# Implementation of An Illustrative Visualization Framework for 3D Vector Fields

COSC 6344 – Visualization, Dr. Guoning Chen

Michael Tran

**Abstract**— 3D vector field visualization techniques suffer from the problem of visual clutter, and it is a challenging task to effectively convey both directional and structural information of 3D vector fields. In this paper, we will implement the approach that has been laid out by Cheng-Kai Chen, Shi Yan, Hongfeng Yu, Nelson Max, and Kwan-Liu Ma in the paper "An Illustrative Visualization Framework for 3D Vector Fields". [1]

## 1 Introduction

As computing power grows and scientists are pushing to simulate and study complex physical phenomena, the need for clear 3D vector field visualization has increased rapidly. Given the sheer size and complexity of data, exploration of 3D vector fields using visualization often results in visual clutter which distracts the user from the important details.

The approach that we will use is guided by the paper "An Illustrative Visualization Framework for 3D Vector Fields" by Chen et all. The paper is inspired by the principles of *abstraction* and *emphasis* used by artists and illustrators to efficiently and precisely convey information. The idea behind this is to display abstrations that preserve certain precise qualities of objects while suppressing or omitting unnecessary details. The resultant visualization can direct the user's focus of attention to salient properties and characteristics of the rendering.

My implementation of illustrative visualization is split into three parts:

- Entropy-Based Streamline Seeding – a method to abstract a 3D vector field by a set of streamlines that are representative and cover the important regions of flow.
- Two-Stage Streamline Clustering – a method to effectively and efficiently decompose a large number of streamlines into different groups. Each group will have a coherent flow structure, which will facilitate abstraction of complex structures.
- Streamtape Generation – using a streamribbon-like visual metaphor, streamtapes concisely depict characteristics of vicinity flow along streamlines.

Our framework will be demonstrated on three different 3D vector fields. The resultant visualizations will show informative depications of the vector fields and highlight important details.

## 2 Related Work

Most of the techniques used in this implementation were taken directly from the paper that this paper is based on. The paper titled "An Illustrative Visualization Framework for 3D Vector Fields" by Chen *et al* details how streamlines are seeded, clustered, and visualized. Some parts of my implementation differ from the original outline and can be credited to Dr. Guoning Chen of University of Houston.

## 3 Methodology and Implementation

The aim of this illustrative visualization framework is to derive abstractions and capture essential flow features while suppressing unimportant details to minimize visual clutter of 3D vector fields.

### 3.1 Overview

Our approach consists of three steps: entropy-based streamline seeding, two-stage streamline clustering, and streamtape generation. Note that there are slight variations from the proposed ideas from the original paper.

### 3.2 Entropy-based Streamline Seeding

For any given flow field, we need to place seeds for streamline generation. The goal of the seeding strategy is to capture the essence of the flow data. We use the concept of Shannon's entropy in information theory to calculate an entropy scalar field. This field measures the information content in each local region of the vector field. Shannon's entropy is defined as:

$$H(x) = -\sum_{x_i \in X} p(x_i) \log_2 p(x_i)$$

Where $H$ is the entropy of a discrete random variable $X$ with a sequence of possible outcome values $\{x_1, \dots, x_n\}$ and $p$ is the probability mass function of $X$. To approximate $p$, we create a histogram-like structure where we bin each selected vector. Instead of vector quantization proposed in the original paper, we use the directions of each vector and bin them into a number of bins $x_i$. The direction of a 3D vector can be described by angles theta ($\theta$) and phi ($\phi$) in Figure 1.

