CIS 572: Streamline via Machine Learning

Algorithmic Development

Steven Walton

June 14, 2019

Abstract

Generating streamlines are an integral part of flow visualization. They enable scientists and engineers to accurately determine how fluids are acting and allow decisions to be made based off of these results. Generating streamlines that show interesting regions, are uniformly distributed, aesthetically pleasing, and do not excessively computationally expensive remains an open problem in scientific visualization. This work attempts to bring light to how machine learning can be used to tackle this problem. A UNet is used to perform image segmentation and ultimately results show that this is not a promising avenue of study. Other models may be more fruitful.

1 Project Description

We attempt to prototype streamline generation by use of UNets. We choose to use a UNet because it is a simple way to encode information. It is not expected to generate an efficient model as this task is extremely difficult. They hypothesis was that we might be able to train a network to identify key segments and segment out key regions. Ultimately this failed and we learned that image segmentation comes nowhere close to providing a reasonable model for streamlines.

2 Background

Scientists use flow visualization techniques to understand the basics of how fluids move in various conditions. This can range from the water flowing in the oceans, global wind data, to how fuel moves inside a rocket engine. Understanding fluid flow has been a great challenge for scientists for a long time.

When solving these systems scientists use vector fields placed at each location within the geometry, denoting the direction and magnitude of the fluid flow. By calculating these vectors a scientist can track how a particle moves about its environment. Computation becomes increasingly difficult when these vector fields start changing as a function of time, these simulations are more representative of real world phenomena.

The difficulty with vector field representations is that it would be impossible to visualize. If we have to draw a vector, with length representing magnitude, at every single point in a geometry then they would completely overlap and you'd just see a completely black image. This can be partially solved by using a sparse representation of vectors. Figure 1 shows an example of this type of sparse representation. In this figure we can see that there are two sinks, areas where flow goes into (such

as a kitchen sink), and many areas where fluid flows at different velocities and directions. There is a lot of information to process in this image and it can be difficult to tell what is happening.

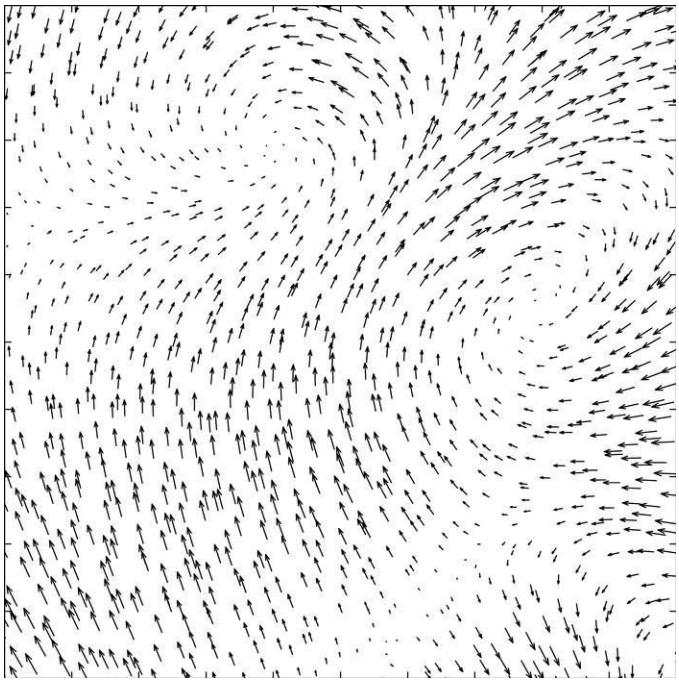


Figure 1: Sample vector field

2.1 Streamlines

A streamline is defined as a line that shows how fluid flows. These are used in scientific visualization to help scientists visualize what is happening within fluids. Generally these are used because the vector field, which is what is used in the simulation data, is too information dense and difficult to make sense of. A good streamline is one that has sparse representation of the fluid but also shows all key features. Figure 2 shows an example of streamlines in a teapot, representing the heat. We can see that while this representation is easier to understand than those from the vector field, it is still cumbersome.

