Diffeomorphic Shape Modeling Tutorial

This tutorial will guide you through one specific shape analysis pipeline, starting from structural images and corresponding binary segmentations. The goal of the tutorial is to familiarize the user with the shape analysis tools and the preprocessing steps necessary to use them correctly.

Data

The data consist of longitudinal observations from subjects in two groups: controls and HD high. Each group contains 7 female subjects around 60 years of age. The number of time-indexed observations per subject varies from 2 to 6, demonstrating how the analysis tools are flexible with respect to staggered or missing data.

Data is organized as **`/tutorial/data/`*`subjectID/scanID/`*.** Each scan session contains a structural image and a label image:

- **`t1_average_BRAINSABC.nii.gz`**
- **`neuro2012_20fusion_merge_seg.nii.gz`**

As you complete stages of the tutorial, additional files and folders will be added.

Scripts

All the processing necessary is automated by a collection of python scripts which are located at **`/tutorial/scripts/`**. As you work your way through the tutorial, keep an eye out for the following:

Run it yourself

These sections will give you instructions to run a specific python script which accomplishes the current task.

Preliminaries: Installing and building necessary software

The shape analysis pipeline requires a variety of tools to be installed:

1) **`Deformetrica`** (longitudinal version). This can be obtained at our private git repository https://github.com/BRAINSia/PHD_ShapeGrant. There are several applications and utilities that are important for this tutorial:

`/deformetrica/app/`

- **`sparseGeodesicRegression3:`** main application for estimating regression model
- **`sparseMatching3:`** aligns a source shape with a target shape

`/deformetrica/utils/`

- **`MapsEllipsoidWithSource:`** aligns an ellipse with a given source shape
- **`extractlabel`**: extracts a desired label as a new image from a label image
- **`surftransform:`** transforms a surface by a given rigid transformation

- **`surfdecimate`**: decimates a surface by a given percentage
- **`surfvolume`**: computes the volume of surface

2) **`BRAINSFit`**. This is used in this tutorial for rigid alignment of images to compute transformation parameters. It included as part of Slicer.

3) **`IRTK`**. This package includes a marching cubes implementation **required** by this tutorial. Download here: https://www.doc.ic.ac.uk/~dr/software/.

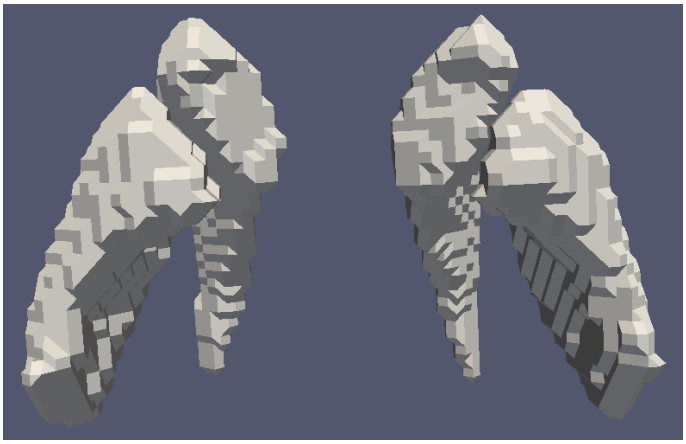
4) Python with numpy and matplotlib. The steps of the tutorial use python scripts to automate processing. Numpy and matplotlib are also used for plotting.

Preprocessing

The following sections cover a use case for getting your data ready for analysis. This includes extracting surfaces, aligning surfaces, and creating baseline shapes for initializing model estimation.

Extract surfaces

For surface extraction, we use the marching cubes implementation in the **`irtk`** package (https://www.doc.ic.ac.uk/~dr/software/). However, any algorithm/implementation which extracts shapes in the correct world coordinates can be substituted. **Note:** The marching cubes filters in VTK do not respect the world coordinates of the image. To follow this tutorial correctly, **`irtk`** should be used.



