KWMeshVisu: A Mesh Visualization Tool for Shape Analysis

Release 1.00

Ipek Oguz[1], Guido Gerig[2], Sebastien Barre[3], and Martin Styner[4]

July 13, 2006

[1]University of North Carolina at Chapel Hill, ipek@cs.unc.edu
[2]University of North Carolina at Chapel Hill, gerig@cs.unc.edu
[3]Kitware Inc., sebastien.barre@kitware.com
[4]University of North Carolina at Chapel Hill, styner@cs.unc.edu

Abstract

Many statistical shape analysis methods produce various types of data about the analyzed surfaces, such as p-value maps, distance maps, 3D difference vectors and local covariance matrices. This data is often too large and thus difficult to be properly evaluated on a qualitative basis. A visual representation of this data strongly simplifies qualitative evaluation by humans and thus greatly enhances the value of the statistical results. In this paper we present a new tool for visualizing various datasets on surfaces represented as triangle meshes.

Our tool, KWMeshVisu, is implemented using the Insight Toolkit ITK, `www.itk.org`, the Visualization Toolkit VTK, `www.vtk.org`, and the KWWidgets user interface toolkit, `www.kwwidgets.org`. The source code for KWMeshVisu, as well as input data used to generate the images in this paper, is provided with this document.

Contents

1 Introduction

KWMeshVisu is designed to visualize various datasets on surfaces represented by meshes. The currently supported data types are scalar maps, vector maps, ellipsoid fields and 3D space curves. KWMeshVisu enables custom visualizations of all these data types, with many parameters that can be fine-tuned to meet the requirements of the specific application. However, all the parameters have sensible default values that are determined based on the dataset; therefore, it is intuitive enough for the most inexperienced user.

In the following sections, we first describe how to use the graphical user interface to visualize various data types as well as the relevant parameters for each of these visualizations. Next, we describe how to use KWMeshVisu as a scripting tool that can process a large population of objects and datasets to produce a set of images; this can be a valuable method for ensuring quality control of the results of a large statistical study. Then, we describe the software requirements for successfully installing KWMeshVisu. Finally, images and descriptions of various visualizations are provided in Section 3 to illustrate the capabilities of KWMeshVisu and its typical usage scenarios. Technical implementation details are provided in Appendix A.

2 Methods

2.1 Visualizing Statistical Data on Meshes with KWMeshVisu

When first launched, KWMeshVisu displays an empty scene, and the user is expected to load an object into the scene, using the "Load Mesh" button. Any file in ITK's MetaMesh format can be loaded into the scene this way.

Once a mesh is loaded into the scene, the user is able to start visualizing data on the mesh surface. The currently supported data types are scalar maps, vector maps, ellipsoid maps, as well as 3D sampled curves.

Visualizing Scalar Maps

A scalar map on a surface represented by a mesh is basically a list of scalar values such that each vertex of the mesh is assigned a value. An intuitive visualization of such a scalar map is to represent the scalars with different colors. In this way, a qualitative understanding of the data may be quickly obtained, and regions with local minima or maxima, and regions of high and low values may be easily detected.

With KWMeshVisu, it is very easy to visualize a scalar map in the way described above. After a mesh is loaded, all that needs to be done is to load the scalar map using the button "Load 1D Attribute". The files that can be loaded this way are simple text files. At the beginning of these files, there is a header section, which should look like the following:

```
NUMBER_OF_POINTS = 4002
DIMENSION = 1
TYPE = Scalar
```

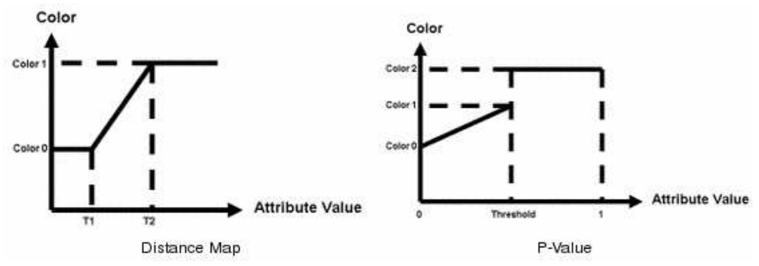

Figure 1: Visualization Modes. The two predefined visualization modes are the distance map, which can be used to visualize the whole range of the data, and the P-value map, which can be used to threshold values.

