SCIENTIFIC COMPUTING AND IMAGING INSTITUTE

ShapeWorks Documentation

ShapeWorks : [USE-CASE] Binary Volumes to Shape Model (Femur Data)

Shireen Elhabian, Praful Agrawal, Riddhish Bhalodia,
Joshua Cates, Manasi Datar, Ross Whitaker

July 22, 2018

Abstract

This demonstrates a use case of femur data in form of binary segmentations and instructions to use the pipeline based script to prepare the data to proper signed distance transform representation for further *ShapeWorks* optimization. It also shows how to perform the *ShapeWorks* optimization and it's subsequent model analysis and visualization.

Contents

1 Data Preparation

Here we describe how to prepare a dataset of femurs for the *ShapeWorks* particle based optimization. Usually the initial data is presented in one of two formats, it's either in form of binary segmentations, or it's in form of surface meshes. Here we start with raw CT data and their corresponding binary segmentations (Figure **??**). We will use a full pipeline script whose usage is given below. Then it will describe the steps in the scripts which can be modified according to the need

1.1 Script Usage

The exemplar command line script for a full prep pipeline form binary segmentations to signed distance transforms is `FullPipelinePrepRunBinarytoDT.sh`. The usage is as follows

./FullPipelinePrepRunBinarytoDT.sh –parent_dir $1 –raw_image_dir $2 –raw_seg_dir $3 –process_raw $4 –img_prefix $5 –seg_prefix $6

Parameter Description :

1. `--parent_raw`: The parent directory where all the generated sub-directories of the pre-processing steps will lie. For example `${PATH}/Prep-Dir/`.
2. `--process_raw`: A flag which allows for processing of the raw CT/MRI images along with the binary segmentation, this is useful in some applications. When set to 1 raw image processing is turned on and it's off if the flag is 0. This example use the value as 0
3. `--raw_seg_dir`: The directory where all binary segmentations lie. For example `${PATH}/Segmentations/`.
4. `--raw_image_dir`: (Optional) The directory where the raw data lies. This not required if the `process_raw` flag is set to 0.
5. `--seg_prefix`: (Optional, DEFAULT = ") The prefix attached to the binary segmentations, in the form of prefix.ID.nrrd . In this case the prefix is 'seg'.
6. `--img_prefix`: (Optional, DEFAULT = ") The prefix attached to the raw images, in the form of prefix.ID.nrrd . In this case the prefix is not needed

1.2 Pipeline Blocks

The scripts have tags with which you can turn on and off each of the steps of the pipeline if you wish to. This case uses all the blocks which are described as follows.

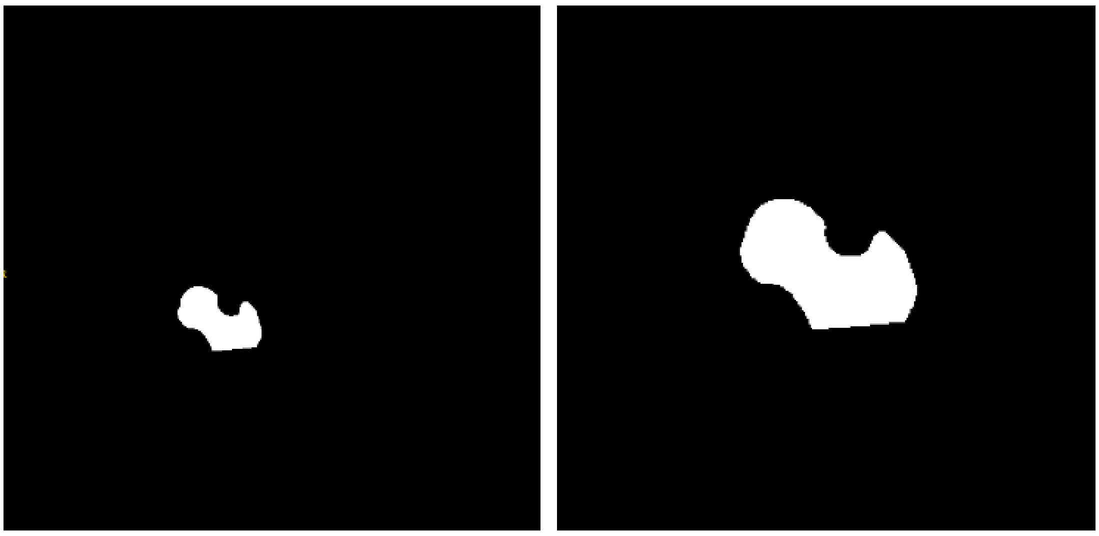


Figure 1: Before (left) and after (right) the center of mass alignment

1. **Isotropic resampling**
   FLAG : `doResample`
   This resamples all the binary volumes which in a raw setting will be in different physical spaces (different dimensions and voxel spacing). This brings all the data to same physical space. The parameters are the sizeX, sizeY, sizeZ which can be determined is usually set to the largest dimensions in the dataset. The isoSpacing parameter determines the voxel spacing, two most used values are 0.625 or 1.0 depending on application.