Run it yourself

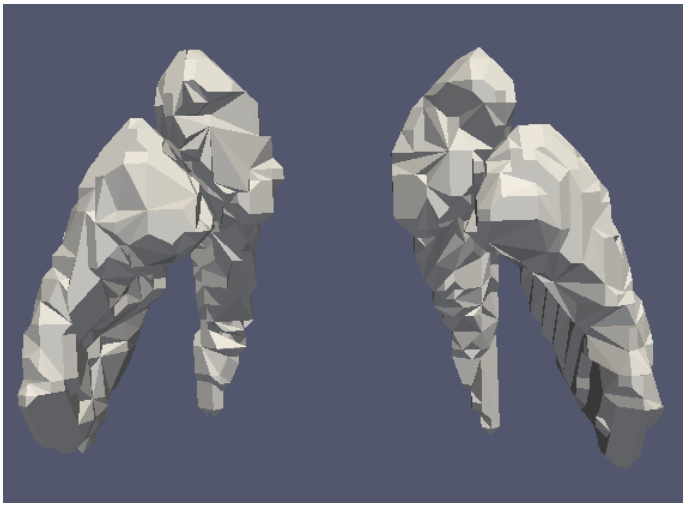
Edit **`mcubes_script.py`** and ensure the paths are correct for your system

- **`mcubes_path`**: the path to **`mcubes`**
- **`extractlabel_path`**: the path to **`extractlabel`**
- **`data_path`**: the base path to the data directory

Run the script **`> python mcubes_script.py`**

Surface decimation

To speed up the processing time for this tutorial, we will decimate the shapes to reduce the number of triangles. For clinical applications it is recommended you run your final experiments on full resolution shapes, as decimation can change both global and local shape features. Decimation is accomplished with the utility application **`surfdecimate`**. Here we reduce the complexity of the shapes by 80%.



Run it yourself

Edit `decimate_surfaces_script.py` and ensure the paths are correct for your system

- **`surfdecimate_path`**: the path to **`surfdecimate`**
- **`data_path`**: the base path to the data directory

Run the script **`> python decimate_surfaces_script.py`**

Alignment of surfaces

Surfaces must be rigidly aligned over time before shape regression models can be estimated. For population studies, we must also align all subjects to a common reference space.

1) Rigidly align images to compute rigid transformation matrix. First, choose an image to use as the global reference (a random subject's first scan works fine). Next, for each subject, align the first time point scan to the global reference. All remaining time points are then registered to the aligned first time point. The application **`BRAINSFit`** performs rigid alignment and outputs transformation parameters.

Run it yourself

Edit `rigreg_script.py` and ensure the paths are correct for your system

- **`brainsfit_path`**: the path to **`BRAINSFit`**
- **`reference_image:`** the path to the global reference image
- **`data_path`**: the base path to the data directory

Run the script **`> python rigreg_script.py`**

2) Apply the transformations to the surfaces. This avoids having to transform and interpolate the label images. This is accomplished with the utility application **`surftransform`**.

Run it yourself

Edit `transform_surfaces_script.py` and ensure the paths are correct for your system

- **`surftransform_path`**: the path to **`surftransform`**
- **`data_path`**: the base path to the data directory

Run the script **`> python transform_surfaces_script.py`**

Organize surfaces as a time-series

Now that we have aligned shapes extracted for each subject, we simply reorganize data from scan directories into a time-series directory. This makes the remaining preprocessing a little easier.

Run it yourself

Edit `move_shapes_to_timeseries_script.py` and ensure the paths are correct for your system

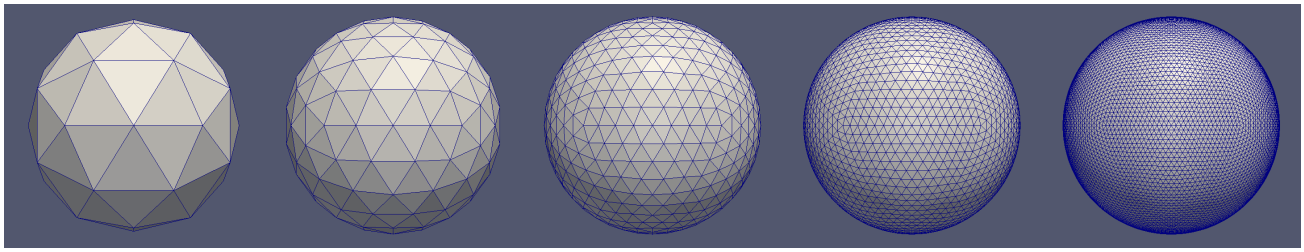
- **`data_path`**: the base path to the data directory