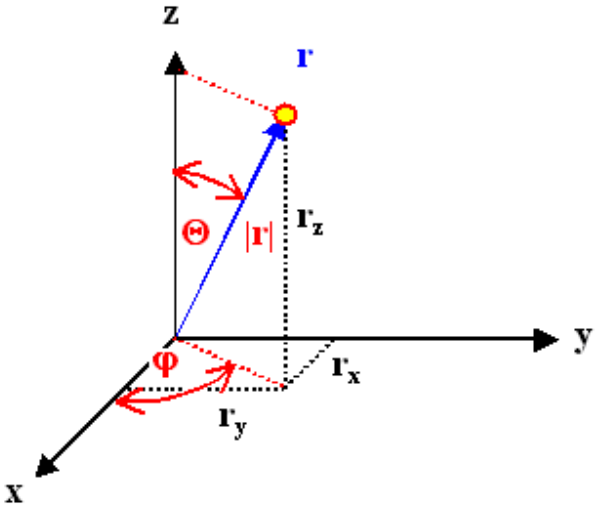


**Figure 1:** Angles of a 3D vector. [2]

Where theta and phi are defined as:

$$\theta = \arccos(z/r)$$
$$\phi = \arctan(y/x)$$

For our implementation, we create 36 different bins that the vector direction can potentially fall in. We then uniformly sample 262,144 points on the vector field and then randomly sample 100,000 points and bin. This ensures that a sufficient amount of points on the vector field is sampled.

The probability function $p$ is defined as follows:

$$p(x_i) = \frac{C(x_i)}{\sum_{i=1}^{n} C(x_i)}$$

where $C(x_i)$ is the number of vectors in the bin $x_i$.

The entropy field is a scalar field which is constructed by calculating the entropy around the local neighborhood of each voxel. (In our case a voxel is defined in a $64\ x\ 64\ x\ 64$ grid) The neighborhood we choose is a $3\ x\ 3\ x\ 3$ cube around each voxel.

We then place a sufficient number of seeds in the field based on the probability associated with each voxel so as to cover as many important areas as we can. To do this, we assign a probability based on the entropy to each voxel giving higher probability to voxels with higher entropy. We then shuffle and randomly sample 1000 voxels which will be used as our initial seeds. Finally, we use the fourth order Runge-Kutta integration to trace the streamlines from the seeds.

Figure 2 shows an example of a volume rendering of the entropy field and the randomly selected streamlines. We can see that areas with high entropy (red) have more streamlines placed in it which ensures that the important flow features are well captured.

