MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Milec Oldřich** Personal ID number: **466801**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Graphics and Interaction**

Study program: **Open Informatics**

Specialisation: **Computer Graphics**

II. Master’s thesis details

Master’s thesis title in English:

Visualization of electromagnetic field around a black hole

Master’s thesis title in Czech:

Vizualizace elektromagnetického pole kolem černé díry

Guidelines:

Analyze the problematic of 3D vector field visualization. Based on the analysis, design and implement application capable of loading and visualizing given 3D vector fields describing magnetic and electric fields around a black hole. The application should allow to see direction and the size of the electric and magnetic forces. Further, the application should allow easy and intuitive interaction with the data (rotation, zoom, filtering, etc.). The goal of the visualization is to allow analyze the properties of electromagnetic field near null points where the field has no effect. Test the application on three given datasets from simulation of three different black holes.

Bibliography / sources:

[1] Tamara Munzner. Visualization Analysis and Design. A K Peters Visualization Series, CRC Press, 2014.
[2] Alexandru C. Telea. Data Visualization: Principles and Practice (2nd edition). CRC Press, 2014.
[3] Oliver Mattausch, Thomas Theußl, Helwig Hauser, and Eduard Gröller. Strategies for interactive exploration of 3D flow using evenly-spaced illuminated streamlines. In Proceedings of the 19th Spring Conference on Computer Graphics (SCCG '03). ACM, New York, NY, USA, 213–222, 2003.

Name and workplace of master’s thesis supervisor:

Ing. Ladislav Čmolík, Ph.D. Department of Computer Graphics and Interaction

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **04.09.2022** Deadline for master's thesis submission: **10.01.2023**

Assignment valid until: **19.02.2024**

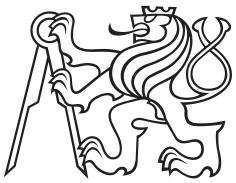
Ing. Ladislav Čmolík, Ph.D. Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

. | Date of assignment receipt | Student's signature

Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Computer Graphics and Interaction

Visualisation of electromagnetic field around a black hole

Bc. Oldřich Milec

Supervisor: Ing. Ladislav Čmolík, Ph.D.
January 2023

Acknowledgements

I would like to thank my supervisor Ing. Ladislav Čmolík Ph.D. for all of his feedback and advice both with designing the visualisation and writing this thesis. Additional thanks also go to all the participants of the usability testing who helped me improve the resulting visualisation.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague 10. January 2023

Signature:.....

Abstract

Electromagnetic fields around black holes are still not completely explored. This work analyzes different approaches of visualising them as 3D vector fields with special focus on their null points. An interactive visualisation was designed and implemented based on this analysis to improve the understanding of these fields. The visualisation uses streamline tracing to display the shape of the electromagnetic field and filtering to easily explore it. The streamlines are evenly-spaced and shaded to help with orientation in the scene. Several additional techniques which can improve the understanding of the vector fields can be enabled, such as visibility impeding halos or transparency mapping.

Keywords: vector field visualization; black hole; null point; electromagnetic field

Supervisor: Ing. Ladislav Čmolík, Ph.D.

Abstrakt

Elektromagnetická pole okolo černých děr zatím nejsou zcela prozkoumána. Tato práce analyzuje rozdílné metody jejich vizualizace jako 3D vektorových polí dr zaměřením na nulové body. Na základě této analýzy byla navržena a implementována interaktivní vizualizace sloužící k lepšímu pochopení těchto polí. Vizualizace využívá metody sledování proudnic pro zobrazení tvaru elektromagentického pole a filtraci pro jeho snadné prozkoumávání. Proudnice jsou pravidelně rozmístěny a stínovány za účelem lepší orientace ve scéně. Navíc je možné zapnout několik dalších technik, které mohou zlepšit porozumění vektorových polí, jako například zvýraznění obrysů nebo mapování na průhlednost.

Klíčová slova: vizualizace vektorového pole; černá díra; nulový bod, elektromagnetické pole

Contents

1 Introduction 1
1.1 Goals 1
1.2 Structure of the thesis 2

2 Analysis 3
2.1 Data 3
2.1.1 Grids 4
2.1.2 Electromagnetic field 5
2.2 Visualisation properties 6
2.3 Glyphs 8
2.4 Color coding 9
2.5 Displacement plots 10
2.6 Stream objects 11
2.6.1 Streamlines 11
2.6.2 Stream ribbons 14
2.6.3 Stream tubes 16
2.6.4 Stream surfaces 16
2.6.5 Seeding strategies 18
2.7 Vector field topology 20
2.8 Line integral convolution 22
2.9 Visibility impeding halos 24
2.10 Stereoscopy 24
2.11 Simultaneous visualisation of two vector fields 25

3 Design 27
3.1 Seeding and generation 27
3.2 Lighting 29
3.3 Visibility impeding halos 30
3.4 Interface and interaction 31

4 Implementation 33
4.1 Stream tubes 33
4.2 Shaders 35
4.3 Interaction 36

5 Results 39
5.1 Datasets 41
5.2 Stream tubes 42
5.3 Magnitude mapping 43
5.4 Filtering 45
5.5 Simultaneous visualisation of both vector fields 48

6 Usability testing 51
6.1 Test design 52
6.2 Test results 53
6.2.1 Comparative testing 56
6.2.2 Usability improvements 57

7 Conclusion 59
7.1 Future work 60

Bibliography 61

A Used acronyms 65

B Structure of attachments 67

Figures

1.1 First image of M87 black hole [1]. 2
1.2 Magnetic field of M87 [2]. 2

2.1 Different types of two dimensional grids, (a) equidistant grid, (b) regular grid, (c) rectilinear grid, (d) structured grid, (e) unstructured grid with varying cell shape, (f) unstructured grid with uniform cell shape [6]. 4
2.2 Different magnitude and identity channels. Magnitude channels are ordered based on their effectiveness [5]. 7
2.3 Vector field visualised using hexagonal pyramids, only one slice is shown to prevent occlusion. The magnitude of the field is mapped onto the size and color channels. [11]. 9
2.4 The same two dimensional vector field visualised using vector color coding. Magnitude is mapped to value in the left example. The right example only shows directions [4].. 10
2.5 Vector field with two displacement objects a sphere and a cube. The boundary of the field flow is visualised using short streamlines [4]. 11
2.6 Streamline visualisation of the electromagnetic field. Magnitude is mapped to color. 13
2.7 Depth mapped to saturation, white to blue. 14
2.8 Depth mapped to opacity, opaque to transparent. 14
2.9 Shaded stream ribbons. 15
2.10 Stream ribbons with visibility impeding halos. 15
2.11 Shaded stream tubes. 16
2.12 Semi-transparent stream surface of a nuclear fusion dataset [15]. . . . 17
2.13 Streamlines with randomly generated seeds. 19
2.14 Streamlines with seeds placed on a grid. 19
2.15 Streamlines with seeds generated by evenly-spaced seeding. 20
2.16 Different types of critical points with corresponding real and imaginary parts of eigenvalues of their respective Jacobi matrix [19]. 21
2.17 Two 2D vector fields with their topological skeletons overlaid on top. Different types of critical points are represented using different colors. Saddle points are yellow, sources red and sinks blue. [20]. 22
2.18 LIC slice of the magnetic field. 23
2.19 Two co-located vector fields overlaid on top of each other [24]. . 25
2.20 Electric (green) and magnetic(red) fields $d_{sep} = 60$ for both. 26

3.1 Initial design of the user interface. Lighter region contains properties of each filtered area. Darker region contains global properties. 32

5.1 Separating distance 120. 39
5.2 Separating distance 60. 39
5.3 Separating distance 30. 39
5.4 Separating distance 15. 39
5.5 Visualisation of the first dataset, only the bottom half is visible. Magnetic field is blue, electric field yellow. The empty area in the center of both images is the location of the black hole. 41
5.6 Visualisation of the second dataset. Magnetic field is blue, electric field yellow. Long electric streamlines curve around the null point separatrice. 42
5.7 Visualisation of the third dataset. Magnetic field is blue, electric field yellow. A separatrice runs diagonally through the magnetic field. 42
5.8 Stream tubes where both diffuse and specular light help to identify the 3D orientation. 43

Tables

5.9 Magnetic field, vector magnitude is mapped onto a linear color scale. Null point can be seen as a red dot. 44
5.10 Electric field, vector magnitude is mapped onto a linear color scale. . 44
5.11 Filtered area around the magnetic field null point. 45
5.12 Two filtered areas with magnetic stream lines. One with high density around the null point. Other with low density in the whole scene. . . . 46
5.13 Using dense transparent lines as context. Left – only context is transparent. Right – both focus and context are transparent. 47
5.14 A filtered narrow region of the magnetic field viewed from the top and from the side. The planar and vertical directions of vectors are visible. 47
5.15 Three areas covering 7 out of the 8 octants of the scene. The areas meet at the null point. 48
5.16 Visualisation of both the electric and magnetic field from dataset 2. Magnetic lines are blue, electric are red. 49
5.17 Visualisation of both the electric and magnetic field from dataset 3. Their intensity is mapped to color and visibility impeding halos are used. 49
5.18 Visualisation of both the electric and magnetic field from dataset 1. Filtering is used to see the streamlines around the surface of the black hole. 50

6.1 Comparison of the same streamlines with flat color and magnitude mapped to color. 55
6.2 Zoomed in region of dense streamlines without and with visibility impeding halos. 56

5.1 Average number of generated streamlines for each separating distance. Calculated from five generations for each dataset. 40

Chapter 1

Introduction

Black holes are fascinating astronomical objects with enormous gravitational pull that even light cannot escape. Because of that it is impossible to directly view a black hole. However, their accretion disk, the matter that closely orbits them but does not reach the event horizon, can be seen. Event horizon telescope [1] managed to take a direct picture of a supermassive black hole M87 in 2017. The rotating accretion disk can be clearly seen in figure 1.1. This accretion disk generates electromagnetic radiation due to its rotation. In 2021 the Event Horizon Telescope team [2] managed to measure parts of the magnetic field of M87 using polarized light. The resulting magnetic field was mapped onto the original image of M87 as seen in figure 1.2.

These strong electromagnetic fields could be responsible for extracting energy from rotating black holes and thus emitting astrophysical jets of ionised matter from their poles. This theory, known as Blandford–Znajek [3] process, is yet to be proven, however, it is the most widely accepted one. Understanding the shape of the electromagnetic field around rotating black holes could help with explaining the formation of astrophysical jets and accretion discs.

1.1 Goals

The aim of this work is to provide an interactive visualisation tool that can help us understand the behaviour of electromagnetic fields around rotating black holes. The direction and magnitude of both magnetic and electric fields, both individually and together, will be visible in any given area. However, the most important areas of interest are around points where the magnetic field intensity approaches zero. It will be possible to focus on these so called null points while keeping the general context of the whole electromagnetic field. Using filtering the user will be able to hide needless information and explore their selected areas of interest. While zooming in on these preferred areas the resolution of the visualisation will dynamically change so it uses the screen area as efficiently as possible without being overly obfuscating while zoomed out.

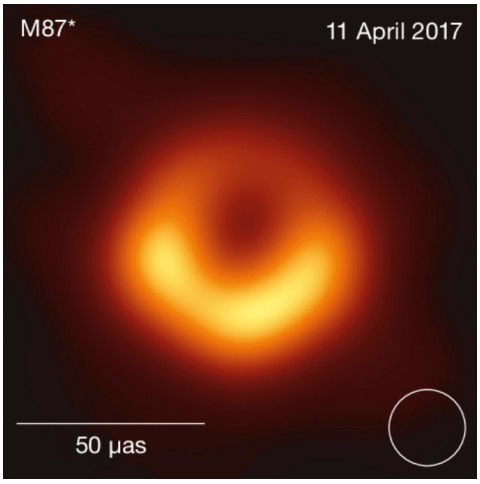


Figure 1.1: First image of M87 black hole [1].

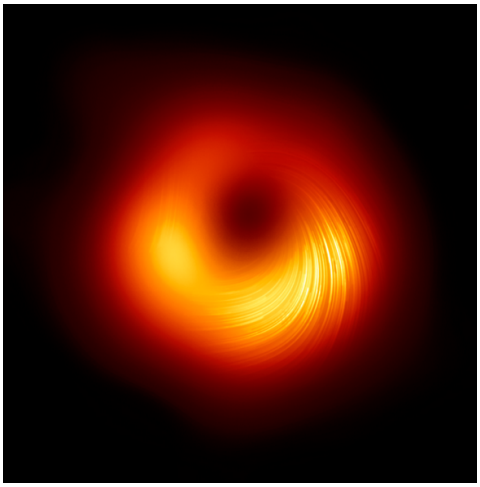


Figure 1.2: Magnetic field of M87 [2].

1.2 Structure of the thesis

The following text of this thesis is structured to 7 chapters. Chapter 2 describes the analysis of three dimensional vector fields and various techniques for their visualisation with regard to the supplied electromagnetic field data. The visualisation itself is designed in chapter 3 based on the results of the analysis. Chapter 4 describes the actual implementation of the visualisation concerning specifically used technology. The results and examples of the visualisation can be found in chapter 5. The design and results of usability testing are described in chapter 6. Chapter 7 is a conclusion of this thesis.

Chapter 2
Analysis

Telea [4] describes visualisation as a tool which helps humans to understand data. It is supposed to provide answers to specific questions of the user or to help them discover new information about the problem of which they were not aware. There are two types of questions the user might have, qualitative and quantitative. Simple quantitative questions such as "What is the population of the largest city?" can be answered using a text based query without the need of a visualisation. Even answers to those questions can be more easily digested when presented using a visualisation. Qualitative questions such as asking for a location of this city, a map is more understandable than coordinates. Moreover, a visualisation helps to provide additional information the user did not specifically request but can still help them understand the broader context of the specific answer. Lastly it can be used by users without any specific questions. These users can gain new insights from a visualisation presenting the data in a way where their pattern recognition helps them understand it. They may realize some unexpected correlation or similarity to previously seen data. Or it may help them actually formulate specific questions about the data. Ideally, the visualisation can provide answers to precise qualitative and quantitative questions and at the same time present the data in a way which generates new often unexpected insights into it.

2.1 Data

Munzner [5] specifies five basic data types: items, attributes, grids, positions and links. Items are discrete entities such as nodes in a graph or rows in a table. Attributes or variables are specific properties which can be qualitative or quantitative. A grid specifies a sampling of continuous data including its topology. Position describes a location in two, three or rarely more dimensional space. A link is a relation between two items.

Four basic dataset types can be described using some of these data types. They are tables, graphs, geometry and field. Tables contain items and attributes. Graphs contain items, attributes and links. Geometry contains items and positions. Fields contain grids and attributes. As the name suggests electromagnetic field is a field dataset and can be represented using grid and attributes.

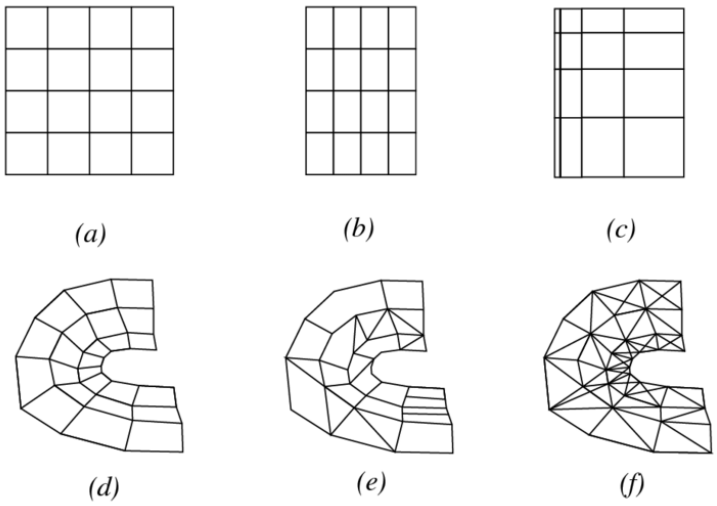


Figure 2.1: Different types of two dimensional grids, (a) equidistant grid, (b) regular grid, (c) rectilinear grid, (d) structured grid, (e) unstructured grid with varying cell shape, (f) unstructured grid with uniform cell shape [6].

2.1.1 Grids

Grids can be equidistant, regular, rectilinear, structured or unstructured as seen in Figure 2.1. Equidistant or uniform grids have the same distance between each pair of neighboring sampled points in all dimensions. In other words, each cell of an equidistant grid is either a square in 2D or a cube in 3D. Thanks to this property, indices of equidistant grids correspond to rounded world coordinates. Only an array of values is needed to represent an equidistant grid.

Regular grids have the same distance between sampled points for each of the three axes. The cells are rectangular cuboids with the same size. In addition to an array of values three additional parameters describing the sampled distance in each dimension are needed.

The cells of rectilinear grids are still rectangular cuboids, however, with varying sizes. The distance of sampled points varies along each axis. To represent this grid an additional array with distances between samples is needed for each dimension.

Structured grids have all cells of the same type which can vary from tetrahedra to hexahedra. Each cell has its own explicit coordinates. The whole grid still represents a matrix-like structure. To represent this grid each cell also needs to store its coordinates.

Unstructured grids specify the coordinates of both vertices and cells. The cells in an unstructured grid can have varying shape, however, usually only one shape is used such as triangles or quadrilaterals. Vertices are defined by their position in space and cells are defined by multiple vertices. While this grid type is the most flexible, it also is the most memory and computation expensive.

Sometimes data points can be sampled without an underlying structure only with positional information. These data sets can either be reconstructed using interpolation with a radial basis function or an unstructured grid can be created using triangulation.

2.1.2 Electromagnetic field

Electromagnetic field is a continuous vector field in 3D space with two vectors, magnetic force and electric force respectively, in any given point. In order to describe and visualise this particular electromagnetic field each of the forces is sampled in an equidistant grid with 512 dimensions in each of the three axes. Therefore the total amount of vectors is two times 134 217 728. This means that the size of the data is around 3 GB for each black hole. Any point in the field outside of the sampled ones can be found using interpolation. The simplest way is to use trilinear interpolation from the 8 surrounding sampled points. In two out of three data set there is exactly one point where the magnetic field intensity approaches zero also known as a null point. These points are particularly interesting because the directional changes of vectors are the most prominent around them as they form a saddle point.

There are two attributes to visualise, vector magnitude and vector direction for each of the fields. At the same time the topology of the grid needs to stay unchanged which means that location cannot be used as a visual channel because it inherently represents the position inside the vector field. At first glance, the color channel is the most likely candidate for visualising vector magnitude. The magnitude is continuous so other channels like size have lower discriminability and possibly even accuracy and separability. Tilt is the most likely candidate for vector direction. The marks representing the vectors need to be tilted in the vector's direction. The tilt of the marks needs to be obvious in 3D space. For example the orientation and length of a simple line is obvious in 2D, however, it is ambiguous in 3D and can represent many different lines with the same screen space projection. Shading or more complex 3D shapes can solve this issue.

Vector fields have several additional properties. Divergence of a vector field V is a scalar field representing the flux outgoing from any given point inside the field. It can be calculated using the equation 2.1 A point with positive divergence is called a source while a point with negative divergence is called a sink. As a scalar field it can be mapped onto a color channel of any mark and visualised. However, initial testing showed that the provided electromagnetic field does not have any areas where visualising divergence shows anything interesting.

$$div(V) = \nabla \cdot V = \frac{\delta V_x}{\delta x} + \frac{\delta V_y}{\delta y} + \frac{\delta V_z}{\delta z} \tag{2.1}$$

Curl of a vector field V also known as vorticity represents the circular motion around any given point. Curl is properly defined only for 3D vector fields and is also a 3D vector field. The direction of each vector forming this field represents the axis of maximal rotation around the point while the

magnitude represents the rotation density. Curl can be calculated using the equation 2.2. Turbulent flows have high vorticity while laminar flows have low. The provided dataset has mostly laminar flow with the exception of the area nearby the null point.

$$curl(V) = \nabla \times V = \left(\frac{\delta V_z}{\delta y} - \frac{\delta V_y}{\delta z}, \frac{\delta V_x}{\delta z} - \frac{\delta V_z}{\delta x}, \frac{\delta V_y}{\delta x} - \frac{\delta V_x}{\delta y} \right) \tag{2.2}$$

Another scalar property closely tied to vorticity is streamwise vorticity Ω which measures the rotation of a vector field around itself. In other words how much the vector field rotates in any given point around the vector in this point. It can be calculated by projecting the vector field onto the curl of the field as seen in equation 2.3.

$$\Omega = \frac{V \cdot curl(V)}{\|V\|} \tag{2.3}$$

2.2 Visualisation properties

Marks and channels are used by Munzner to describe how is the data converted to an image. Marks are graphical elements which form the whole visualisation such as points, lines, shapes, 3D objects and so on. Visual channels alter the appearance of the mark without changing the geometric primitive. There are many channels including position (horizontal, vertical and depth), size (length, area, volume), hue, tilt and motion as seen in Figure 2.2.