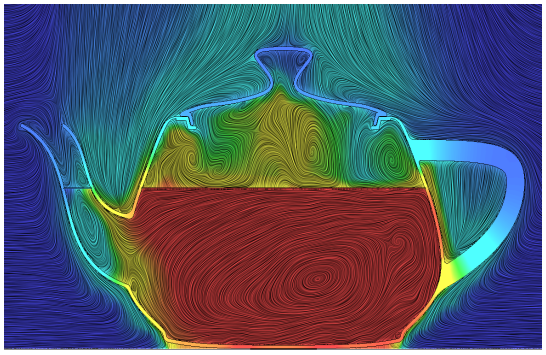


Figure 2: Streamlines Depicting Heat in a Tea Pot

Generating streamlines that are both sparse and representative is an unsolved problem in scientific visualization because it is hard to define what a good representation is. Key features are also

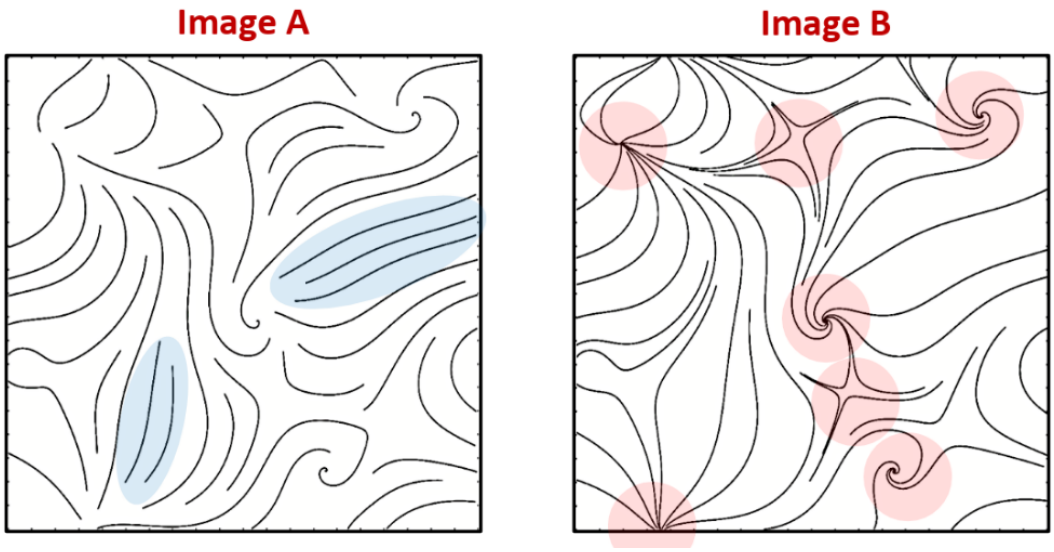


Figure 3: Different Streamline Representations of the Same Fluid Flow with 50 Streamlines[7]

difficult to define. Features that are important are sources, sinks, and flow direction. What can be difficult is the interaction between these features. Straight flow lines bend around sources and sinks even if they do not enter them. This creates a lot of open questions. Such as: "How different do streamlines need to be?" and "At what point do we start drawing and terminate a line?". Verma eta al.[8] were the first to define the desired characteristics of streamlines, naming the three criteria as: Coverage; Uniformity; and Continuity. Coverage refers to ensuring that streamlines convey key features, or regions of interest. Uniformity is defined such that streamlines should be uniformly distributed over a field, meaning that there shouldn't be large empty spaces in some areas and highly dense areas in others. Continuity is defined as an aesthetic interest, mainly in that longer streamlines are preferred. Figure 3 shows how two different streamline representations can convey different information. The red areas in Image B show different sources and sinks that aren't as apparent or sometimes non-existent. In the blue regions of Image A we can see some repetitive lines, ones that aren't in Image B.

