GRAM User's Manual

Apr 2010

Jihun Hamm, Donghye Ye, Ragini Verma, and Christos Davatzikos
(Jihun.Hamm/Dong.Ye/Ragini.Verma/Christos.Davatzikos@uphs.upenn.edu)

SBIA, Department of Radiology, University of Pennsylvania

1. Introduction

This document describes the installation and the usage of GRAM package.

What is GRAM?

GRAM (Geodesic Registration on Anatomical Manifolds) is a framework for groupwise registration of medical images, described in the paper by Hamm et al. (MedIA 2010, submitted).

Why use GRAM?

Challenges in medical image registration

The goal of registering medical images is to find biologically plausible transformations between two images or from a template to multiple images. For the two images such as T1-weighted brain MRIs to have anatomical one-to-one correspondence after registration, the transformation has to be topology-preserving (=diffeomorphic). However, when two images differ greatly in shape, it is difficult to find a topology-preserving transformation algorithmically.

Geodesic registration on the manifold of diffeomorphism

Several approaches such as LDDMM (Large Deformation Diffeomorphic Metric Mapping) have been proposed to remedy this problem. These methods provide mathematically solid solutions to the large-deformation registration problems, by finding geodesic paths of transformations on the manifold of diffeomorphisms. However, such transformations are computationally very costly to compute. Ever more importantly, such mathematically-defined transformations may not coincide with biologically plausible transformations between real brain anatomies.

Geodesic registration on anatomical manifolds

Ideally we want to calculate geodesics on the manifold of transformations that represent only the biologically relevant variations. However, such manifold cannot be represented analytically.

GRAM is a registration framework toward this goal. The key idea of GRAM is that we approximate analytical geodesic paths with finite sequences of small deformations observed in the actual anatomies in data. In another point-of-view, we are constructing empirical manifolds from data, a technique known as manifold learning, instead of dealing with an analytical manifold of diffeomorphisms. In particular we borrow an idea from Isomap algorithm Tenenbaum et al. (2000), which replace the geodesic path of the analytical manifold by the shortest path on a k-nearest-neighbor (kNN) graph that approximates the metric structure of the empirical manifold.

What GRAM is not

GRAM is **not** an algorithm to register two images. It is a framework for registering a large number of images to a group template. GRAM uses any pairwise registration algorithm as an interchangeable component. In this package Demons is the default component for pairwise registration. The package will be extended to other component algorithms such as Freeform registration, HAMMER, or DRAMMS in the future.

2. Installing GRAM

Requirements

a) ITK: For the package to build properly, you need to have ITK library built with "review" option on (this is standard for SBIA environment). For more information on installing ITK, go to http://www.itk.org/ITK/resources/software.html. The package was test with ITK 3.14.
b) MATLAB: The package uses MATLAB scripts. The package uses Matlab Graph Library (http://www.stanford.edu/~dgleich/programs/matlab_bgl) and NIFTI toolbox (http://www.rotman-baycrest.on.ca/~jimmy/NIfTI), which are freeware. These toolboxes are included in the package. The package was tested with MATLAB R2009B, but it will probably work with older versions as well.
c) CMake: CMake (http://www.cmake.org) is used create cross-platform makefiles.
d) Compiler and linker for Windows: You can use the freeware version of Visual Studio (http://www.microsoft.com/express/). The package was test with Visual Studio 2008 Express.

Operating Systems

The package was tested on linux and Windows XP.

Source Code for GRAM

The source code for the package is in the SVN repository (https://sbia-svn.uphs.upenn.edu/projects/GRAM)

Folder Structure in SVN repository

a) Code
   a. Core: Core algorithms
   b. ITK: ITK library
   c. matlab_bgl: Matlab library for graph theoretic algorithms
   d. NIFTI: Matlab library for manipulating medical images in NIFTI formats
b) Demo:
   a. subject: example dataset of 2D brain patches
   b. result: intermediate and final registration results of the demo dataset will be saved here
c) Doc :
   a. GRAM User's Manual: This document
   b. GRAM papers: please refer to the paper when using the package

Installations

1. Building ITK-dependent binaries [linux]
   a) Install ITK with the "review" option on. Skip this step if ITK is already in the system.
   b) Go to GRAMROOT/Code/ITK , where GRAMROOT is the path for the package.
   c) Create ITK binary directory by `mkdir bin`
   d) Go to ITK binary directory by `cd bin`
   e) Run `ccmake ..` (Configure, accept warnings, configure and generate makefiles).
   f) Build by `make`, this will generate following binaries in the bin folder.