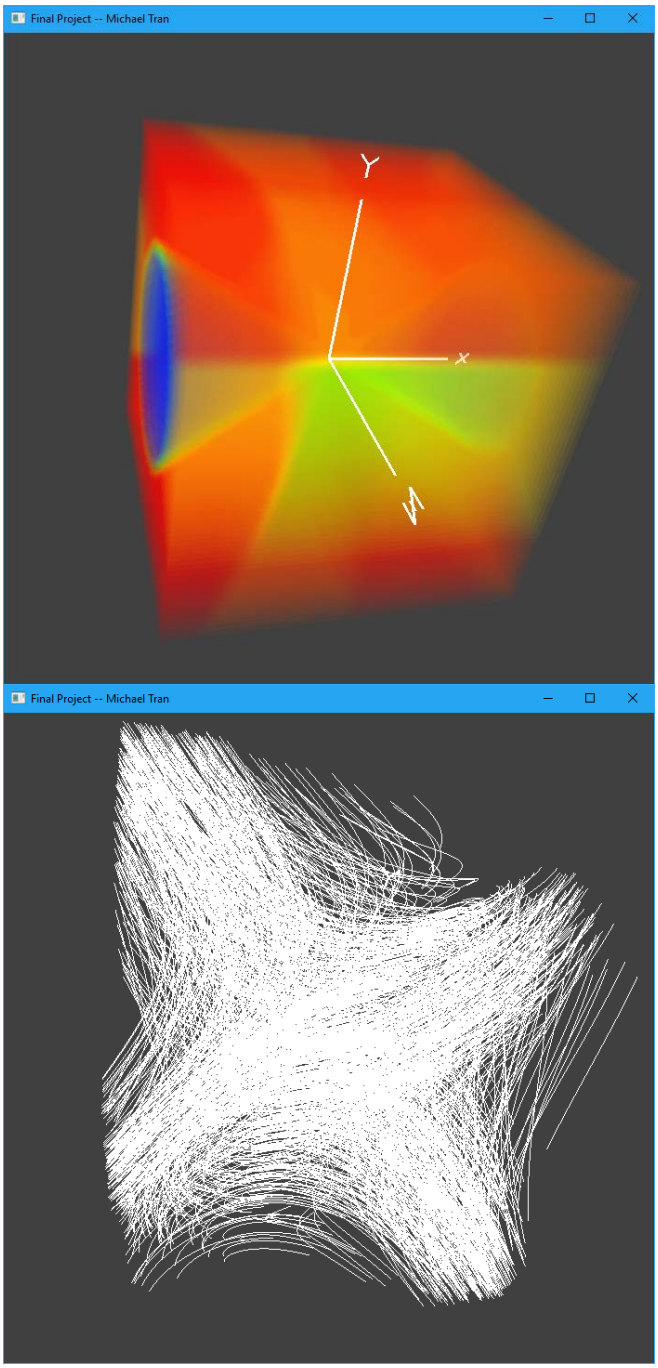


**Figure 2:** Entropy field (top), randomly selected streamlines (bottom)

### 3.3 Two-Stage Streamline Clustering

After obtaining a number of streamlines based on our entropy-based seeding strategy, we need to consider how to effectively draw them to capture the essence of the underlying field.

To do this, we simplify the streamlines through clustering using the k-means algorithm. The result of the k-means algorithm partitions the streamlines into $k$ sets. Our implementation is split into two stages.

For the first stage, we want to obtain an overview of the flow patterns. To do this we partition the streamlines into $p$ bundles. For the k-means algorithm in the first stage, we set the clustering property based on the vector *spatial properties*, which is composed of the start point, middle point, and end point. Thus, we obtain an initial

