Constructing 3D Elliptical Gaussians for Irregular Data

Wei Hong[1], Neophytos Neophytou[2], Klaus Mueller[3], and Arie Kaufman[4]

[1] Center for Visual Computing and Department of Computer Science Stony Brook University weihong@cs.sunysb.edu
[2] Center for Visual Computing and Department of Computer Science Stony Brook University nneophyt@cs.sunysb.edu
[3] Center for Visual Computing and Department of Computer Science Stony Brook University mueller@cs.sunysb.edu
[4] Center for Visual Computing and Department of Computer Science Stony Brook University ari@cs.sunysb.edu

Summary. Volumetric datasets obtained from scientific simulation and partial differential equation solvers are typically given in the form of non-rectilinear grids. The splatting technique is a popular direct volume rendering algorithm, which can provide high quality rendering results, but has been mainly described for rectilinear grids. In splatting, each voxel is represented by a 3D kernel weighted by the discrete voxel value. While the 3D reconstruction kernels for rectilinear grids can be easily constructed based on the distance among the aligned voxels, for irregular grids the kernel construction is significantly more complicated. In this paper, we propose a novel method based on a 3D Delaunay triangulation to create 3D elliptical Gaussian kernels, which then can be used by a splatting algorithm for the rendering of irregular grids. Our method does not require a resampling of the irregular grid. Instead, we use a weighted least squares method to fit a 3D elliptical Gaussian centered at each grid point, approximating its Voronoi cell. The resulting 3D elliptical Gaussians are represented using a convenient matrix representation, which allows them to be seamlessly incorporated into our elliptical splatting rendering system.

1 Introduction

Direct volume rendering is an important technology in the fields of computer graphics, as well as scientific and medical visualization. It allows the user to comprehend and visualize a volumetric dataset directly, without requiring the generation of a polygonal iso-surface. Volumetric datasets are commonly classified as *rectilinear* or *non-rectilinear*, according to their grid structure. Here, both the curvilinear and the unstructured grids belong to the class of non-rectilinear grids, while cubic grids are the simplest case of rectilinear grids. The volumetric datasets obtained from scientific simulation and partial

differential equation solvers are typically given in the form of non-rectilinear grids.

The straightforward method to visualize non-rectilinear grids is to resample them into a rectilinear grid [1], where usual rendering methods readily apply. However, as a non-rectilinear grid may consist of cells of drastically different sizes, the resampling approach may either cause a loss of important information or result in a huge dataset. Thus, several techniques have been developed for the direct volume rendering of non-rectilinear grids, i.e. ray casting, cell projection, and splatting. Ray casting is the most popular direct volume rendering technique where volume rendered images are generated by casting rays from the viewer's eye, through the screen pixels, into a 3D volume, and compositing the contributions of all sample points taken along each ray into the corresponding screen pixel. Many algorithms for the ray casting of non-rectilinear grids have been developed [2] [3] [4]. Since a non-rectilinear grid may be composed of cells of drastically different sizes, sampling with a constant interval along a ray is not desirable. Therefore, sample points are usually taken at the intersections of rays and cells, which tends to be very time-consuming. In the cell projection technique [5], a cell in a volume is projected onto the screen, and its contribution to the pixels under its projection extent is calculated and composited with the contributions from the previously projected primitives. Cell projection algorithms need to obtain the proper cell visibility ordering to generate the correct compositing result. Here, the cell visibility ordering itself is not trivial and can be rather time consuming.

The splatting technique has become quite popular for directly rendering volumetric datasets of various grid structures. The original algorithm, first proposed by Westover [6] for rectilinear grids, projects each voxel onto the image plane and composites the result into an accumulation image. As each voxel is projected onto the image plane, the voxel's energy is spread over the image raster using the 2D projection of a 3D reconstruction kernel, which is centered at the voxel's projection point. For regular grids, the 3D reconstruction kernel, also called a *splat*, is spherically symmetric and centered at a voxel. Since the splat is reconstructed into a 2D image raster, it can be implemented as a 2D reconstruction kernel called a "footprint function", containing the integration of the 3D kernel along the projection direction. By ways of two-dimensional texture mapping, rectilinear grids can be quickly rendered with a single polygon per voxel and using a single Gaussian kernel for all voxels. The direct extension of this technique to non-rectilinear grids is not straightforward, because the appropriate kernels for non-rectilinear grids are not easy to calculate. In this case, the splats are arbitrary ellipsoidal kernels, with their shape being defined by the local grid structure.

Both ray-casting and cell projection algorithms have been extended for the volume rendering of non-rectilinear grids. Recently, graphics hardware has been used to accelerate ray-casting [7] and cell projection [8] [9] algorithms for irregular grids. However, both of these modalities have some limitations. For the cell projection algorithm, the piecewise linear interpolation may result in

banding at cell boundaries, degrading the quality of the resulting image. In addition, cell projection approaches are limited by the cell visibility sorting, which prevents the current graphics hardware from running at full capacity. For ray-casting algorithms, the ray-cell intersection test, the identification of the face of the cell through which a ray exists, and the interpolation from the surrounding grid points are very expensive operations. Even the hardware accelerated ray-casting algorithm [7] can not achieve interactive rendering speed.