      ConcatenateFields
      WarpImage
      JacobianField
      DemonsRegistration_dong
      DemonsRegistration_dong_n

      Please check that the binaries are built without errors.
   g) Set environmental variables for your shell. If you use bash, type
      `gedit $HOME/.bashrc` and insert
      `PATH=$PATH:GRAMROOT/Code/ITK/bin` at the end. If you use csh, type
      `gedit $HOME/.cshrc` and insert
      `setenv PATH ${PATH}:GRAMROOT/Code/ITK/bin` at the end.
      Save and exit.
   h) To apply this change in the shell, either type `bash` (or `csh`) or log-in again. To check the path is correctly included, type `DemonsRegistration_dong` in the shell. If you see a list of arguments and help texts, you are all set. If not, check the path again.

2. Building ITK-dependent binaries [MS Windows 32bit]
   a) Install ITK with the "review" option on. Skip this step if ITK is already in the system.
   b) Go to GRAMROOT/Code/ITK, where GRAMROOT is the path for the package.
   c) Create ITK binary directory by mkdir bin
   d) Launch cmake-gui.exe and set the source dir as GRAMROOT/Code/ITK and set the bin dir as GRAMROOT/Code/ITK/bin
   e) Configure and generate solution files for Visual Studio.
   f) Launch Visual Studio and build the solution in Release mode. This will generate following binaries.

      ConcatenateFields

      WarpImage

      JacobianField

      DemonsRegistration_dong

      DemonsRegistration_dong_n

      Please check that the binaries are built without errors.
   g) Set environment variables: go to My Computer->Properties->Advanced->Environment Variables-> User variables. Add GRAMROOT\Code\ITK\bin\Release to the PATH variable.
   h) To check the path is correctly included, open command prompt and type DemonsRegistration_dong. If you see a list of arguments and help texts, you are all set. If not, check the path again.

3. Including Matlab Libraries [OS-independent]

   Launch MATLAB and include the three folders in the Matlab path:
   a) GRAMROOT/Code/Core
   b) GRAMROOT/Code/matlab_bgl
   c) GRAMROOT/Code/NIFTI

3. Testing GRAM

The script GRAMROOT/Code/Core/test_GRAM.m demonstrates the usage of GRAM for groupwise registration of medical images. For testing purpose, simulated 2D brain patches are included in the package.

Before running test_GRAM.m, open it in MATLAB by typing edit test_GRAM.m. You need to change the following directories in the script to your installation folder:

```
%% Directories
dirSubject = 'GRAMROOT/Demo/subject';
```

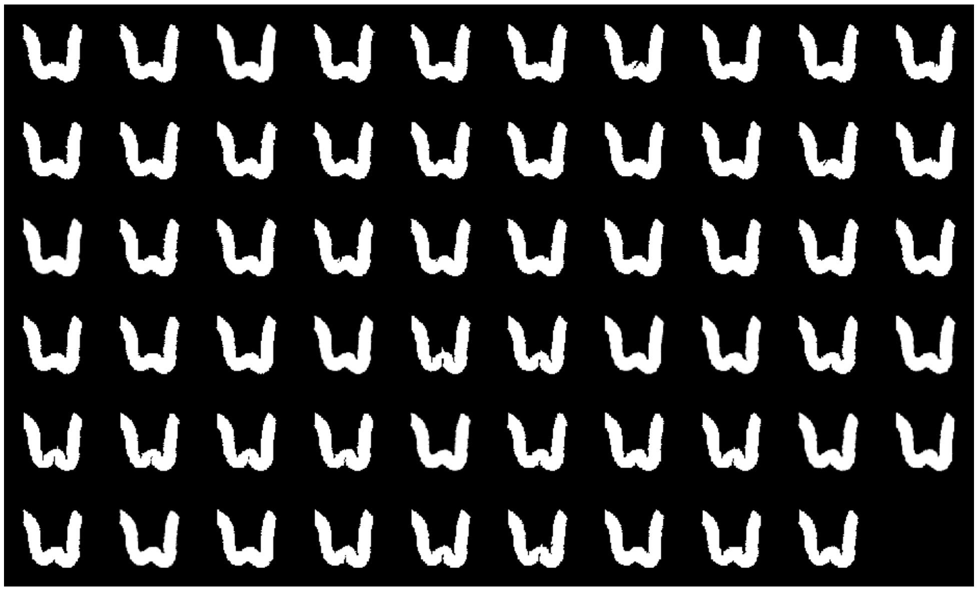
```
dirResult = 'GRAMROOT/Demo/result/geo';
dirDirectResult = 'GRAMROOT/result/direct';
```

Parameters for the algorithms can be changed. Please refer to GRAM.m or type help GRAM.m
For now, accept the default parameters in the script:

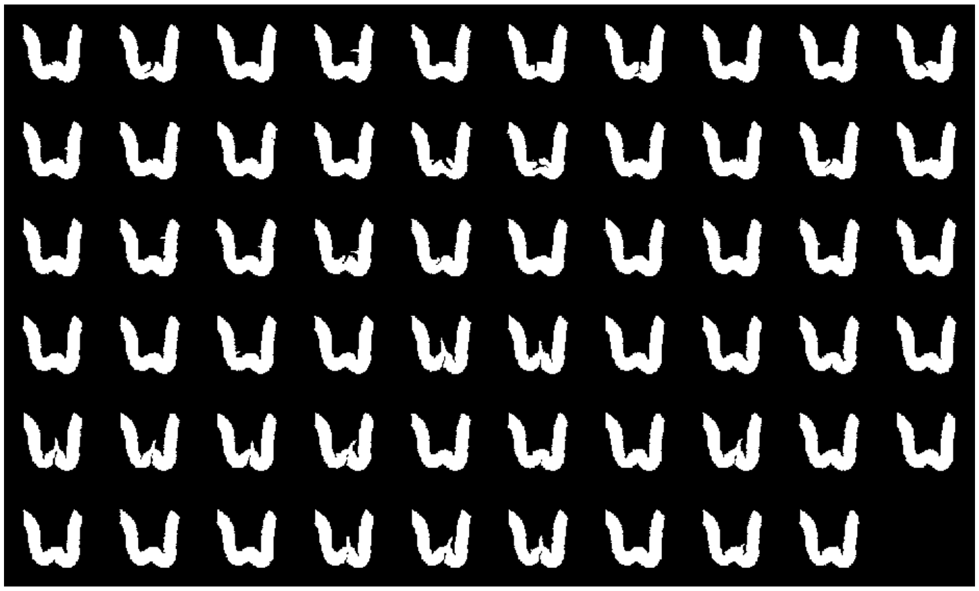
```
%% Parameter
w = .75;
NNiter = '15x10x5';
NNsmoothing = '1.5';
Fineiter = '5';
Finesmoothing = '1.5';
Directiter = '15x10x10';
Directsmoothing = '1.5';
```

Type test_GRAM to run the script. This will take less than an hour to finish. If you're connected via SSH and is automatically disconnected sooner than an hour, try launching Matlab in the background by typing **matlab &** before starting. No user input is required for test_GRAM.m. When the script has finished running without error, you will see the following four figures.

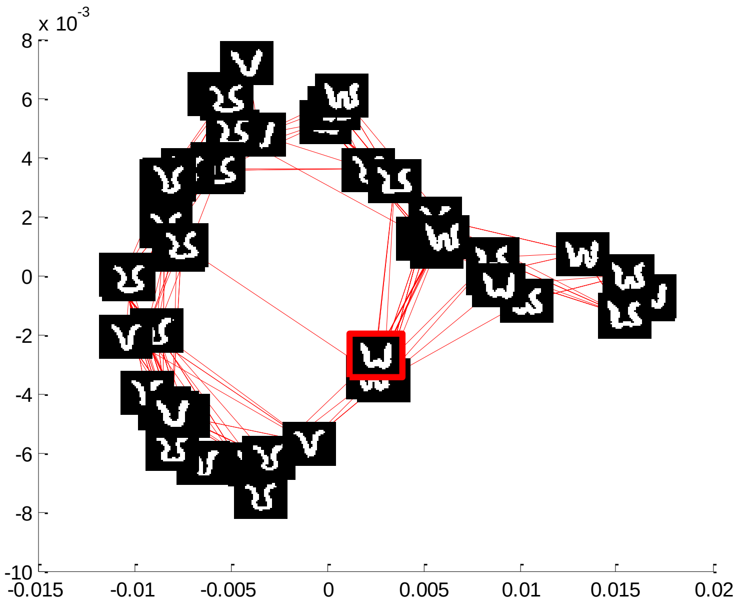
GRAM Result: These are 60 images registered to a W-shaped template by GRAM method.



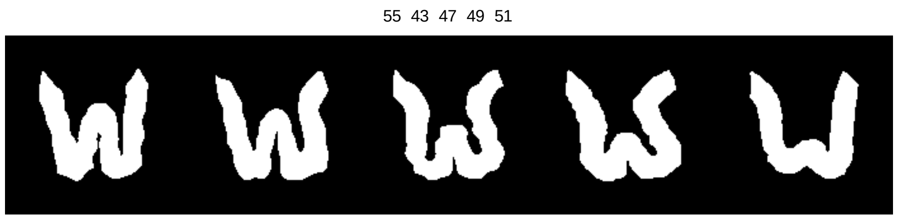
Diffeomorphic Demon Result: These are 60 images registered to the same template using Demons method directly. Compare this figure with the figure above.



ISOMAP: This figure shows 2-dimensional embedding of the data using ISOMAP algorithm. Red lines show nearest-neighbor relationships of the images. The template image is marked by a red box.



Geodesic Path in Anatomical Manifold: This shows an example of a registration path found by GRAM. The leftmost image (source) is registered to the rightmost image (target) sequentially via the path shown below.