These channels are used to encode attributes. Each attribute can be represented using one or more channels. Using more channels can improve visual perception of these attributes while at the same time limiting the amount of attributes which can be shown at the same time. Not all channels can be used to represent quantitative attributes. Munzner separates them into identity channels and magnitude channels. Identity channels are used for qualitative attributes and answer questions such as what is something or where it is. These channels can be hue, position or shape. Magnitude channels are used to represent quantitative attributes and answer questions how much of something there is. Magnitude channels are size, tilt, saturation and brightness.

Not all channels are effective at encoding certain attributes, therefore it is necessary to analyze them based on several criteria. Munzner uses accuracy, discriminability, separability, popout and grouping.

Accuracy measures how well can users read the information encoded in the channel. According to Stevens's power law [7] the apparent magnitude of a channel is not linear, instead it follows a power function. The only exception is one dimensional length which is linear. Certain channels such as area or brightness are compressed while others such as color saturation are magnified.

To improve accuracy of color channels Crameri et al. [8] suggest several criteria for selecting color spaces. The used color mapping should be perceptually uniform. Even though the representation of the colours is not linear

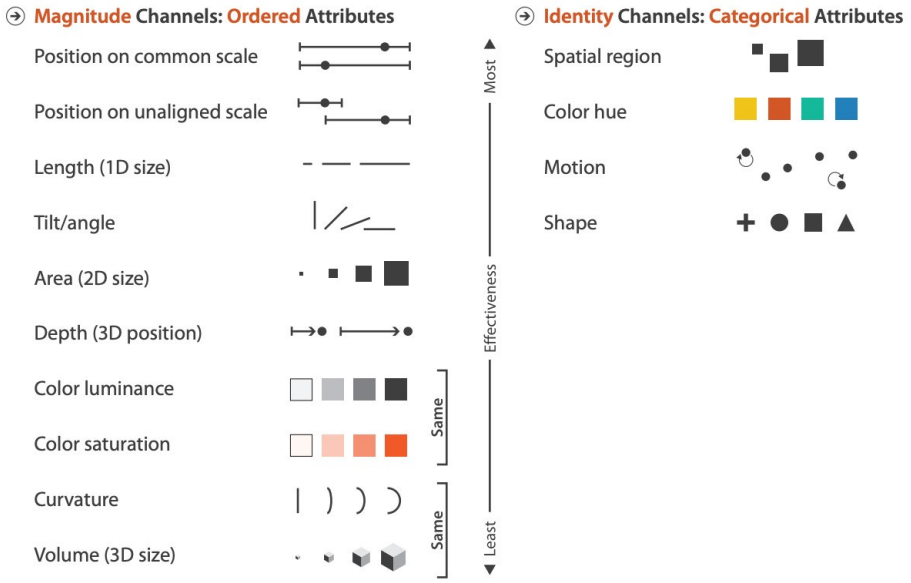


Figure 2.2: Different magnitude and identity channels. Magnitude channels are ordered based on their effectiveness [5].

their visual perception is. It should not contain red and green colours with similar saturation and brightness. These colours are hard to distinguish for a non-negligible part of the population because of red-green color blindness. Do not use color maps with larger amount of hues, such as the commonly used rainbow color map, for quantitative attributes. The changes in hues create artificial edges in the visualisation which do not exist in the data. In addition there is not an inherent meaning in the spectral order of the hues which makes intuitive understanding of magnitudes difficult.

Discriminabilty measures how many steps of the attribute can the channel represent. According to Veras and Collins [9] low disciminability indicates that changes in data are not reflected as changes in the visualisation. Therefore some areas of the visualisation are ambiguous and do not properly represent the original data. When working with large datasets it is important to aggregate or sample the data so the visualisation does not saturate. Ellis et al. [10] define saturated visualisations as those where data points or lines are overplotted, clustered or even overlapping and any patterns may be hidden. Sometimes coloring can be used to indicate the amount of overplotting and help differentiate the individual plots. However, this is not possible if the color channel is already used to represent an attribute of the data. The other issue is that the order of rendering individual marks can affect the resulting visualisation. The proposed solution is to sample saturated areas and therefore improve discriminability. Different channels have varying discriminability. Line width has the lowest with only three discernible levels before lines start to appear as areas. Sometimes, however, that can be enough and line width can be the correct choice for a specific visualisation. It is important to correctly match the range of the data with the range of the visual channel.

Separability measures how different channels interact with each other. Munzner calls channels which do not interact at all separable channels and channels which are always intertwined integral channels. The negative interaction between channels is called interference. Pairs of channels can have varying level of interference ranging from fully separable to fully integrated. Position in 2D and hue is an example of fully separable pair (or even a triple when considering horizontal and vertical position) of channels. Size and hue introduces slight amount of interference due to the fact that the hue of smaller marks can be harder to distinguish and in some cases can be perceived differently due to the background color. Any pair of three RGB color channels are fully integrated and cannot be perceived individually. In 3D the position and size have major interference as any mark that is large but far from the camera can appear the same size as these that are near but are smaller.

Popout measures how much a distinct mark stands out from multiple other distractor marks. The interesting and useful fact is that the amount of distractor objects does not increase the time that it takes to find the distict object. This is done almost automatically by the visual system and is called preattentive processing. Depending on the selected channels popout can be faster or slower. It is very effective when using marks with distinctive hues, while it is slightly slower for marks with same hues but different shapes. There are many different channels which support popout such as size, proximity, hue, tilt, shape and even motion. Popout can be used to quickly identify outliers or user defined items.

Certain channels are better for representing groups of items. Munzner describes four levels of grouping. The most effective is to use links between grouped objects forming a graph like structure or to contain the grouped object in one area specified by a boundary. While this provides the best distinction of grouped objects it also adds additional clutter to the scene and can reduce discriminability. The other approach is to use identity channels for groups, this approach is not as effective and can reduce separability, however it also does not need additional clutter. The third possibility is to group objects by proximity. The last approach uses similarity of hue or shape channels.

Telea [4] proposes several methods for vector visualisation such as glyphs, color mapping, displacement plots, stream objects and texture based visualisation.

2.3 Glyphs

Glyphs are simple shapes or in case of 3D vector fields simple objects which are placed rotated according to the direction of vectors in each sampled point. There are multiple sampling strategies 2.6.5 which vastly differ in output readability. The simplest shape to use is a line which works well in 2D but in 3D their orientation becomes ambiguous. Using shapes with inherent orientation information such as cones or arrows helps to visualize

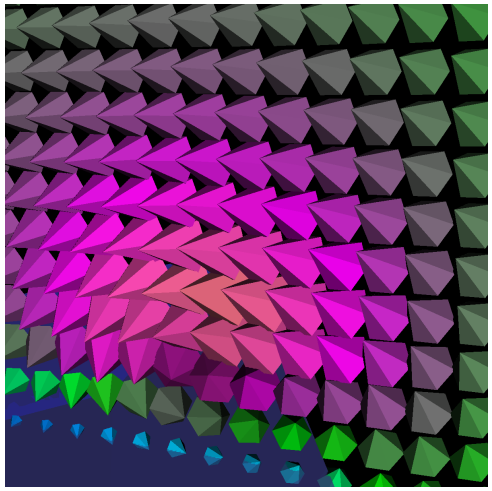


Figure 2.3: Vector field visualised using hexagonal pyramids, only one slice is shown to prevent occlusion. The magnitude of the field is mapped onto the size and color channels. [11].

vector direction in each sampled location. However these shapes also use up more space which can clutter the visualisation so a lower amount of samples is needed. Using more complex and specialized shapes can help to encode additional scalar attributes of the field. The color channel of the glyph can be used to convey some other attribute of the field such as magnitude or divergence. Alternatively the magnitude can be mapped to the size of the glyphs.

The biggest advantage of glyphs is the simplicity and performance of the implementation. A single object gets copied and rotated to each sampled point. The main drawback of glyphs is their discrete nature. It can be difficult to understand the flow of the field using glyphs since they are not directly connected. Another problem arises in 3D only where glyphs will occlude each other in screen space. This issue can be partially solved by using semi-transparent glyphs or subsampling the field which in turn reduces the understanding of the flow even more. Because of these drawbacks glyphs are not useful for visualising electromagnetic field.

2.4 Color coding

Sparse visualisations such as already discussed vector glyphs have several issues which can be alleviated using dense visualisations. Color coding is one of the possible techniques to densely visualise vector fields. For each point from the field a hue is selected based on the vector orientation. Using the HSV representation of colors the hue can become circular representing any angle. The saturation or value can be also used to represent the magnitude of the vector. Both of these variants can be seen in Figure 2.4. This approach is usable with two dimensional vector fields where the vector direction can be seen as a planar angle.

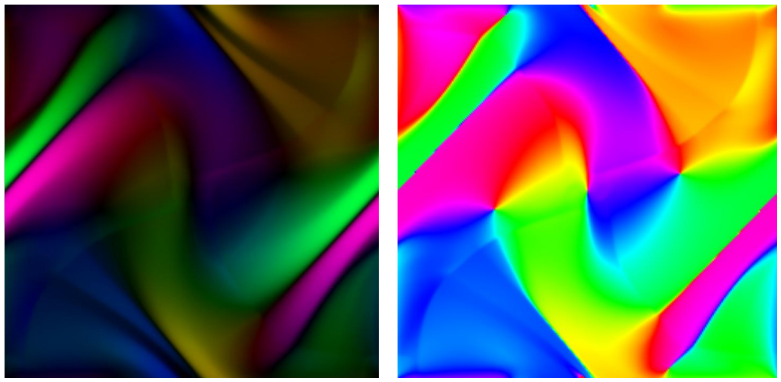


Figure 2.4: The same two dimensional vector field visualised using vector color coding. Magnitude is mapped to value in the left example. The right example only shows directions [4].

The color coding is more complicated for 3D vectors. Hue can be used again to represent a planar angle, usually the direction in the x, y plane. The z component needs to be represented using saturation or value. These channels are not fully separable and introduce interference. This makes it very hard to visually decode the vector direction in any given point. In addition 3D dense visualisations do not work as well due to occlusion. Therefore color coding can only be use in addition to some other technique such as glyphs. Another option is to filter the dataset to only show a user defined slice. This fixes the occlusion issue, though even in a 2D slice the visualised vectors are still three dimensional. Slices can help with understanding the vector field in a specific area, however, the context of the whole field is lost. Color coding has too many issues to be effective at 3D vector visualisation and will not be used in this work.

2.5 Displacement plots

Vector fields can be also viewed as a force field which deforms anything inserted into the field. Displacement plots are objects such as a plane, cube or sphere which are placed into the field and at each sampled point moved in the direction of the vector. The distance of the displacement is relative to the magnitude of the vector. Telea formally describes it as an surface $S \epsilon D$ where D is a vector field domain and surface S is represented as a set of sampled points p_i. Each point from S is displaced using equation 2.4 creating a displacement plot S' where k is a displacement factor and $V(p_i)$ is a vector at the point p_i. Displacement factor affects the strength of the displacement effect. When it is too small, relative to a specific vector field, the surface does not change enough to produce any usable result. While if it is too large the displacement plot can become too stretched when the difference between neighboring sampled points becomes too large.

$$p'_i = p_i + kV(p_i) \tag{2.4}$$

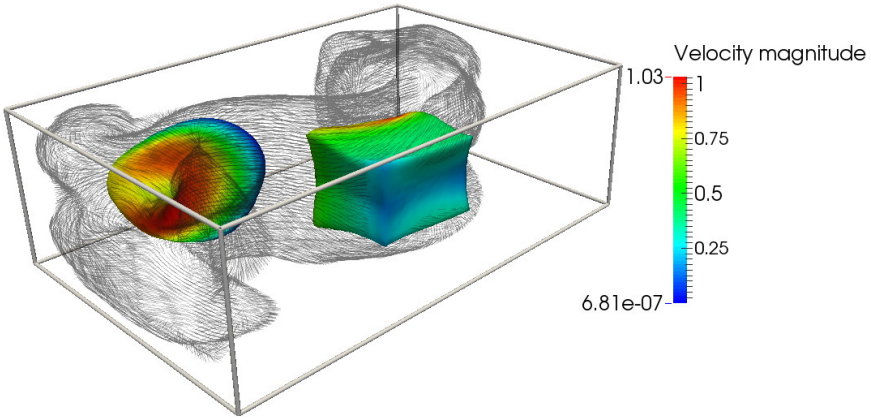


Figure 2.5: Vector field with two displacement objects a sphere and a cube. The boundary of the field flow is visualised using short streamlines [4].

It is important to note that the resulting plot does not represent the shape of the vector field, it only shows the direction and magnitude of the vectors on the surface S in the original location. This can be useful for vector fields representing flow where the user is interested in the cross section of the flow. Selection and placement of the surface also needs to be done carefully. Planar surfaces are the usual choice because the result is easy to interpret. However, flat surfaces perpendicular to the flow will only move a little and can self intersect. A sphere is symmetrical so it does not have this issue, however, parts of the sphere can intersect if the displacement factor is not carefully selected.

2.6 Stream objects

Stream objects try to visualise a vector field by tracing lines as if particles are being carried away by the vector field. There are multiple techniques which all visualise the vector field as a collection of trajectories. Some possible examples are streamlines, stream ribbons, stream tubes and stream surfaces.

2.6.1 Streamlines

Streamlines are curves whose tangent at any given point is the vector in the same point. They are generated by numerically integrating the vector field from certain sampled points also called seeds. The selected integration method and seeding method 2.6.5 can greatly affect the outcome so it is necessary to carefully consider the best approach for the specific problem. The most common integration methods are Euler, Runge-Kutta and adaptive.

Euler integration repeatedly takes the vector at the current point and repeatedly advances for a given time Δt in the direction of the vector both forward and backward. In each step the vector needs to be interpolated from eight nearby samples. Each step is a straight line, however, the underlying vector field usually contains smooth curves. Therefore, each step is made

with error which is approximately Δt^2. The error accumulates with each step and the resulting global error is $T\Delta t$ where T is the total time reserved for the streamline. This means that Δt needs to be small relative to magnitudes of the vector field in order for the error to stay low. The smaller the Δt the more integration steps need to be performed; therefore, accurate integration can be quite slow, even though this is the simplest method. The exact computation of one Euler iteration can be seen in equation 2.5. The current point is $p(t)$ and the calculated point is $p(t + \Delta t)$. The function $V(p)$ returns the interpolated vector at point p.

$$p(t + \Delta t) = p(t) + \Delta t \times V(p(t)) \tag{2.5}$$

Second order Runge-Kutta integration consists of two vector calculation in each step. Two euler steps are computed and the resulting vector is the average of these two steps. It is then added to the original point and the whole process repeats. One iteration can be seen in equation 2.6. This method reduces local error to Δt^3 and global error to $T\Delta t^2$.

$$\begin{aligned} \vec{v_1} &= \Delta t \times V(p(t)) \\ \vec{v_2} &= \Delta t \times V(p(t) + \vec{v_1}) \\ p(t + \Delta t) &= p(t) + \frac{1}{2} \times (\vec{v_1} + \vec{v_2}) \end{aligned} \tag{2.6}$$

Fourth order Runge-Kutta integration expands this method to four vector calculations in each step. These four vectors are then combined using weighted average. The computation can be seen in equation 2.7. When using this method the global error is reduced to $T\Delta t^4$.

$$\begin{aligned} \vec{v_1} &= \Delta t \times V(p(t)) \\ \vec{v_2} &= \Delta t \times V(p(t) + \frac{\vec{v_1}}{2}) \\ \vec{v_3} &= \Delta t \times V(p(t) + \frac{\vec{v_2}}{2}) \\ \vec{v_4} &= \Delta t \times V(p(t) + \vec{v_3}) \\ p(t + \Delta t) &= p(t) + \frac{1}{6} \times (\vec{v_1} + 2\vec{v_2} + 2\vec{v_3} + \vec{v_4}) \end{aligned} \tag{2.7}$$

Adaptive methods use changeable step size based on error estimation. Liu, Moorhead and Groner [12] use error estimation formula visible in equation 2.8. It is to be used with fourth order Runge-Kutta integration. It compares the vector at the expected next point to the vector at the actual computed next point. If the difference ϵ is too large the step Δt is halved. If it is too small it is doubled.

$$\epsilon = \frac{\vec{v_4}}{6} - \frac{V(p(t + \Delta t)) \times \Delta t}{6} \tag{2.8}$$

It is important to consider the criteria which dictates when the tracing should stop. One possibility is to have constant maximum time T reserved for each streamline. That way the amount of integration steps is the same for

each streamline unless an adaptive integration method is used. Nevertheless, the resulting length of each streamline varies based on the changing vector magnitude. This can lead to vastly different lengths of streamlines in vector fields with large fluctuations of magnitude. Usually it is more desirable to have all streamlines with the same or at least similar length. To achieve this the length of the streamline needs to be independent of the integration time T. Instead the integration needs to be terminated based on the integrated distance. The magnitude of the currently processed vector multiplied by δt is added in each step and when it reaches the desired total length the integration is terminated. The resulting streamlines all have the same length unless they leave the boundary of the sampled vector field. If the maximum length is low the visualisation looks similar to vector glyphs.

Longer streamlines provide more insights into the shape of the vector field. However, they also overlap more and introduce more clutter. Additionally long streamlines may blend together in areas where the vector field converges. To solve this issue an additional stopping criterion needs to be introduced which terminates the integration when the streamline gets too close to other already integrated streamlines. This approach is computationally expensive unless a specialized data structure is used for storing already traced streamlines. In addition the integration of an individual streamline is no longer independent which heavily affects parallel tracing. It is still possible to trace streamlines simultaneously, however, due to the stopping criterion several race conditions are introduced and need to be properly dealt with.

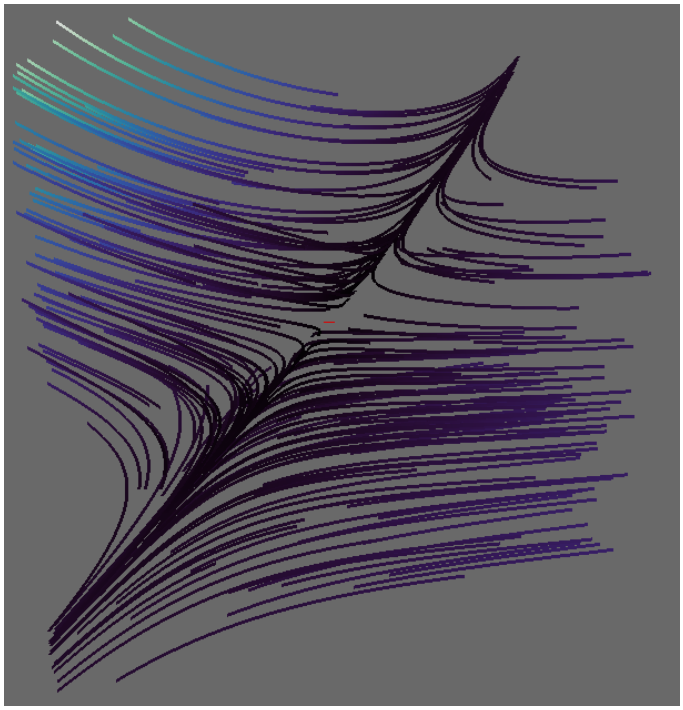


Figure 2.6: Streamline visualisation of the electromagnetic field. Magnitude is mapped to color.