2.2 Techniques to Generate Streamlines

Sane [7] collected a large number of modern techniques used for streamline selection for his area exam. Within this work he discusses the use of seed placement and streamline selections. That is by placing a point within a vector field, following its movement and selecting proper streamlines to draw based on these seed placements. In this paper he lists the common practices for generating streamlines. The three major categories are: density based techniques, feature based, and similarity based techniques. Density based focuses on creating evenly-spaced streamlines within the image space. Feature based techniques are formed from vector fields and focus on identifying key features, such as: centers, sources, sinks, repelling-spirals, attracting-spirals, and saddles. See Figure 4.

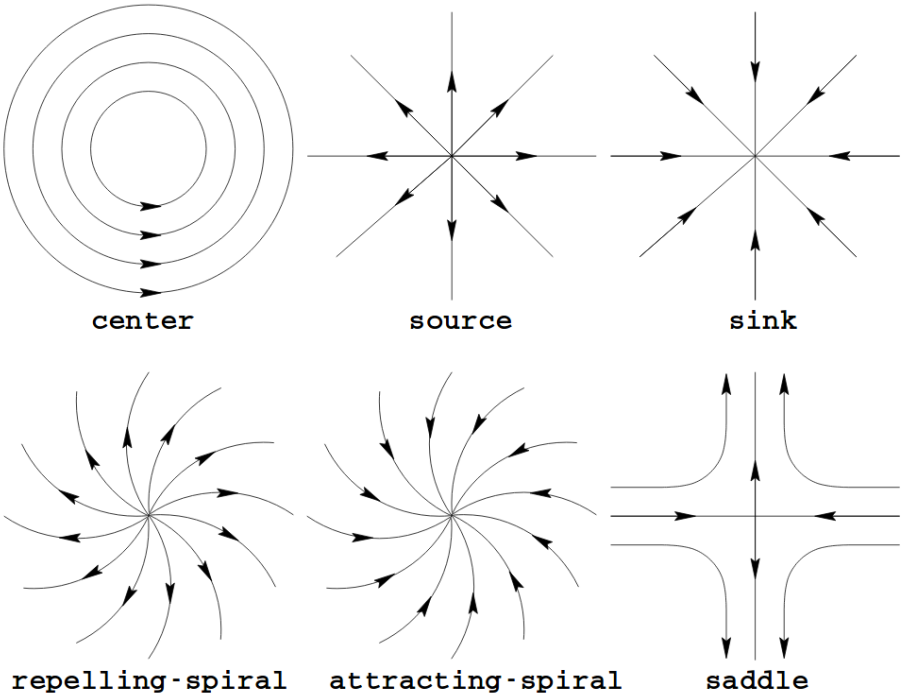


Figure 4: Types of Features [8]

Similarity based techniques focus on reducing the number of streamlines to reduce the similarity between different lines. Figure 5 shows three different streamlines that have different types of similarities. It is clear that in (a) that these lines are redundant because p_i is close to q_i. Conversely we can see that while (b) and (c) look similar they go in different directions. For similarity based techniques we not only care about the shape of the streamlines, but also the orientation similarity.

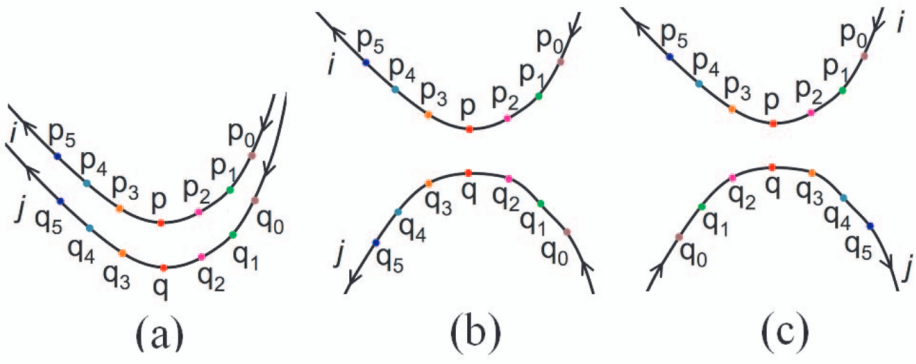


Figure 5: Showing Similarity in (a) and differences in (b) and (c) [1]

