Visualization of Stellar Models

Christian Gray
RIT Computer Science Masters Project Proposal
July 31, 2005

1. The Problem

Stellar modeling on the galactic level involves the calculation of millions of data points. To make use of these datasets they must be effectively visualized so that sense can be made from the raw numbers. Visualization allows the observer to see information that is not normally visible. For stellar models this could be information such as gravity. Some challenges exist with the visualization of a truly three-dimensional dataset of this size.

Traditional methods of displaying datasets of this size suffer from over-plotting. Over-plotting occurs when the number of data points contained in one area exceeds our ability to discern or the medium's ability to display individual points. Graphs such as a scatter-plot become single blobs, losing all detail and structure of the data. Another problem stems form the three-dimensional nature of the data. Since we can only see in 3D, data points can potentially block other data points. We perceive only the outer boundary of this cloud not its insides.

2. Overview of Solutions

To tackle this problem I will explore three volume rendering techniques. The first one will generate 2D slices of the dataset. This technique will offer very accurate detail of the data, but suffers from only being able to see a small subset of the data at a time. Then next techniuque will be the generation of iso-surfaces from the volume data. These surfaces will represent the boundary levels between two values of interest. By using shading multiple surfaces can be combined to show a larger view of the data, but some preciseness is lost through obstruction. The final method will be a direct volume rendering. In this method the data from the volume is rendered directly using all the volume data. This technique works well for very fluid or gas like data and where values can be easily classified. While visualizing the most information it is also the least precise of the three methods.

3. Methods

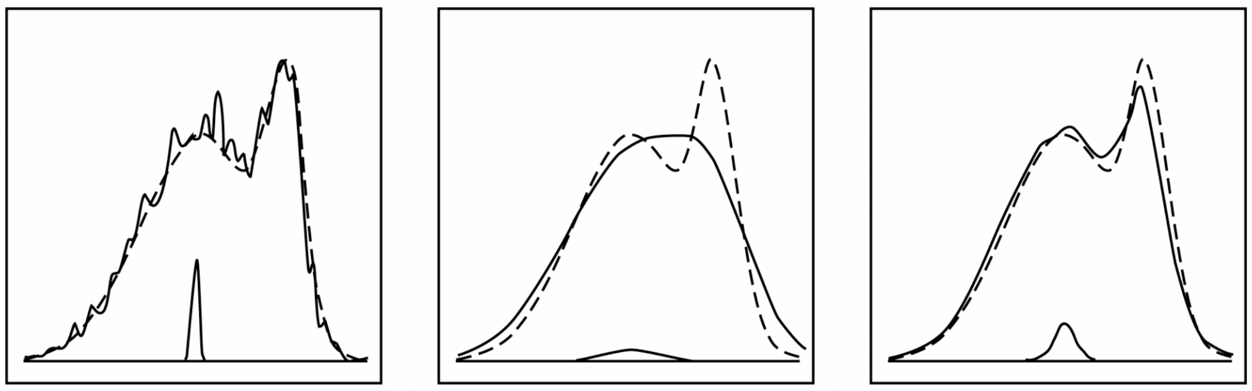
3.1 Kernel Density Estimator

To overcome the effects of over-plotting and to transform the dataset into a volume data set I will use a density map. The density map will be created using a kernel density estimator. Density maps allow the underlying structure of the data to be maintained though individual points are still lost. A kernel density estimator performs a local averaging on a dataset to smooth out and fill in missing data. The formula for a kernel density estimator is given as follows [5]:

$$\hat{f}(x;h) = (nh)^{-1} \sum_{i=1}^{n} K\left\{\frac{x - X_i}{h}\right\}. \qquad (1)$$

K is the kernel, it is a unimodal probability density function, satisfying ∫ K (x) dx = 1. Bandwidth is represented by h. Bandwidth determines the amount of smoothing that takes place.

A small bandwidth produces a density estimation with a large amount of variance. A large bandwidth can oversmooth the data causing a loss in structure. A good bandwidth maintains the structure of the data while smoothing out the local variance in the data. The choice of bandwidth



is perhaps the single most important factor in the kernel density estimation, in my case the choice will be largely subjective and adjusted primarily on the quality of the image produced rather then how well it reproduces some underlying function.