Run the script **`> python move_shapes_to_timeseries_script.py`**

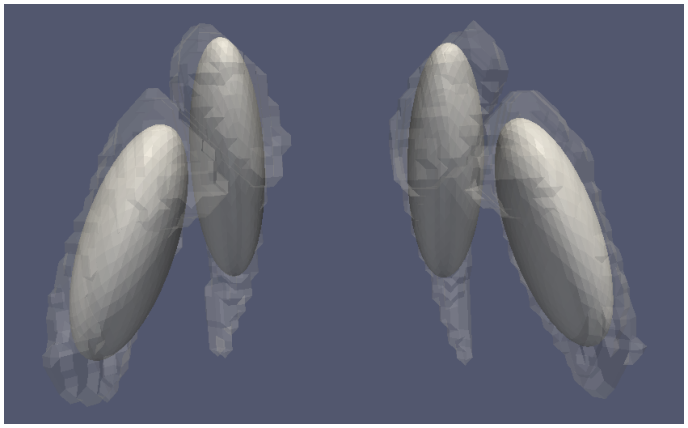
Create baseline shapes

`Deformetrica` requires the user provide an initial baseline shape. This initial shape configuration is refined during model estimation. It is possible to initialize with the shape observation earliest in time. However, here we will initialize with a generic representation which is shared across the population.

1) Choose a sphere mesh of desired complexity (to represent the anatomical shape of interest). Sphere meshes (.vtk) can be found in your **`Deformetrica`** build at **`/utils/meshes/`**. For this example we will work with **`sphere1280.vtk`**.



2) Transform a sphere into an ellipse aligned with each shape in your multi-object complex. This is accomplished with the utility application **`MapsEllipsoidWithSource`**. The figure below shows the baseline ellipses (solid) aligned with the observations earliest in time (transparent).



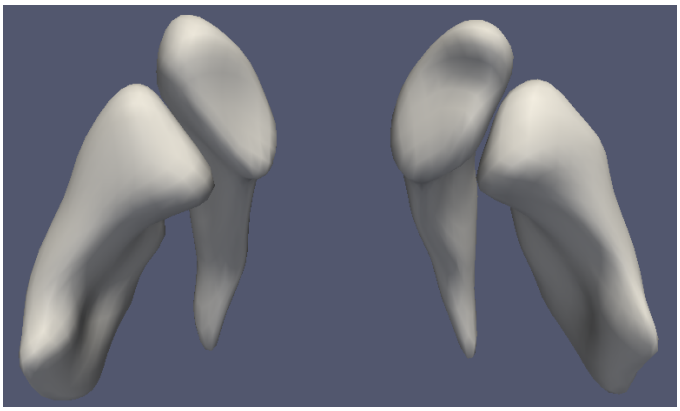
Run it yourself

Edit `create_init_baselines_script.py` and ensure the paths are correct for your system

- **`align_elipse_path`**: the path to **`MapsEllipsoidWithSource`**
- **`data_path`**: the base path to the data directory
- **`sphere_path`**: the path to the sphere mesh (.vtk)

Run the script **`> python create_init_baselines_script.py`**

3) Refine the ellipses to roughly match the earliest shape observations. This is accomplished with the Deformetrica application **`sparseMatching3`**. The figure below shows the initial baseline shapes for a given subject, which are now ready to be used as input for shape model estimation.