In this header, the `NUMBER_OF_POINTS` entry represents how many scalar values there are in the file. Note that this number must match the number of vertices in the mesh. Following the header, there is a simple list of scalars, with a new line used as separators after each value. The total number of lines after the header should therefore be equal to the number of vertices in the mesh.

Once the map is loaded, the user has the option of fine-tuning the visualization using the left panel of the tool. The visualization mode can be set to be either a distance map, or a P-value map. The distance map assigns two different colors to two values, and any value that is in this range is assigned an interpolated color. As the name implies, it is most convenient to use this mode for distance maps. A P-value map, on the other hand, has three end points, the middle one being used as the threshold of “significance”. A good usage example would be a significance map where values between 0 to 0.05 are decreasingly significant, and any value above 0.05 is insignificant. Figure 2 shows the visualization of such a significance map, and compares the resulting image with the distance map mode.

In both of these visualization modes, the scalar values of the end points as well as the colors assigned to these values can be specified through KWMeshVisu’s user interface. In the color transfer function editor in the left panel, double clicking on the end points will let the user assign colors; dragging the points around or entering a value will change the available range. Note that in the predefined P-value map color transfer function, the two end points are constant and set at 0 and 1; therefore, the only variable range parameter is the threshold value. For distance maps, both end points can be modified.

As an initial value, the distance map automatically chooses a value which includes all of the available scalar range. The user can later change these values if he/she wants to zoom in on a particular part of the range.

Visualizing Vector Maps

Another common data type is vector maps, which consist of a 3D vector defined at each vertex of the mesh. Examples of such datasets are difference vectors, growth vectors, principal directions, etc. It is very intuitive to visualize these vectors as oriented lines or arrows, originating from the surface point.

To load such a vector map with KWMeshVisu, the user should simply click on the “Load 3D Attribute” button. The supported file format is very similar to 1D case, with a slight change in the header:

```
NUMBER_OF_POINTS = 4002
DIMENSION = 3
TYPE = Vector
```

Following the header section, each line consists of 3 values representing the x, y, z dimensions of the vector corresponding to the current vertex. The values are separated by a space character.

In the visualization, both the arrow lengths and coloring are done in proportion to their vector scalar lengths. This way, it is possible to quickly observe where the vectors are large (big arrows of a particular color) and where they are small (small arrows of a different color). The coloring parameters can be changed in the same way as the scalar map case.

Note that if a scalar map and a vector map are loaded on the same mesh, the scalar map visualization will become static and it will not be possible to change it any further. This is done to ensure that the scalar map and the vector map can have different visualization parameters. The user should first load the scalar map, adjust parameters until satisfied with the result, and then load the vector map and adjust the parameters for the vector visualization. An example of the simultaneous visualization of a scalar map and a vector field is shown in Figure 6.

When working with vector maps, two other features of KWMeshVisu become relevant. The first one is the "Mesh Opacity", which determines the appearance of the mesh surface. If some of the vectors are directed towards the inside of the mesh, then making the mesh transparent greatly improves the visualization. Figure 3 compares the visualization on an opaque mesh with the visualization on a transparent mesh. The second important option for visualizing vector maps is the "Feature Scale", which lets the user choose a factor to scale all the vectors. Therefore, it is possible to zooming in and out of the arrows without having to alter the surface visualization or the attribute file.

Visualizing Ellipsoid Fields

The next common type of data is ellipsoid fields. The most common usage of ellipsoid fields is to represent local covariances on a surface. In KWMeshVisu, the button "Load Ellipsoid Field" enables the user to visualize such a map.