In an attempt to overcome these problems, we propose a new approach that utilizes splatting, in conjunction with arbitrarily shaped elliptical Gaussians, for the rendering of irregularly gridded data. Our splatting approach offers the following advantages: (i) its smooth and overlapping kernel functions will reconstruct a smooth representation of the grid-sampled signal, without the artifacts of the piecewise linear representations of the cell projection approaches; (ii) it promises to be more efficient than ray casting due to the ease of footprint rasterization, especially when implemented in hardware; (iii) it is also more efficient than other splatting approaches for irregular grids, since the space-filling kernels are only required at the grid points, and thus the rendering complexity matches that of the grid. Finally, apart from non-rectilinear gridded data, our method also supports collections of scattered data points.

The main topic of this paper is the method for constructing arbitrarily oriented elliptical Gaussians from irregular grid topologies. Once the 3D reconstruction kernels are found, the software rendering is straightforward. We can either use the sheet buffer algorithm for composited rendering [10], or we can just splat the whole kernel for X-ray type rendering.

2 Previous Work

Only a limited amount of work has been done so far on how to use the splatting algorithm for the rendering of irregular grids. Meredith and Ma [11] proposed a spherical Gaussian splat-based rendering method for irregular data. In this method, they use an octree with roughly the same number of data points stored at each leaf node. No connectivity information is stored for the data points. For any given viewing parameters, they calculate the projected size of any octree node on the screen. Then they divide the screen area based on the number of data points within that octant to calculate the approximate kernel size. This method can only give a rough estimate of the kernel size.

Mao et al. [12] [13] presented a method that resamples irregular grids with a set of new points whose energy support extents in the 3D physical space can be approximated by spheres or ellipsoids. To approximate the scalar field represented in the original grid as accurately as possible without using too many sample points, an adaptive three-dimensional stochastic sampling method called Poisson sphere/ellipsoid sampling is employed. Then, after the new splat distribution has been calculated, the original splatting algorithm can

be used to render the irregular grid. The disadvantage of this method is that the original grid must be resampled to compute the scalar value for the new sample points. The error caused by this non-regular resampling potentially degrades the quality of the resulting images. In addition, this method also generates considerably more splats than the original number of grid points. For example, the NASA Blunt Fin dataset with resolution of $40 \times 32 \times 32$ is resampled with $79,971$ sphere points and $5,041$ ellipsoid points, which more than doubles the number of points. Moreover, this method cannot be used to render scattered data.

Jang et al. [14] developed a procedure, based on the Voronoi cell describing the region around a point, to place and orient Gaussian kernels to give more uniform coverage in non-uniform cells. However, they did not specify how they construct these splats. This method also need to resample the original non-uniform cells. Jang et al. [15] performed a global optimization method to fit radial basis functions (RBFs) to irregular data. In this paper, we propose a method to construct 3D ellipsoidal kernels for irregular grids without the need for an error prone resampling of the original grid, since our 3D reconstruction kernels are still centered at the original grid points. Instead, a weighted least squares method is used to fit a single ellipsoidal kernel to the Voronoi cell at each grid point, which is a a local optimization method. In our method, we do not interpolate any additional data points.

3 Creating 3D Elliptical Gaussian

The shape of a 3D elliptical Gaussian kernel centered at the origin can be modeled via the implicit equation of an ellipsoid:

$$Ax^2 + By^2 + Cz^2 + 2Dxy + 2Exz + 2Fyz - 1 = 0 \tag{1}$$

This equation has six unknowns and represents a quadric surface. The quadric surface can be represented by using matrix notation, giving rise to a 3×3 symmetric quadric matrix Q:

$$Q = \begin{vmatrix} A & D & E \\ D & B & F \\ E & F & C \end{vmatrix} \tag{2}$$

The quadric surface represented by Q can be easily translated, scaled, and rotated by multiplying it with a transformation matrix. Given a 3×3 affine transformation matrix M, the transformed quadric surface $Q^{'}$ is given by:

$$Q^{'} = (M^{-1})^t \cdot Q \cdot M^{-1} \tag{3}$$

With this representation we can create an arbitrarily oriented elliptical Gaussian by applying the scaling and rotation transformations contained in

matrix $S = \{a, b, c\}$ and R on a unit sphere, respectively, as described in the following equation:

$$Q = (R^{-1})^t \cdot (S^{-1})^t \cdot I \cdot S^{-1} \cdot R^{-1} = R \cdot (S^{-1})^2 \cdot R^t \tag{4}$$

Here I is the identity matrix which represents the unit sphere, and $(S^{-1})^2 = \{1/a^2, 1/b^2, 1/c^2\}$ is a diagonal matrix, which can be thought of as a scaling matrix. It represents an axis aligned ellipsoid. The rotation matrix R is an orthogonal matrix representing the ellipsoid orientation, which can be defined by three rotation angles α, β, and γ along the three axes. Instead of directly fitting an ellipsoid using Equation 1, we fit the scaling matrix S and rotation matrix R separately, using Equation 4. S and R are decided by the three scaling factors and the three rotation angles, respectively. Due to this matrix representation, the resulting ellipsoidal kernel can easily be incorporated into our rendering algorithm, which represents the elliptical splats using a rotation and a scaling matrix.