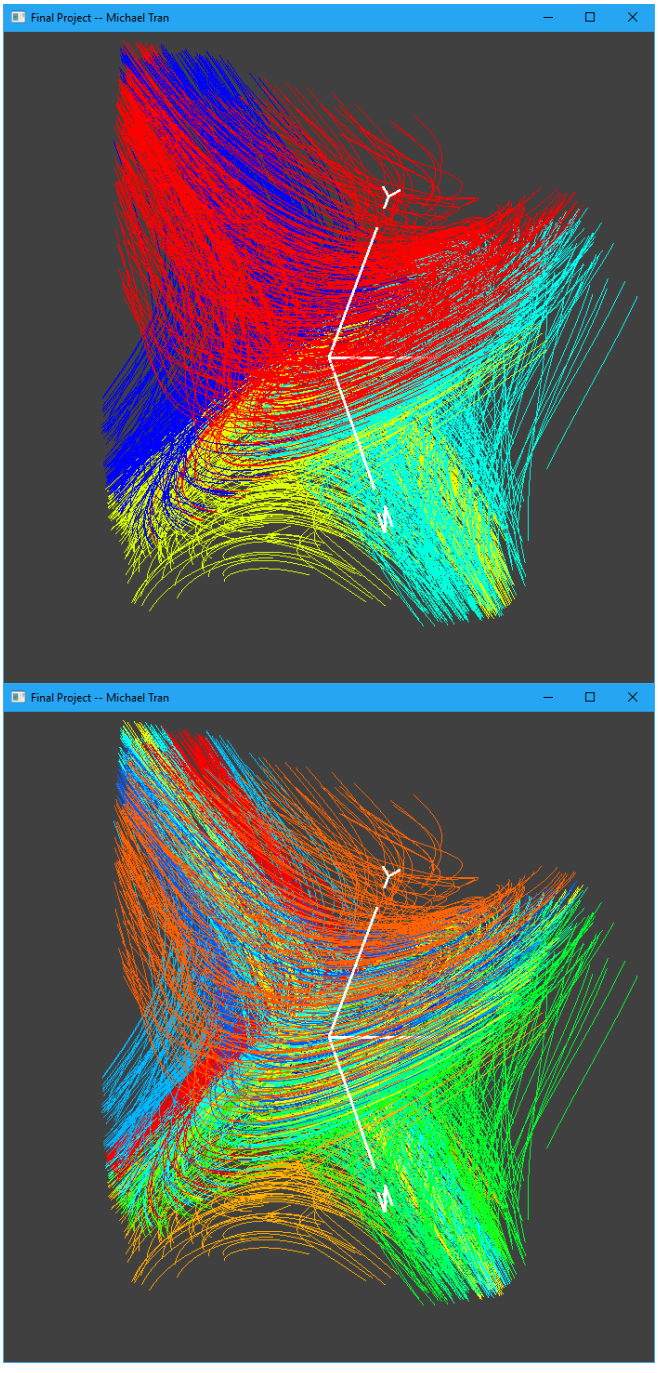


**Figure 3:** Stage 1 Clustering (top), Stage 2 Clustering (bottom)

partition containing $p$ bundles, which each bundle has a roughly coherent structure in space.

In the second stage, for each bundle generated in the first stage, we further partition its streamlines into $q$ sub-bundles. For the k-means algorithm in the second stage, we set the clustering property based on the vector *shape properties*, which is composed of the linear and angular entropy values of a streamline.

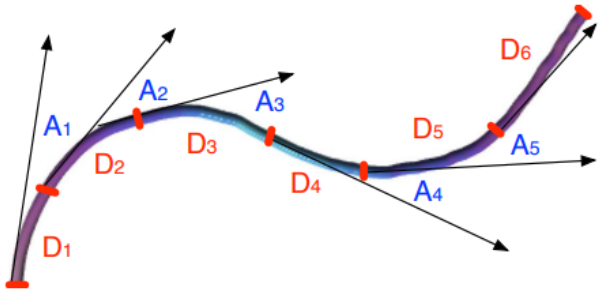


**Figure 4:** Parameters used to compute the entropy criteria. The streamline is depicted in purple and the distance ($D_j$) and angular ($A_j$) parameters are shown. [3]

The linear entropy $E_L$ quantifies the amount of information in the flow magnitude along the streamline. It is defined as follows:

$$E_L = -\frac{1}{\log_2(m+1)} \sum_{j=0}^{m} \frac{D_j}{L_S} \log_2 \frac{D_j}{L_S}$$

where $m$ is the number of streamline segments, $D_j$ is the length of the $j$-th segement and $L_S$ is the total length of the streamline.

The angular entropy $E_A$ quantifies the amount of angular variation along a streamline. It is defined as follows:

$$E_A = -\frac{1}{\log_2(m)} \sum_{j=0}^{m-1} \frac{A_j}{L_A} \log_2 \frac{A_j}{L_A}$$

where $m$ is the number of streamline segments, $A_j$ is the absolute value of the angle at the $j$-th streamline joint and $L_A$ is the total angular variation along the streamline.

The combination of linear and angular entropy values for a given streamline can be used to discriminate stream line properties and detect flow phenomina.

Once we have our clusters from stage 2 clustering, we bundle them and pick a representative streamline from each cluster. To do this, we select the streamline that is closest to each given cluster centroid. The total number of streamlines chosen to represent the 3D vector field equals to $p \times q$. At run time the user can specify the values of $p$ and $q$. The default values of our implementation are $p = q = 4$, which creates 16 representative streamlines.

Figure 3 shows an example of stage 1 and 2 clustering. We can see how the streamlines are clustered after each stage. Our algorithm detected a total of 16 different clusters for the given field and $p$ and $q$ values. Figure 5 shows the result of choosing representative streamlines from each cluster.

### 3.4 Streamtape Generation

To render the representative streamlines, we use a streamribbon-like visual metaphor, the streamtape. This visual metaphor leverages the advantages of both streamlines and integral surfaces. It generates both meaningful and visually pleasing visualization results.