2. **Padding with constant value**
   FLAG : `doPad`
   Pads the volume uniformly with a constant value, in this case we do zero padding. The padding size and padding value are the two parameters in this step.

3. **Center of mass alignment**
   FLAG : `doCOM`
   This is a basic alignment step which aligns the center of mass of the binary shape in each image to the image center. This is shown in figure 1.

4. **Rigid Alignment**
   FLAG : `doAlign`
   This rigidly aligns all the scans to a reference scan whose id is provided in the script. These alignment is powered through ANTS registration package. The most important parameter is the id of the scan to be

kept as reference.

5. **Extract largest bounding box**
   FLAG : `doLargestBB`
   This takes in all the aligned segmentations and finds the bounding box which encloses all the shapes, this is necessary as it is used in conjunction with the cropping step.
6. **Crop volumes**
   FLAG : `doCrop`
   This takes in the information produced by the largest bounding box step and crops all the volumes to the largest bounding box and can also apply padding if needed. This takes in the padding value as parameter again
7. **Groom and convert to mesh**
   FLAG : `doGroomAndMesh`
   This performs the ShapeWorks grooming step which comprises of closing holes, anti-aliasing, smoothing and using fast marching to convert the binary scans to signed distance transforms, it is followed by topology preserving smoothing. It also converts the distance transforms to meshes which may be useful for other applications. Shown in figure 2. This has several parameters which are set in subsidiary scripts (check more on ShapeWorks documentation).

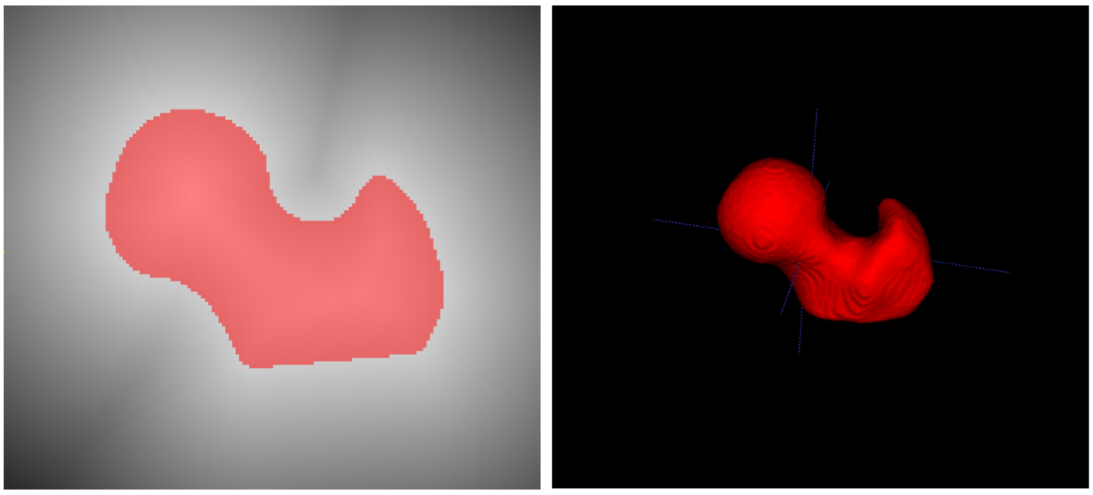


Figure 2: (Left) distance transform overlayed with a segmentation. (Right) 3D mesh of the processed scan

At the end of the pipeline in the specified parent directory there will be a sub-directory called `DistanceTransforms`, this contains the processed output which will serve as input to *ShapeWorks* particle based optimization.

2 Particle Based Optimization

After the generation of the distance transform we now need to perform the particle based optimization in a single stage format (which means that there is a fixed number of particles specified and we do a single run of ShapeWorks, for more details refer to ShapeWorks run documentation).