The irregular grids are always described in terms of their grid structure. However, in our algorithm we are only interested in the grid points. In that respect, we treat an irregular grid as a volumetric point cloud. Our algorithm only uses these grid points as input for generating the 3D ellipsoidal kernels. In the following sections, we describe our approach to fit the 3D ellipsoidal kernel using the matrix representation.

3.1 Guide Points

As is well known, the dual of the Delaunay triangulation is the Voronoi diagram, which consists of cells around the data points such that any location in a particular cell is closer to that cell's generating point than to any other. Thus, the shape of the Voronoi cell can give us a clue about the shape of the reconstruction kernel. The main idea of our algorithm is to fit elliptical Gaussian kernels to the grid points by approximating their Voronoi cells. We show a 2D example in Figure 1, in which the Voronoi cell of grid point V_0, shown in red, is approximated with an ellipse, shown in blue. The Voronoi cell of V_0 is obtained by connecting the circumcenters between pairs of Delaunay triangles that are adjacent and both incident to V_0.

As the first step of our algorithm, we apply the 3D Delaunay triangulation algorithm to the input grid points. Through the 3D Delaunay triangulation, we obtain for each grid point a list of tetrahedra incident to it. The circumcenters of these tetrahedra are the vertices of the Voronoi cell generated for that grid point, i.e. the cell's generating point. In the ideal case, each circumcenter is shared by four reconstruction kernels, with each of these contributing 25% to it. This would mean that the elliptical Gaussian kernel passes through these circumcenters with the 0.25-valued iso-contour. Furthermore, in this ideal case, the 0.5-valued iso-contour of the Gaussian kernel should pass through the midpoints of the edges joining the cell's generating point.

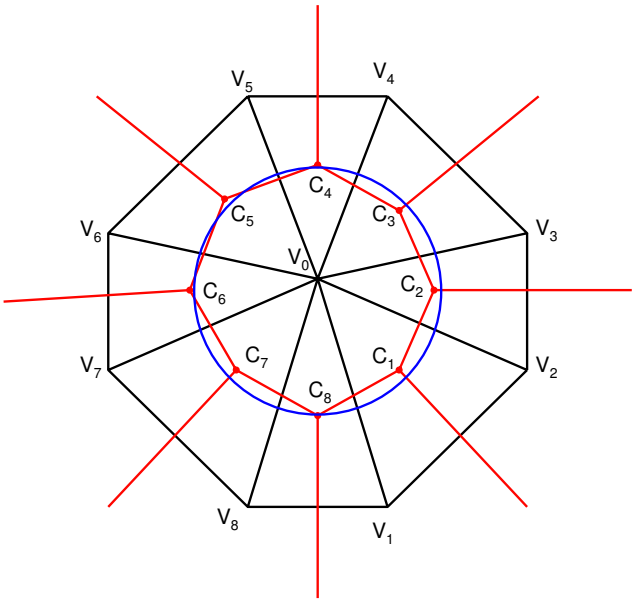


Fig. 1. A Voronoi cell is approximated by an ellipse shown in two-dimensions. V_0 is the grid point for which the Voronoi cell is constructed, $V_1, \ldots, V_8$ are neighboring grid points, and C_i is the circumcenter of triangle $V_0V_iV_{i+1}$.

However, in the general case these edge midpoints do not capture the shape of the Voronoi cell as well as the circumcenters. This is illustrated in Figure 2, where the edge midpoints are shown as red dots, and the ellipse fitted from these midpoints is also shown in red. The ellipse fitted from the circumcenters is shown in blue. From this example, we can see that the blue ellipse approximates the Voronoi cell, shown in green, much better than the red ellipse. Therefore, we use the circumcenters of the incident tetrahedra as guide points for fitting the ellipsoid that approximates the 0.25-valued iso-contour of an elliptical Gaussian kernel.

If a tetrahedron is almost flat, its circumcenter is located far away from this tetrahedron. In this case, the Voronoi cell is an inferior shape for fitting the grid points kernel. Thus, if the circumcenter is too far away from the center grid point, we use the circumcenter of the triangle opposite to the center point as the contour guide point.

For each such guide point, we use its corresponding solid angle in the tetrahedron as the weight. Thus, for each grid point we have a list of weighted guide points associated with it. This list of the weighted guide points are then fed to a weighted least squares algorithm to fit the elliptical Gaussian kernels.

