras[B])

4.8.5 Composing transformations (-rc)

-rc <warp_image>

In addition (or instead of) reslicing images, you can use the reslice mode to compose multiple transforms or to convert an affine transform into the corresponing (linear) warp image. For example:

> greedy -d 3 \

-rf fixed.nii.gz \

-r warp1.nii.gz warp2.nii.gz affine.mat \

-rc composite_warp.nii.gz

The **-rc** command can be used on the same command line with **-rm** and **-rs** commands.

4.8.6 Warping meshes and point sets (-rs)

-rs <input_mesh> <output_mesh>

The transform chain specified with -r can be applied to points in a mesh. However, whereas image intensities are mapped from moving space into fixed space, coordinates of vertices are mapped from fixed space to moving space.

> greedy -d 3 \

-rf fixed.nii.gz \

-rm moving.nii.gz resliced.nii.gz \

-rs fixed_mesh.vtk output_mesh.vtk \

-r warp1.nii.gz warp2.nii.gz affine.mat

4.9 Matching by Moments of Inertia

4.9.1 Moments mode (-moments)

-moments <1|2>

-o <moments_matrix>

Matching by moments can be an effective strategy when two images have similar content, but are so misaligned that the affine and rigid modes fail. Matching by moments is particularly useful for binary objects, e.g. two hippocampal segmentations. Matching by moments line up the centers of mass of the two images, and (optionally) match the second momentum tensors.

- **If the argument to -moments is 1, only centers of mass are** matched.
- **If the argument to -moments is 2, the second moment tensors are** also matched.
  - **These is ambiguity with respect to reflection when matching the** second tensors. Greedy will consider all possible reflections (e.g., in 3D there are 8 possible reflections) and choose the one that minimizes the metric between fixed and moving images.

The output is a matrix file, just as in affine and rigid registration. However, unlike rigid and affine modes, the matrix may also include a coordinate flip (reflection).

4.9.2 Restricting flipping in moments mode (-det)

-det <1|-1>

For a 3D image there are 8 ways to line up second momentum tensors, since the direction is along each momentum vector is arbitrary. Four of these ways involve flipping, and four do not. By default, the alignment of tensors that gives the best metric value is used. However, you can force flipping to always occur (e.g., when you know that you are matching a left hippocampus mask to a right hippocampus mask) by setting **-det -1**. Likewise you can prevent flipping by setting **-det 1**. This option has no effect when using **-moments 1**.

4.9.3 Disabling rotation in moments mode (-cov-id)

-cov-id

This option sets the second moment tensors to have identity covariance, which means the matching will not perform any rotation, only alignment of centers of mass and flipping. Note that **-moments 2 -cov-id** will allow flipping, whereas **-moments 1** only aligns the centers of mass.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

This work is funded by the NIH/NIBIB under grants R01 EB-017255 and R01 EB-014146