For each streamline chosen to represent its cluster, we

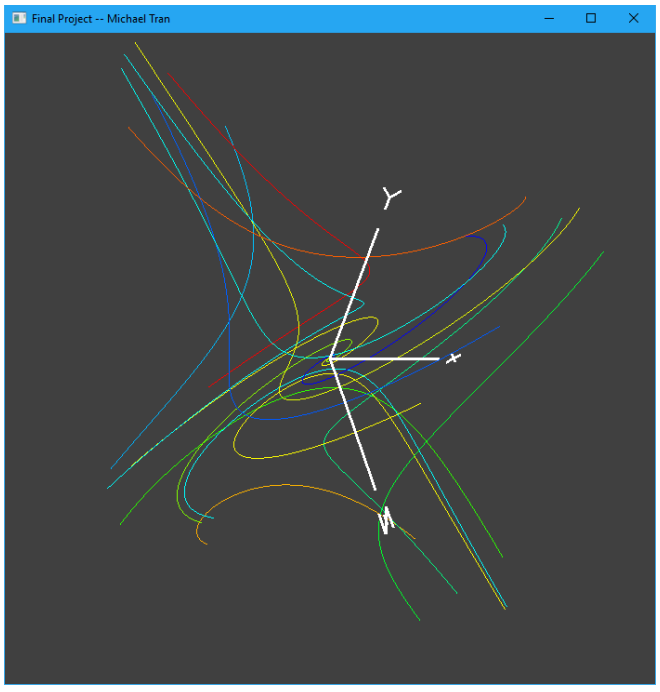


**Figure 5:** Representative streamlines chosen after two-stage clustering.

calculate the binormal vector $B$ at each point of the streamline $r(s)$:

$$B = T \times N$$
$$T = dr/ds$$
$$N = dT/ds$$

where $T$ and $N$ are the tangent and normal vectors of the local field at that point, respectively. The binormal vector $B$ determines the orientation of each streamtape segment.

To determine the width of each streamtape segment, we calculate the torsion $\tau$ of the local field at each point on the streamline. Torsion is a measurement of how sharply the local field twists and is given by:

$$\tau = \frac{\det(r', r'', r''')}{\left|\left|r' \times r''\right|\right|^2}$$

where $r', r'', r'''$ are the first, second, and third derivatives of r(s), respectively.

Given torsion, we then calculate streamtape width $\omega$ to determine the width of each segment of the streamtape:

$$\omega = 1.0 - \tau$$

To finish off our implementation of streamtapes, we also add arrows oriented by the binormal vector to help convey the direction of the vector field.

Figure 6 shows an example of streamtapes with constant width and without constant width. We can see how much clearer the visualization of the 3D vector field is when compared to Figure 5 and previous examples.