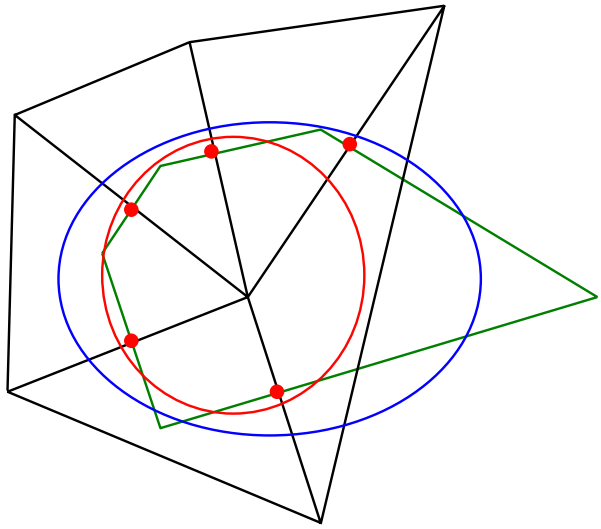


Fig. 2. In more general cases, as shown here, the edge midpoints do not capture the shape of a Voronoi cell as well as the circumcenters.

3.2 Initial Guess

Before we use the least squares method to fit an ellipsoid at each grid point based on the generated weighted guide points, we analyze the guide points using the Principal Component Analysis (PCA) [16] method to estimate the ellipsoid defined by the guide points. A PCA analysis of the guide points performs an eigen-decomposition of the covariance matrix of the guide points. This produces three eigenvalues and corresponding eigenvectors, which in 3D define a local orthogonal coordinate system related to the ellipsoid induced by the guide points.

Suppose the given grid point is v and the related N guide points are $u_i, i = 1, 2, \ldots, N$. We use the following equation to compute the covariance matrix M:

$$M = \sum_{i=1}^{N} (u_i - v)(u_i - v)^t \tag{5}$$

where M is a 3×3 matrix. From this 3×3 covariance matrix, we can compute the three eigenvalues and the corresponding eigenvectors. We use the three eigenvalues as the initial guess for the three scaling factors. The corresponding eigenvectors form a rotation matrix, which yields the initial guess for the three rotation angles. The PCA analysis makes the minimization process convergence faster by providing a good guess of the ellipsoid.

3.3 Energy Function

Given a set of weighted guide points $(w_i, u_i), i = 1, 2, \ldots, N$, in order to use the weighted least squares method to fit them with equation 4, we need to design an energy function for minimization. For this purpose, we use the sum of the weighted distances from the guide points to the quadric surface Q. This yields the energy function:

$$E(a, b, c, \alpha, \beta, \gamma) = \sum_{i=1}^{N} (w_i \times d_i^2) \tag{6}$$

where w_i is the weight of the guide point u_i, and d_i is the distance from point u_i to the ellipsoid Q, which is the length of the shortest line segment connecting u_i to any point on Q. For a given guide point $u_i = (x_i, y_i, z_i)$, Hart [17] proposed an algorithm to compute the closest point $u_i' = (x_i', y_i', z_i')$ on an axis-aligned ellipsoid defined by equation $f(x, y, z) = (x/a)^2 + (y/b)^2 + (z/c)^2 - 1 = 0$. As we know, the vector $\overrightarrow{u_i' u_i}$ is normal to the surface defined by $f(x, y, z)$ at u_i', which satisfies the following equation:

$$x_i - x = t\frac{x}{a^2}, y_i - y = t\frac{y}{b^2}, z_i - z = t\frac{z}{c^2} \tag{7}$$

Plugging this equation into $f(x, y, z)$ confines the point to the ellipsoid, producing:

$$\frac{a^2 x_i^2}{(t + a^2)^2} + \frac{b^2 y_i^2}{(t + b^2)^2} + \frac{c^2 z_i^2}{(t + c^2)^2} = 1 \tag{8}$$

This equation is equivalent to a sixth degree polynomial, obtained by multiplying through by the denominators. The largest root of this polynomial corresponds to the closest point on the ellipsoid. There are no closed formulas for the roots of such polynomials. We use a Newton's iteration method to find the largest root. When we obtain the largest root t_0 of this polynomial, the closest point $u_i' = (x_i', y_i', z_i')$ on the surface of the ellipsoid is obtained by plugging t_0 into Equation 7, which yields the following equation:

$$x_i' = \frac{a^2 x_i}{t_0 + a^2}, y_i' = \frac{b^2 y_i}{t_0 + b^2}, z_i' = \frac{c^2 z_i}{t_0 + c^2} \tag{9}$$

Then, the distance from the point u_i to the ellipsoid is exactly the distance between u_i and u_i'.