In the similarity based techniques approach we find a subsection of machine learning that includes work with SVM's [4] and DNN Feature Descriptions [3]. Li's [4] work focuses on uses binary support vector machines to determine which streamlines are similar and removes ones that are considered too similar. This method work does not account for what humans consider to be important. Han et al.'s [3] work uses a CNN across 3D image data and then uses an autoencoder to learn what the minimum number of streamlines are needed to represent a cluster of streamlines. They use the

sum of the Euclidean distance to all other points in a cluster and identify its minimum. While this technique was successful it takes a significant amount of time to train and does not generalize to arbitrary streamlines. Figure 6 shows the machine learning model used by Han et al. The voxel information is the 3D image data that is being used as an input into the 3D convolutional layer. The inability to create a generalized model is the major drawback of this technique.

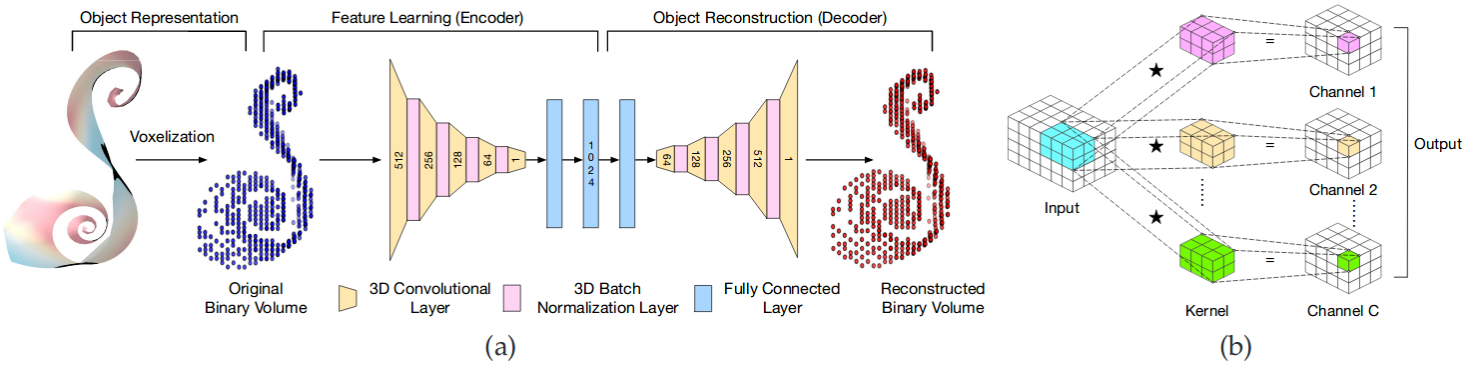


Figure 6: FlowNet Model

3 Model

In this work we use a UNet and attempt to perform image segmentation a set of representative streamlines for two common flow types: ABC and Radial. Figure 7 shows an example of a UNet model. A UNet was chosen because it has an encoding and decoding side. One the input it down samples an image and on the output side it upsamples, training on different input and output images, commonly called masks.

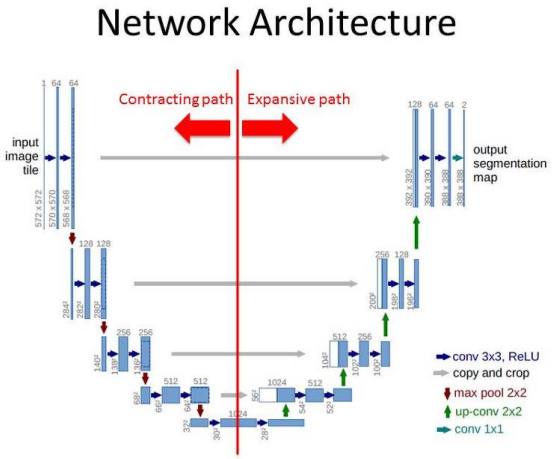


Figure 7: Unet model [6]

