SCIENTIFIC COMPUTING AND IMAGING INSTITUTE

THE UNIVERSITY OF UTAH

ShapeWorks Documentation

ShapeWorks : [USE-CASE] Surface Meshes to Shape Model (Femur Data)

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

July 22, 2018

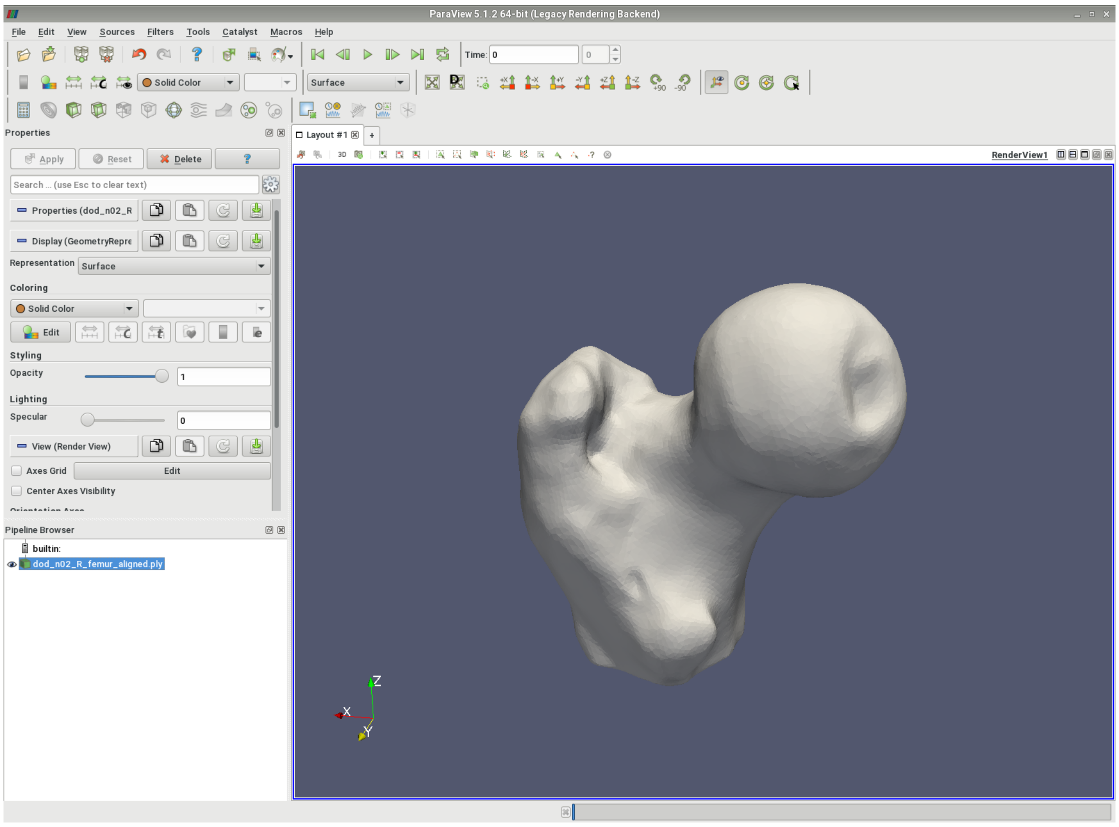
Abstract

This demonstrates a use case of femur data in form of surface meshes 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 surface meshes (Figure 1). 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



ParaView 5.1.2 64-bit (Legacy Rendering Backend)
File Edit View Sources Filters Tools Catalyst Macros Help
Properties
Apply Reset Delete ?
Pipeline Browser
builtin:
dod_n02_R_femur_aligned.ply
Layout #1
RenderView1
Z
X
Y

Properties (dod_n02_R
Display (GeometryRepre
Representation Surface
Coloring
Solid Color
Styling
Opacity 1
Lighting
Specular 0
View (Render View)
Axes Grid Edit
Center Axes Visibility
Time: 0