Figure 2.6 shows a simple streamline prototype using black hole magnetic field data. The general shapes of magnetic field lines are visible. However, there are several issues. The depth information is lost, the lines overlap and sometimes even merge into large indistinguishable areas. Another problem lies in discerning the direction and orientation of individual lines.

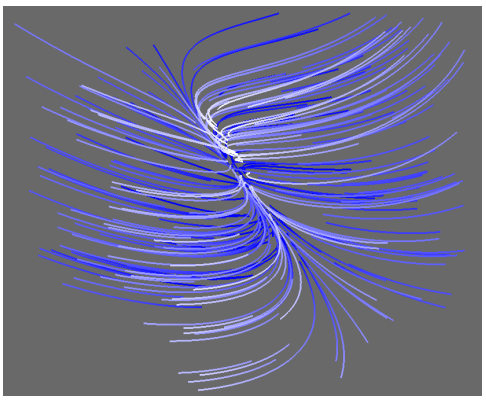


Figure 2.7: Depth mapped to saturation, white to blue.

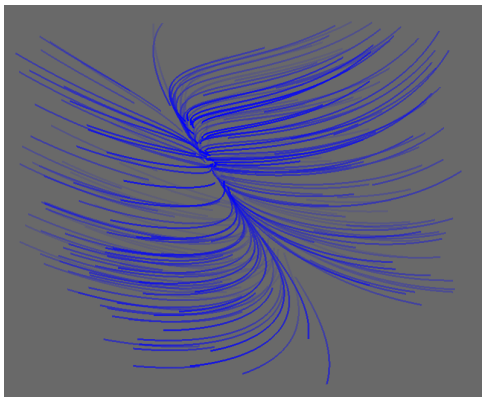


Figure 2.8: Depth mapped to opacity, opaque to transparent.

The depth information can be mapped either to the color or opacity channel. This mapping needs to run in real-time and adjust itself whenever the camera is changed. The results can be seen in Figure 2.7 and 2.8 respectively. The distance from camera can now be easily seen. However, the color channel can no longer be used to visualise the magnitude of the vector field. Moreover, the orientation and overlapping of streamlines with similar depth is still hard to distinguish. Converting the streamlines into proper 3D objects 2.6.3 and adding lighting and shading could help improve this issue. The other method which can improve the readability of streamlines visualisation is to select a better seeding strategy 2.6.5.

2.6.2 Stream ribbons

Streamlines are not particularly good for displaying local vorticity, or in other words how the vector changes when moving perpendicularly to the direction of the vector field. Stream ribbons help with visualising vector fields with high vorticity. Ribbons consist of two streamlines traced from seeds close to each other rendered as a sequence of quadrilaterals. As seen in figure 2.9 the shading can help discerning their orientation, however the distance from camera and overlapping problems are still present. Furthermore, a new problem arises when tracing stream ribbons. The streamlines that form the ribbon can diverge and the ribbon quickly becomes too wide obscuring the whole scene behind it. On the other hand they can also converge and degrade into a simple streamline. Both of these issues can be solved by checking the distance between the current points of the ribbon in each step and ending the tracing prematurely if it becomes too large or too small.

Matthausch, Theubl et al. [13] suggest using visibility impeding halos to better distinguish where lines or ribbons overlap. These halos are simple solid color lines following the edges of a ribbon. This effect can be seen in figure 2.10. When two or more ribbons overlap it is now instantly visible which one is in front of the others. This technique produces visually better results than simple streamlines, however, it is not too well suited for this problem since the vorticity is low. When visualising the magnetic field vector field ribbons often converge close to a plane running through the null point collapsing into a single streamline. A multi-resolution tracing generating narrower ribbons only around these areas could solve this issue.

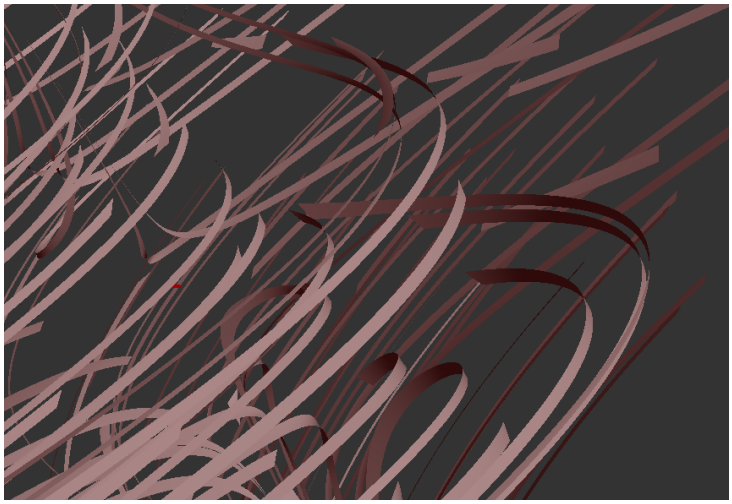


Figure 2.9: Shaded stream ribbons.

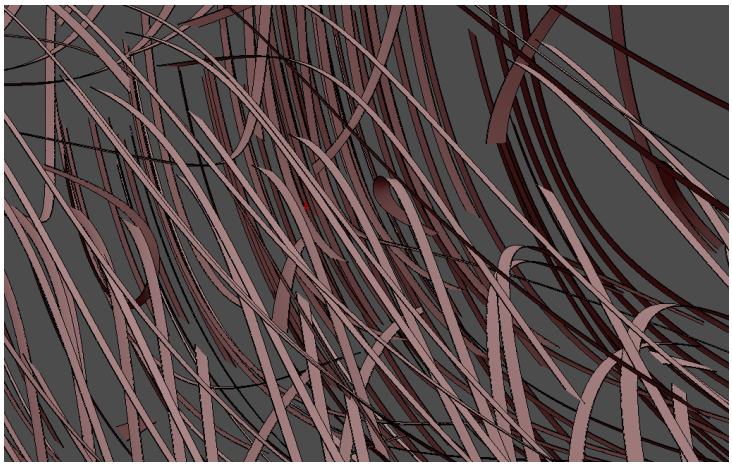


Figure 2.10: Stream ribbons with visibility impeding halos.

2.6.3 Stream tubes

Another problem with simple streamlines is that they are only 2D objects and cannot be shaded. Stream tubes are generated exactly the same way as streamlines except they are or appear as a solid 3D object. They can be represented as polygonal tubes or just as polylines cleverly shaded using a method introduced by Zockler, Stalling and Hege [14]. In which the lines are shaded using a light texture map. Shaded tubes provide more information about orientation of the streamline, while perspective projection also helps with identifying depth. Compared to stream ribbons, stream tubes shading generally looks better thanks to their round shape. Same as before visibility impeding halos can be used to better distinguish overlapping tubes.

Figure 2.11: Shaded stream tubes.

Simple prototype of shaded stream tubes can be seen in figure 2.11. There is only one global directional light which does not produce the best results. Various other light setups would probably improve the visualisation. Stream tubes seeding is random, a better seeding strategy 2.6.5 is necessary for a less cluttered output. Stream tubes are not particularly good at visualising fields with high vorticity perpendicular to the main direction of the streamline. This dataset does not have that property, therefore stream tubes do not have any major disadvantage compared to other stream objects. The issue of overlapping stream tubes still persist and needs to be solved in some other way such as filtering or transparency.

2.6.4 Stream surfaces

The seeding area can also form an arbitrary curve. Stream surfaces integrate lines from each seed of the curve and then merge the result into a polygonal surface as seen in Figure 2.12. Telea formally defines it as a stream surface S_Γ which is in any point tangent to the vector field and contains the seed curve Γ. This technique can be used to visualise how the seed curve advects over time. Since the stream surface is always tangent to the flow represented

by the vector field it can be used to separate the field into disjoint areas. The flow from one area cannot reach the other area because it cannot cross the stream surface, at most it can only converge to it. The simplest construction of the surface is to trace streamlines and then connect the points to form a polygonal mesh. Points can be connected to the nearest points of neighbor streamlines. Alternatively each point also keeps track of its line distance from the seed curve and neighboring points with the same distance are connected.

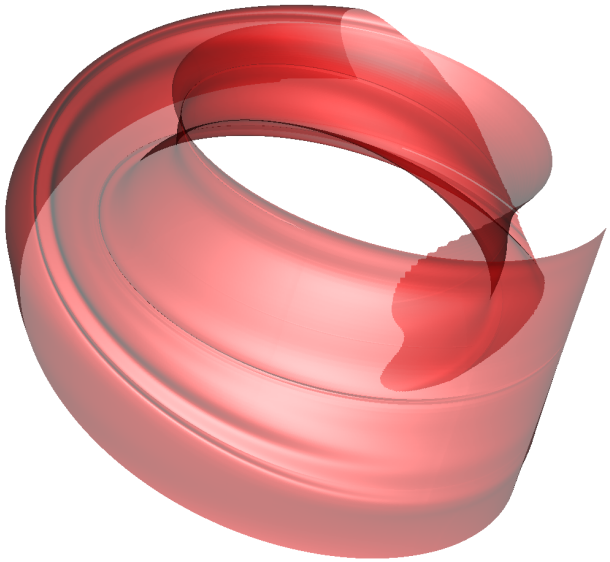


Figure 2.12: Semi-transparent stream surface of a nuclear fusion dataset [15].

Regions where the streamlines converge or diverge need to be treated with care. Convergences are not that much problematic, the only issue is a needlessly large amount of extremely small polygons which can be replaced by fewer larger ones. Divergences pose way more problems. When the streamlines diverge, polygons get larger and larger and stop following the actual flow of the vector field. There are two possible solutions to this issue. Either to stop connecting points with too large distance between them or to add new seeds between the diverging streamlines. The first solution is simpler, however, it introduces tears and holes into the polygonal surface. Tears are created by diverging lines while holes are created when diverging lines converge again and get merged creating polygons once more.

The seeding curve also needs to be selected with care. A similar problem as with displacement plots arises, if any parts of the curve are tangent to the vector field that part of the surface collapses. Int the worst case scenario the seeding curve is a streamline and the surfaces becomes only a streamline as well. A solution is to let the user freely select and change the seeding curve. They can simply redefine it if the resulting surface is not useful. Another approach is to automatically select one or several interesting locations. There are several methods which consider multiple curves and select the best one based on the direction of vectors along the curve. Ideally the direction is

always perpendicular to the curve. This approach does not take into account the global flow of the field which means that the resulting stream surface can converge even if the curve is locally optimal.

Esturo et al. [16] propose a global selection method which looks at the whole vector field and tries to find the optimal stream surface instead of an optimal seeding curve. The domain of the vector field is subdivided into cubes with additional diagonal edges. Then a stream ribbon is integrated from each of these edges. This is the most time consuming part of the algorithm, however, ribbons can be traced in paralleled. Each edge is assigned a weight which is based on the quality of the stream ribbon. The quality is determined by the alignment, curvature and area of the ribbon. The final seed curve is obtained by finding a simple path with minimal sum of weights. The authors use simulated annealing in this step. The seed curve is then reconstructed from the path using corner-cutting subdivision.

Stream surfaces are useful for visualising the most interesting areas of the vector field. They cannot show the whole shape of the field except in special cases. Because of that they are most useful for visualising flow or other vector fields with inlets and outlets where the surface will span the majority of the field. Displaying multiple surfaces to try to visualise larger area of the field is not particularly useful as one surface occludes the rest. There does not seem to be any obvious feature of the provided vector field which would benefit from being visualised by this method. Therefore it will not be considered in the design of the visualisation.

2.6.5 Seeding strategies

Streamlines and all of their derived stream objects need a point where the integration is supposed to start. There are several different ways how to find these points. Random seeding, Grid seeding, Evenly-Spaced seeding and Multi-Resolution seeding will be further explained.

Random seeding is a fast method which generates a random point as a start for each streamline. The main issue lies in skewed density and distribution of the lines in space. There are too many overlapping lines in certain areas while others can remain empty. This is especially pronounced in diverging vector fields. The result of random seeding can be seen in figure 2.13, the discussed problems are clearly visible. The top left area contains only two streamlines while the center area contains multiple streamlines which are not only overlapping in screen-space (which is inevitable) but also in 3D space.

Grid seeding can help with evenly distributing streamlines in space. Seeds are placed on a regular grid with specified distances between seeds. Even though streamlines generated in this way tend to fill the scene more evenly they also create unwanted patterns which can be distracting or even misleading as seen in figure 2.14. This approach also does not help with converging vector fields as streamlines can still easily overlap.

A more complex method introduced by Jobard and Lefer [17] evenly distributes streamlines into the scene using all previously created streamlines as a reference. A separating distance d_{sep} is the minimal distance between

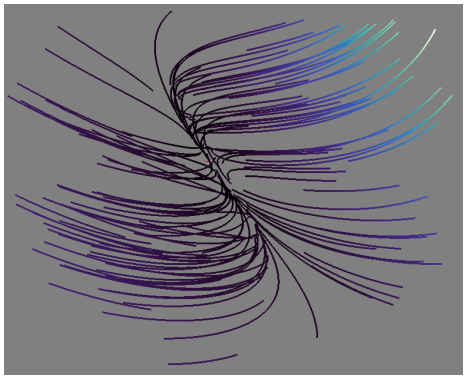


Figure 2.13: Streamlines with randomly generated seeds.

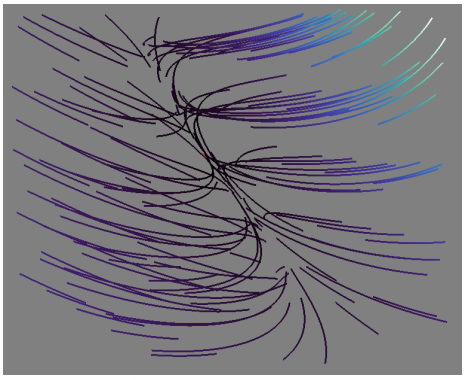


Figure 2.14: Streamlines with seeds placed on a grid.

a streamline and the newly generated seed. A testing distance d_{test} is the minimal distance between a newly traced point and all other already generated points. In other words the streamline generation stops when a new point would be closer than d_{test} to any point of any existing streamline.

The algorithm itself starts with generating a randomly seeded streamline and placing it into a queue. Until the queue is empty a streamline is taken from the queue and all seed points which are exactly d_{sep} away from the current line and at least d_{sep} away from all the other lines are found. In 2D there are exactly 2 candidate seed points for each point of the streamline, the points perpendicular to it. In 3D there are infinite candidate points forming a circle perpendicular to the streamline. Matthausch, Theubl et al. [13] use 6 points forming a regular hexagon because the neighboring points are d_{sep} from each other. The whole process is sequential which means that when a new valid seed is found a new streamline is immediately traced and added to the queue. The points of this new streamline contribute to distance comparisons when searching for additional seed points from the streamline which is still being processed.

A lot of distance calculations need to be made both during seeding and tracing. To reduce this number a regular grid is built over the scene which contains cells of size d_{sep}. Points from new streamlines are added to their corresponding cells. Each new candidate seed only needs to be compared with points in its cell and 8 neighboring cells for 2D fields or 26 for 3D fields.

The result of this approach can be seen in figure 2.15. No streamlines overlap in 3D space since they cannot be closer than d_{test}. The whole scene is more evenly filled without generating misleading patterns. In this prototype candidate seeds were simply selected as random points d_{sep} from each processed point. Using regular hexagon as described above would make the streamlines slightly more evenly spaced.

Multi-resolution extension of evenly-spaced seeding algorithm, as proposed by Matthausch, Theubl et al. [13], generates evenly-spaced streamlines multiple times with decreasing value of the separating distance d_{sep}. The output of the previous resolution is taken as an input of the currently computed

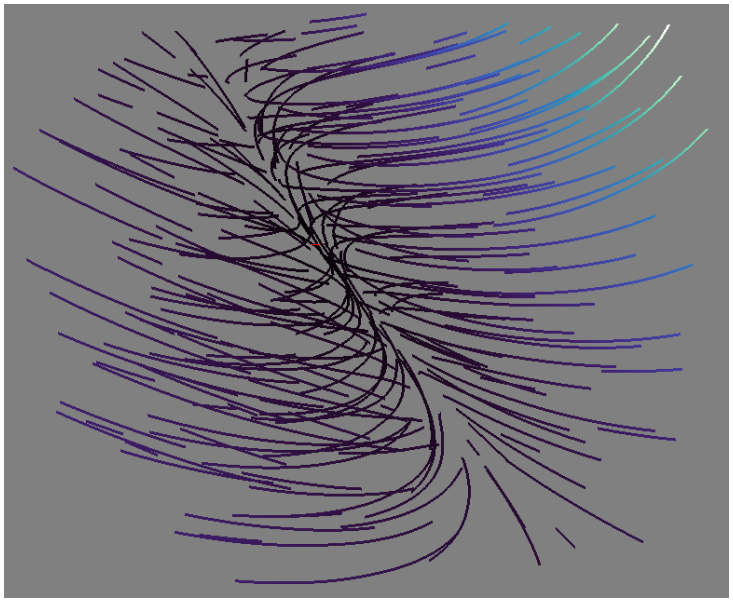


Figure 2.15: Streamlines with seeds generated by evenly-spaced seeding.

resolution. This way increasing resolution keeps the original streamlines and only adds additional ones between them. Different resolutions can be displayed together using focus and context. A smaller focus area is displayed in higher resolution and the rest in lower resolution serving as context.

2.7 Vector field topology

Critical points are the most interesting areas of most vector fields. Helman and Hesselink [18] describe them as points where the magnitude of the field vanishes and classify them into distinct cases which can be characterized by the behaviour of nearby curves as seen in Figure 2.16. To analyze this behaviour a Jacobi matrix of first derivatives is constructed using equation 2.9 and its eigenvalues and eigenvectors are computed. Real eigenvectors are tangent to the vector curves which end in the critical point while the eigenvalues define their characteristics.

$$\left[\frac{\delta(u,v,w)}{\delta(x,y,z)}\right] = \begin{bmatrix} \frac{\delta u}{\delta x} & \frac{\delta u}{\delta y} & \frac{\delta u}{\delta z} \\ \frac{\delta v}{\delta x} & \frac{\delta v}{\delta y} & \frac{\delta v}{\delta z} \\ \frac{\delta w}{\delta x} & \frac{\delta w}{\delta y} & \frac{\delta w}{\delta z} \end{bmatrix} \tag{2.9}$$

A real part of the eigenvalue represents the divergence of the critical point. A positive value means that the point is a sink and attracts vectors while negative value means it is a source and repels them. When this value is zero the point can either be a saddle point and both attracts and repels vectors at the same time or it is a center of circulation. This can be distinguished

using the imaginary part of the eigenvalue because it represents circulation or vorticity where the sign of the imaginary value indicates the direction of the rotation. A saddle point has zero vorticity therefore the imaginary part is also zero. When both parts of the eigenvalue are non-zero the vectors are attracted or repelled in a vortex-like shape also known as a focus. Sources and sinks with zero vorticity are sometimes referred to as nodes.