2.1 Script Usage

For this we use the script with 512 particles as follows
./SingleScaleShapeWorksRun.sh –data_dir $1 –out_dir $2 –verbosity $3 –model_suffix $4 –num_particles $5 –start_reg $6 –end_reg $7 –procrustes_interval $8

Parameter Description:

- `data_dir`: Path to the directory containing the processed data in form of signed distance transform. In this case `${PATH}/Prep-Dir/DistanceTransforms/`.
- `out_dir`: The directory where you need to save the output produced by the ShapeWorks optimization. In this case `${PATH}/Prep-Dir/PointFiles/`.
- `verbosity`: (Optional) 0 : almost zero verbosity, 1: minimal verbosity, 2: Default verbosity, 3: full verbosity.
- `start_reg`: (Optional) Sets the starting regularization value (refer to the parameter description documentation of *ShapeWorksRun*).
- `end_reg`: (Optional) Sets the ending regularization value (refer to the parameter description documentation of *ShapeWorksRun*).
- `model_suffix`: (Optional) (default : model)
- `num_particles`: Number of particles, a power of 2 like 256, 512...
- `procrustes_interval`: (Optional) (default = 0) Flag for the turning on and off the procrustes correction during optimization.

For the femur data all the tags which are needed is `data_dir`, `out_dir` and `num_particles`.
For more details about the parameters (which are specified in) and their specific intuition please refer to the parameter description document in the *ShapeWorks* run documentation.

2.2 Output Description

The output of the ShapeWorks is a set of particle files with extensions '.local.particles' and '.world.particles'. In this example they both will be exactly same (a list of 512 3D coordinates). They are same because we set the procrustes flag to be 0, which turns of the stage wise particle alignment off. If it is turned on then the shape analysis should be performed on the '.world.particles' which is the world space with rotation and translation elements removed. The correspondence model is shown in figure **??**.

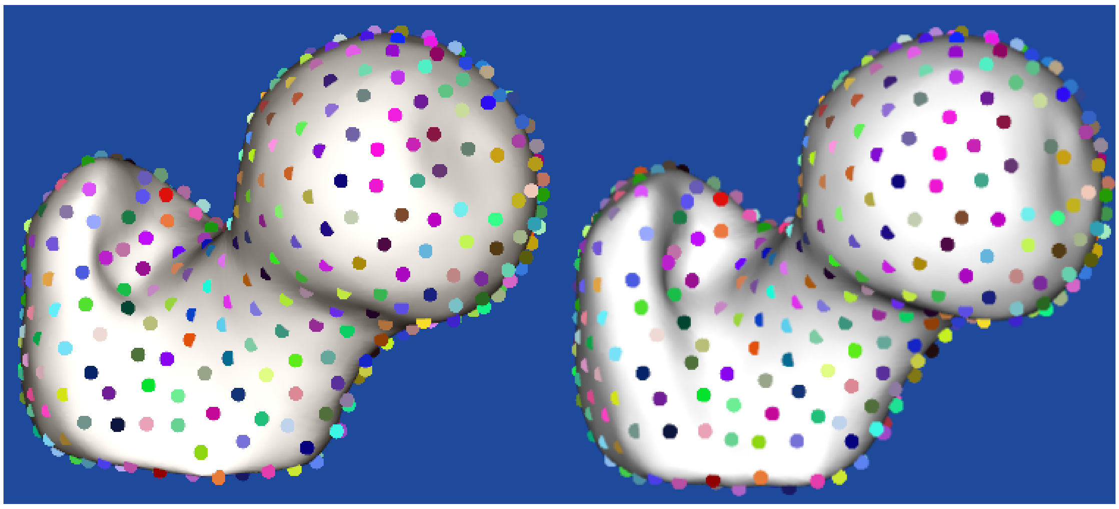


Figure 3: The output of ShapeWorks optimization showing correspondences on two of the femurs in the dataset. This are screenshots taken from *ShapeWorksPost*

3 Shape Analysis and Visualization

Now we have a correspondence model for the population of femurs, we now perform some statistical analysis like PCA and visualize these modes of variation. We provide two different tools for the visualization, one is *Shapeworks View2* which is based on VTK library and is fully developed. The second tool is more easy to develop and use and can provide more functionality in future but is in a prototype stage called ShapeWorksPost. We will describe both these tools.