The choice of the function for K is secondary though related to the bandwidth selection. Any function that satisfies the specifications above can be used; for this project I'll use the normal density function,

$$f(x) = (2\pi)^{-1/2} e^{-x^2/2}. \quad (2)$$

This function gives the standard bell shaped curve centered at 0. Substituting equation 2 into equation 1 gives:

$$\hat{f}(x;h) = (nh)^{-1} \sum_{i=1}^{n} (2\pi)^{-1/2} e^{-\left(\frac{x-X_i}{h}\right)^2/2}. \quad (3)$$

Since we are interested in the density of the number of stars in the neighborhood of some point P_a (x_a, y_a, z_a) we can substitute the distance between the point of interest and the positions of the stars for $x - X_i$,

$$\hat{f}(x;h) = (nh)^{-1} \sum_{i=1}^{n} (2\pi)^{-1/2} e^{-\left(\frac{\sqrt{\left((x_a-x_i)^2+(y_a-y_i)^2+(z_a-z_i)^2\right)}}{h}\right)^2/2}. \quad (4)$$

The bandwidth determines how far away we want to look for determining the density at P_a (x_a, y_a, z_a). The result from the kernel density should be a value between 0 and 1.

The kernel density estimator will be used to sample the original dataset on a regular three-dimensional grid. Since the display device dictates the maximum resolution capable of being displayed the grid will be calculated on a grid twice the resolution of intended output. If we intend to display at 640 x 480 then our density estimation would be based on a grid 1280 x 960 x 1280. The choice of twice the resolution is not arbitrary. This choice is based on the Nyquist Theorem from sampling theory. The Nyquist Theorem states that a signal can be recreated if the sample was captured at a frequency at least twice the highest frequency of the original signal. In this case our highest frequency is our display resolution. Using this theorem will allow us to save processing time by preventing over sampling and will provide us with good detail by not under sampling.

3.2 Slicing

The first method for display of the data will be the use of slices. These slices are planer cross sections of the data. This technique is widely used in medical imaging to display cross section of internal organs.

3.2.1 Slicing Algorithm

3.2.1.1 Define Plane

The first step in the algorithm is to define the plane we are interested in. A plane can be defined by a point $P_a(x_a, y_a, z_a)$ and a normal vector N(A, B, C) that is perpendicular to the plane.

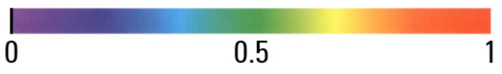
3.2.1.2 Determine Data on the Plane

All data points from the kernel density estimation some threshold *t* from the plane will be used for rendering. Given a data point $P_b(x_b, y_b, z_b)$ and the plane given above we calculate the minimum distance D from the plane and a point as follows:

$$D = \frac{A(x_b - x_a) + B(y_b - y_a) + C(z_b - z_a)}{\sqrt{A^2 + B^2 + C^2}}$$

