**EVENLY SPACED STREAMLINES(3D)**

**1. ABSTRACT**—This project implements Flow visualization which concerns with vector quantities, i.e., magnitude and direction. Flow visualization can be characterized into two categories, namely texture based flow visualization and geometry based flow visualization. The main challenge in geometric flow visualization is the seeding strategy used to place the streamlines within the data domain. The placement of the streamlines depends on the seed location and unsatisfactory results may be obtained for arbitrary seed locations. The difficulty level increases in 3D because a balance of field coverage, occlusion and visual complexity must be maintained. The position of the seed affects time-dependent data. The number of streamlines is also important as cluttered streamlines makes it difficult to distinguish flow vector and critical features are missed with very few streamlines. So a new seeding strategy using two parameters dsep and dtest are used to control the density of the displayed streamlines. The existing streamlines are used to seed new streamlines where dsep is the distance between the new streamline and the candidate seed point dtest is used to control the closest distance the streamlines are allowed to one another.

**Index Terms**—Keywords should be taken from the taxonomy (http://www.computer.org/mc/keywords/keywords.htm). Keywords should closely reflect the topic and should optimally characterize the paper. Use about four key words or phrases in alphabetical order, separated by commas (there should not be a period at the end of the index terms).

**2.INTRODUCTION**

Scientific Visualization, is branch in computer science which bounds different kinds of data, algorithms, computer graphics. It helps analyze and explore the data to provide insights into the structure and composition of the data and for this reason, it can also be termed as Data Visualization. These visualization techniques are different for different forms of data. Not all data can be represented with one single technique. Generally, data can be categorized as scalar data, vector data and tensor data. Scalar data points are discrete points in data space which usually represent quantities like temperature, density etc. Scalar fields can be represented with techniques such as volume rendering and iso-surfacing. Volume Rendering is used to project 3d objects onto 2d image plane. Iso-surfacing done using marching squares to connect data points which are at equal level. Vector fields show the underlying physical properties of flow structures. Vector fields have both magnitude and direction associated with them and represent movement or flow like winds, velocity etc. It is usually termed as flow visualization. Visualizing vector fields is completely different and challenging when compared to scalar fields since the direction of every point must be represented without cluttering the visual display to the user. Many techniques are available at present to visualize simulation data like, drawing streamlines or glyphs but choosing one among them is not easy and depends on the properties we are trying to visualize. Also, approaches have to be changed depending on the dimension of data. Visualizing 2D data is completely different from visualizing 3D data. Flow field data can be visualized in various categories such as direct visualization technique, texture-based visualization technique, geometric and feature-based techniques. Point-based visualization techniques consider vector data at a point along with its neighborhood to visualize it. Here, no intermediate data processing is done, that is the vector field is directly mapped to the plane. Arrow plots is an example for point based flow visualization technique on glyphs. They are applied to time dependent vector fields such that they adapt to the velocity at a given time. Sometimes, the flow features can be mapped to a data point and scalar field methods can be applied.  Also, in some cases, data must be preprocessed in order to perfectly visualize it. Density is an important factor to be considered in some cases. It is useful in some techniques such as particle tracing which is nothing but streamlines in geometric based flow field visualization.Texture-based techniques are efficient in showing flow characteristics but are more suited for 2D domain. In geometric-based techniques, Streamlines are a used to represent flow of the vector data.  They are nothing but tangents drawn to the vector fields. A starting point is selected and numerical integration is done to join all the other points along the flow to draw a streamline. The starting points are called seed points. Seed points are used to draw streamlines or stream-ribbons at every point in the direction of the flow. There are two ways to define seed points: one way is to allow the user to provide the seed points. The second way is automatically place the seed points. The interactive method is efficient and makes algorithm easy.  They are easy and fast to calculate and show the exact flow structure of the field. Streamlines can be drawn for both 2D as well as 3D data. Data can be time dependent or time independent and they must be differentiated. The type of grid used also affects the visualization. Grids can be uniform, rectilinear, etc and can affect storage of data. In 2D data, calculating streamlines is easy and they are projected on to an image plane directly. There are few challenges in drawing streamlines for 3D data. The seed placement determines the quality of the visualization. If too many seed points are placed in a small region, it looks cluttered and clumsy. On the other hand, placing few streamlines does not show the flow properly. Also, the streamlines must lie within the vector field and should not go outside the field. There are few solutions to these problems. In order to reduce the cluttering, streamlines are drawn at a distance of d-separation and also, bounding box or plane can be placed to restrict the streamlines. Another method is to add streamlines with light intensity to solve the depth issue.