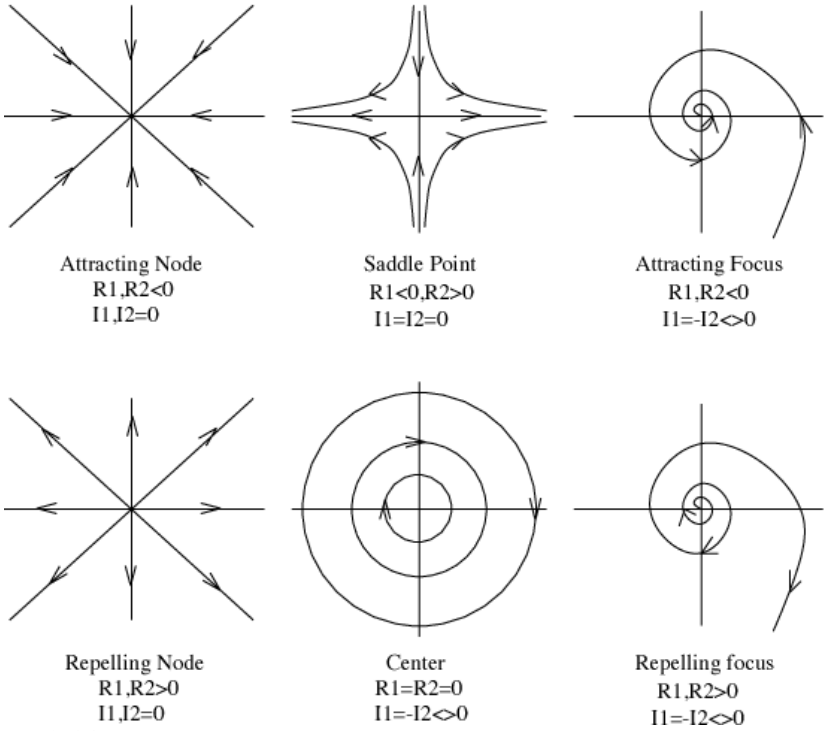


Figure 2.16: Different types of critical points with corresponding real and imaginary parts of eigenvalues of their respective Jacobi matrix [19].

The saddle points have an interesting property that only two tangent vectors run through them in case of 2D vector fields. Tracing these vectors as streamlines divides the plane into four different regions. The vectors from each neighboring pair of regions diverge at the saddle point. In addition the curve either reaches another critical point or it leaves the sampled area. Thanks to this property the topology of the vector field can be reconstructed by tracing these curves from each saddle point and visualising them together forming a graph. The nodes of this graph are the critical points except for centers of circulation which cannot be reached by any curve. The edges are the asymptotic lines running through each saddle point also known as separatrices. They divide the space into areas where all curves have similar properties.

The critical points become more complex in 3D vector fields. Each point can be a combination of the 2D cases for each of the three planes. Since there are three eigenvalues and eigenvectors each pair represents the behaviour of the critical point compared to a plane. For example in two planes the

point behaves as a saddle point while in the last one it is a node. In 3D the separatrices also need to become surfaces instead of curves to properly divide the space. It is possible to either visualise them only as lines connecting the saddle points and lose some information about exact borders of the regions or to visualise them as surfaces which occlude each other. None of these approaches is strictly better and an interactive approach allowing to switch between them seems to be the most useful solution. The critical points themselves can be visualised as 3D glyphs with different shapes based on the type of the critical point.

Each of the provided datasets contains only one critical point, the null point of the electromagnetic field. In all cases it is a saddle point so it can be used to generate separatrices and visualised as a topological skeleton. However, since there is only one critical point the topology will not provide any additional or unexpected insights into the vector field. Therefore, this approach is not considered to be of any use.

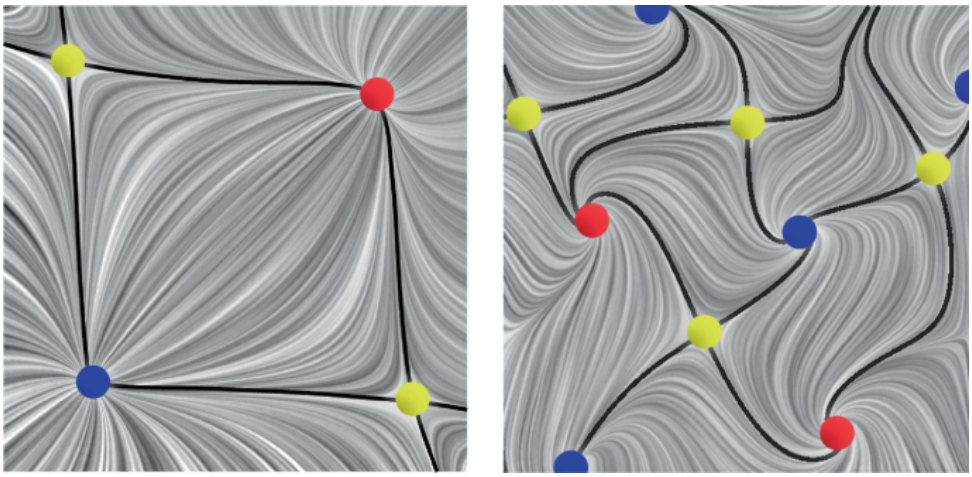


Figure 2.17: Two 2D vector fields with their topological skeletons overlaid on top. Different types of critical points are represented using different colors. Saddle points are yellow, sources red and sinks blue. [20].

2.8 Line integral convolution

Line integral convolution (LIC) is a dense vector visualisation technique which generates a texture corresponding to the vector field. First a white noise texture with preferred resolution is generated. Then, for each pixel of the texture, a streamline with a specified length is traced forward and backwards. White noise pixels, which the streamline overlaps, are summed according to a convolution kernel. It can be a box filter for a simple average or a more sophisticated one such as time varying periodic filters for animated visualisation. The result of this convolution is stored as the value of the output texture. The technique works because the convolution output is almost identical for adjacent pixels on the same streamline. Therefore, the resulting

color of pixels lying on this line is very similar. This creates visible patterns corresponding to the vector field. The output is a grayscale image and its hue channel can be used to map some other property. Cabral and Leedom [21] blend the LIC result with the vector magnitude mapped to color. The result shows a dense representation of both direction and magnitude.

It is possible to use this technique for 3D vector field visualisation in several ways. The simplest one is to take a slice of the vector field and calculate LIC for this specific 2D slice as seen in figure 2.18. This method has a lot of issues since the calculated streamlines for each pixels are 3D and need to be projected onto the slicing plane. The result correctly displays the orientation of the field in directions parallel to the plane, however, it completely discards all information about the distance of the streamline from the slicing plane and the angle between them. Multiple different vector fields can therefore produce identical slice which can be misleading.

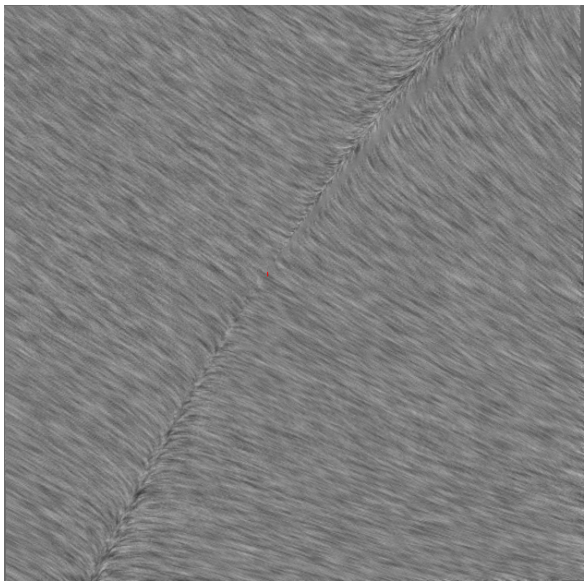


Figure 2.18: LIC slice of the magnetic field.

A better way is to generate a 3D white noise texture and trace a streamline for each voxel of this texture. Since the output of this method is a 3D texture it needs to be filtered before it can be drawn. The simpler method is to show only 2D slices of this 3D texture. Interrante and Grosch [22] propose a filtration technique which uses a masking function for selecting a specific region of interest. The masking is usually done using a scalar value of the field such as magnitude or divergence and selecting only voxels where this value is lower or higher than a specified constant. Masking the original white noise texture generally produced better results than masking the output. Even with masking it is still hard to see inside the visualised 3D object so a better approach is suggested. The original 3D white noise is replaced by a sparse set of points. Using 3D LIC on this sparse texture now produces output consisting of multiple streamlines which are equidistantly placed.

The black hole electromagnetic field does not have any obvious characteristics which can be successfully used as a masking function and 2D slicing can be misleading and does not provide any additional information compared to previously discussed techniques. Therefore, this method is not considered useful for this particular problem.

2.9 Visibility impeding halos

Visibility impeding halos as briefly described above serve to separate same color streamlines or other stream objects which overlap and blend together in the scene. They are not needed in 2D vector fields visualisations since the whole field is visible in the scene and nothing overlaps. While in 3D halos can greatly improve discriminability of the stream objects. In case of stream ribbons halos are easy to generate as simple black lines tracing the edges of the ribbons, which are the originally traced streamlines.

Their generation is more complex for actual 3D objects with volume such as stream tubes or 3D glyphs. In this case the halos do not have a fixed position and need to update in real time as the scene rotates. The edges are detected using any edge detection convolution filters such as the Sobel filter. However, this filter cannot be applied to the rendered image because there still are overlapping objects with similar colors where no edges can be detected. Instead it is used on the depth buffer to detect silhouette edges which form the halos. Nienhaus and Dollner [23] apply this approach in real time using depth buffer saved to a texture map and linearly filtered using multiple render passes. This operation is not computationally demanding since 3D rasterization already needs to calculate the depth values for each fragment. The filtering itself can be done for each pixel in parallel only needing a small amount of GPU computations. The same approach used with the normal buffer can detect crease edges on the surface of polygons, however, these are not needed for visualising the halos.

2.10 Stereoscopy

Two-dimensional screens have issues properly displaying depth. The human eyes use parallax to perceive depth and a flat screen always provides a very similar view to each eye. There are several techniques which alleviate this issue using varying degree of complex hardware and software solutions. Stereoscopy is one of the most common ways of displaying 3D scenes on a 2D display with improved depth perception. Stereoscopy as the name suggest uses two images rendered for the left and right eye separately.

One of the simplest approaches is called Anaglyph. The image needs to be rendered from two slightly different camera positions. Both of these renders are then overlapped over each other while the left eye image is filtered to remove blue and green and the right eye image is filtered to remove red. Then another physical red and cyan filter in the form of glasses is used to look at

the image. Thanks to these filters each eye only sees its respective rendered image. The brain combines both of these images into a 3D image. This approach works well for black and white images, however, due to the nature of filtering the red color is not visible, only blue and green.

Head mounted displays are another form of stereoscopy. This time the display is so close to the eyes that each eye can have its individual screen while the device blocks any other incoming light. Similarly as before the resulting image appears to be 3D which helps with depth perception.

2.11 Simultaneous visualisation of two vector fields

The electric and magnetic field need to be visualised simultaneously to properly show their relationship. There are several techniques which visualise co-located vector fields with varying levels of accuracy, discriminability and separability.

The simplest approach displays the vector fields side by side in two separate scenes. This method is easy to implement and works well both in 2D and 3D, if the visualisations in both scenes move and rotate together. While it does not introduce any additional clutter and both vector field are always clearly visible, it can be difficult to compare the fields in specific points. Additionally the side-by-side view limits the screen area reserved for the visualisation.

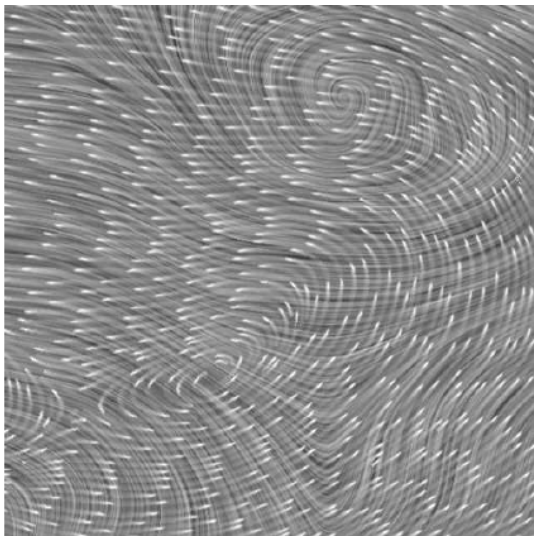


Figure 2.19: Two co-located vector fields overlaid on top of each other [24].

Another possibility is to freely switch between these two fields in the same scene. This approach also works well for both 2D and 3D vector fields and utilizes the screen area in a more efficient way. Comparison of the fields in a specific point can be easily done by focusing on that point and switching back and forth between the two fields. On the other hand, comparison of global features of these fields becomes more difficult and requires memorizing one of these fields.

The best spatial awareness can be achieved by visualising the vector fields overlaid on top of each other. However, these fields need to be visualised in a way which makes both of them visible and understandable. This task can be very difficult, especially for 3D vector fields. Urness, Interrante et al. [24] discuss various techniques for visualising co-located vector fields using texture based techniques such as LIC and sparse techniques such as glyphs and streamlines. While overlaying texture based approaches with glyphs or streamlines works well for 2D vector fields, as seen in Figure 2.19, it cannot be applied to 3D vector fields for the same reasons as with LIC alone. Therefore, the only option is overlaying glyphs or streamlines over each other. The prominence of these fields needs to be balanced so one does not overpower the other. At the same time, there should be high contrast between these representations to easily distinguish which line belongs to a certain field. It can be achieved using different colors or alpha values for each field. The occlusion issue already encountered when visualising a single 3D vector field is magnified. To prevent it the shapes of both vector fields need to be considered when selecting seed points. Real-time filtering of the visible 3D region can help as well.

Electromagnetic field has a useful quality, that electric and magnetic forces are always perpendicular to each other. This means that visualising both of them at the same time creates almost a grid-like pattern and the fields do not converge into each other. This is an ideal relationship for visualising two co-located 3D vector fields as the vectors never run in parallel. Still, a large area of the fields is hidden and needs to be visualised by making one or both of the vector fields transparent or low in density. The prototype rendering of electric and magnetic field using evenly spaced seeding can be seen in Figure 2.20.

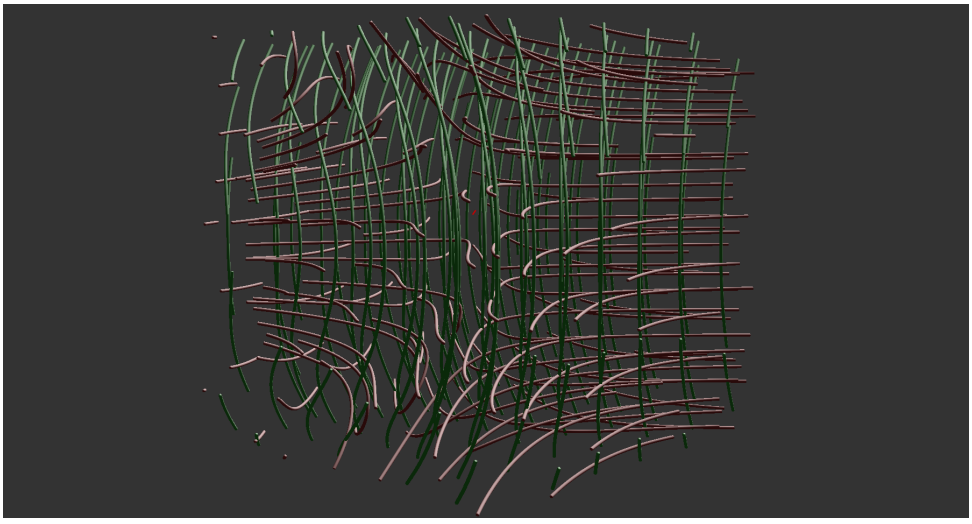


Figure 2.20: Electric (green) and magnetic(red) fields $d_{sep} = 60$ for both.

Chapter 3

Design

Multiple different techniques were analysed and the most suitable one is to use evenly-spaced shaded stream tubes which fill the whole scene as proposed by Mattausch, Theußl et al. Several different areas of interest can be defined using rectangular cuboids. Multiple resolutions of streamlines are generated in advance, which enables real-time swapping of streamline density. Each specified area can be assigned different streamline density and other additional properties. This approach allows the user to define one large region with low density as context and one or more smaller regions with higher density as areas of interest. In addition each area can visualise either electric or magnetic field. Therefore, using multiple areas both of these fields can be visualised at the same time, which makes their relationship more easily understandable. Visibility impeding halos are used to more easily distinguish overlapping streamlines. Several additional user interaction techniques are used to help with the exploration of the dataset.

3.1 Seeding and generation

Stream tubes seeding is done using the evenly-spaced seeding briefly described above. The exact steps can be seen in Algorithm 1. The only non-trivial step is finding points forming a regular hexagon perpendicular to the streamline in a certain point (see line 9). First tangent vector $\vec{t}$ is computed as a difference between the previous and the next point. Next an orthonormal basis $(\vec{b_1}, \vec{b_2}, \vec{t})$ is found from this tangent using Frisvad's method [25]. Finally the points forming the regular hexagon are computed using equation 3.1. Where p is the original point and θ is an angle which goes from 0 to 2π in 6 equidistant steps.

$$p(\theta) = p + d_{sep} \times (\vec{b_1} Cos(\theta) + \vec{b_2} Sin(\theta)) \tag{3.1}$$

Each point forming this hexagon is a candidate seed point c. Any seed point closer to an already existing point than the distance d_{sep} is immediately discarded. To reduce the amount of comparisons a regular 3D grid G with cells of size d_{sep} is constructed. Each point only needs to be compared to points in its own and neighboring cells. This is a significant speedup as in

most cases points are evenly spread among the cells. When a candidate seed does not have any point around it a streamline can be immediately traced and added to the queue. The next point from the original streamline is processed after that. The streamlines are traced immediately in order to add the points to the grid G and to affect the evaluation of the rest of the candidate points around the original streamline. In laminar and converging flows only the first set of candidate points actually generates streamlines the rest will always be too close to the newly generated neighboring streamlines. In diverging flows a new streamline is inserted between the neighboring streamlines when their distance reaches $2 \times d_{sep}$.

```
   Input: d_sep, d_test
   Output: array of streamlines S
 1 Q = queue
 2 G = empty grid with d_sep sized cells
 3 f = traced streamline with a random seed
 4 Place f into Q
 5 Place all points from f into G
 6 repeat
 7 |   Take streamline s from Q
 8 |   foreach point p in s do
 9 |   |   Find 6 candidate seeds C forming a regular hexagon
   |   |    perpendicular to s in p
10 |   |   foreach candidate seed c in C do
11 |   |   |   N = neighboring cells to c from G
12 |   |   |   if the distance between all points from N and c is greater
   |   |   |    than d_sep then
13 |   |   |   |   Trace a new streamline n forward and backwards from c
14 |   |   |   |   Stop tracing when any point in G is closer than d_test
15 |   |   |   |   Place n into Q
16 |   |   |   |   Place all points from n into G
17 |   |   |   end
18 |   |   end
19 |   end
20 |   Add s into S
21 until Q is empty;
```

Algorithm 1: Evenly-spaced streamline generation

This algorithm is sequential due to the fact that streamlines are traced immediately before processing additional points. In a way it is similar to a seed fill algorithm, it starts with one point and expands to evenly fill the whole space. Parallelizing this algorithm is problematic as the even distribution of the streamlines is dependent on the sequential processing. If four starting points, each with its own thread, are used, the algorithm generates four regions with evenly placed streamlines. However, the borders where these regions meet are noticeable as the gap between them is somewhere between d_{sep} and

$2 \times d_{sep}$ because the regions cannot align perfectly. Another possibility is to start with only one streamline and split the space based on this streamline. Each part then gets assigned its own thread. This might work in 2D because streamlines cannot cross each other. There is no such guarantee in 3D as streamlines can rotate around each other and are impossible to separate into disjoint sets without additional computations.

Streamlines themselves are traced using second order Runge-Kutta method with constant Δt. The magnitude is saved in each traced point to be used later for magnitude mapping. The tracing stops whenever a new point is closer to any existing point than the distance d_{test} which is described as a percentage of the separating distance d_{sep}. When all streamlines are traced they are converted into solid 3D objects (stream tubes) so they can be more easily used by graphics frameworks. In each point of the generated streamline n new points are created forming a perpendicular circle around this point. This is done using the same method as for generating the regular hexagon. These points are connected by triangles to form tubes. Each point is also assigned a normal facing outwards from the tube so they can be smoothly shaded.

In some cases an extremely short streamline can be generated. Streamlines consisting of one or two points need to be discarded as they are problematic to visualise. However, the points forming these short streamlines still need to be added to the grid G, otherwise, they would be generated and deleted over and over infinitely looping.