Run it yourself

Edit `match_init_baselines_script.py` and ensure the paths are correct for your system

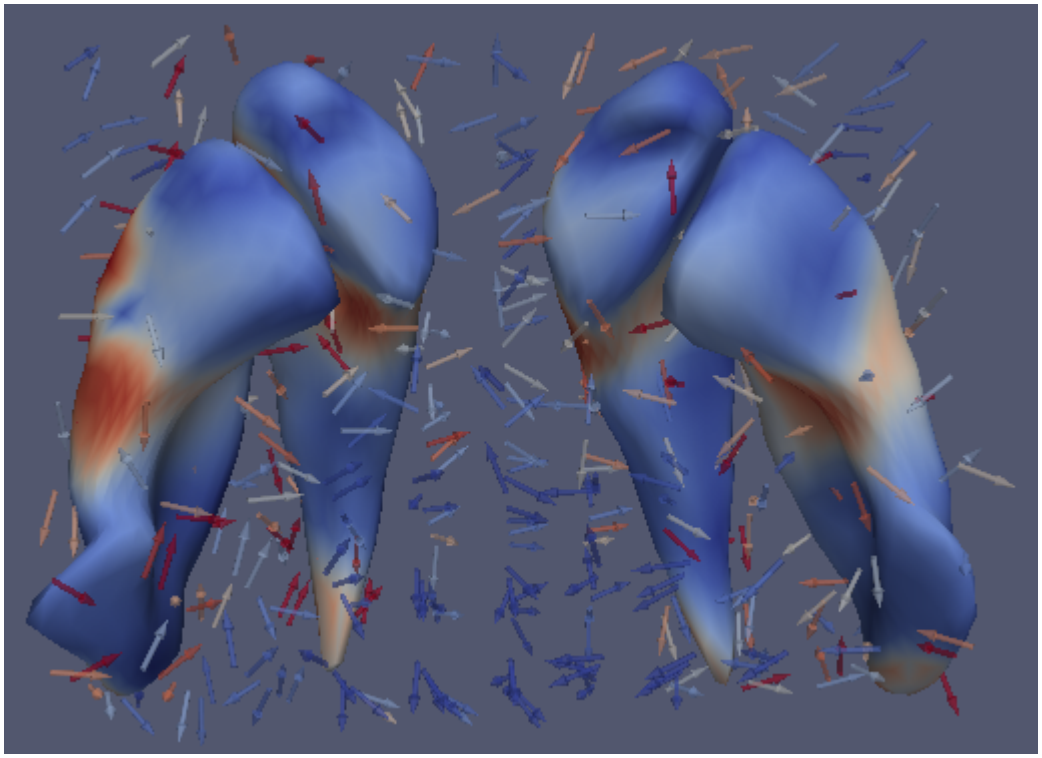
- **`match_path`**: the path to **`sparseMatching3`**
- **`data_path`**: the base path to the data directory

Run the script **`> python match_init_baselines_script.py`**

*****This step may take several hours depending on your computer*****

Estimating subject specific shape models

With the preprocessing done, we can estimate subject specific diffeomorphic shape models. We choose deformation kernel width of 6 mm, shape matching kernel width of 2 mm, and data-matching/regularity trade off parameter 0.1. We discretized time into 30 steps, so the output from our shape model will be a sequence of 30 time-indexed shapes. It is recommended to run overnight, as it may take several hours.