The supported file format is very similar to the scalar and vector map cases. The header now looks like the following:

```
NUMBER_OF_POINTS = 4002
DIMENSION = 9
TYPE = Ellipsoid
```

The visualization is very similar to the vector map case, with the ellipsoids drawn to scale. The coloring is done based on the largest eigenvalue of the ellipsoid. All visualization parameters (scale, opacity, coloring range and end points) can be changed in the same way as before. Figure 4 illustrates this type of visualization.

Visualizing Space Curves

The last supported data type is 3D space curves. A typical usage can be to visualize the integral curve of some vector field on the mesh, like the principal curves of the surface.

The file format is similar, although the header is slightly different. In this case, the `NUMBER_OF_POINTS` entry shows how many samples on the curve are provided. Therefore, this number does not have to match the number of vertices on the mesh, and can be any integer value. The lines after the header each include the three dimensional coordinates of the samples on the curve.

```
NUMBER_OF_POINTS = 1442
```

```
DIMENSION = 3
TYPE = Curve
```

For the time being, this feature offers less visualization options as compared to the other data types. The coloring parameters can not be altered, even though it is possible to change the surface opacity.

Figure 5 illustrates a space curve visualization.

2.2 Other Features

There are a few other features in KWMeshVisu that are meant for advanced users. These include camera specification and saving/loading the visualization as a single data file.

In the user interface, it is easy and intuitive to use the mouse for obtaining the desired viewpoint and camera angle to achieve the desired visualization. However, if KWMeshVisu is used as a scripting tool, as described in Section 2.3, the camera specification can be tricky. It is suggested that one does at least one visualization for each dataset through the GUI, and then use the “Get Camera” button to grab the current camera parameters. These can then be supplied to the command line tool as parameters to reproduce the same viewpoint for the subsequent visualizations.

KWMeshVisu is also capable of saving/loading the currently visualized mesh and dataset as a single file rather than two separate files (.meta and .txt files). This can be useful for easily reproducing the visualization in the future. Simply use the “Save Attributed Mesh” to save; then, the “Load Attributed Mesh” button can be used at any time for loading the current mesh and dataset at once. Note that currently only scalar data can be saved together with the mesh.

2.3 KWMeshVisu as a Scripting Tool

In statistical shape analysis studies, it often happens that the analysis produces a few different datasets for every object in a population. In such a scenario, it is very inconvenient to go through each object and each dataset, just in order to inspect the overall quality of the analysis. Moreover, when doing these visualizations one by one manually, it is much more likely that a small difference in the fine-tuning of the visualization parameters will result in a mistaken conclusion. Therefore, it would be desirable to automize this process as much as possible to make the task easier and less error-prone.

The solution is to use KWMeshVisu as a command line tool that can be used in a script that automatically generates visualizations of the dataset. All of the functionality described in the previous section is also available as command line options. The following is a list of these command line options and their descriptions.

- `--help`: Displays a brief list of the available command line options.
- `--mesh [filename]`: Loads the mesh from the specified file. Note that this is a required option, since the visualization requires at least the presence of the object.
- `--scalar [filename]`: Loads the scalar map from the specified file.
- `--vector [filename]`: Loads the 3D vector field from the specified file.
- `--ellipsoid [filename]`: Loads the ellipsoid field from the specified file.

- `--image [filename]`: Generates a bitmap (.bmp) image of the specified visualization.
- `--pvalue [threshold]`: Selects the visualization mode to be P-value map, with the specified threshold value.
- `--distMapMin [min]`: Selects the visualization mode to be distance map, and specifies the lower end point. Note that this option should be used with the –distMapMax option to produce the expected results.
- `--distMapMax [max]`: Specifies the higher end point for the distance map visualization mode.
- `--opacity [o]`: Specifies the opacity of the mesh surface. 1 corresponds to completely opaque, and 0 corresponds to completely transparent.
- `--scale [s]`: Specifies the factor by which the arrows/ellipsoids will be scaled in the visualization. Default is 1.
- `--br [red] --bg [green] --bb [blue]`: Specifies the background color. Default background color is black.
- `--posx [x] --posy [y] --posz [z]`: Specifies the position of the camera.
- `--focx [x] --focy [y] --focz [z]`: Specifies the focus point of the camera.
- `--upx [x] --upy [y] --upz [z]`: Specifies the up vector for the camera.
- `--zoom [x]`: Specifies the zoom amount for the camera.