After generating a single resolution the next one is generated with current streamlines already stored in the queue and the grid. The separating distance d_{sep} needs to be selected with regard to the separating distance of the already generated resolution. All of these resolutions are stored independently, only storing additionally generated streamlines to save space. Any resolution can be shown by displaying it and all resolutions levels above it.

3.2 Lighting

Ambient, Diffuse and Specular Phong shading is used to improve orientation readability. The lighting is done using three lights as described by Birn [26]. This technique originally comes from actual studio lighting but can be as easily applied to digital rendering. In order to enhance the shapes of illuminated objects three light sources are used.

The key light is the main light providing the most illumination. The direction of the key light defines the dominant angle of the lighting. Usually it is placed to the side of the camera and slightly above it. The key light can sometimes also be thought of as natural sunlight.

The fill light simulates indirect illumination or secondary light sources and is usually dimmer than the key light and placed on the other side of the camera. If the key light is above the object the fill light should be slightly lower. Birn recommends to have a slight overlap in key and fill light to make sure that no discontinuities appear on the illuminated object. It is also

recommended to avoid unnatural symmetry and do not place the fill light to a mirrored position of the key light. The intensity of the fill light compared to the key light is called the key-to-fill ratio and defines the contrast between the brightly lit areas and dimly lit ones.

The back light separates the object from the background. It is placed at the back of the scene facing the camera. It only lights the edges of the objects which are almost facing away from the camera. This creates a brightly lit rim of the objects which makes them visually stand out. In a way this effect can be similar to visibility impeding halos as only the edges of the objects are lit. This effect is not always desirable which makes the back light an optional addition. Birn also notes that representing the back light as only one light during rendering can pose problems and it is often instead simulated as a group of lights close to each other.

Directional lights are used as there is no need for point or area lights. This way the shading is more consistent and depth is conveyed by other means than distance from the light. By default the key light is coming from the upper left side of the camera while the fill light is coming from the right side and is slightly lower. The back light comes from the front of the camera and points slightly below it. The direction, intensity and color of the lights are adjustable.

3.3 Visibility impeding halos

Visibility impeding halos are added using the depth buffer. First the depth buffer is stored to a texture. In order to only get the depth of fragments which represent the actual stream tubes any other 3D elements need to be hidden. Next for each pixel of the depth texture the neighboring pixels are examined based on the separability distance. The maximal difference between these pixels is found and compared to a constant. If it is higher the pixel is colored black otherwise it stays transparent. The depth values are not linear, making the differences closer to the camera greater. This can be a good thing as this effect is most needed close to the camera. The fact that the halos are found using a 2D filtering method leads to all black lines being the same width. This can make distant streamlines look thicker than they actually are. The solution is to use non-linear filtering which changes the filter size based on the depth of the fragment. The filter is largest near the camera and gets smaller with increasing depth. However, the thinnest possible halos work the best to enhance the edges without obscuring the scene too much. If one pixel wide halos are drawn near the camera the halos away from it cannot be thinner. In practice the halos are two pixel wide as both the pixel on the edge of the tube and on the edge of the background are coloured. This can be fixed by discarding either the background fragments or the tubes fragments.

3.4 Interface and interaction

The whole windows is separated into two parts, the left area is reserved for user interface elements while the right contains the rendered scene. The generated scene can be interacted with by zooming, rotating and panning. By default the camera is always facing the null point of the vector field so it is easy to zoom into it and the whole scene rotates around it. The initial UI design can be seen in Figure 3.1.

Multiple areas can be defined by axis aligned rectangles which are visible as white wire-frame outlines. Their dimensions can be adjusted in real time affecting the content. Each of these areas is independent and has additional properties which can be turned on and off. First each box can be hidden or shown based on a checkbox. The resolution in each of the boxes can be changed using a slider or a dynamic resolution can be selected which updates the resolution based on the distance from the camera. By default each area contains magnetic field lines, these can be individually swapped to electric field using a checkbox. Visibility impeding halos can be turned on and off for each area individually using another checkbox. Transparency can be turned on either using a flat transparency slider or by enabling mapping depth to transparency with a switch. The last checkbox can hide the bounding box of each area. This box serves as a visual aid to help the user when they change the dimensions of the area. However, it can be distracting during exploration of the rendered scene. The color mapping can be changed using a switch, either a flat color or vector magnitude mapped to a linear scale can be selected. If flat color is selected it can be changed using red, green and blue sliders. Thickness is a global property of all areas which can be adjusted in real time using a slider.

Three additional buttons are at the top of the UI. These are used for saving the state of the visualisation, loading an already saved state or regenerating the streamlines. Saved state remembers the dimensions and properties of each area. It also remembers the already generated streamlines which only need to be converted to 3D objects. This significantly speeds up the time it takes to start the visualisation.

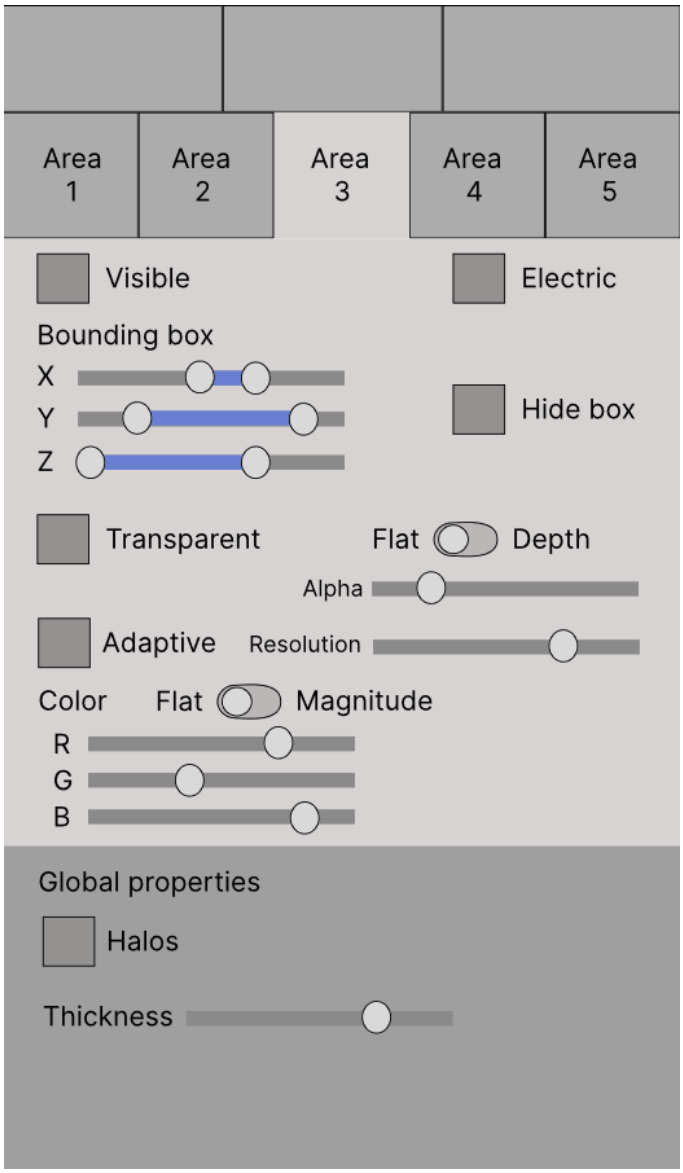


Figure 3.1: Initial design of the user interface. Lighter region contains properties of each filtered area. Darker region contains global properties.

Chapter 4

Implementation

The implementation was written in Python 3.10.6 using Panda 3D 1.10.12, an open-source framework for 3D rendering, because it is simple to prototype with. It is separated into two python scripts *gui.py* and *generator.py*. *Gui.py* contains all rendering and interaction functions which need Panda3D to work, while *generator.py* calculates streamlines and other stream objects using different implemented seeding strategies. These streamlines are simple python lists containing pairs of 3D points and corresponding vector field magnitudes in these points. This way the streamlines data is not tied to any specific technology and can be visualised using any framework or library if the need arises.

4.1 Stream tubes

The visualisation itself is done in multiple steps. First, streamlines are seeded and traced using the function $findStreamlines$. By default this function uses equidistant seeding using the algorithm 1 already described above but it can also place the seeds randomly, on a regular grid or around a point. The vector in each traced point is calculated using trilinear interpolation in functions $findPoint$ and $trilinearInterpolation$. First the position of the point is rounded down to find the integer indices and the offset which defines the interpolation, then for each component x, y and z interpolate the corresponding 3D grid of vector components. The streamline is first traced forward and then backward, once both parts are done the backwards part is reversed and prepended to the forward part, this way a continuous streamline is formed. The starting seedpoint needs to be removed from one of these parts, otherwise it would be represented twice. Before the streamline gets added to the resulting list it is also simplified by the $simplyfyLine$ function. It calculates the angle in each point of the streamline and removes points which do not contribute to the shape of the streamlines. The points are removed dynamically during the simplification in order to correctly compute the angle of the next point.

If evenly spaced seeding is selected the algorithm first constructs a 3D python list with cell size based on the separating distance d_{sep}. Each cell of this list can contain any number of points. The lower the test distance d_{test}

is the more points the cells usually contain. The function $findNextSeed$ takes not yet processed queue of streamlines, the 3D grid and both distances. It calculates candidate points around each point of the streamline and calculates the distance to points from nearby cells. It also needs to remember the position in the currently processed streamline so it can return to it when another seed point needs to be found. When another level of resolution is traced the algorithm takes the already traced streamlines and puts them all into the queue. It also needs to create a new grid with lower resolution and place all points from the original grid to the new one.

When all streamlines are traced, they are passed to a visualisation function which converts them to 3D tubes. Panda3D supports procedural geometry generation using $GeomVertexFormat$ which specified the required parameters of each vertex. In this case two three dimensional vectors are used for vertex position and vertex normal and one four dimensional vector is used for vertex color. The colour of each vertex corresponds to the vector field magnitude in that point. It is obtained from a linear color scale using the function *speedColor*. An additional $GeomVertexFormat$ containing only vertex position is used for real time animation of the stream tube thickness. It contains the same amount of points as the other one and each point is very close to the original streamline point.

Input: number of cross section points n, streamlines S
Output: triangles T
$cnt = 0$
foreach s in S **do**
  foreach point in s except the last one **do**
    for *i from 0 to n-2* **do**
      Add triangle (cnt, cnt + 1, cnt + n + 1) into T
      Add triangle (cnt, cnt + n + 1, cnt + n) into T
      cnt += 1
    end
    Add triangle (cnt, cnt - (n - 1), cnt + 1) into T
    Add triangle (cnt, cnt + 1, cnt + n) into T
    cnt += 1
  end
  cnt += n
end

Algorithm 2: Create triangles to form 3D tubes.

Vertices are generated based on two parameters, number of cross section points (how many points are generated on the perpendicular circle) and maximal thickness of the tube. Once the vertex array is filled with data, primitives connecting the vertices need to be specified. Triangles are chosen using $GeomTriangles$ and specified using indices of previously created vertex array using algorithm 2. Generated tubes with unwanted resolution can be simply not rendered using the Panda3D *hide*() function and quickly rendered when requested using the *show*() function.

4.2 Shaders

Per vertex Phong shading is implemented using GLSL shaders *shader.vert* and *shader.frag*. A key, fill and back lights are used as described in section 3.2. Each of these lights is specified in a configuration file *lights.config*. Each light is a directional light and is described by its color and direction. The color vector serves as an intensity as well so its components can be higher than one. The direction is in camera space and is loaded straight into the fragment shader. This means that the lights rotate with the camera so the tubes with the same direction relative to the screen always have the same shading. In addition the vertex shader also handles the clipping of the stream tubes based on the defined bounding box of each area. This is done by discarding vertices which lie outside of the specified area. While this is a very fast approach it also leaves a jagged edges on the boundary where whole triangles disappear. Another slower but smoother way is to perform this filtering in the fragment shader. Clipping planes are defined as the boundaries of the filtered area and each fragment outside of these planes is discarded. Each area can have independent boundaries as they are sent to the vertex shader as a custom input. Another input of the fragment shader is a true or false flag which defines whether the depth of each fragment should be mapped to transparency. If flat transparency is instead selected it only changes the alpha value of each fragment.

Another set of shaders is used for rendering visibility impeding halos. Another Panda3D camera needs to be specified in order to render the contents of the scene in a separate buffer instead of on the screen. The first pair of GLSL shaders *depth.vert* and *depth.frag* is used to render the depth of each fragment into the buffer. These shaders are applied to a dummy *PandaNode* so all objects in the scene can be taken into account. It is important to mask the box objects using a bitmask which is applied to this camera only. This way their depth is not rendered into the buffer and no distracting black outline appears around them. Another issue arises from the stream tube objects already having a shader. This is fixed by assigning a higher priority to the depth shader. It overwrites the lighting shader only for this camera and the depth information can be extracted. The last shader *halos.sha* is a combined Cg shader as it was taken from a panda3D example cartoon shader and modified. It loads the generated depth buffer and finds the edges which are assigned black color. Finally a 2D rectangle is placed in front of the camera with the buffer as its texture.

This approach has an issue with rendering halos in multiple areas simultaneously as only one clipping boundary can be specified in the shader *depth.vert*. A possible workaround applies this filter independently to each box with their own clipping boundaries. However, this approach runs into another issue with the halos being rendered into a 2D quad in front of the camera. This way tubes from one area hidden by tubes from the other one still produce halos which do not make any sense on the screen as their tubes are obscured. Therefore, this solution does not work.

To actually solve this issue a *setShaderInput* property of Panda3D nodes needs to be utilized in a clever way. This property specifies a custom named input of a shader assigned to this node. It is only tied to a specific node regardless of its shader and persists when the shader is swapped or removed. This means that using a shader input with the same name in both *depth.vert* and *shader.vert*, which specifies the filtered boundary, fixes this issue. The dummy node with the depth shader takes all non-masked nodes in the scene and applies this shader to them. Each of these nodes remembers the original shader input and applies it to this shader as it has the same input name. The resulting halos cover all visible stream tubes, even with disjoint areas, without covering any other objects such as the boxes.

4.3 Interaction

Camera control is done in spherical coordinates, using elevation angle, azimuth angle and distance from the center. Center of rotation and zooming is set to be the null point. Camera panning moves the center of zooming in the direction opposite to the direction of mouse movement. Initially the direction is in 2D screen coordinates and needs to be converted to 3D world coordinates. This is done by finding a plane perpendicular to the current camera direction and placing the 2D vector onto that plane. Finally the center of rotation and in turn the camera itself are both translated by this 3D vector. The left mouse button is used for camera rotation, the mouse wheel for zooming and the right mouse button for panning.

The division of the screen into the right side 3D area reserved for stream tubes and the left side 2D area reserved for UI is done using *displayRegions*. The 3D *displayRegion* has its own camera which inherits from the main render camera, this means that anything attached to the Panda3D render node is rendered only in the right window scaled based on the region size. The aspect ratio of this area needs to be manually updated every time the screen changes otherwise the content is unnaturally stretched.

There are five areas which can be used to visualise the vector field and interact with it. Each area is represented using a dictionary with following properties[1]:

- visible – whether the area is rendered
- halos – whether visibility impeding halos are rendered in the area
- transparent – whether to apply transparency to this area
- depth – whether to map depth to transparency or select it using a slider
- alpha – float from 0.0 to 1.0, defines the amount of flat transparency
- boundary – three pairs of integers, defining the boundaries of the area

[1]These properties are boolean unless otherwise specified.

- box – whether to draw simple white box around the area
- resolution – integer, defining the resolution of tubes inside the area
- electric – whether to render electric field instead of the magnetic field
- adaptive – whether adaptive density is turned on
- color – four dimensional vector, the flat color of the area
- mapping – string, defining what is mapped onto the color channel

These properties are mostly saved only for generating save file of the visualisation state and for correctly setting the values of UI elements when a different area is selected by the user. The actual effects which these properties represent happen immediately when interacting with the UI and this dictionary is not even needed for them. Another list contains the generated stream tubes objects or in case of higher resolutions a list of stream tubes object forming the selected resolution. These are complex Panda3D objects which are harder to save using pickle. When loading a state the object for each area needs to be reconstructed based on the loaded properties.

The graphical user interface which controls the areas and other aspects of the visualisation is realized using the Panda3D *DirectGUI* module. The positioning of these elements is not simple as their origin and size changes based on their text.

Five *DirectButton* buttons at the top of the UI change the currently selected area. The selected button has the same color as the background of all area related UI elements to visually connect the area button to everything that affects the area. Not selected buttons have different color and are clickable while the selected button is disabled. Selecting a different area also changes the value of all UI elements to the values of the newly selected area. These buttons themselves do not change anything inside the actual 3D region.

Other *DirectButtons* are the save, load and generate buttons. There are three save slots which can be used by any of the three datasets. As no file dialog has been implemented, keeping more than these three saves needs to be done manually. When the application is launched for the first time only load and generate buttons are clickable. The generate button brings up new dialog consisting of three buttons each representing one of the three provided datasets. The selected dataset is then generated and visualised. As this is a lengthy process a *DirectWaitBar* represents its completion.

The generation and loading run as a Panda3D task in their own thread which means that the process is not blocking the main render thread and the window can be interacted with. The magnetic field stream tubes are immediately visualised once they are generated and can be interacted with before the electric stream tubes are generated. The button which swaps the area to visualise electric field only appears once all electric field resolutions are generated. Simultaneous generation of electric and magnetic lines cannot provide any significant speedup because of Python's global interpreter lock which only allows one python thread to run at a given time. The most

computationally demanding operation is the streamline generation itself. The conversion into 3D object is significantly faster and saved files can contain visualised electric field data. Therefore, both magnetic and electric lines are loaded together in a single Panda3D task.

The area boundary, transparency, resolution, RGB and thickness sliders are generated using the *DirectSlider* element. The resolution slider is a special case, as it is supposed to be an integer only slider with four steps. However, *DirectSlider* does not allow for integer sliders so the value needs to be rounded to the nearest integer. Additionally to better represent the selected value the handle is moved to the rounded position. However, a small issue with Panda3D sometimes prevents the handle to be moved in the code if the mouse is hovering over it. This means that sometimes the handle stays between integer positions even though the actual value of the slider is an integer.

The boundary slider has range from negative 260 to positive 260. This is slightly higher than the 512 range of the vector field since the points on the edge of the field can be transformed into tubes partly lying outside of it. It has negative and positive values to be easily applied to filter the area as the center of the vector field is located in the point $[0, 0, 0]$. The red, green, blue, alpha and thickness sliders all have their range from 0 to 1. These sliders are hidden when other color mapping than flat color is selected and when transparency is not selected.

There are multiple *DirectCheckButton* checkboxes which toggle visibility, halos, transparency, electric field, rendering of area boundary and adaptive resolution. The halos checkbox is unique since, as discussed above, it is a global property for all areas simultaneously. The last two elements are *DirectRadioButton* radio buttons. One pair of them allows the selection of color channel mapping, flat color and magnitude mapping can be chosen. The other pair is for selecting the used transparency method, either flat transparency or depth mapped to transparency.

The box around each area is formed by a simple white line segments. Each area has its own box and multiple can be rendered at the same time. A minor issue arises when using halos and boxes at the same time as the halos are a 2D layer in front of everything else and ignore the boxes. The resulting halos overlap the boxes in areas with stream tubes behind the boundary lines. The null point itself is drawn as a small shaded red sphere so the user can easily locate it. Adaptive density checks the distance to the center of the dataset every time the zoom changes. If it passes certain thresholds the density of the visualisation changes on its own by swapping to a different resolution. These thresholds are consistent and always change the resolution at the same distance.