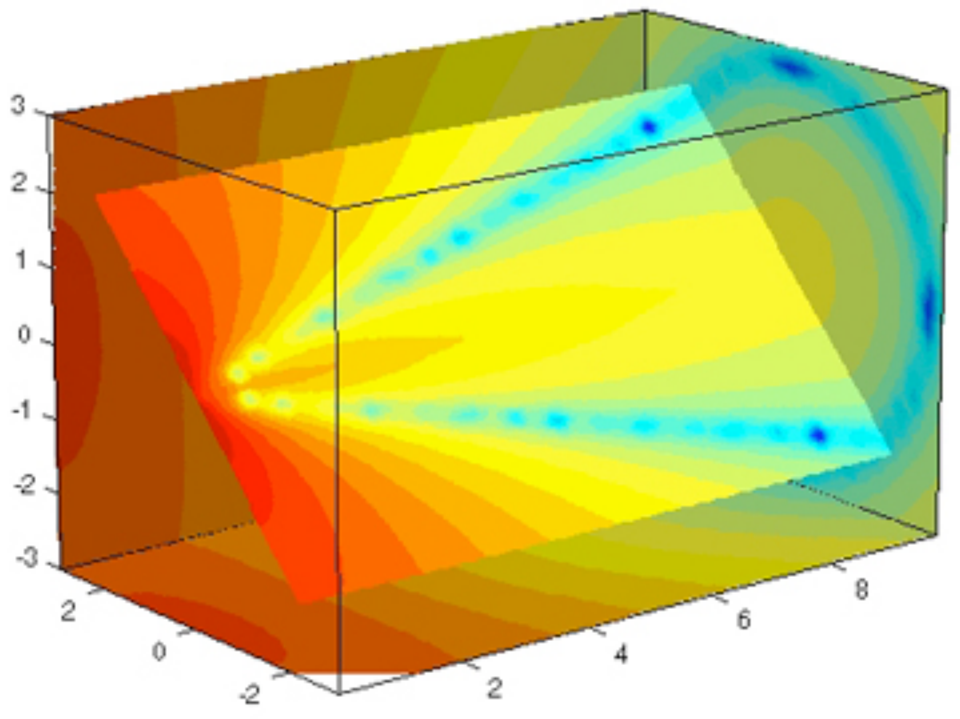
3.2.1.3 Map Values

Data from the kernel density estimation will contain values ranging from 0-1. In this step those values are mapped to a color for display.



3.2.1.4 Render Plane

The final step is to render the plane using the value at each intersecting point and looking up the color to be rendered in the color map. Multiple planer slices can be rendered to produce a more three-dimensional feel for the data. The picture below shows an example of this technique being used in a plot of fluid flow.



3.3 3D Contours

The second technique I use will be to extract a triangle mesh representing three-dimensional contours of similar density. One can think of these contours as thin shells representing the boundary between two densities. These meshes should have several important properties. First, they should represent the contour as accurately as possible. Second, they should be constructed using a minimum number of triangles. Finally the algorithm for extracting the contours should allow for both opened and closed surfaces that may be non continuous. In research I found a paper that details an algorithm that should be adaptable to my needs.

3.3.1 Mesh Extraction Algorithm

"Fast Extraction of Adaptive Multiresolution Meshes with Guaranteed Properties from Volumetric Data" by Gavrilu, Carranza, Breen, and Barr, details an algorithm for extracting adaptive multiresolution triangle meshes from volume datasets [1]. It provides fast extraction times comparable to the most widely used method for extracting meshes from volume data, Marching Cubes, as well as the properties listed above.

The Algorithm has four parts. In part one the volume dataset is parsed to produce a connectivity graph. Part two, extracts a course disk coverings of the connectivity graph. Part three, refines the course disk covering to the desired accuracy. Finally, in part four the Voronoi Neighborhoods of the disk covering are found. The dual of a Voronoi diagram is a Delaunay diagram which gives us the triangulation mesh.

3.3.1.1 Stage One – Candidate Vertex Connectivity Graph

The first stage of producing the connectivity graph is a refined version of the Marching Cubes algorithm. The surface of the contour will intersect with the cubes created by the grid. By determining which intersections of the grid are inside and which are outside the conceptual surface patches, surfels [1][6], are created. The intersections of these surfaces with the grid are call nodes. With each surfel a vertex, c-vertex [1], is associated. The coordinates of a c-vertex are the average of its associated nodes. C-vertices are neighbors if they share one or more nodes. The connectivity graph consists of c-vertices, the edges of which connect neighbors and are weighted by the Euclidean distance between them

3.3.1.2 Stage Two – Simple Disk Covering

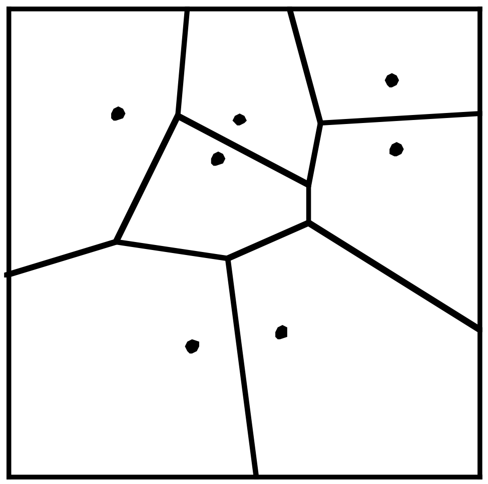
Part two extracts a simple disk covering from the connectivity graph. A simple disk covering consists of simple disk neighborhoods. A disk neighborhood consists of all the c-vertices who have a [v, w]-geodesic with a length less then the radius α. A [v, w]-geodesic is the shortest path between c-vertices v and w in the connectivity graph. A disk neighborhood is simple if for all the c-vertices in the disk neighborhood there exists only one [v, w]-geodesic. A disk covering of the connectivity graph is the set of c-vertices and real number pairs [v, α] such that all c-vertices w are in the disk neighbor hood consisting of vertex v and radius α.