Figure 1: Raw surface mesh sample of a femur in PLY format shown in the software Paraview, it's a good software to load and debug mesh as well as counters

1.1 Script Usage

The exemplar command line script for a full prep pipeline from surface meshes to signed distance transforms is `FullPipelinePrepRunMeshtoDT.sh`. The usage is as follows

./FullPipelinePrepRunMeshtoDT.sh –parent_dir \$1 –raw_data_dir \$2 –mesh_extension \$3

Parameter Description :

1. `--parent_raw`: The parent directory where all the generated subdirectories of the pre-processing steps will lie. For example `${PATH}/Prep-Dir/`.
2. `--mesh_extension`: This can be 'vtk' or 'ply', the default is 'ply'.
3. `--raw_data_dir` (Optional): The directory where all raw images lie. For example `${PATH}/RawData/`. This is not needed in this use case.

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.

1. **Mesh Decimation Using Preview**
   FLAG : `doPreview`
   This uses a package called previewCmd to perform mesh based quality control as well as mesh decimation which helps in making the subsequent processing faster. The parameters are `decimation_decimal` and `decimation_percentage` which determines the fraction and the percent of the mesh vertices to be retained respectively. Default is 0.95 and 95 respectively.
2. **Mesh Smoothing**
   FLAG : `doSmooth`
   Performs mesh based smoothing, the parameters are `smoothing_iterations` which determines the number of smoothing passes over each mesh, and , `relaxation_factor` which is a smoothing parameter between 0 and 1, default is 0.5 (recommended).
3. **Mesh Based Rigid Alignment**
   FLAG : `doAlign`
   This rigidly aligns all the input mesh to a specified reference mesh. The reference mesh is specified through it's id. There are two parameters `reference_meshID` which obviously takes the id of the reference mesh, and `registration_mode` which determines the metric used for mesh based registration, the recommended and default is set to 'similarity'.
4. **Find the largest bounding box**
   FLAG : `doSizeOrigin`
   This computes the dimensions as well as the origin of the largest bounding box which encloses all the surface meshes. This is an required step for the subsequent fids based conversion of meshes to distance transforms.

5. **Convert meshes to distance transforms**
   FLAG : `doMeshesToVolumes`
   This uses a Fids based algorithm to convert meshes into distance transforms, this is a really time consuming step and can act as a big bottleneck for large datasets, this is why the initial mesh decimation using Preview is encouraged. The algorithm and the parameters for usage are described in a separate document in the *ShapeWorks-Prep* repository, for this example all the default setting will work perfectly.
6. **Fix DT Leaks**
   FLAG : `doFixFidsDT`
   The above conversion to distance transforms sometimes cause artifacts in the shape representation, which we call leaks. This is the step which fixes these leaks
7. **Topology Preserving Smoothing**
   FLAG : `doTPSmoothDT`
   This performs the final smoothing step such that it does not remove any shape based features.

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. After the preparation weather starting from mesh as in this example or from binary segmentation, ones we have the processed distance transforms the subsequent *ShapeWorks-Run* and shape analysis is exactly same.

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 **??**.

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 *ShapeworksView2* 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.