Chapter 5

Results

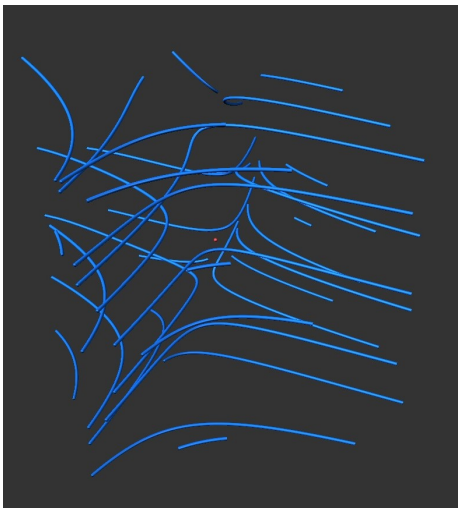


Figure 5.1: Separating distance 120.

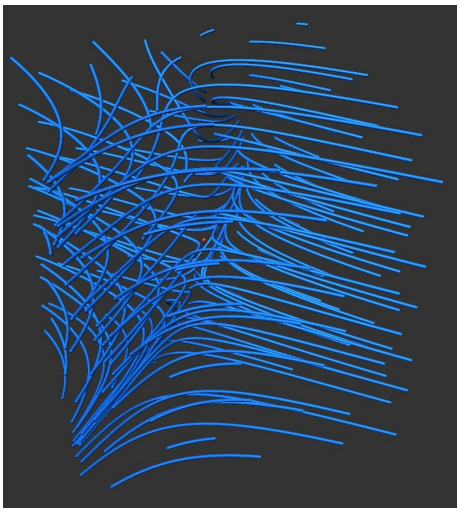


Figure 5.2: Separating distance 60.

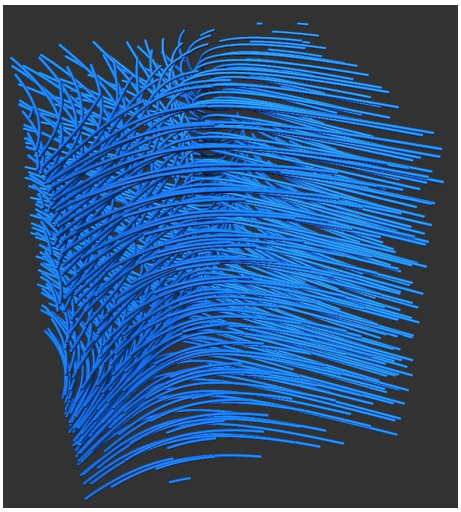


Figure 5.3: Separating distance 30.

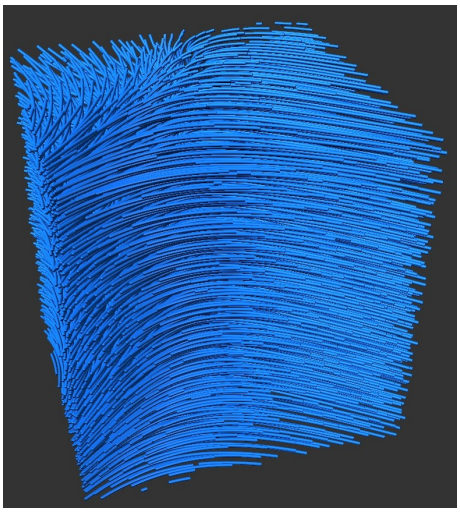


Figure 5.4: Separating distance 15.

The resulting visualisation has four levels of stream tubes density. Multiple different separating distances were tested. Lower values provide more detail which can be filtered and visualised while also taking much longer to generate. When the separating distance is halved, the amount of streamlines does not increase two times but closer to four times. The average numbers of traced streamlines can be seen in Table 5.1. It is interesting to note that, except for dataset 1, the electric field consistently generates significantly lower amount of streamlines, because the electric field is similar to a laminar flow and does not converge anywhere. This means that longer parallel streamlines are generated, which fill the whole space using lower amount of lines. The tracing of electric lines also runs faster as most of these lines run out of bounds before they can be stopped by the testing distance. Therefore, the space is not as filled as when generating the magnetic field streamlines.

In the first dataset both the electric and magnetic field behave mostly like a laminar flow. Thus, there is not a large difference in the number of generated streamlines. The actual amount of streamlines changes when the same dataset is regenerated. However, it only depends on the selection of the first seed point, the rest of the generation is deterministic. Nevertheless, the difference between the amount of generated streamlines can be up to 6% just by changing this initial seed.

The final separating distance values to generate four different resolutions are 120, 60, 30 and 15 with testing distance 0.2, 0.2, 0.3 and 0.5. The testing distance increases as the separating distance decreases to avoid intersection of converging streamlines when they are converted into 3D objects. The resulting rendered resolutions can be seen in Figures 5.1, 5.2, 5.3 and 5.4 respectively, where the same dataset is viewed from the same angle. It is obvious that the occlusion quickly becomes a major issue starting from $d_{sep} = 30$ and going as far as only the surface being visible when setting $d_{sep} = 15$. Both magnetic and electric field generation uses these separating distances.

	d_{sep}			
	120	60	30	15
Magnetic field 1	20.8	58.2	256.4	1102
Electric field 1	21	68.8	295	1273
Magnetic field 2	31.8	100.2	389	1517.4
Electric field 2	24.4	69.8	276	1099.4
Magnetic field 3	38.4	98.6	359	1635.4
Electric field 3	27.8	84.2	335	1301.6

Table 5.1: Average number of generated streamlines for each separating distance. Calculated from five generations for each dataset.

5.1 Datasets

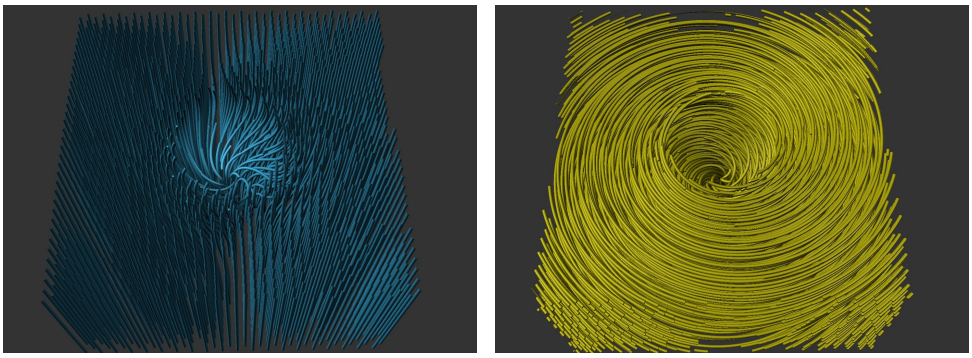


Figure 5.5: Visualisation of the first dataset, only the bottom half is visible. Magnetic field is blue, electric field yellow. The empty area in the center of both images is the location of the black hole.

Three datasets were provided, however, only two are simulations of an electromagnetic field which contains a null point. On the other hand, the last dataset contains NaN values which represent the black hole itself. The results from second and third datasets are similar in nature especially when compared to the first dataset which has its own unique properties. The main difference is that when generating streamlines from the first dataset the NaN values need to be ignored which creates a large hole in the middle of the scene. The resulting stream tube visualisation can be seen in Figure 5.5. Both the electric and magnetic field generates streamlines which are mostly parallel except for the area around the black hole itself. The streamlines rotate around the black hole as they avoid it. It is interesting to note that this dataset is vertically symmetrical with the axis of symmetry being the x, y plane exactly in the middle of the z axis.

The second and third dataset have similar properties and can be mostly described together. They both contain a null point close to the center of the magnetic field. Topologically these null points are saddle points and generated streamlines never pass through them. The separatrices of these points serve as asymptotes where the streamlines converge. The second dataset has the highest difference between the amount of generated electric and magnetic streamlines. This can be explained by looking at Figure 5.6 where both of these fields are side by side. The electric streamlines are curved and on average very long, which means that one streamline fills a significant amount of 3D space. On the other hand, the magnetic field streamlines converge which makes them shorter and more numerous. One of the magnetic field separatrices is curved in the vertical axis and the the electric field mirrors this curve. The third dataset can be seen in Figure 5.7. Again, the electric field stream lines are longer, however, the whole electric field turns twice and additional shorter streamlines need to be generated to fill the gaps. The magnetic field is diagonally divided by a notable null point separatrice.

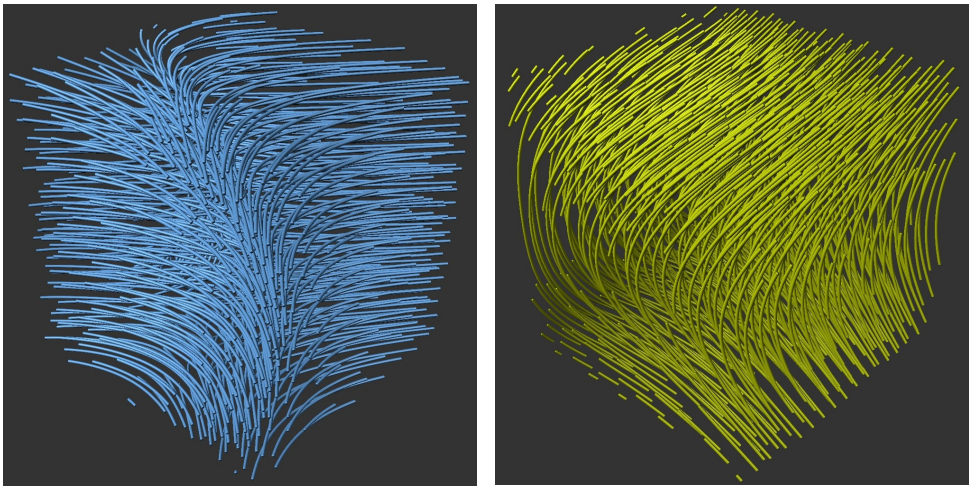


Figure 5.6: Visualisation of the second dataset. Magnetic field is blue, electric field yellow. Long electric streamlines curve around the null point separatrice.

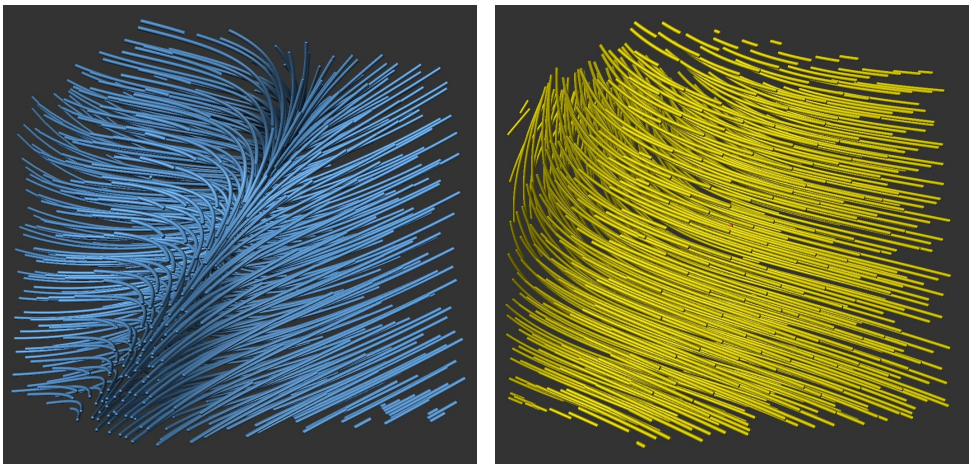


Figure 5.7: Visualisation of the third dataset. Magnetic field is blue, electric field yellow. A separatrice runs diagonally through the magnetic field.

5.2 Stream tubes

The stream tubes themselves are shaded using three lights with diffuse and specular lighting. The back light does not provide specular lighting as the steep angle of the light direction towards the camera creates unnatural artifacts. The default lighting works best with darker colors as the lights are bright to work well with selected color scales. Figure 5.8 shows two example stream tubes where the effect of shading on perception of shapes is significant. The wider shadow at the right side of the tubes and specular highlights indicate that the tubes are bending upwards almost running parallel with the screen and then again sharply bending away from the screen. The diffuse shading itself indicates the shape of the tube, while the specular highlights can help identifying regions with changes in the direction of the tubes.

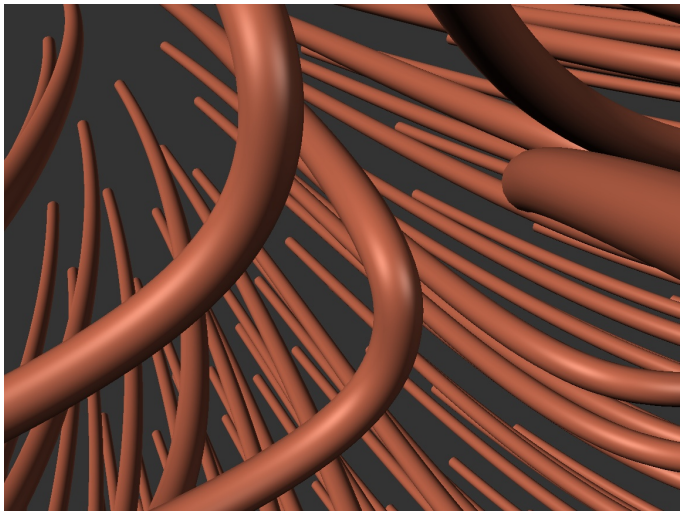


Figure 5.8: Stream tubes where both diffuse and specular light help to identify the 3D orientation.

5.3 Magnitude mapping

The intensity of both magnetic and electric fields can be seen by mapping it onto the color channel of the tubes using a linear color scale. A different color scale is used for each vector field as can be seen in Figures 5.9 and 5.10. Both visualise an identical view of the same dataset. The lowest values are unsurprisingly around the null point, but also along one of the separatrices running from the saddle point. The other areas have mostly constant magnitude lower than the field's median magnitude. The highest intensity appears as a bright area in both vector fields as the color scales are linear both in color and brightness. This means that the mapping is perceptually uniform even in a grayscale picture or for users with color vision deficiency.

As expected, both the electric and magnetic fields have their maxima in the same region and the intensity of both of these fields increases at the same pace. The color scale label for each color map is visible at the left and right side of the screen respectively. The minimum and maximum vector value is visible at the bottom and top of each label. The magnetic field minimum is very close to zero which makes sense as it approach zero in the null point. There is only one label for all resolutions as the difference between extrema of different resolution is insignificant. Filtering, on the other hand, changes the extrema present in the scene significantly. Currently the vector intensity mapping is not updated whenever an area is filtered, hidden or shown. This feature is desirable and can be included in future improvements, because it improves the separability of intensity mapping for regions which look similar with the global mapping.

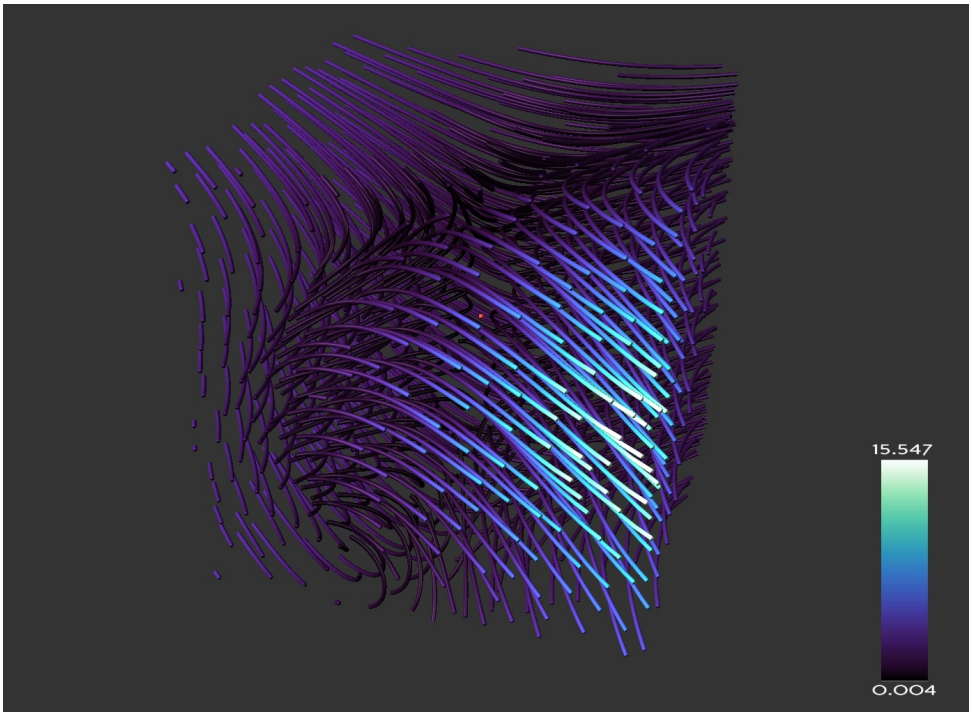


Figure 5.9: Magnetic field, vector magnitude is mapped onto a linear color scale. Null point can be seen as a red dot.

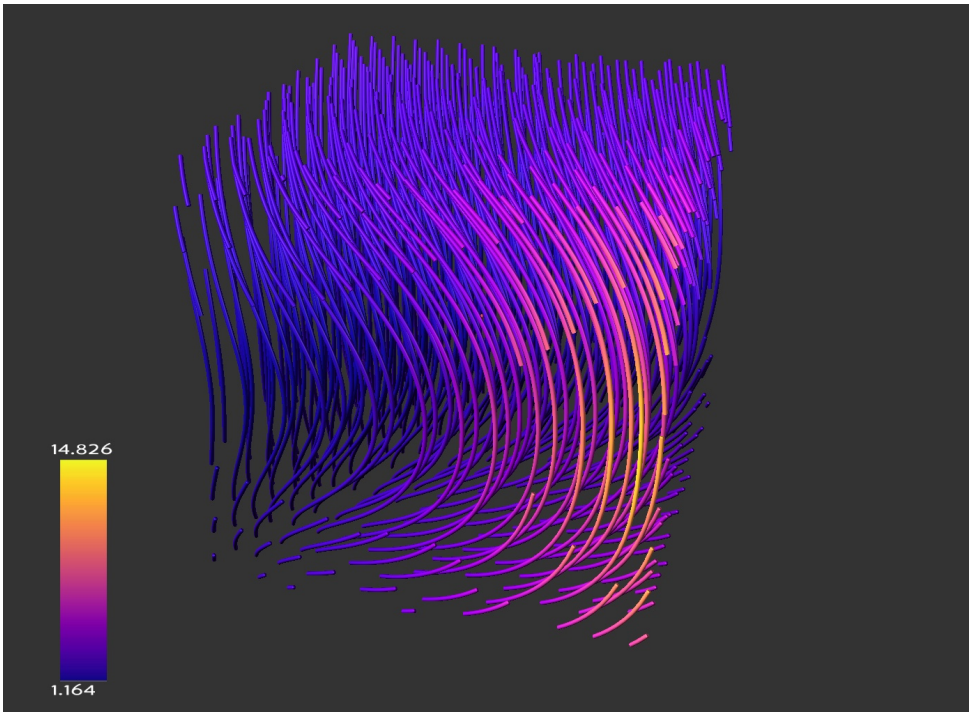


Figure 5.10: Electric field, vector magnitude is mapped onto a linear color scale.

5.4 Filtering

In order to avoid occlusion and cluttering both electric and magnetic fields can be filtered using cuboid regions. Figure 5.11 shows the magnetic field with high streamline density filtered only around the null point and viewed from top. The white boundary lines help with specifying the filtered area and can be turned off. The shape of a saddle point is clearly seen in the filtered magnetic field. The shape is especially visible as the streamlines barely change in the vertical direction and are aligned with the filtered box.

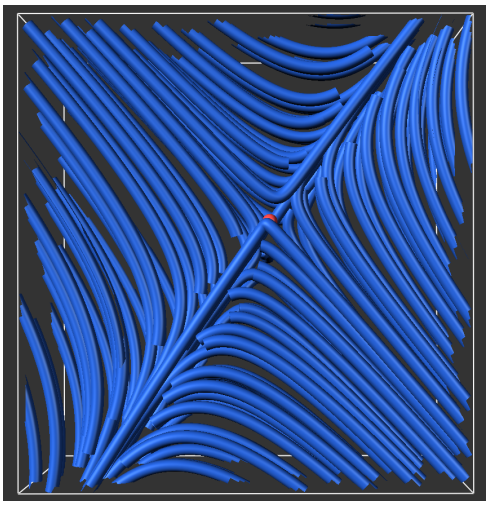


Figure 5.11: Filtered area around the magnetic field null point.