3.1 *ShapeWorksView2* visualization framework

This is well tested platform for visualization based on VTK but have limited scope in add more visualization functionality in future. The functionality available with *ShapeWorksView2* are the following

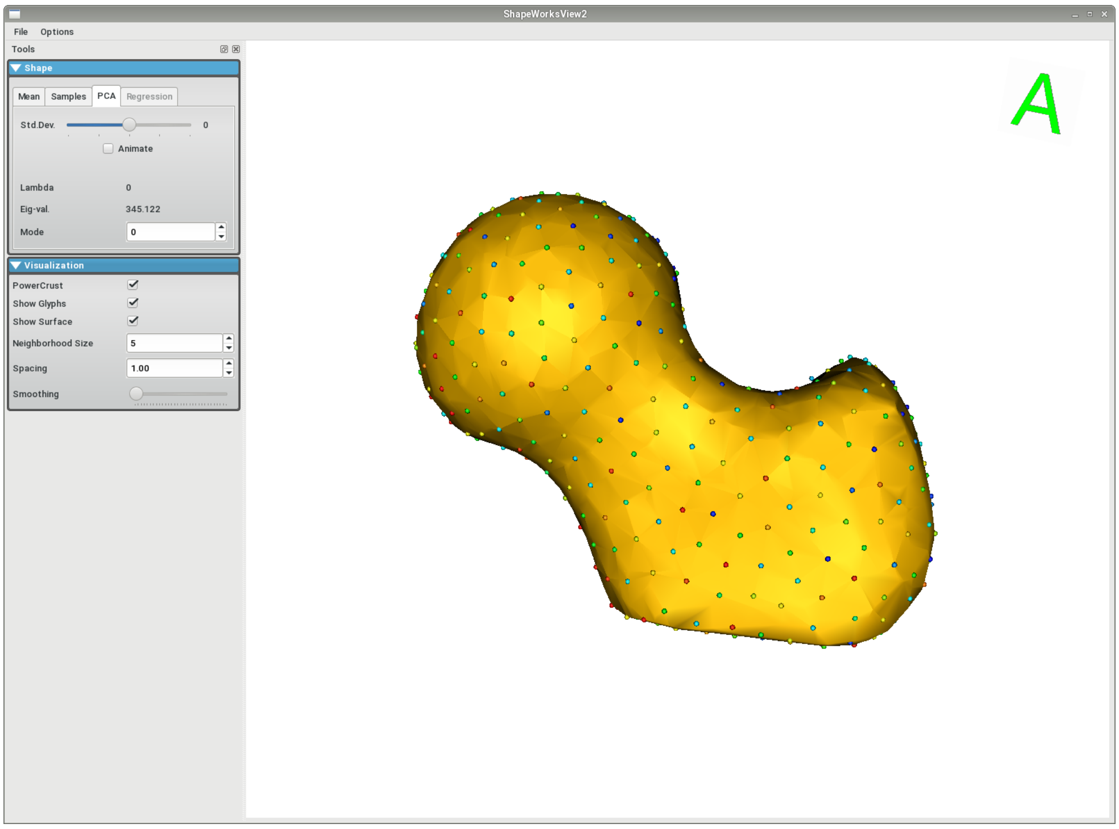
- It visualizes the particles on one shape at a time with in built surface reconstruction, and can scroll through the dataset.
- It computes the PCA of the correspondence data and allows for animating the variation of shape along a principle component.

- It can visualize the mean shape.
- It can save the eigenvalues and the PCA loadings of each shape in an excel sheet for analysis.
- If there are groups in the data it can visualize group differences

To execute the GUI it takes in an XML parameter file which is generated by the *ShapeWorks-Run* script executed in the previous step, it's called `ShapeWorksView-Analyze.xml`. The execution of the GUI is as follows:

```
./ShapeWorksView2 ShapeWorksView-Analyze.xml
```

The usage of this tool is intuitive with the provided GUI, a screenshot of the app is given in figure 4.



Std.Dev. 0
Lambda 0
Eig-val. 345.122
Mode 0
PowerCrust ✓
Show Glyphs ✓
Show Surface ✓
Neighborhood Size 5
Spacing 1.00
Smoothing

Figure 4: A snapshot of *ShapeWorksView2* with femur data

3.2 *ShapeWorksPost* visualization framework

This is a developmental prototype platform based on OpenGL and libigl, it incorporates faster processing and better mesh reconstruction algorithms.