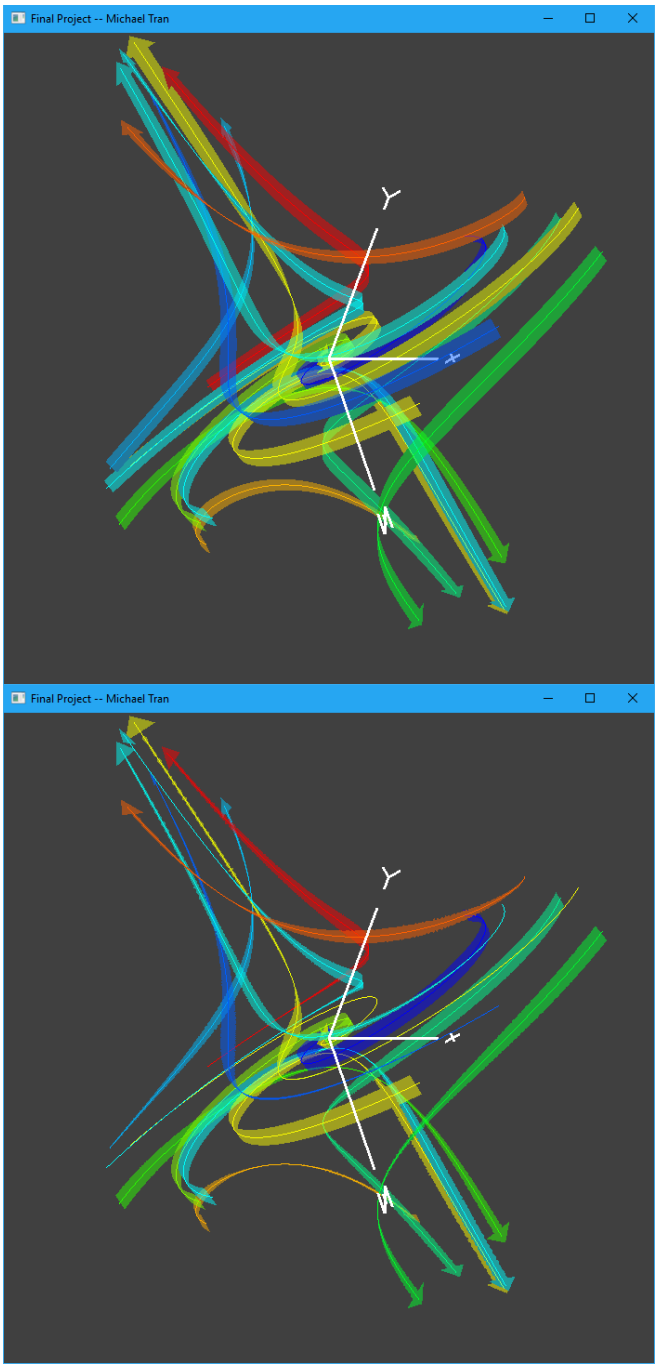


**Figure 6:** Streamtapes with constant width (top), Streamtapes with calculated width (bottom)

## 4 Results and Discussions

In our framework, entropy field calculation, streamline generation, k-means clustering, and streamtape generation are implemented on the CPU on a single-thread. The program performs very quickly with the entire visualization rendered in under a second on an Intel® Core™ i7-7700K CPU @ 4.20 GHz machine with 16GB of memory and a NVIDIA GeForce GTX 1080 graphics card with 8GB of video memory. Our framework can quickly compute these calculations on all three fields given with comparable times.

The part that caused our framework to take much longer times was based on how many random streamlines we decided to choose. The processing time is proportional to number of streamlines chosen. We started with generating over 100,000 random streamlines which took over 30 seconds to compute, in the end we chose

1000 random streamlines. Two of the main reasons were because it performed much faster, and that it yielded similar results, therefore it was unnessesary to compute so many streamlines. However, it should be noted that due to the randomness of our framework, choosing less streamlines also have its disadvantages. With less randomly chosen streamlines, it is possible (albiet very unlikely) that we sample only streamlines in low entropy portions of the vector field. This will yield some unsuccessful visualizations of the 3D vector field. Therefore, if time is not an issue, or we make the framework more parallel with CUDA or OpenMP, the amount of randomly sampled streamlines should be increased to lower the chances of that occurring.

For the quality of visualization of our framework, we can see from Figure 6 that our visualization is very clean, especially when compared to our previous figures of the vector fields. We can see that the vector field swirls into the center and also pushes away from the center if not on its swirl axis.

We also have to be very careful when choosing a $p$ and $q$ value as that will increase the amount of streamtapes generated. Choosing too high of a value could cause more visual clutter to start occurring and thus ruining the advantages of this framework. Likewise, choosing too low of a value could cause not enough streamtapes to be generated and not cover enough of the vector field to yield meaningful results.

## 5 Conclusions and Future Work

In conclusion, we implemented our own version of 'An Illustrative Visualization Framework for 3D Vector Fields' by Cheng-Kai et all. Our framework conbines an entropy-based seeding strategy, a two-stage streamline clustering approach, and a streamtape-like visual metaphor to generate concise and informative visualizations for complex flow structures. Starting from a (formulaized) 3D vector field, our framework generates an initial set of representative streamlines based on the entropy in the field. We then employ a two-stage k-means clustering algorithm to cluster these streamlines and then bundle them into representative streamlines that capture local flow characteristics and features. This framework can provide people with a powerful visualization tool for displaying complex flow fields and allows for efficient interpretation of the data.