Multiple filtered regions can be visualised at the same time. Each of them with their own boundaries, streamline density, color, selected vector field and more. An example of two regions with different density can be seen in Figure 5.12. In addition to the null point filter from previous paragraph a global region with low density is added. The red streamlines with lower density serve as a context of the whole vector field, while the blue denser streamlines are the main focus. As the bigger region contains the smaller one some of the streamlines pass through both of them. Those streamlines change from red to blue once they pass the boundary of the smaller region and back to red once they leave it. It is because the resolutions are copied from all previous streamlines with different separating distances. Once they are copied they overlap each other in the scene and only one color is visible. Having multiple objects exactly in the same space does not create any artifacts, incorrect shading or flickering. The only issue is with the visible color in each region. Usually, the last region interacted with has all of its stream lines visible with the color specified for this region. Therefore, controlling the visible color is not too complex with two overlapping regions as the color does not change on its own when rotating or zooming. However, the color of stream tubes simultaneously inside three or more regions is very difficult to control and either magnitude mapping or transparency should be used.

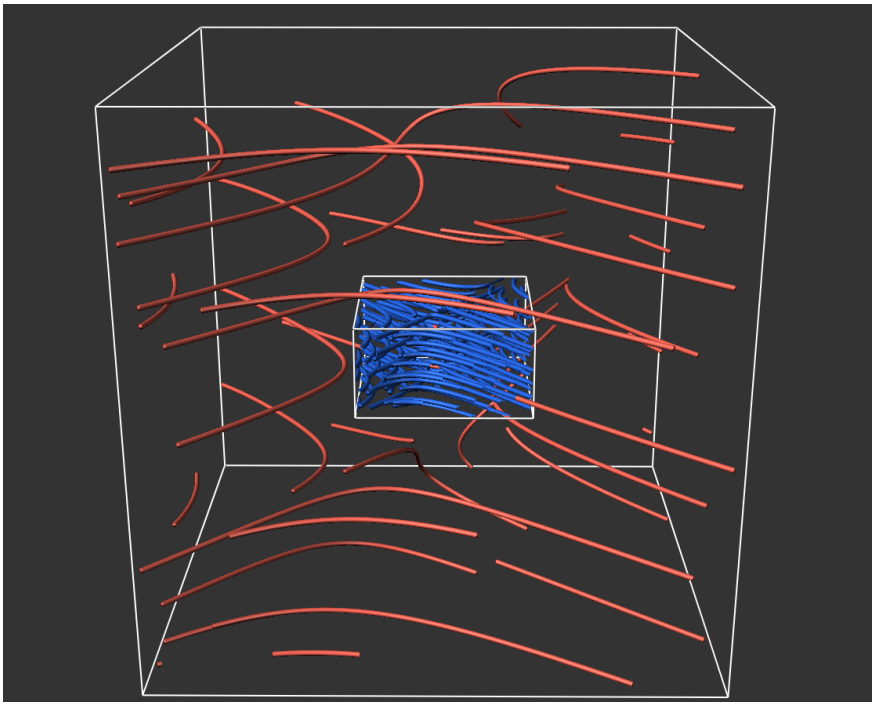


Figure 5.12: Two filtered areas with magnetic stream lines. One with high density around the null point. Other with low density in the whole scene.

Another possibility of adding context to a filtered area is to use dense but transparent region. Two different options are visible in Figure 5.13. The first one keeps the smaller area of interest opaque while adding a transparent context area. The smaller area is still perfectly visible and can be explored while the transparent area provides context. A small issue is the potential loss of depth perception of the small area. It is not clear where exactly it is located in the scene as it is much stronger than any transparent lines in front of it. The other approach makes both of the areas transparent. This makes the depth more easily understandable at the cost of general visibility of the focused area. Same as with color before, the overlapping transparent objects can have issues with correctly blending together. When the streamlines visible on top of other ones lower their opacity the ones hidden beneath them emerge, however, as they may be discarded during the depth test only the background color gets drawn. The object which is being covered blends well with the object which is over it, but not the other way around.

The filtering can also be used similarly to slicing in 3D scalar fields. A narrow region of the field is filtered out and treated similarly to a 2D vector field. When viewed from the top the planar directions of the vectors can be seen. The vertical directions can be seen when viewed from the side, depending on the thickness of the slice and nature of the field. Even thin slices can provide vertical information as can be seen in the Figure 5.14. The shape of the vector field is clearly visible in the top view without any clutter. The side view shows that on one side of the area the vectors are turning upwards.

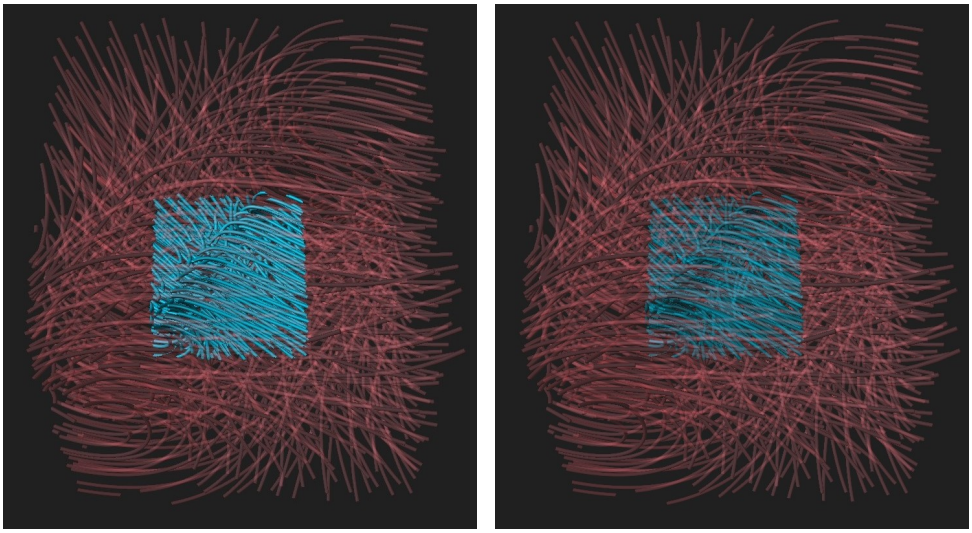


Figure 5.13: Using dense transparent lines as context. Left – only context is transparent. Right – both focus and context are transparent.

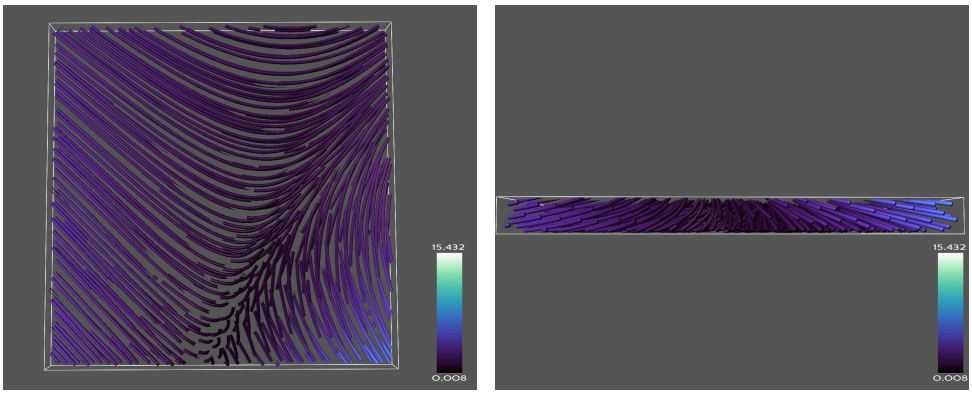


Figure 5.14: A filtered narrow region of the magnetic field viewed from the top and from the side. The planar and vertical directions of vectors are visible.

Another use for the filtering is exploring the shape of the field around a point using three areas which all touch at the point. This way the behaviour in each axis can be visible without any occlusion. The result can be seen in Figure 5.15. The bottom area clearly showcases the divergence of vectors around the null point while the left area shows the curve of the separatrice mapped in darker color as the vectors there have low intensity. In this case the white bounding boxes actually help with 3D scene orientation as they establish the planes around the null point. This approach can be used only with two areas with similar effect.

There are many more uses for the filtering especially when visualising both fields at the same time. For example, more than one focus area can be specified to analyze several features of the field. Another use for the area filtering is to create multiple regions with different parameters and then quickly cycle through them without actually visualising them simultaneously.

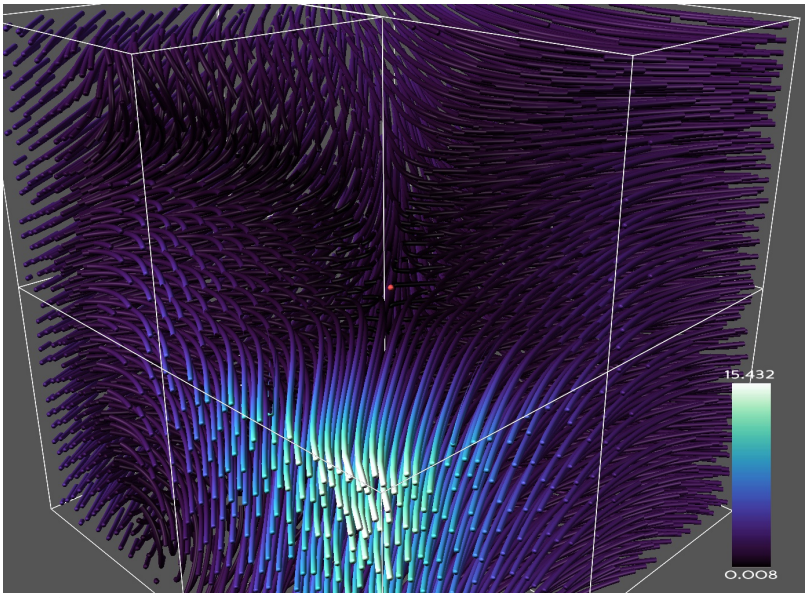


Figure 5.15: Three areas covering 7 out of the 8 octants of the scene. The areas meet at the null point.

5.5 Simultaneous visualisation of both vector fields

The electric and magnetic field are perpendicular to each other and can be visualised simultaneously, without needing any complex filtering, just by using reasonable density. The simplest approach uses two contrasting colors to distinguish each vector field. Figure 5.16 shows this visualisation where magnetic lines are blue and electric are red. The perpendicular relationship between these fields is clearly visible. The electric field even mirrors the curve of the magnetic field separatrice. The streamlines resolution level is second out of four as anything more clutters the screen too much and only surface of the domain is visible. A slightly different approach can treat one of the fields, in this case the electric, as context and visualise it transparently while the magnetic field remains opaque as the main focus of interest.

Visualising electric and magnetic field simultaneously with intensity mapping enabled provides good results as well. The intensities of both fields can be easily compared in one scene as can be seen in Figure 5.17. The almost white areas of the magnetic field and yellow areas of the electric field are aligned in the corner of the domain an represent the maxima of the vector fields. The grid is not as pronounced as in the second dataset, however, the streamlines are still visibly perpendicular. Visibility impeding halos are used to separate nearby streamlines. It is interesting to note, that the halos work very well even if the actual crossings of streamlines fail to generate them as they are too close in depth. It is because the human brain is designed to fill gaps to create patterns. The halos not being generated properly especially happens in this case, as the electric and magnetic field streamlines often intersect each other.

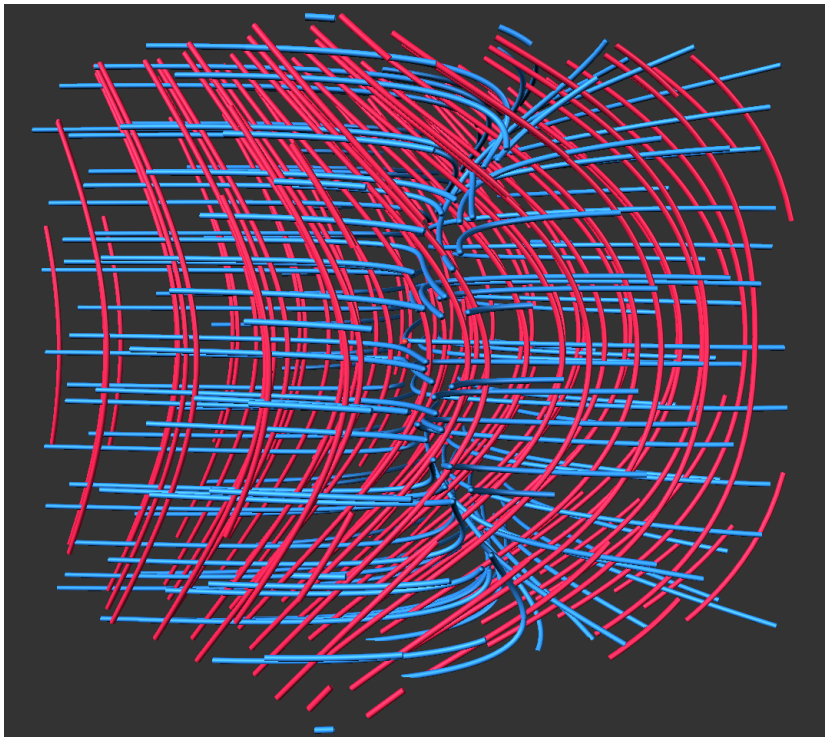


Figure 5.16: Visualisation of both the electric and magnetic field from dataset 2. Magnetic lines are blue, electric are red.

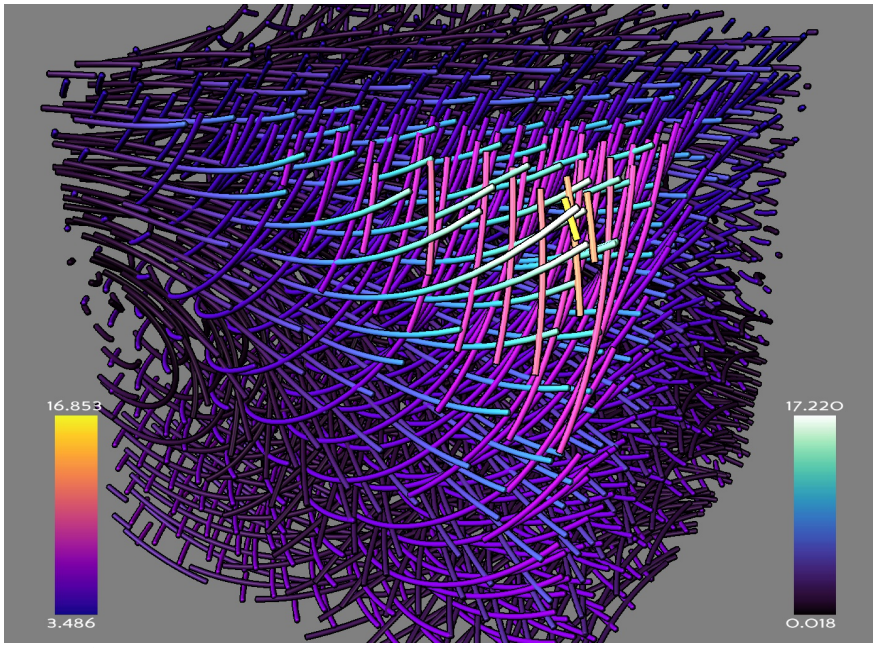


Figure 5.17: Visualisation of both the electric and magnetic field from dataset 3. Their intensity is mapped to color and visibility impeding halos are used.

The last Figure 5.18 shows both the electric and magnetic fields visualised in conjunction with area filtering. The most interesting part of the first dataset is in the center around the black hole, however, it is mostly hidden by a large number of parallel streamlines which do not provide any interesting insights into the behaviour of the field. Therefore, five faces of the area filter are positioned closely around the black hole itself and the last face cuts the hole in half. This way the rotation of the magnetic lines at the top and bottom of the black hole and the symmetric skew of the electric lines can be seen. The largest intensities of the vector fields are located around the equator of the black hole.

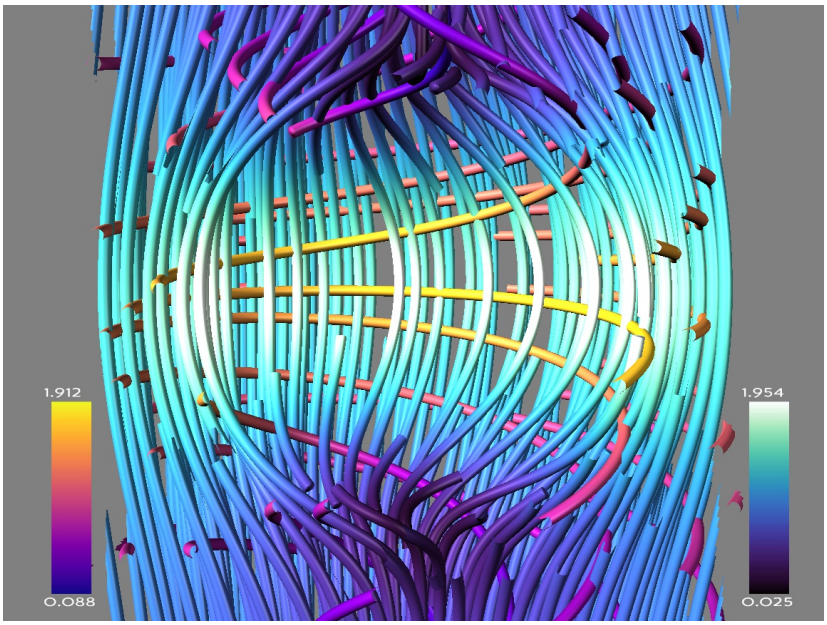


Figure 5.18: Visualisation of both the electric and magnetic field from dataset 1. Filtering is used to see the streamlines around the surface of the black hole.

Chapter 6

Usability testing

In order to test and improve the usability of the visualisation, qualitative user experience test is necessary. Dumas and Redish [27] recommend five steps for a successful usability test.

The primary goal of the test needs to be improving the usability of the product. This is a broad statement so some specific sub-goals should also be defined. These vary based on the product being tested and the specific needs of the currently designed test. There can be multiple tests of the same product with greatly different goals. Since this project is not that complex user experience as a whole can be analyzed in one test.

The users participating in the test need to represent real users of the product. Finding a domain expert for this specific project, who would agree to do a usability test did not turn out as planned. So the participants for this test are mostly students. This fact slightly devalues the outcome of the test, however, the most egregious issues should still be revealed and the overall user experience should improve.

The tasks need to represent real tasks. In order to do a usability test a list of tasks is prepared beforehand which the participants perform. The tasks need to reflect real usage of the product. They should be relevant, realistic and should correspond with the specific goals of the test.

The course of the test needs to be observed and can be recorded. The outcome of the tasks is not the only indicator of any problems with the product. The participant's comments and even non verbal communication are indicative of any issues or misunderstandings. Participants should be encouraged to think out loud to better understand their thought processes. After the tasks are finished a questionnaire or an interview should follow in order to gain additional insights.

Lastly the collected data needs to be analyzed and compared across the users. Some participants can have vastly different opinions on the same part of the project. That is okay as users are not homogeneous and can interact with the product in different ways. The issues which were found also need to be addressed and fixed otherwise the usability test is meaningless.

The usability test should mostly provide qualitative data based on the participant's experience. Some quantitative data can be obtained in the following questionnaire where participants try to quantify some aspects of their

interaction with the product. This quantitative data can be misleading as users quantify their experience differently. It can be more valuable in test with larger sample sizes, however, for usability testing Nielsen [28] recommends to have 5 participants in order to find the majority of problems with the product. The main drawback of larger amount of participants is that most of the findings are repeating. It is not a bad thing to see that multiple users are having the same issues. Nevertheless, it is enough to see it repeated two to five times anything more is just a waste of time of both the participants and the researcher.