Also adding more visualization functions is smooth and easy. The functionality available with *ShapeWorksPost* are the following

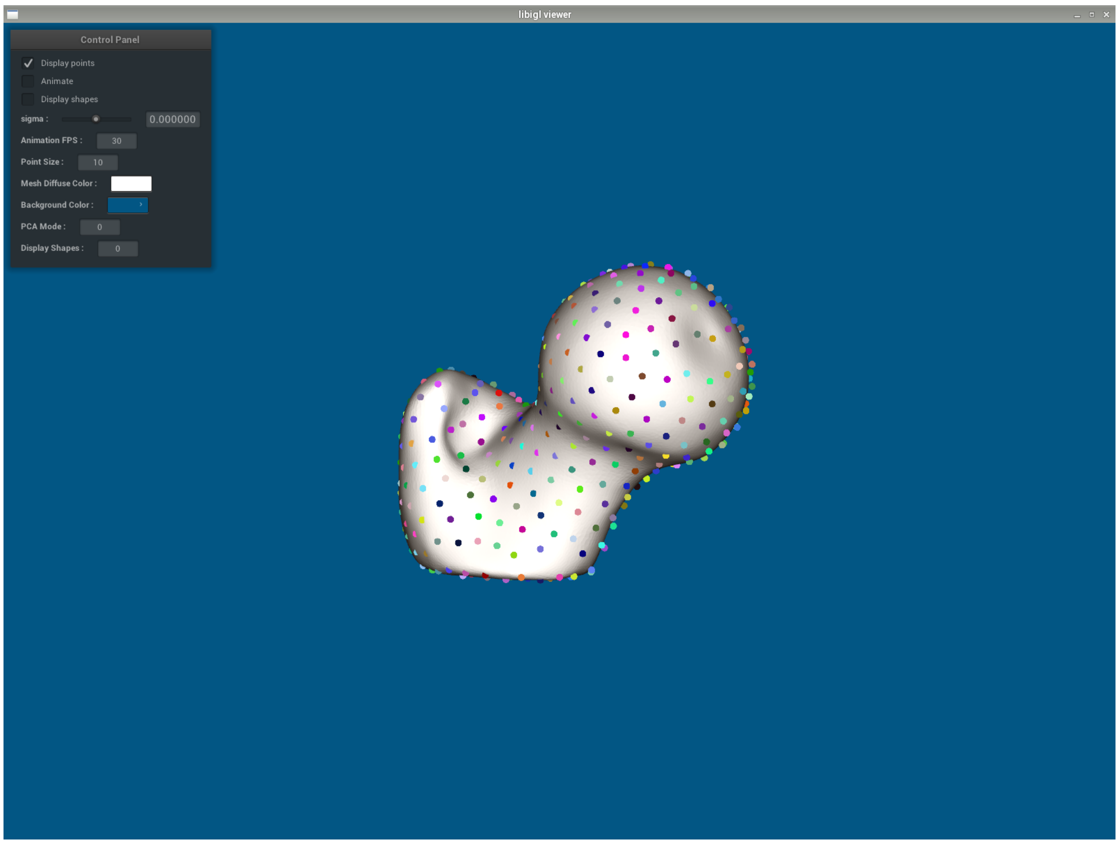
-
- It visualizes the particles on one shape at a time with in built surface reconstruction, and can scroll through the dataset.
- It computes the PCA of the correspondence data and allows for animating the variation of shape along a principle component.
- It can visualize the mean shape.
- It enables the graphics based mesh functionality of changing the mesh, background, texture and animation formats.

To execute the GUI it takes in an xml parameter file which is generated by the *ShapeWorks-Run* script executed in the previous step, it's called `ShapeWorksPost-Analyze.xml`. The execution of the GUI is as follows:

./shapeworkspost ShapeWorksPost-Analyze.xml

In this parameter file you will notice it takes in one of the distance transforms and it's corresponding particle as a reference, instead of the distance transform one can modify the file to use the associated VTK mesh as well. There is also a functionality of mesh decimation, which is useful if the number of vertices of the computed mesh is too much which causes an initial time overhead for the algorithm.

The usage of this tool is intuitive with the provided GUI, but for more info check out the documentation for ShapeWorks Post, a screenshot of the app is given in figure 5.


libigl viewer

Control Panel

- [✓] Display points
- [ ] Animate
- [ ] Display shapes

sigma : 0.000000

Animation FPS : 30

Point Size : 10

Mesh Diffuse Color : ⬜

Background Color : 🟦

PCA Mode : 0

Display Shapes : 0

Figure 5: A snapshot of *ShapeWorksPost* with femur data