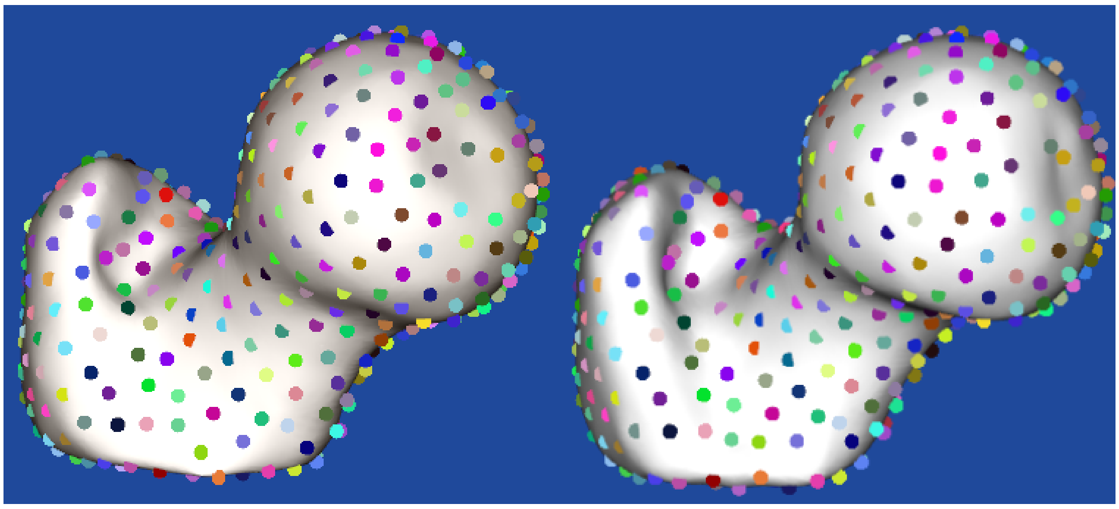


Figure 2: The output of ShapeWorks optimization showing correspondences on two of the femurs in the dataset. This are screenshots taken from *ShapeWorksPost*

3.1 *ShapeWorksView2* visualization framework

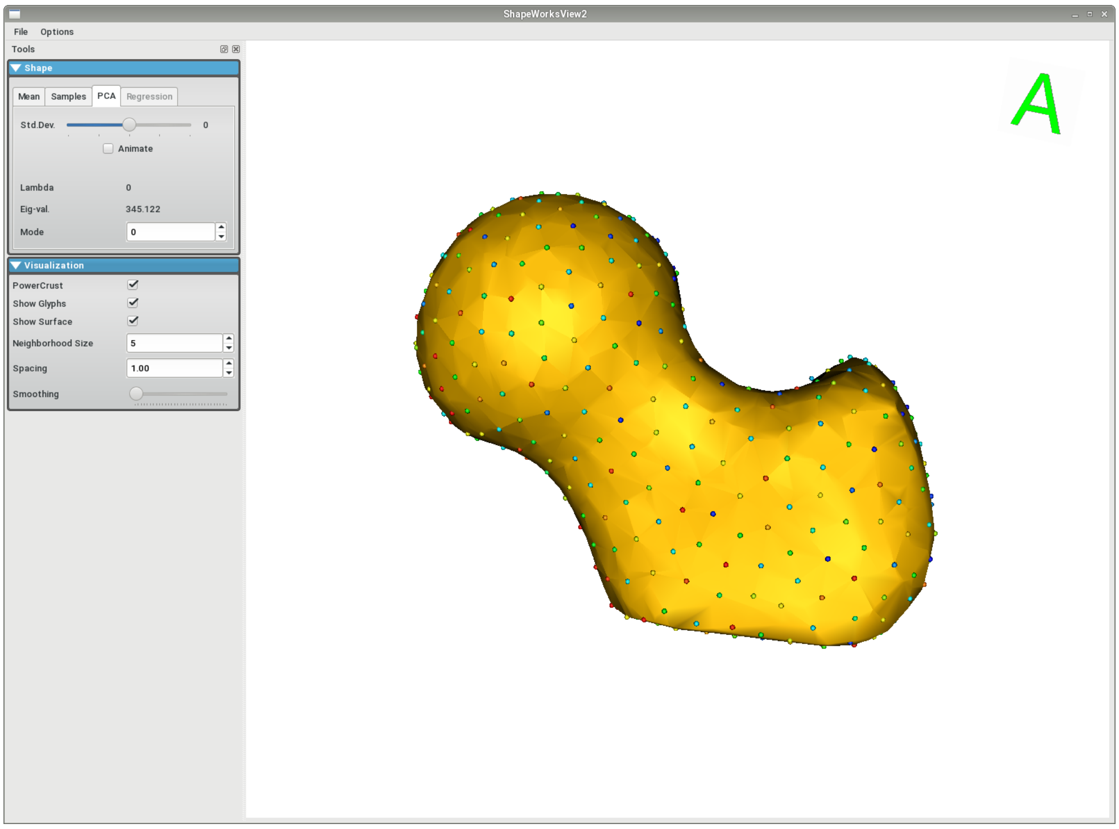
This is well tested platform for visualization based on VTK but have limited scope in add more visualization functionality in future. The functionality available with *ShapeWorksView2* are the following

- 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 3.