To compute the distance from guide point u_i to an arbitrary oriented ellipsoid $Q = R \cdot (S^{-1})^2 \cdot R^t$, we transform Q and u_i to $Q' = (S^{-1})^2$ and $u_i' = R^{-1} u_i$ respectively by applying matrix R^{-1}, where Q' is an axis aligned ellipsoid centered on the origin. Then, the distance from u_i to Q is the distance from u_i' to Q' in the new coordinate system, which can be computed using the above equations. Next, we employ an iterative method to compute the minimum of E.

3.4 Minimization

The energy function of Equation 6 is a very common unconstrained minimization problem. Powell [18] proposed a minimization method to solve this kind of problem without calculating derivatives. Powell's method ensures convergence in a finite number of steps, for a positive definite quadratic function, by making use of some properties of conjugate directions. However, this method sometimes results in search directions that become linearly dependant. The simplest way to avoid linear dependance of the search directions with Powell's basic procedure, retaining quadratic convergence, is to reset the search directions to the columns of the identity matrix after every n or $n+1$ iterations, where n is number of unknowns in the system. However, the restarting may slow down convergence, because information built up about the function is periodically thrown away. Thus, we use a modification of Powell's basic procedure proposed by Brent in [19] to solve the minimization problem. In consequence, we obtain the scaling matrix S and the rotation matrix R of the 0.25-valued iso-contour for each elliptical Gaussian kernel, which give the shape and orientation of the elliptical Gaussian kernel.

4 Evaluation

The straightforward way to evaluate the resulting 3D elliptical Gaussian kernel configuration is to resample the irregular grid data into a $N \times N \times N$ regular grid R. In the ideal case, if the grid point is inside one of the tetrahedron, the contributions from all kernels to this point sum to one. In practice, the contributions from all kernels to a grid point do not always sum to one. Therefore, the volume rendering image generated with the splatting algorithm may look blotchy. Normalizing the reconstructed value at each grid point by the contribution of reconstruction kernels can alleviate the problem.

The sum of the contributions of all kernels to each grid point can be used to evaluate the quality of the fitted 3D elliptical Gaussian kernels. Suppose the set of regular grid points inside the tetrahedra mesh is V. The standard deviation is computed as follows:

$$S = \sqrt{\frac{\sum_{v \in V} (C_v - 1.0)^2}{|V|}} \tag{10}$$

where C_v is the sum of the contributions from the reconstruction kernels to grid point v. The standard deviation S with a small value indicates a better quality of the reconstruction kernel ensemble, constructed via our fitting procedure.

5 Rendering

Our rendering system uses the sheet-buffered image aligned splatting algorithm introduced in [20] [10] and further refined in [21] for the 4D case. The system was extended in order to rasterize ellipsoids of varying size and orientation. Following this method, the elliptical kernels of the volume are sliced into image aligned sheet buffers. The slices are then shaded per-pixel and composited front-to-back onto the final image. The ellipsoids are defined by a 3×3 rotation matrix and a diagonal 3×3 scaling matrix, as produced by the fitting algorithm described above. Similar to 3D and 4D regular splatting where this method was used, it produces crisp fully shaded images. The process, however, is slightly more demanding when rendering unstructured grids using elliptical splats, because the produced sheet buffers have to be normalized before shading and compositing.

6 Implementation and Results

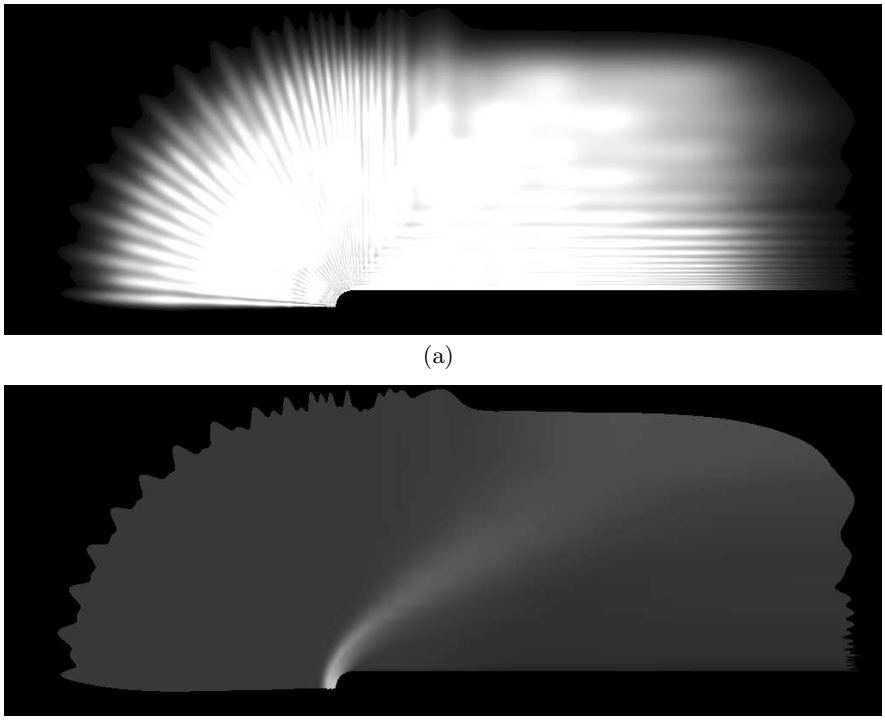
In this section, we present some implementation details and testing results. Our algorithm is implemented using C++ on the Windows platform, and the CGAL C++ library [22] is used to perform the 3D Delaunay triangulation in the preprocessing step. All of the experiments have been conducted on a 3.0 GHz Intel Pentium IV PC running Windows XP with 1G RAM. We list the datasets used in the experiments, the kernel fitting time, and the standard deviation, and max weight in Table 1. Our fitting algorithm can, on average, fit $1,300$ points per minute.