6.1 Test design

The test is performed independently with 5 participants. First, each participant is briefed about the nature of the visualisation and its underlying vector field without showing or explaining anything about the application itself. Next the participants perform the tasks, some tasks contain subtasks which the participants are told to complete if they struggle with their current task:

- Generate new visualisation of dataset number 2.
- Describe the shape of the magnetic field around the null point compared to general descriptions of vector field singularities such as saddle points, sources, sinks etc.
  - Localize the null point of the magnetic field.
  - Filter the size of the area to be only in proximity of the null point.
  - Change the resolution of the filtered area from 1 to 4 or 8.
  - Rotate, zoom and filter the area to see the behaviour of the magnetic field from each side of the null point. Use visual aids such as halos or opacity mapping to help with this task.
- Visualise both the electric and magnetic field at the same time in the whole area and describe their relationship.
  - Hide the currently visualised area.
  - Add new area for visualising the magnetic field
  - Add new area for visualising the electric field
  - Set the color of each of these areas to a different flat color.
- Find the region where the magnetic or electric field has the highest intensity.
  - Change the color of one area to represent the magnitude.
  - Find the region with the highest visible magnitude based on the color scale.

- Define one small region around the null point with high density and another global region with low density.
  - Change the boundary of one magnetic field region to be around the null point. Rotating and zooming needs to be utilized to make sure all axes are close to the point.
  - Add a new area with magnetic low density of magnetic field streamlines.
  - Change the color of each region to distinguish them in the scene.
- Save the state of the visualisation, close it and load the saved state again.

The next set of tasks is about comparative qualities of different techniques of visualisation. The participants look at one or more pictures or visualisation states and answer the following questions, for each question multiple different scenarios are presented:

- In which direction is a specific stream tube heading?
  - Stream tubes with flat color and magnitude mapped to color.
  - Stream tubes with and without specular light.
  - Stream tubes with different thickness.
  - Stream tubes in an interactive visualisation.
- Which of the following stream tubes is further from the camera?
  - Stream tubes with and without halos.
  - Stream tubes with depth mapped to opacity.
  - Stream tubes in an interactive visualisation.

After all these tasks are completed the participants are interviewed about their experience with the application, where they struggled, what worked and what did not.

6.2 Test results

Multiple issues were found when trying to generate streamlines in the first task. There was no visual feedback that the generation was ongoing and multiple participants tried to generate it again. In addition the dataset options panel could not be easily closed and caused additional issues. As the generation runs in its own thread new generation or loading can be done from the main thread. This results in a buggy undefined behaviour and should not be allowed by the UI.

The second task was completed by all participants without any critical issues. All of them understood that the red point is the null point on their own without any explanation. They all used zooming, rotating, panning and resolution swapping to see the behaviour of the magnetic field around the

null point. Only some of the participants used area filtering to help them with this task. The function of the boundary axis slider were not obvious enough to try them on their own or if they tried filtering they did not find it useful for the task. However, when they were later shown how to use the filtering to easily see the saddle point of dataset 2, they all agreed it was the most straightforward technique to achieve the specified task. The main problem with the filtering was the absence of any information about the axes in the scene. Therefore, moving a desired side of the bounding box resulted in trial and error. One participant suggested having axis aligned arrows in the top right corner of the scene which also can serve to align the camera exactly with the preferred axis. The bounding box sliders themselves could be more intuitive so it is obvious that each of them moves one of the faces of the cuboid. This can be achieved by customizing the handles to convey some additional information, for example they can look like inward facing arrows to signify that they affect the opposite sides of the region. Another participant only used the filtering of the sides of the bounding box which point in the negative axes as the right side sliders did not seem to be related to the left side ones.

Another small issue was the behaviour of the resolution slider which is supposed to be rounded to the nearest integer. A Panda3D bug prevents the movement of the handle whenever the mouse hovers over it. Therefore, the resolution slider sometimes jumped to the nearest rounded location and sometimes did not. The resolution slider also needs to be more prominent as it is one of the most important UI elements.

The third task was the most problematic as several participants failed to visualise both vector fields at the same time without additional hints. The function of the five area buttons was not obvious as nothing visible happened when clicking between them as the visibility checkbox was unchecked by default. The name of the buttons themselves also was not very clear and participants guessed what it means and does. By default the buttons in Panda3D have a shaded 3D outline, which directly interferes when trying to use them as selectable tabs. The color of the buttons still changes to the color of the region with area setting, however, the shading and 3D border make them look disconnected. Better buttons without any distracting outlines need to be used to fix this issue. When the function of the area buttons and visibility checkbox were explained the participants managed to complete the task without any additional issues and saw the relationship between the electric and magnetic lines.

The fourth task had the least amount of issues as most of the participants managed to use the magnitude button beforehand. They all quickly identified where the maxima lie for each vector field and that the location is identical for both of them. One participant suggested using more distinct color scales for each field without any overlapping colors as these are misleading and lower the separability of the mappings. A small issue arose with the color selection radio buttons. Their default Panda3D look is too close to checkboxes and does not indicate any exclusivity between them. Therefore, most participants tried

to disable magnitude mapping by clicking on it again and not by selecting flat color. The fifth task was completed easily as the participants already had all the necessary knowledge from previous tasks. All of them managed to specify the focus and context areas and change their resolution to achieve the optimal result. Some participants even tried using transparency to improve the context of the visualisation.

The final task had similar issues as the first one. The save and load dialog could not be cancelled. And loading did not have any immediate feedback. Additional issued arose when interacting with the UI during loading as some important structures, which the UI tried to use, were not loaded yet and the whole visualisation crashed. Therefore, the UI needs to be either disabled during loading or additional checks for loaded elements need to be added.

The adaptive zoom option which changes the streamline density based on the distance from the center of the scene was barely used as participants preferred having control over the density themselves. Only one participant liked this feature and tried to use it. The first two resolutions work nicely with adaptive zoom, however, the step to the third one is too large and happens when the user is too close to the streamlines and they clutter the screen too much. The distances between the steps of adaptive zoom need to be more consistent for it to be more useful.

Several participants wanted to change the background color. The default one works well with certain streamline colors but not so well with others. It also heavily affects the outcome of transparent streamlines as the background color blends with the lines themselves changing their visibility. A simple grayscale slider which changes the background color in real time should be sufficient to help with this issue.

The last issue was with the readability of saved files. Their names are generated by concatenating strings containing the selected save slot, selected dataset, current day and time. This string is displayed when loading a file. While the actual save can be recognized it is not easy and more complex parsing of saved file names and multi line text need to be implemented to fix this issue.

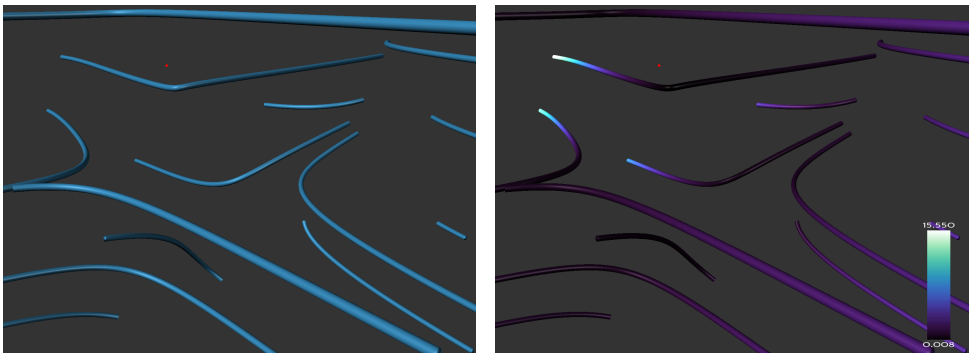


Figure 6.1: Comparison of the same streamlines with flat color and magnitude mapped to color.

6.2.1 Comparative testing

The perception of direction was comparable in streamlines with flat color and with magnitude mapped to colour as seen in Figure 6.1. Most participants slightly preferred the flat color as the shading was more consistent. However, the difference in the actual understanding of streamlines direction was negligible.

The participants were slightly divided on specular highlights. Some only liked them with darker colored stream tubes where they helped distinguish their shapes and directions. Others considered them beneficial in any situation as they are not too powerful and can help identify stream tubes with similar angle towards the user. Overall, highlights were considered to be mostly positive and remain in the visualisation.

Four levels of streamline thickness were shown to the participants. While the thickest streamlines were most beneficial for understanding their shape, they occluded the scene too much. On the other hand, the thinnest streamlines were considered to be the most accurate by some participants, however, their shapes were hard to distinguish. Therefore, most participants preferred the second or third thickness, which is also closest to the default thickness of the visualisation which lies somewhere between them.

Visibility impeding halos as seen in Figure 6.2 were viewed overwhelmingly positively. All participants liked how they help with separating individual streamlines in a dense visualisation. Some participants even believed halos help them with the orientation in the 3D space. Participants also liked to use halos both with sparse and dense streamlines as the streamlines on top are identified automatically without focusing on them explicitly.

The depth mapped to opacity was not very beneficial in helping with discerning the depth of specific streamlines. Most users preferred opaque streamlines and reading the depth from their shading and size in perspective projection. Rotating and zooming in the actual interactive visualisation by far provides the best depth perception.

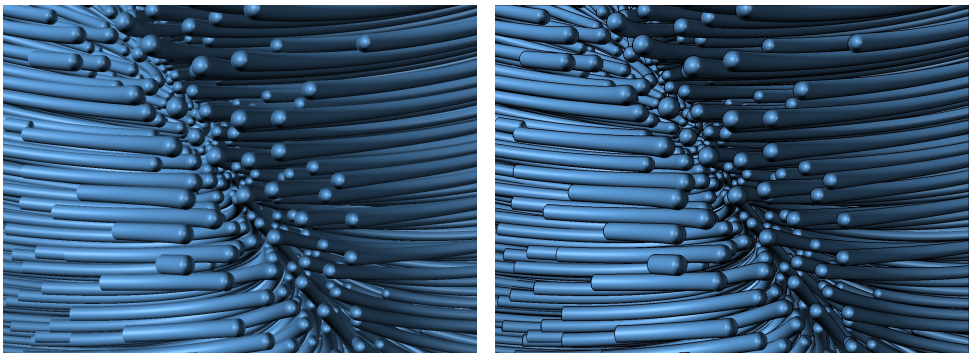


Figure 6.2: Zoomed in region of dense streamlines without and with visibility impeding halos.

6.2.2 Usability improvements

Found issues were classified by their severity and fixed in that order. All of the critical errors which crash the visualisation, such as interaction with not fully loaded streamlines, were fixed. The UI elements are now disabled during loading and generating as they do not serve any purpose at that time and only can be sources of issues. Once the magnetic field streamlines are generated all UI elements become interactable except for the electric streamlines switch which only becomes interactable once the electric streamlines are generated. Loading creates other issues as the saved state can contain areas with visible electric lines, therefore, the visualisation needs to wait until both magnetic and electric lines are loaded to show them all simultaneously. Once that happens the UI can be interacted with again.

The second category of issues contains major usability problems. The generation and loading progress is now visible in a proper loading bar which displays the percentage completion and its text changes based on the currently generated or loaded vector field. The Area buttons are arranged to neatly align with the box beneath them to create tabs. This way the settings for each area are visually connected with its respective button which no longer has any 3D effects or borders. A semitransparent clickable region is now added behind the options menu visible when selecting a save file or a dataset to load. Clicking on that region hides the dialog again. None of the UI elements below the region are clickable when it is present. This way the user can change their mind and return when accidentally clicking one of the three top buttons.

All radio buttons now have a custom circular image to differentiate them from checkboxes. Visibility and electric lines checkboxes are replaced by radio buttons which additionally also include the magnetic lines. This way each area has three possible states – hidden, visualising electric lines and visualising magnetic lines. Several additional UI elements now have improved position, readability or name. The rest of the found issues need to be fixed in the future.

Chapter 7
Conclusion

The goal of this thesis was to design and implement an interactive application for visualising electromagnetic vector fields around black holes to make their exploration easier, especially focusing on the area where the magnetic field approaches zero also known as a null point. Therefore, various techniques for visualising three dimensional vector fields were analyzed. Texture based approaches such as line integral convolution proved to be unsuitable for this task even when using volumetric textures, as multiple vector fields needed to be visualised simultaneously. Sparser stream objects like streamlines or stream ribbons appeared to be a better choice as their density can be controlled. Based on the analysis, evenly-spaced illuminated stream tubes were chosen as the best approach for this problem. They uniformly fill the scene and can be shaded to help distinguish their direction.

The application itself was designed to allow visualisation of multiple regions of stream tubes with different parameters such as density, color and transparency. Boundaries of these regions can be changed in real-time to only view required areas of interest. Each region can swap between visualising the electric and the magnetic field. Both of these fields can also be visualised at the same time enabling direct comparison. Vector magnitudes can be mapped to the color of the stream tubes using a different linear color scale for each field. Visibility impeding halos can be used for better orientation in the scene by separating close stream tubes.

The implementation was done in Python with the help of the Panda3D library. Streamline seeding and tracing runs only in Python using the NumPy library for handling the large datasets. The calculated streamlines are transferred to Panda3D to be transformed to solid 3D objects, in order to be drawn and shaded. A simple GUI was designed and implemented using DirectGUI from Panda3D to enable interaction with the visualisation.

Usability testing was performed with five participants who completed several tasks, such as describing the shape of the magnetic field around the null point or finding the exetrema of both fields. While all of the tasks were in the end completed, issues of varying importance were discovered and a large number of them subsequently fixed.

7.1 Future work

In the future the application can be improved in several ways. First the remaining issues revealed by the usability testing need to be fixed and another round of testing concluded. Afterwards, the performance of streamline generation and loading can be improved. The optimization was not a main priority as the generation only happens once at the beginning and the rest of the visualisation runs in real-time without any issues. Other improvements can include real-time rescaling of the magnitude color map based on the filtered area, filtered area of different or arbitrary shapes and visualisation of animated streamlines.

Bibliography

[1] GODDI, C. and CREW, G. and IMPELIZZERRI, C. et al. *First M87 Event Horizon Telescope Results and the Role of ALMA*. The Messenger No. 177. 2019.

[2] The Event Horizon Telescope Collaboration et al. *First M87 Event Horizon Telescope Results. VIII. Magnetic Field Structure near The Event Horizon.* The Astrophysical Journal Letters, Volume 910. 2021.

[3] BLANDFORD, R. D. and ZNAJEK, R. L. *Electromagnetic extraction of energy from Kerr black holes*. Monthly Notices of the Royal Astronomical Society, Vol. 179. 1977.

[4] TELEA C. A. *Data Visualization - Principles and Practice*, 2nd edition. CRC Press: 2014. Chapter 6 Vector Visualization. ISBN-13: 978-1-4665-8526-3.

[5] MUNZNER, T. *Visualization Analysis and Design*. Boca Raton, Florida: A K Peters/CRC Press, 2015. ISBN 9781466508910.

[6] YAGEL, R., REED, D. M. et al. *Hardware assisted volume rendering of unstructured grids by incremental slicing*. Proceedings of 1996 Symposium on Volume Visualization. 1996.

[7] STEVENS, S. S. *To honor Fechner and repeal his law: a power function, not a log function, describes the operating characteristic of a sensory system*. Science 133, 80–86. 1961.

[8] Crameri, F., Shephard, G.E. and Heron, P.J. *The misuse of colour in science communication*. Nat Commun 11, 5444. 2020.

[9] VERAS, R and COLLINS, C. *Discriminability Tests for Visualization Effectiveness and Scalability*. IEEE VIS. 2019.

[10] ELLIS, G. and BERTINI, E. and DIX, A. *The sampling lens: Making sense of saturated visualisations*. Conference on Human Factors in Computing Systems. 2005.

[11] ROSSI A. R. *Glyphs, Streamlines, Stream Tubes, Stream Surfaces.* [online] Available from: `http://ryanrossi.com/sv4.php`. Cited 23. 12. 2022.

[12] LIU Z., MOORHEAD R. and GRONER J. *An Advanced Evenly-Spaced Streamline Placement Algorithm.* IEEE Transactions on Visualization and Computer Graphics, vol. 12, no. 5. 2006.

[13] MATTAUSCH O., THEUBL T et al. *Strategies for Interactive Exploration of 3D Flow Using Evenly-Spaced Illuminated Streamlines.* SCCG '03: Proceedings of the 19th Spring Conference on Computer Graphics. 2003.

[14] ZOCKLER M., STALLING D.and HEGE H. *Interactive visualization of 3D-vector fields using illuminated stream lines.* Proceedings of Seventh Annual IEEE Visualization. 1996.

[15] CAMP, D., CHILDS, H. et al. *Parallel Stream Surface Computation for Large Data Sets.* In Proceedings of the IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV). IEEE Press. 2012.

[16] ESTURO, J.M., SCHULZE, M., RÖSSL, C. and THEISSL, H. *Global Selection of Stream Surfaces.* Computer Graphics Forum, 32: 113-122. 2013.

[17] JOBARD B. and LEFER W. *Creating Evenly-Spaced Streamlines of Arbitrary Density.* Proceedings of the eight Eurographics Workshop on visualization in scientific computing. 1997.

[18] HELMAN, L. J., HESSELINK, L. *Visualizing vector field topology in fluid flows.* IEEE CG&A, 11(3): pages 36-46. 1991.

[19] SCHEURMANN, G., TRICOCHE, X.M. and HAGEN, H. *C1-interpolation for vector field topology visualization.* Proceedings Visualization '99. 1999.

[20] GÜNTHER, T., BAEZA ROJO, I. *Introduction to Vector Field Topology.* In: HOTZ, I., et al. *Topological Methods in Data Analysis and Visualization VI.* Mathematics and Visualization. Springer, Cham. 2021. 289-326. ISBN: 978-3-030-83500-2.

[21] CABRAL, B. and LEEDOM, L.C. *Imaging vector fields using line integral convolution.* In Proceedings of the 20th annual conference on Computer graphics and interactive techniques (SIGGRAPH '93). Association for Computing Machinery, New York, NY, USA. 1993.

[22] INTERRANTE, V. L. and GROSCH, C. E. *Recent advances in visualizing 3D flow with LIC.* National Technical Information Service. 1998.

[23] NIENHAUS, M. and DOELLNER, J. *Edge-Enhancement – An Algorithm for Real-Time Non-Photorealistic Rendering.* Journal of WSCG. 2003, vol. 11, no. 1-3. 2003.

[24] URNESS, T., INTERRANTE, V. et al. *Strategies for the visualization of multiple 2D vector fields*. IEEE computer graphics and applications, vol. 26. 2006.

[25] FRISVAD J. *Building an Orthonormal Basis from a 3D Unit Vector Without Normalization.* Journal of Graphics Tools. 2012.

[26] BIRN, J. *Digital Lighting and Rendering*. First edition. New Riders Publishing. 2000. Chapter 3 Three point lighting. ISBN 9781562059545.

[27] DUMAS, J. and REDISH, J. *A practical guide to usability testing*. Intellect ltd. 1999. ISBN 1841500208.

[28] NIELSEN, J. *Why You Only Need to Test with 5 Users*. Nielsen Norman Group. 2000. [online] Available from: `https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/`. Cited 20. 12. 2022.

Appendix A

Used acronyms

- **2D** Two Dimensional
- **3D** Three Dimensional
- **CG** C for Graphics
- **GLSL** OpenGL Shading Language
- **GPU** Graphics Processing Unit
- **GUI** Graphic User Interface
- **LIC** Line Integral Convolution
- **RGB** Red, Green, Blue
- **UI** User Interface

Appendix B

Structure of attachments

- app — Executable files for different systems
  - macosx_10_9_x86_64 — Mac OS executables
  - manylinux_x86_64 — Linux executables
  - win_amd64 — Windows executables
- doc — Documentation
- img — Example images of all 3 datasets
  - dataset1
  - dataset2
  - dataset3
- manual — User manual
- src — Source codes
  - data — Datasets in numpy readable format
  - resources — Resources used in the visualisation
  - saves — Saved states of the visualisation
  - shaders — Shaders used in the visualisation
- text — Text of this thesis