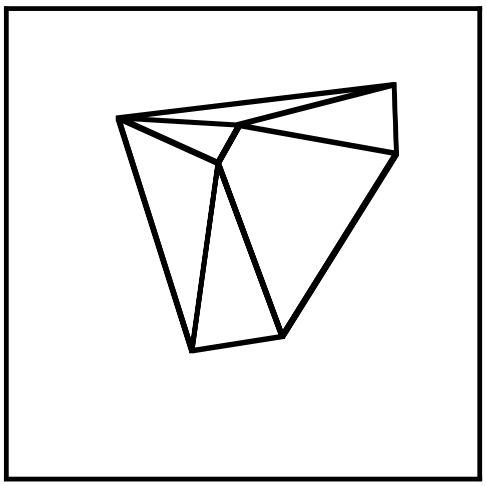
3.3.1.3 Stage Three – Refine Disk Covering

The third part of the extraction algorithm refines this course disk covering by adding more disk neighborhoods to the covering. A user supplies a maximum mesh error ε. The algorithm loops through all the disk neighborhoods in the disk covering splitting and adding more until all disk neighborhoods are less then the error. Several different error metrics can be applied. The simplest calculates the maximum distance form the surface. Another useful metric calculates the curvature of the disk covering, and refines the mesh in areas of high curvature.

3.3.1.4 Stage Four-Triangulation

In the final stage of the algorithm the Voronoi covering of the connectivity graph is computed using the refined disk covering. Taking the dual of the Voronoi (left diagram below) covering gives the Delaunay triangulation (right diagram), which is the triangle mesh.

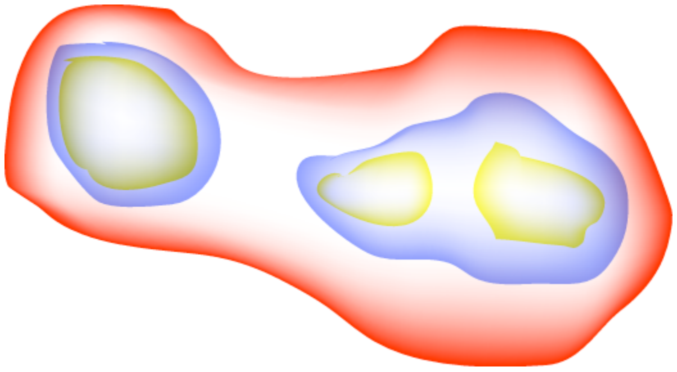




3.3.2 Mesh Rendering

Once the mesh is complete the final rendering can now be done. As mention above the a benefit of the extra work above the Marching Cubes algorithm is that the final mesh has much fewer triangles this will hopefully allow us to render the mesh in real time while rotating it so that the observer can explore the complexities of any contour extracted.

The shader will emphasize the edges and areas of curvature in contours. The shader will use the angle of incident from the ray cast from the viewer. As the angle of incident goes from 90˚ to 0˚ the opacity will go from 0 to 100. The results should look something like the following with nested contours:



With the capability to rotate the viewer will be able to look at multiple contours and be able to discern the three dimensional shape of each one.