Table 1. Kernel fitting times (in minutes), standard deviation, and max weight for two different datasets.

Dataset	Points	Tetrahedra	Fitting Time	Standard Deviation	Max Weight
Blunt Fin	40,960	187,395	27.8	0.57	2.85
Combustion	47,025	215,040	40.8	0.48	3.15

We use the NASA Blunt Fin dataset with $40,960$ grid points in our first experiment. To perform the numerical comparisons, we use the fitted 3D elliptical Gaussian kernels to resample it into a regular grid. One slice of the resampled regular grid is shown in Figure 3 with two images: (a) the weight image of that slice, and (b) the density image with normalization applied. Both the weight image and the density image look smooth, but somewhat fuzzy at the boundary. The weight image is the key for quality evaluation. The more homogeneous the quality of resulting kernels is the better. We show the 3D elliptical Gaussian kernels in Figure 4 (a) using a surface rendering method. Figure 4 (b) is the volume rendered image using our software splatting algorithm for elliptical splats. We observe that there are some large elliptical

Gaussian kernels located at the boundary, which cause the fuzziness of the volume rendering at the boundary.



(a)

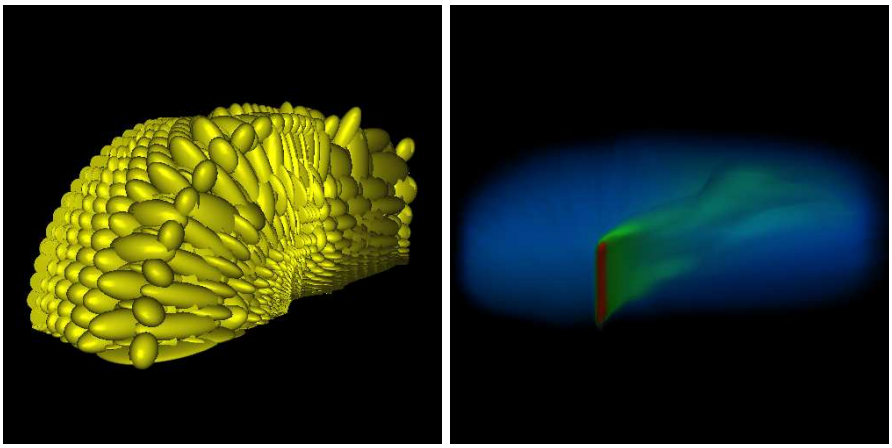
(b)

Fig. 3. Blunt Fin dataset: (a) Weight image and (b) density image for one slice of the regular grid samples evaluated using the fitted 3D elliptical Gaussian kernels of the corresponding irregular grid.

The Combustion Chamber dataset is from the Visualization Toolkit (Vtk). It consists of $47,025$ grid points. One slice of the resampled regular grids is shown in Figure 5. The resulting elliptical Gaussian kernels and volume rendered image with the splatting algorithm for the Combustion Chamber are shown in Figure 6.

7 Conclusion and Future Work

In this paper, we have presented a method to construct an ensemble of 3D elliptical Gaussian kernels for irregular data. Our method does not resample the irregular grids to generate regular grids. Instead, we create the 3D elliptical kernels centered on the original grid points using a weighted least squares



(a) (b)