Run it yourself

Edit `run_regression_script.py` and ensure the paths are correct for your system

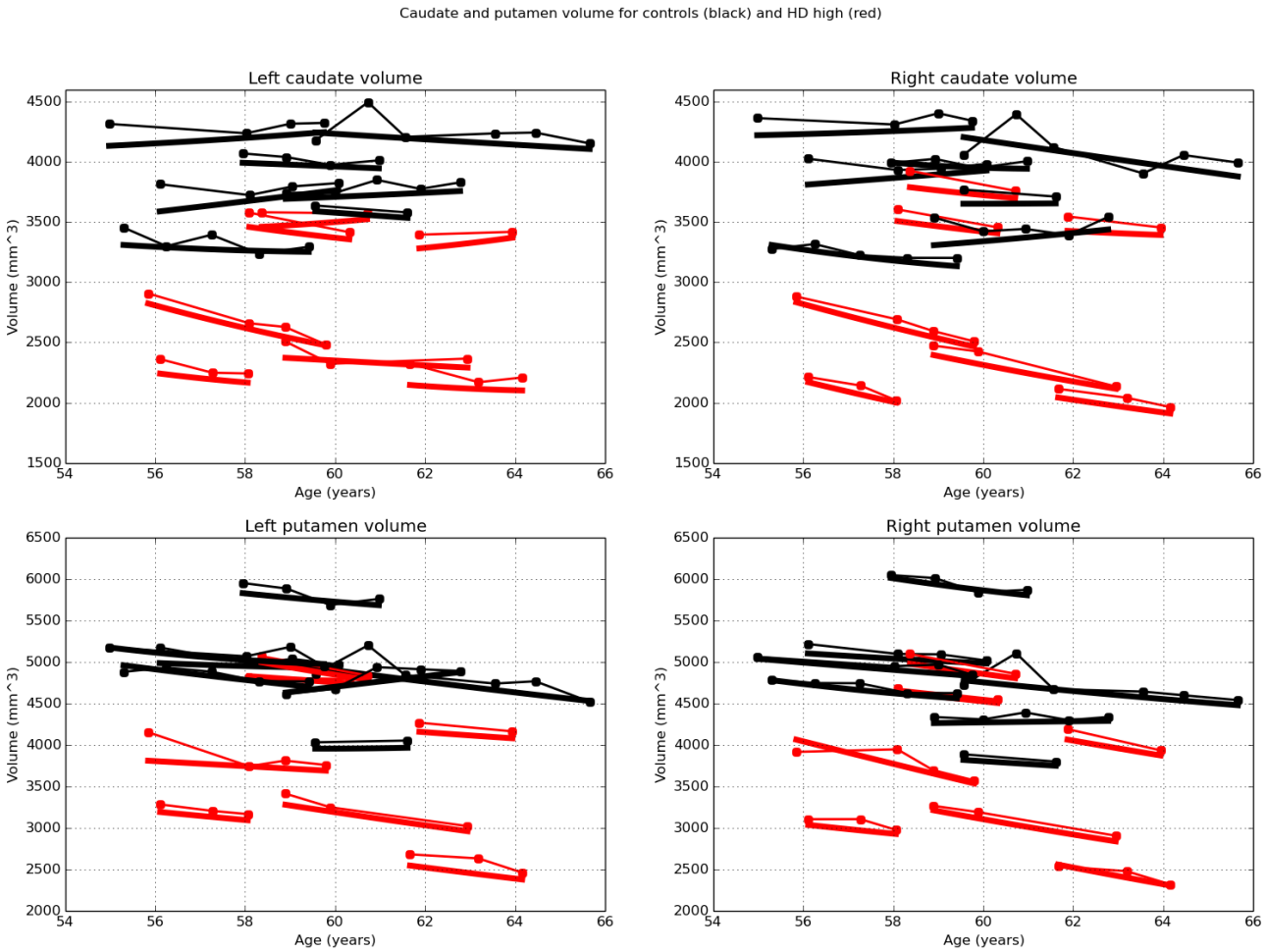
- `deformetrica_path`: the path to `sparseGeodesicRegression3`
- `data_path`: the base path to the data directory
- `output_data_path`: the base path for regression output

Run the script `> python run_regression_script.py`

*****This step may take several hours depending on your computer*****

Computing and plotting volume

Now that subject specific diffeomorphic shape models have been estimated, we can extract volume continuously for each structure. To make an informative plot, we compute the volume of the observations as well as the volume after regression.



Run it yourself

Edit `compute_obs_volume_script.py` and ensure the paths are correct for your system

- **`surf_volume_path`**: the path to **`surfvolume`**
- **`data_path`**: the base path to the data directory

Run the script **`> python compute_obs_volume_script.py`**

Edit `compute_regression_volume_script.py` and ensure the paths are correct for your system

- **`surf_volume_path`**: the path to **`surfvolume`**
- **`regression_path`**: the base path to the regression directory

Run the script **`> python compute_regression_volume_script.py`**

Edit `plot_regression_volume_script.py` and ensure the paths are correct for your system

- **`data_path`**: the base path to the data directory
- **`regression_path`**: the base path to the regression directory

Run the script **`> python plot_regression_volume_script.py`**