**3. RELATED WORK**

A plethora of papers have been written in the past two decades related to flow visualization. Various techniques are available for flow 2D fields. An image based streamline generation and rendering technique is presented[1]. Here, the seed points are first projected on the 2D image plane and then are put back to the 3D space before integrating the streamlines. A depth map is maintained to know the depth of the points in 3D. This method assures the minimum distance between streamlines. An algorithm for evenly spaced streamlines is proposed for 2D streamlines[2]. In this algorithm, seed points are calculated at a distance of d separation and the streamlines are placed in a queue. This particular algorithm is being extended to 3D streamlines in this project. A lot of research has been done on the techniques of visualization and is drafted for clear understanding of the processes and algorithms[3]. Similarity between streamlines is used to grow streamlines from a set of seed points[4]. Distance between a point on one streamline to the other streamline is calculating by considering a window which is the distance from the point to another point at a certain distance on the streamline. A feature guided placement strategy is proposed for 2D vector data.Another strategy calculates points which are very far from each other to draw streamlines and manhattan or euclidean distance formula is used to find seed points. Dual streamline seeding is proposed by Christoph Petz where streamlines are drawn orthogonally instead of tangential.[5]

**4. METHODOLOGY AND IMPLEMENTATION**

*Proposed Algorithm :*

According to the algorithm a seed point is taken randomly and a streamline is computed. This streamline is placed into a queue and is considered as the initial streamline.

Now we select a candidate seed point at a given distance *dsep* from the current randomly generated streamline and we add it to the queue. But before adding it to the queue we check if the streamline is valid or not and if the streamline is within the bounding box or not. This can be done by computing the minimum and the maximum points in the dataset and then comparing the current seed points with those minimum and maximum values. To check the validity of the streamline we consider if the points on the streamline are also at a distance of *dsep* from the initial streamline. We compute multiple streamlines but place only one valid streamline into the streamline queue. If no valid streamline is found we check the queue and select one entry from the queue as our streamline and then add it. If there are no streamlines left in the queue it means that we have calculated all valid streamlines from the data and the remaining streamlines are not valid(evenly-spaced). For every iteration we consider the next streamline on top of the queue as the current streamline and continue. The above process is repeated until all the valid streamlines are generated.

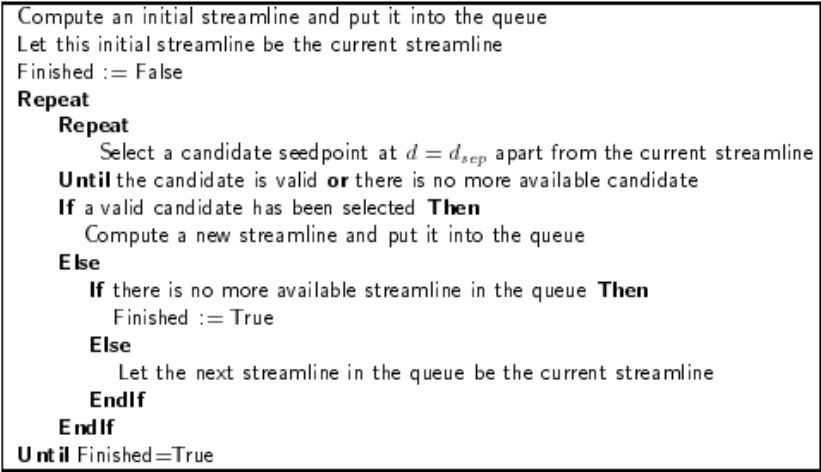
*Seed point calculation*- Initial seed points are calculated using the function mesh grid. It takes the x, y, z co-ordinates of the starting seed point. This seed point is placed in a queue. MATLAB does not have a queue data structure, so a matrix is considered to represent a queue. The initial seed point is added to the matrix. Every time a seed point is placed in the matrix, the boundary conditions are checked that is, the point lies between the minimum and maximum data points. Using this seed point, other points are computed at a distance *dsep* and this initial point is popped out, in this case, removed from the matrix and a streamline is drawn using this point. In MATLAB, streamline function is used to integrate the streamline. Once constructed, set method is used to draw the streamline.

Since storing the entire streamline is a very tedious and computationally expensive process, we store only the initial seed points in the Queue structure and move forward.

*Calculating dsep and controlling its value*- Density is a global feature however it has to be expressed as a local feature in order to control the texture. The control is done during the construction of the streamlines such that every streamline should be separated by a distance of *dsep*. This means that a new point is valid only if it is separated from the previous line by a distance of *dsep*. If the grid is denser the comparisions are lesser. This separating distance can be determined by the used according to how densely the streamlines have to be placed.

*Streamline Integration* – Generally streamline integration us done by Euler integration or Runge-Kutta integration of the fourth order. Here we use the function streamline where we give input the data points of the bounding box, i.e the array of points of the input data and the initial seed point. The streamline is then computed using this data and then we use set method to draw the streamline on the output bounding box. The properties like color width are set so as to display it on the output figure.

Texture – This is controlled by using the value of the *dsep.* For larger distances the streamlines are located farther away to each other and for smaller values they are located close by. This helps in generation of the texture density.



# 5. RESULTS AND CONCLUSION

We can conclude that evenly-spaced streamlines can be generated if we have a control on the separating distance between the streamlines. By varying this distance we can change the density of the streamlines. This method of storing only the seed points is computationally much less expensive when compared to storing all the points of the streamlines in the queue. The seed points can be randomly given to the algorithm to begin from different locations.