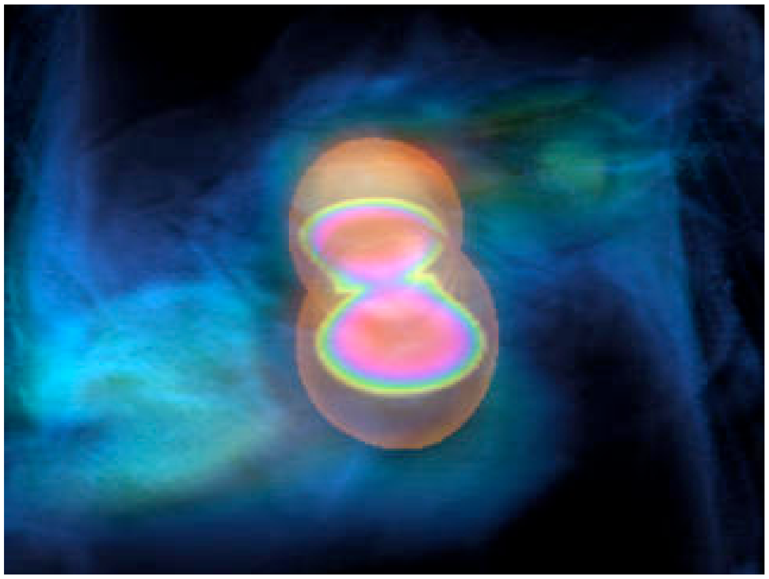
3.4 Direct Volume Rendering

The last method of visualization to be tried will render the volume data directly. In this method rays are cast through the volume accumulating color and opacity at each intersected voxel. This method is most useful when values of interest are known ahead of time.

3.4.1 Ray Casting Algorithm

Cast a ray through each pixel of the image plane. For each voxel $v(x_i, y_i, z_i)$ along the ray look up the value in the classification table. If the classification has changed compute the normal and color value. Next weight the color by the opacity value from the classification table. Finally, you add this contribution to the accumulated values so far.

This technique works best with data that can be easily classified. In medical imaging scans of the human body can easily classified as bone, muscle, skin, etc. That means that this technique may have limited application to our volume. It will be able to show the fluid nature of our volume more readily then the other techniques and in combination with the other techniques (figure below) will add an additional level of information.



4. Measure of Success

A hard metric cannot be generated for the effectiveness of the methods detailed in this paper. This is due to the subjective nature of images. To determine their effectiveness the generated images will be show to experts in the field of astrophysics. These experts will be able to study the images and determine how well the images convey information found in the data.

5. Schedule

Proposal Approved

Kernel Density Estimation Algorithm (1 month)

Slicing Algorithm (2 months)

Mesh Extraction (3 months)

Ray-Casting (2 months)

Final Report (1 month)

7. References

[1] Gavriliu, Marcel, et al. "Fast Extraction of Adaptive Multiresolution Meshes with Guaranteed Properties from Volumetric Data." VISUALIZATION. In Proceedings of the Conf. on Visualization '01 (San Diego, California, October 21 26, 2001). Washington, DC: IEEE Computer Society, 2001. 295 303. ACM Portal. Wallace Lib., Rochester Institute of Technology, Rochester, NY. 12 July 2005 <http://portal.acm.org>.

[2] Klein, Rolf. Concrete and Abstract Voronoi Diagrams. Ed. G. Goos and J. Hartmanis. Lecture Notes in Computer Science 400. Berlin, Germany: SpringerVerlag, 1989.

[3] Lorensen, William E., and Harvey E. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm." Proceedings of the 14th Annual Conf. on Computer Graphics and Interactive Techniques. International Conf. on Computer Graphics and Interactive Techniques. New York, NY: ACM, 1987. 163169. ACM Portal. Wallace Lib., Rochester Inst. of Technology, Rochester, NY. 12 July 2005 <http://doi.acm.org/10.1145/37401.37422>.

[4] Scott, David W. Multivariate Density Estimation: Theory, Practice, and Visualization. New York: John Wiley & Sons, Inc., 1992.

[5] Wand, M. P., and M. C. Jones. Kernel Smoothing. Boca Raton: Chapman & Hall/CRC, 1995.

[6] Wood, Zoe J., et al. "Semi Regular Mesh Extraction from Volumes." VISUALIZATION. In Proceedings of the 11th IEEE Visualization 2000 Conf. Washington, DC: IEEE Computer Society, 2000. ACM Portal. Wallace Lib., Rochester Inst. of Technology, Rochester, NY. 12 July 2005 <http://portal.acm.org>.