4. References

1) The main function: GRAM.m

   i) Below is the description of GRAM.m inside the code.

```
% function [pdist,gpath,gmean,gdist,gMSE,gHE,dMSE,dHE]=GRAM
% (dirSubject,dirResult,dirDirectResult,
w,NNiter,NNsmoothing,Fineiter,Finesmoothing,Directiter,Directsmoothi
ng)
%
% GRAM does the following things: 1) performs pairwise registration
of all image to analyze the
% structure of data, 2) finds the geodesic mean template and the
geodesic
% paths, 3) performs the proposed geodesic registration of all
images to the
% template, and 4) performs direct registration of all images to the
% template for comparison.
%
% %Dependencies:
% GRAM uses the following four binaries for registration of images
and manipulation of
% deformation fields:
% ConcatenateFields, WarpImage, JacobianField,
DemonsRegistration_dong, DemonsRegistration_dong_n
% GRAM uses matlab_nifti toolbox to read and write .nii images.
% GRAM uses matlab_bgl toolbox to compute shortest-paths.
%
% % Output parameters
% pdist : distance between all pairs of images computed from
pairwise registration
% K : minimum size of the connected k-nearest-neighbor graph
% gpath : shortest-paths from all images to the template
% gmean : template number chosen from the dataset.
% gdist : geodesic distances between all images
% gMSE : Mean-squared error resulting from geodesic registration
% gHE : Harmonic Energy resulting from geodesic registration
% dMSE : Mean-squared error resulting from direct registration
% dHE : Harmonic Energy resulting from direct registration
%
% % Input parameters
% dirSubject: directory for storing image dataset
% dirResult: directory for storing registration results from
geodesic
% method. gHE.mat, gMSE.mat, pHE.mat, and pMSE.mat files are
created.
% Inside dirResult, four subdirectories are created:
%   Field: stores deformation field from the template to each image
%   Jacobian: stores Jacobian of the fields
%   NNField: intermediate deformation fields along the geodesic
paths
%   Warped: stores warped(=registered) images
% dirDirectResult: directory for storing registration results from
% direct method. dHE.mat and dMSE.mat files are created.
% Inside dirDirectResult, three subdirectories are created similarly
to
% dirResult directory:
%   Field: stores deformation field from the template to each image
%   Jacobian: stores Jacobian of the fields
%   Warped: stores warped(=registered) images
% w: weight between MSE and HE in distance definition
% NNiter: iteration number of NN registration
% NNsmoothing: regularization parameter for NN registration
% Fineiter: iteration number for fine tuning Finesmoothing:
regularization parameter for fine tuning
% Directiter: iteration number for Diffeomorphic Demons
% Directsmoothing: regularization parameter for Diffeomorphic Demons
%
```

b) The workflow of GRAM consists of the five steps:
   i) GRAM_PairwiseDistance: Calculate distances (weighted sum of MSE and HE) between two images
   ii) GRAM_FindK: Find the minimum K which makes the k-nearest neighbor graph connected
   iii) GRAM_GeodesicPath: Find geodesic paths in Anatomical Manifolds
   iv) GRAM_GeodesicRegistration: Concatenate small diffeormophic fields along geodesic path
   v) GRAM_FineTuing: Fine-tune the resulting field
   vi) GRAM_DirectRegistrationForComparison: Apply Demons directly for comparison
      (a) Please refer to the paper for details.

c) The following ITK binaries are used internally :
   i) ConcatenateFields – Concatenate deformation fields
   ii) WarpImage – Apply deformation field to a moving Image
   iii) JacobianField – Calculate the Jacobian determinant
   iv) DemonsRegistration_dong – Register images by Demons
   v) DemonsRegistration_dong_n – Register images by Demons with NN interpolation.

2) test_GRAM.m

a) Below is the description of test_GRAM.m inside the code.

```
% This script is a demonstration of GRAM framework. It shows the
usage of function  GRAM.m
```

```
% for groupwise registration of simulated 2D images resembling
cortical patches.
%
% To run this script with default parameters, change the directories
to fit your downloaded folder.
% It is recommended that the path names do not include spaces.
% After changing the directories, run test_GRAM.m. The script runs
in less than an hour on a PC
% without user actions. This will output four figures, whose
description is
% in the text.

% To use GRAM.m for other image databases, the directories and the
parameters need
% to be set properly. Descriptions of the directories and the
parameters can be found by typing
% help GRAM, or by looking into the GRAM.m function.
```

b) test_GRAM.m uses the following utility functions to visualize the results.
   i) GRAM_Montage: Display warped Images
   ii) GRAM_ISOMAP: Compute and display the ISOMAP embedding
   iii) GRAM_ShowPath: Display geodesic paths in the Anatomical Manifold