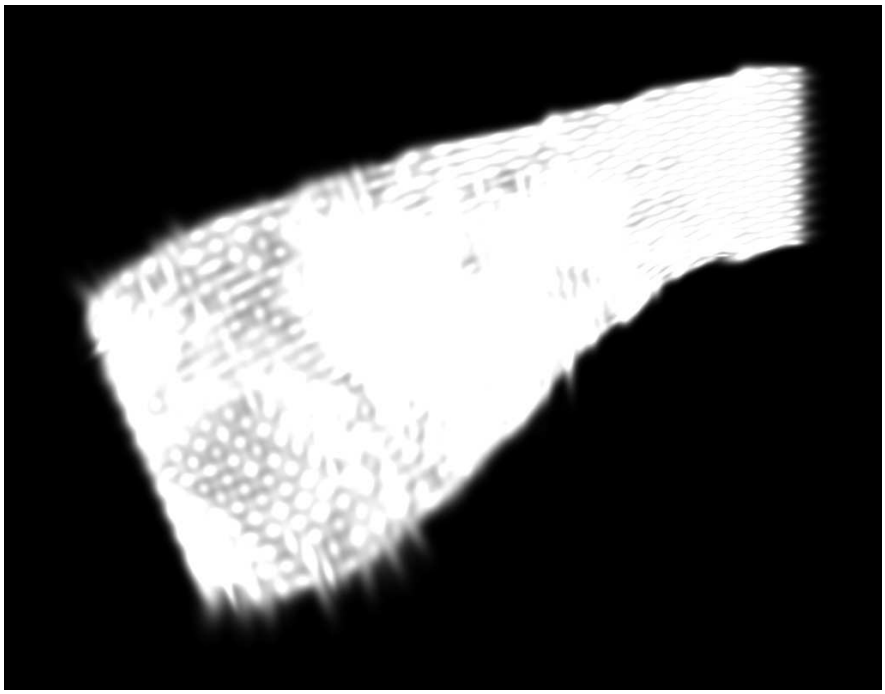
Fig. 4. Blunt Fin dataset: (a)The ensemble of fitted 3D elliptical Gaussian kernels, and (b) a volume rendered image using the elliptical splatting algorithm.

method to fit ellipsoids. We perform PCA on the guide points to provide an initial guess for the minimization process to obtain faster convergence. The resulting kernels are arbitrarily oriented elliptical Gaussians modeled via a matrix representation. The kernels are seamlessly incorporated into our splatting rendering system.

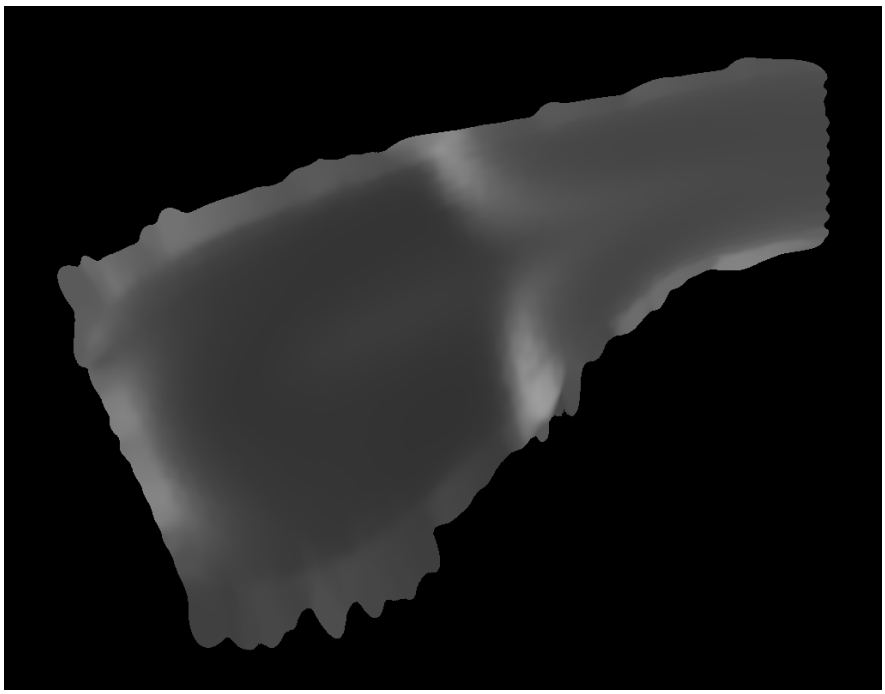
Our method has some limitations. The quality of the resulting kernels is affected by the shape of the Voronoi cells. It would require some amount of resampling in locations that are not covered well by the resulting kernels. In this case, the global optimization methods will do better. Our experiment results show that the boundaries of the irregular grids are not preserved well, appearing somewhat fuzzy. Two methods are possible to be used to solve this problem. One method is to subdivide the boundary tetrahedron. But this would require the resampling of more points and thus more splats would be generated. Another method is to add a layer of "ghost splats" outside the tetrahedra mesh to solve this problem. In future work, we would like to study where to place these ghost splats in order to preserve the boundary well. In our current implementation, the rendering is implemented using software. In subsequent, we would like to exploit the power of GPUs to accelerate the rendering. Here, the main feature of floating point blending will be highly beneficial.