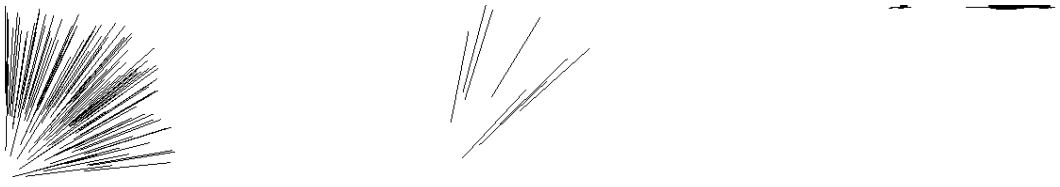
In this work we generated dense and “sparse” streamlines, using the sparse as the image mask.

3.1 Creating Streamlines

Included in the code package is a C++ file called "advect.cpp" that was written to generate streamline files. At the top of this code are options to change how and what streamlines are generated and where the files will be written to. This file contains functions to generate ABC, Radial, and Gyre flow fields. Generally we used 100 lines for the dense representation and 10 for the sparse. Other sizes were tried but these sizes led to fast computation and did not change the results of the network.
The C++ file generates files with the extension ".lines". These files can be used to visualize our streamlines in a program such as Visit [2] A python script, called "visit_script.py", was used to generate all images from a specified directory and output them in another directory. Finally a script, called "convertToPIL.py", was used to convert these images into Python's Imaging Library (PIL) format, which consists of 3-Channels and can be easily read by PyTorch. This convert file was used because it simplified the process and greatly reduced the size of the data, as the dense streamline files can easily take a few hundred gigs of storage space. These files also allowed us to modify images fairly easily.

3.2 Training and Evaluation

For the UNet model we used Pytorch-UNet [5] as a base and modified the training and prediction algorithms. Changes were made to the network to both update for the newer versions of PyTorch, changes to the network were made to allow for use of our input images and changing kernel sizes, as well as options were added to test different different hyperparameters, and of course debugging the original author's work. This lead to a much better understanding of PyTorch than the authors of this work previously had.
Initially the network was trained on large images, size 1024x1024. It was found that this was taking substantial time, was not training well, and did not fit on the GPU. Images were then tried at 512x512 and finally 184x184. The final size was found to fit on the GPU, still contained images where the operators could determine streamlines, and new test cases could be generated quickly.
The work started by focusing on Radial images, as these were the simplest. Initial test sizes included about 5000 input images and corresponding masks. A validation size of 20% was constantly used throughout the experimentation. An epoch size of 1000 was set to run over night and took many hours to complete. Ultimately this did not prove fruitful and unfortunately the loss did not vary significantly between epochs. Figure 8 shows a sample image from the input set 8a, a sample mask 8b, and the resultant output from the test 8c.



(a) Radial Input (b) Radial Mask (c) Output from Network

Figure 8: Input and mask of Radial Streamlines

As this did not go well an attempt was made using ABC inputs and masks. Similarly this did not result in a good output. Figure 9 shows a representative out output of the absolute value of the loss vs the number of epochs the network was trained on. All test cases showed similar results.