For future works, there are some improvements that can be made to our framework. As alluded to in the previous section, we can parallelize the framework on CUDA and OpenMP to a speed the computations up dramatically. The paper also suggests the use of Elbow Criterion to suggest the optimal values of $p$ and $q$ instead of manually choosing the value. This would be a good quality of life improvement to our framework. The original paper also suggests the use of Phong Shading to help improve visualization. If that were to be implemented, that would improve depth perception in our visualizations.

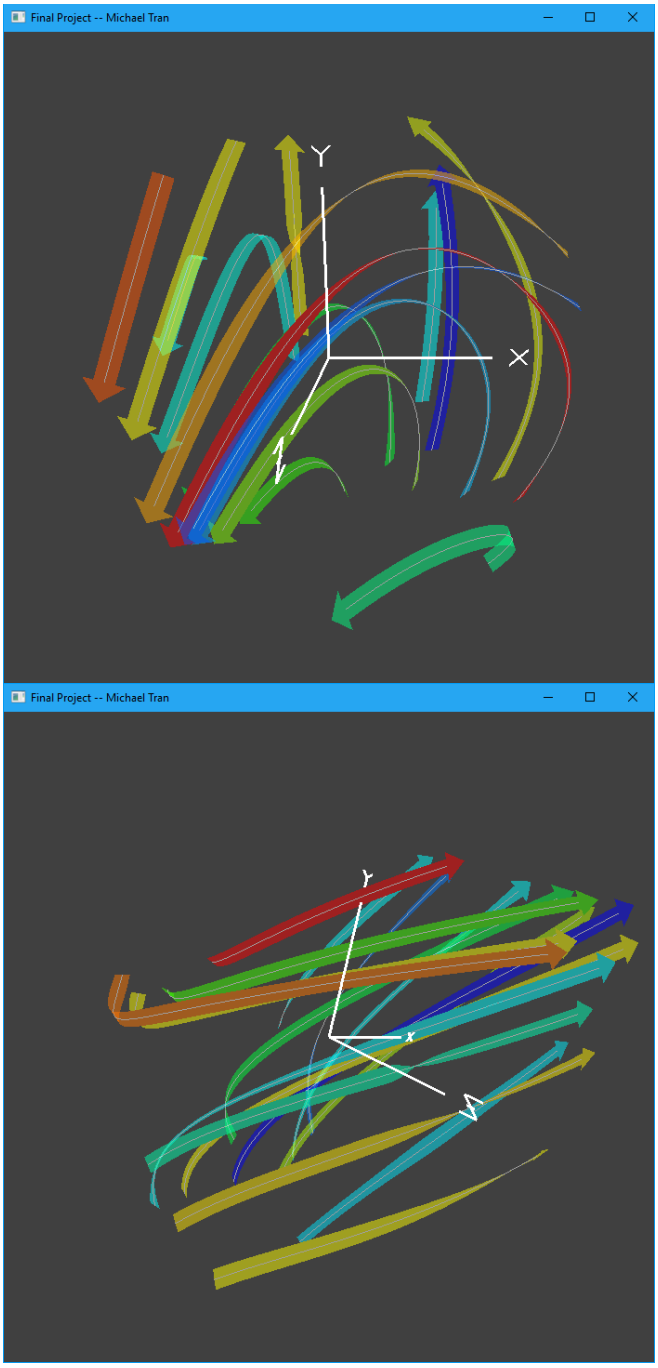


**Figure 7:** Results using 'Field 1' (top), Results using 'Field 2' (bottom)

## References

[1] Cheng-Kai Chen, Shi Yan, Hongfeng Yu, Nelson Max, Kwan-Liu Ma, 'An Illustrative Visualization Framework for 3D Vector Fields'

[2] Laurent Crivello, 'Angles of a known 3D vector', 2017. [Online] Available: https://math.stackexchange.com/questions/2247039/angles-of-a-known-3d-vector. [Accessed: 03 – Dec- 2018]

[3] Stephane Marchesin, Cheng-Kai Chen, Chris Ho, Kwan-Liu Ma, *'View-Dependent Streamlines for 3D Vector Fields'*