References

1. Fruhauf, T.: Raycasting of Nonregularly Structured Volume Data. Eurographics, C294–C303 (1994)

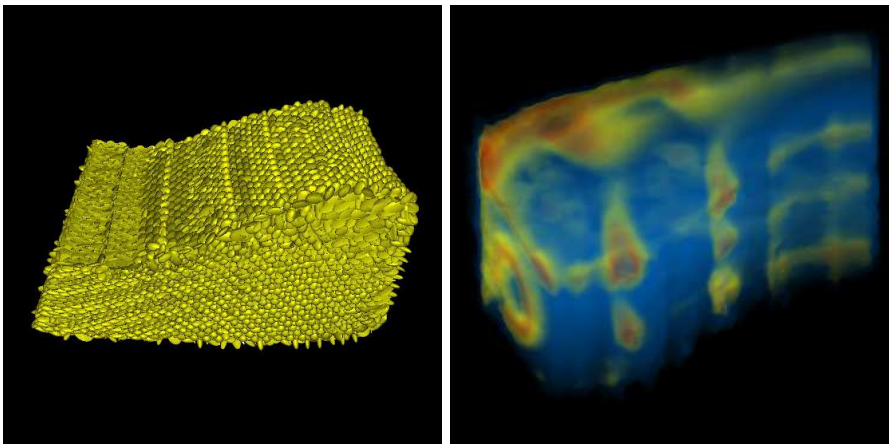


(a)



(b)

Fig. 5. Combustion Chamber dataset: (a) Weight image and (b) density image for one slice of the regular grid samples evaluated using the fitted 3D elliptical Gaussian kernels of the corresponding irregular grid.



(a) (b)

Fig. 6. Combustion Chamber dataset: (a)The ensemble of fitted 3D elliptical Gaussian kernels, and (b) a volume rendered image using the elliptical splatting algorithm.

2. Garrity, M.: Raytracing Irregular Volume. Computer Graphics, Vol. 24, No. 5, 35–40 (1990)
3. Ramamoorthy, S. and Wilhelms, J.: An Analysis of Approaches to Ray-Tracing Curvilinear Grids. Tech. Report UCSC-CRL-92-07, Univ. of California, Santa Cruz (1992)
4. Farias, R. and Silva, T.C.: Out-of-Core of Large, Unstructured Grids. IEEE Computer Graphics and Applications, 21(4), 42–50 (2001)
5. Max, N., Hanrahan, P., and Crawfis, R.: Area and Volume Coherece for Efficient Visualization of 3D Scalar Functions. Computer Graphics, 24(5), 27–33 (1990)
6. Westover, L.: Footprint Evaluation for Volume Rendering Computer Graphics. 24(4), 367-376 (1990)
7. Weiler, M., Kraus, M., Merz, M., and Ertl, T.: Hardware-Based Ray Casting for Tetrahedral Meshes. In Proceedings of IEEE Visualization, 333–340 (2003)
8. Röttger, S., Kraus, M., and Etrl, T.: Hardware-Accelerated Volume and Isosurface Rendering Based On Cell-Projection. In Proceedings of IEEE Visualization, 109–116 (2000)
9. Weiler, M., Kraus, M., and Ertl, T.: Hardware-Based View Independant Cell Projection. In Proceedings of IEEE Symposium on Volume Visualization, 13–22 (2002)
10. Mueller, K., Möller, T. and Crawfis, R.: Splatting without the blur. In Proceedings of IEEE Visualization, 363–371 (1999)
11. Meredith, J., Ma, K.L.: Multiresolution View-Dependent Splat Based Volume Rendering of Large Irregular Data. Proceedings of the IEEE symposium on parallel and large-data visualization and graphics, (2001)
12. Mao, X., Hong, L., and Kaufman, A: Splatting of Curvilinear Volumes. In Proceedings of IEEE Visualization, 61–68 (1995)

13. Mao, X.: Splatting of Non Rectilinear Volumes Through Stochastic Resampling. IEEE Transaction on Visualization and Computer Graphics, 2(2), 156-170 (1996)
14. Jang, J., Shaw, C., Ribarsky W. and Faust N.: View-Dependent Multiresolution Splatting of Non-Uniform Data. Eurographics-IEEE Visualization Symposium, 125–132 (2002)
15. Jang, Y., Weiler, M., Hopf, M., Huang, J., Ebert, D.S., Gaither, K.P., and Ertl, T.: Interactively Visualizing Procedurally Encoded Scalar Fields. Joint Eurographics-IEEE TCVG Symposium on Visualization (2004)
16. Jolliffe, I.T.: Principal Component Analysis. Springer-Verlag, New York, NY (1986)
17. Hart, J.: Computing Distance betwwen Point and Ellipsoid. Graphics Gems IV, Academic Press, Boston, MA, 113–119 (1994)
18. Powell, M.J.D.: An efficient method for finding the minimum of a function of several variables without calculating derivatives. Comp. J. 7, 303–307 (1964)
19. Brent, R.P.: Algorithms for Minimization Without Derivatives. Dover Publications, Mineola, NY (1973)
20. Mueller, K., and Crawfis, R.: Eliminating popping artifacts in sheet buffer-based splatting. In Proceedings of IEEE Visualization, 239–245 (1998)
21. Neophytou, N. and Mueller, K.: Space-time points: Splatting in 4D. Symposium on Volume Visualization and Graphics, 97–106 (2002)
22. CGAL C++ library www.cgal.org