As mentioned previously, in order to find sensible values for the camera parameters, it is suggested to manually make one visualization, and when the desired view is obtained, use the "Get Camera" button on the GUI for grabbing the current camera parameters. If omitted, default values for the camera are used, which usually center the object.

2.4 Software Requirements

In order to succesfully compile and use KWMeshVisu, you need to have the following software installed:

- Insight Toolkit - ITK
- Visualization Toolkit - VTK
- KWWidgets
- Tcl/Tk
- CMake 2.2

Note that KWWidgets is not part of the current environment for Insight Journal. However, it can be obtained from `http://www.kwwidgets.org/Wiki/KWWidgets`.

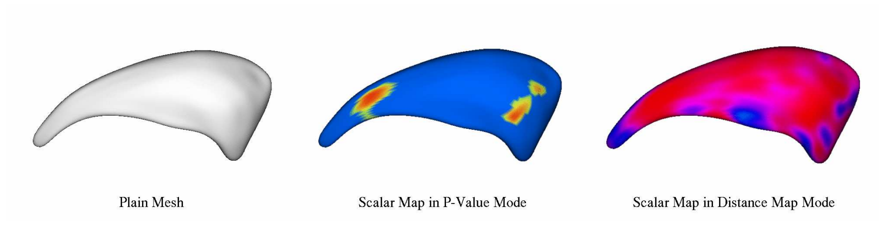

Figure 2: Visualizing a scalar map on a mesh. The scalar map visualization clearly enables the user to quickly detect the patterns in the map, and which regions on the surface these patterns correpond to.

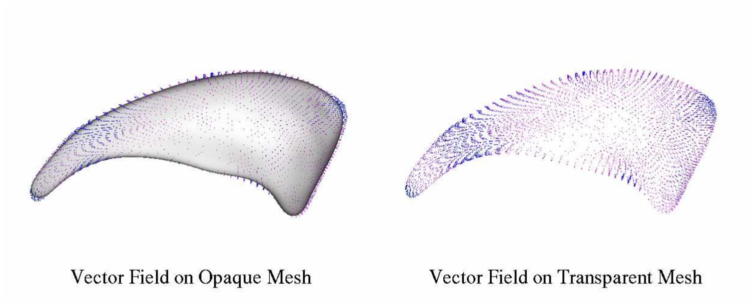

Figure 3: Visualizing a vector field on a mesh. The mesh opacity can be adjusted depending on the particular application. Here, the same vector field is shown on a completely opaque and a completely transparent mesh, to contrast the two ends of the available range.

3 Results

In this section we present several images generated using KWMeshVisu to illustrate the different types of visualizations that can be achieved with our tool.

Figure 2 illustrates the scalar map visualization. The image to the left is showing a caudate mesh before the scalar map is loaded; to the center, a statistical P-value map is visualized on the same surface. The threshold on this visualization is 0.05, which is a standard value for P-value analysis. Given this visualization, it is easy to locate where on the caudate surface the P-value is significantly low (red regions) for this analysis, and where it is too high to be significant (blue regions). To the right of Figure 2, the same scalar map is visualized in Distance Map mode, with the range 0-1. Here, the overall trends in the data are more visible than specific regions of significance. Clearly, the two images are compatible with each other, although they can lead to different conclusions.

Figure 3 illustrates the vector field visualization. The same data set is shown on an opaque mesh to the left, and on a transparent mesh to the right. Clearly, this visualization makes it intuitive to inspect the vector field. It should be obvious that depending on the nature of the dataset and on the purpose of the visualization, the mesh opacity should be adjusted. Usually, fully transparent meshes make it very difficult to grasp the whole data structure, whereas fully opaque meshes make it impossible to inspect inward-pointing arrows. A value around 0.5 usually gives the best results, and this is the default value when a vector field is loaded into KWMeshVisu.

Figure 4 shows an ellipsoid field visualized on a mesh. The coloring is done according to the largest eigenvalue of each ellipsoid.