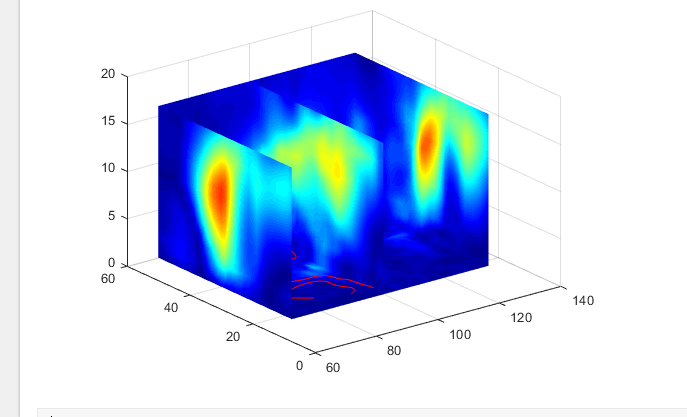


fig 5.1 Generation of a single streamline

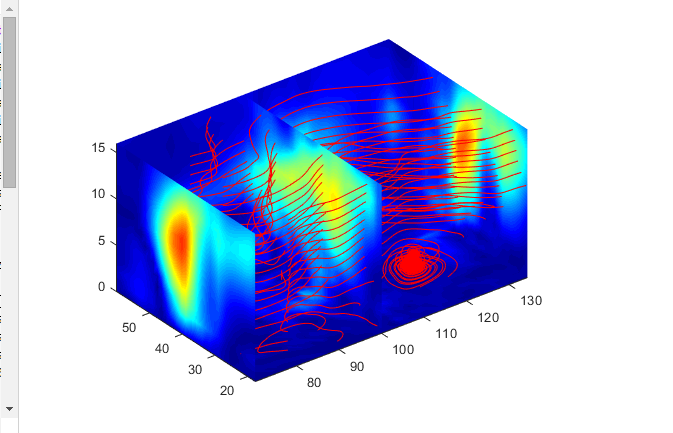


fig 5.2 After generating multiple streamlines

For different values of *dsep*

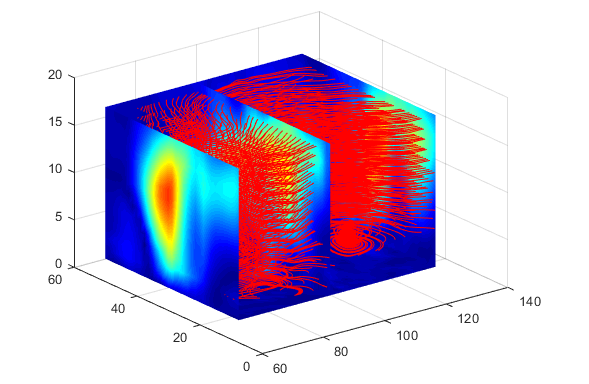
**

fig 5.3 For very small *dsep* = 0.3, cluttered streamlines.

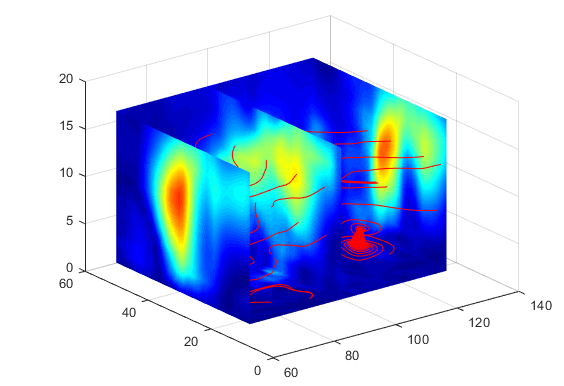


fig 5.4 For larger values of *dsep* = 7, streamlines are very far.

**6. FUTURE WORK**

We have implemented VTK(Visualization Toolkit) using C++, the results were not as we desired. We wish to get similar results in VTK as well.

**7. REFERENCES**

 1.   Liya Li and an-Wei Shen: Image Based Streamline Generation and         Rendering [IEEE Transactions on Visualization and Computer Graphics](http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=2945) in 2007

2.      B.Jobard and W.Lefer, "Creating Evenly Spaced Streamlines of Arbitrary Density,",1997

3.     T.McLoughlin, Over Two Decades of Integration-Bases, Geometric Flow Visualization

4.    Yuan Chen, Student Member, IEEE, Jonathan D.Cohen, and Julian H. Krolik: Similarity-Guided Streamline       Placement with Error Evaluation, 2007.

5. Olufemi Rosanwo, Christoph Petz, Steffen Prohaska,Hans-Christian Heges and Ingrid Hotz Zuse Institute Berlin, Germany: Dual Streamline Seeding [Visualization Symposium, 2009. PacificVis '09. IEEE Pacific](http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=4816119)

Authors-

1.Priscilla Likhita Roy Imandi – Master in Computer Science, University Of Houston. Designed the algorithm and implemented it. Wrote Methodology and Implementation, Results and Conclusion, Future Work in the current paper.

2.Abheesta Reddy Jakka – Master in Computer Science, University Of Houston. Designed the data structure and implemented it. Wrote the abstract, Introduction and Related Work in the current paper.