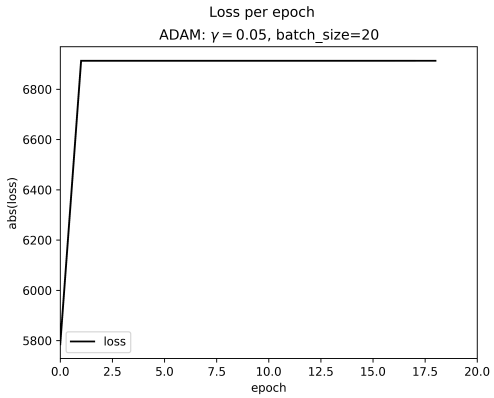


Figure 9: Loss per epoch using ADAM

After ABC and Radial failed an experiment was done just to see if have a a combination of the two would produce different results. 500 images from each were randomly selected and used to train, for a total of a thousand images. Unfortunately this did not result in a change in the loss. We tried SGD as well as ADAM and batched up a process to test vary learning rates, momentum, and batch sizes. The number of epochs was reduced to 20 as it was judged that if learning did not start by that point it was unlikely to learn in longer iterations. We also tried to change the depth of the network but this did not prove to be fruitful. Adding an extra layer did not change the loss in a significant way nor did it show that there was any improvement between epochs. It was decided to not test the Gyre images as these were more complicated and if the UNet could not learn a simple Radial streamline it was not going to learn more complex features.

4 Conclusion

Unfortunately this work did not prove fruitful. While it was not expected to show groundbreaking work, we did not expect the results to be as terrible as they turned out to be. From these results we can conclude that a UNet is not the appropriate model to use for generating anything close to streamline images nor for identifying regions of interest. We believe that a substantially more complicated network will need to be used to make progress.

4.1 Future Work

Being an open problem and that machine learning in scientific visualization is a relatively unexplored work there is more than ample reason to continue to pursue this problem. While replicating the model from Han et al was outside the scope of this class, it seems like a reasonable next step. Verifying and testing their model would be the best stepping stone to work from. This would be good work to attempt over the summer. It would be interesting to test variational autoencoders and test against the results of FlowNet. This is suspected to be a significant undertaking and that tuning the hyper-parameters will take significant time. Work also needs to be done to generalize a machine learning model to generate streamlines. This is the major downfalls of the works referenced in the Background section. For a machine learning technique to truly be considered successful it needs to be able to generalize its solution. While the referenced work was insightful and shows promise, there is much work to be done before we see machine learning used in the generation of streamlines.

While Han et al. and Li et al.'s work focused on similarity based techniques there are, to the author's knowledge, no research done in exploring machine learning through density or feature based streamline generation. It also appears that work has not been done to directly take in vector field data and generate streamlines from this. Since one can think of streamlines as an encoding of vector fields it may be interesting to explore the use of various types of encoders, in combination with other networks, to generate streamlines. Specifically one can think about features being the key features that need to be encoded, so it may be possible to be extract features from vector field data. Potentially some form of CNN may be able to extract these features as well. We believe that the promising direction for generating streamlines from vector field data would be using a feature based technique, where a model could identify the key features previously listed. If a model only showed these features this would be considered major progress and show promise for this type of work.

References

[1] Yuan Chen, Jonathan D. Cohen, and Julian H. Krolik. Similarity-guided streamline placement with error evaluation. *Visualization and Computer Graphics, IEEE Transactions on*, 13:1448–1455, 12 2007.

[2] Hank Childs, Eric Brugger, Brad Whitlock, Jeremy Meredith, Sean Ahern, David Pugmire, Kathleen Biagas, Mark Miller, Cyrus Harrison, Gunther H. Weber, Hari Krishnan, Thomas Fogal, Allen Sanderson, Christoph Garth, E. Wes Bethel, David Camp, Oliver Rübel, Marc Durant, Jean M. Favre, and Paul Navrátil. VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*, pages 357–372. Oct 2012.

[3] J. Han, J. Tao, and C. Wang. Flownet: A deep learning framework for clustering and selection of streamlines and stream surfaces. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2018.

[4] Yifei Li, Chaoli Wang, and Ching-Kuang Shene. Extracting flow features via supervised streamline segmentation. *Computers and Graphics*, 52:79 – 92, 2015.

[5] Milesial. Pytorch-unet. `https://github.com/milesial/Pytorch-UNet`, 2019.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[7] Sudhanshu Sane. Strategies for seed placement and streamline selection. Available at `http://www.cs.uoregon.edu/Reports/AREA-201904-Sane.pdf` (2019/06/14), 2019. Area Exam.

[8] Vivek Verma, David Kao, and Alex Pang. A flow-guided streamline seeding strategy. In *Proceedings of the Conference on Visualization '00*, VIS '00, pages 163–170, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.