Figure 5 illustrates how a curve is visualizeAd on a mesh. The curve used on this image is a principal

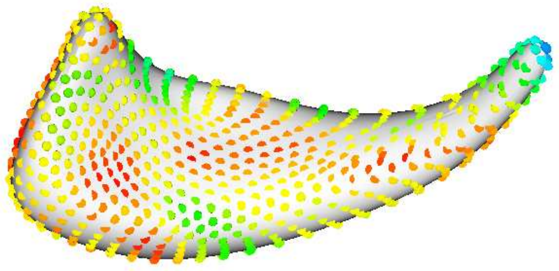

Figure 4: Visualizing an ellipsoid field on a mesh. Ellipsoid fields typically represent local covariances for a population. The coloring is based on the largest eigenvalue of each ellipsoid.

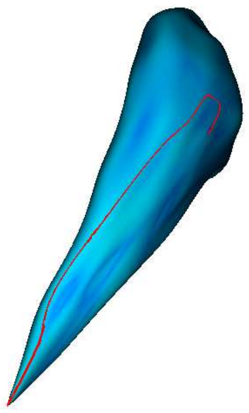

Figure 5: Visualizing a space curve. A curve is shown on top of a scalar map. The dataset simply consists of samples along a smooth curve.

integral curve of the shown caudate, whereas the scalar map is showing local curvature.

It is also possible to simultaneously visualize different types of data to enhance the analysis and to compare how the different datasets compare to each other. However, it should be noted that putting too much data in one image can complicate the images rather than improving the understanding. Undersampling the mesh and the data usually is a good solution to this kind of problem. Also, choosing different colorings for the different datasets can be useful. Figure 6 shows a rather crowded visualization with both a scalar map and a vector map, using the same color schemes. The two datasets are closely matching each other, since the scalar map simply consists of the vector length of the arrows; however, it is hard to assess any more information from this image. Figure 7, on the other hand, is showing an undersampled dataset with different color schemes for the ellipsoid and the scalar maps. The data is much easier to inspect in this manner.

Other images of various visualizations of the results of shape analysis studies, as well as studies where our tool could have been successfully used, can be found in [1, 2, 3, 4, 5, 6, 7, 8]. Clearly, KWMeshVisu is a valuable tool for evaluating and communicating the results of shape analysis studies.

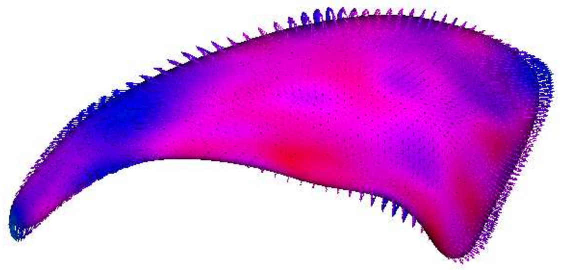

Figure 6: Visualizing two datasets on a mesh. In this particular example, the two datasets closely match each other, since the scalar map is defined by the length of the vector associated with the same location.

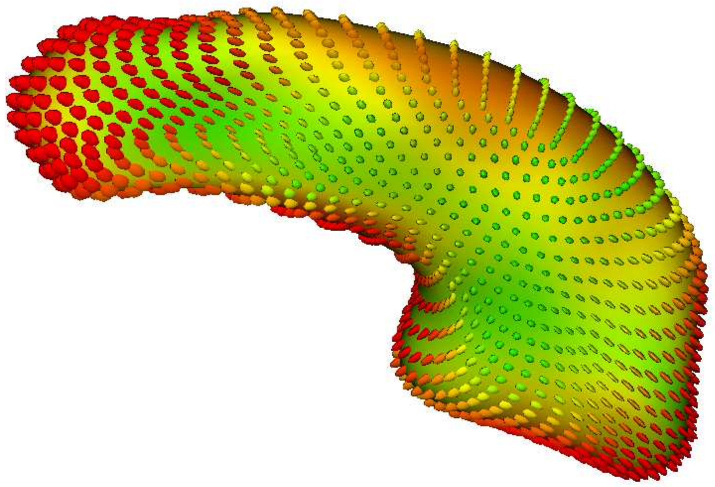

Figure 7: Visualizing two datasets on a mesh. Notice how undersampling the data can help the visualization quality, as compared to a very crowded visualization.