Std.Dev. 0
Animate
Lambda 0
Eig-val. 345.122
Mode 0

PowerCrust ✓
Show Glyphs ✓
Show Surface ✓
Neighborhood Size 5
Spacing 1.00
Smoothing

Figure 3: 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.

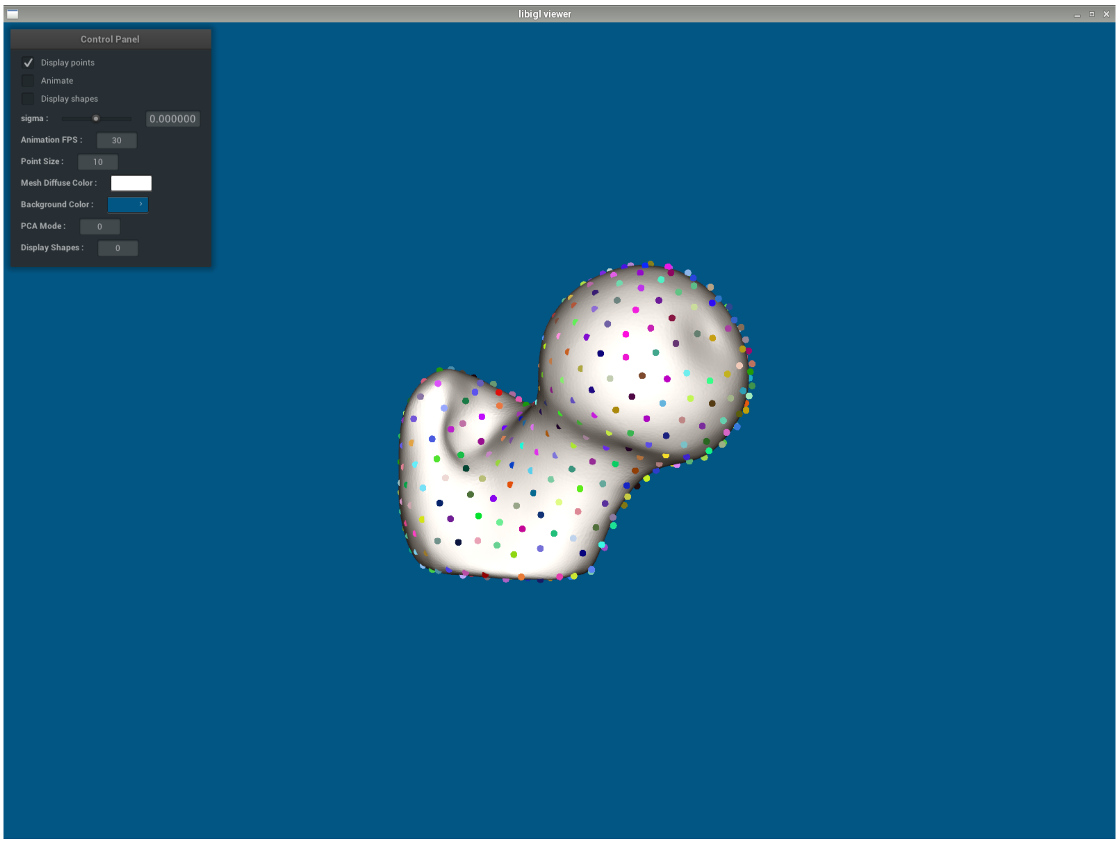
To execute the GUI it takes in an xml parameter file which is generated by the *ShapeWorks-Run* script executed in the previous step, it's called

`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 4.



✓ 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 4: A snapshot of *ShapeWorksPost* with femur data