4 Conclusion

We presented an open source visualization tool for inspecting various data sets on surface meshes. KWMeshVisu supports the most commonly encountered data types, such as scalar maps, vector fields, ellipsoid fields and space curves. We have shown how this tool is being used in shape analysis studies. As demonstrated in this paper, KWMeshVisu is a powerful tool since it addresses the visualization needs for various common datatypes in a single environment through an intuitive user interface. It also has scripting abilities that are convenient for automation purposes.

Future work includes integrating KWMeshVisu to Slicer 3.0 environment, to further facilitate its usage as a step in a shape analysis pipeline.

5 Acknowledgements

The brain image segmentations used in this manuscript were provided by Jim Levitt and Martha Shenton, (Dept. Psychiatry, VA Boston Healthcare System, Brockton Division, Harvard, Boston). This research is supported by the NIH Roadmap for Medical Research Grant U54 EB005149-01, UNC Neurodevelopmental Disorders Research Center HD 03110, and the NIH NIBIB grant P01 EB002779.

A Implementation Notes

KWMeshVisu is a typical example of how various open source libraries can be used together to create a tool. In this work, various components from ITK, VTK and KWWidgets are used together to form a powerful visualization tool. The mesh is read through ITK meta mesh I/O interface, and is internally converted to VTK PolyData format using the `itkMeshTovtkPolyData` class from InsightApplications toolkit. The datasets are stored as PointData in the PolyData structure. For the vector field and ellipsoid field visualizations, VTK's Glyph structure is used to efficiently render the numerous objects that are introduced to the scene. KWWidgets greatly simplifies the user interface issues since it has built-in support for many of the VTK components that are used, like the `ColorTransferFunction` used for the coloring parameters of the visualization.

References

[1] LH Cavidanes, M Styner, and WR Proffit. Image analysis and superimposition of 3-dimensional cone beam computed tomography models. *American Journal of Orthodontics and DentoFacial Orthopedics*, 129(5):611–618, 2006. 3

[2] D Pantazis, RM Leahy, TE Nichol, and M Styner. Statistical surface-based morphometry using a non-parametric approach. In *IEEE International Symposium on Biomedical Imaging*, pages 1283–1286, 2004. 3

[3] M Styner, M Jomier, and G Gerig. Closed and open source neuroimage analysis tools and libraries at unc. In *IEEE International Symposium on Biomedical Imaging*, pages 702–705, 2006. 3

[4] M Styner, JA Lieberman, RK McClure, DR Weinberger, DW Jones, and G Gerig. Morphometric analysis of lateral ventricles in schizophrenia and healthy controls regarding genetic and disease-specific factors. *Proc. of the National Academy of Sciences*, 102(13):4872–4877, 2005. 3

[5] M Styner, JA Lieberman, RK McClure, DR Weinberger, DW Jones, and G Gerig. Morphometric analysis of lateral ventricles in schizophrenia and healthy controls regarding genetic and disease-specific factors. *Proc. of the National Academy of Sciences*, 102(13):4872–4877, 2005. 3

[6] M Styner, JA Lieberman, D Pantazis, and G Gerig. Boundary and medial shape analysis of the hippocampus in schizophrenia. *Medical Image Analysis*, 8(3):197–203, 2004. 3

[7] M Styner, I Oguz, RG Smith, C Cascio, and M Jomier. Corpus callosum subdivision based on a probabilistic model of inter-hemispheric connectivity. In *Medical Image Computing and Computer Assisted Intervention*, pages 765–772, 2005. 3

[8] S Xu, M Styner, B Davis, S Joshi, and G Gerig. Group mean differences of voxel and surface objects via nonlinear group averaging. In *IEEE International Symposium on Biomedical Imaging*, pages 758–761, 2006. 3