Extraction and Visualisation of Vortices from Instantaneous Particle Image Velocimetry

Oliver H. Eisenberg

Bachelor of Science in Computer Science with Honours
University of Bath
May 2020

Extraction and Visualisation of Vortices from Instantaneous Particle Image Velocimetry

submitted by Oliver H. Eisenberg

Declaration
This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Abstract

Within Formula One, data gathered from Computational Fluid Dynamics (CFD) does not always correspond to the telemetry collected on the racetrack, where a higher correlation between the sources allows for CFD to carry more weight resulting in successful aerodynamic packages. To improve correlation, Formula One teams use wind tunnels with a scale model of their car to reproduce conditions in both CFD and the racetrack. This paper focuses on the visualisation of data measured from the tunnel using Particle Image Velocimetry. Flow visualisation has undergone lots of research and this document explores the current state of the art and applies vortex extraction and visualisation techniques to a time-dependent data set, provided by Scuderia Toro Rosso, to allow for a meaningful interpretation of data.

Impact by COVID-19

COVID-19 has reduced the amount of work possible during the project. The data initially obtained was two dimensional and was meant to provide a baseline before extending work to the third dimension. Unfortunately due to the degradation of contact with Scuderia Toro Rosso this possible extension was halted and alternative avenues were explored.

Contents

Acknowledgements

I would like to thank the Aerodynamics department at Scuderia AlphaTauri, formally Toro Rosso (2006 - 2019), for the data provided and inital guidence on avenues to explore with this project. I would also like to thank my University project supervisor, Yongliang Yang, for his advice and counsel during the course of the project.

List of Figures

List of Tables

List of Code snippets

Chapter 1

Introduction

1.1 The Formula One World Championship

Formula One (F1) is a single-seater racing series first held in 1950. In the early 1950s, manufacturers such as Mercedes-Benz and Alpha-Romeo populated the sport but by the 1970s it had become the domain of specialist constructors [8], where a 'constructor' is someone who designs key parts - as determined by the Federation Internationale de l'Automobile (FIA). Since 1981 the base of the car, the chassis, was included as a mandatory build part for each team, making the terms 'constructor' and 'team' interchangeable [9].

Red Bull Racing and Scuderia Toro Rosso (Toro Rosso) were two teams owned by the Austrian beverage company Red Bull. Toro Rosso, based out of Faenza, functioned as a junior team to develop skills of their drivers and in the 2007 season competed using the same chassis made by its parent Red Bull Racing. In 2010 the technical regulations were updated to require each constructor to own the intellectual property rights to their chassis and in 2020, the team was renamed to Scuderia AlphaTauri to compete as one of ten teams in the 2020 championship season.

Wind tunnels have a critical role in the Formula One World Championship [10] where one of the largest contributions to performance is aerodynamics. An aerodynamicist will design concepts, based on simulations, to create or manage flow features to produce pressure differentials along the car. These form suction surfaces [11] which create downforce where, in 1992, a 1% coefficient was observed to reduce the lap time by 0.1s around the Silverstone Grand Prix Circuit [12]. In the tunnel, engineers can rapidly test and refine their concepts on a scaled model before releasing them, optimised for specific circuits [11].

At Toro Rosso, pressure data was gathered from sensors embedded in the model and Particle Image Velocimetry (PIV) is used to take external flow field measurements. PIV is an optical method of flow visualisation used to obtain instantaneous velocity fields in turbulent flow. Currently, their PIV data is time-averaged and interpreted as a series of two-dimensional vector planes. Images

are then analysed, to determine the path of a vortex.

The series of data is repeatedly analysed as a vector field requiring the manual searching for vortices and tracing during bursts. This has the potential to be improved using algorithms to extract, track and visualise the results for better understanding.

1.2 A Foreword on Existing Systems

The visualisation of flow data is not a novel concept, but has had extensive research to illustrate the complex flow information. However, due to the rarity of the data collection method and bespoke data format used, no commercial systems to visualise the data exist. At Toro Rosso, animations from two-dimensional Quiver plots are generated and interpreted by aerodynamicists to extract the positions of vortices.

1.3 Project Aims and Objectives

The project aims to investigate and explore techniques to make an informed decision on extracting, tracking and visualising vortices from the data provided by the Formula 1 constructor Toro Rosso. There is lots of work surrounding the topic of flow visualisation, however, the use of PIV and therefore parameters available in the data make efficient visualisation harder to undertake. The project should produce an application that builds on the existing work and can be used by aerodynamics to aid their comprehension of the data. The high-level objectives for this dissertation have been outlined:

1. To research existing algorithms in two and three dimensions within the flow visualisation field;
2. To investigate the time-dependent vector data produced by PIV;
3. To extend and apply existing work to be able to identify a vortices from the data;
4. To produce a system capable of extracting and visualising vortices from the data.

1.4 Contributions

The main contributions of this project include:

1. A summary of research into flow visualisation;
2. A system that makes large, complex, time-dependent data comprehensible;
3. An assessment of the system.

1.5 Structure of this Dissertation

The structure of this dissertation has been outlined by chapter below:

1. **Introduction**
   This chapter provides an overview of the problem and outlines the project's aims and contributions.

2. **Literature Review**
   This chapter provides an overview of the state of the art of flow visualisation and extraction techniques.

3. **Requirements**
   From the research, the project aims are developed further in the form of software requirements which guide the development detailed in the Design chapter.

4. **Design**
   This chapter details the development process of the system, split by front-end and back-end, to fulfil the requirements outlined in the Requirements chapter.

5. **Implementation**
   This chapter provides an overview into how the system was implemented as structured by the Design Chapter.

6. **Discussion**
   This chapter provides further detail about how the system was implemented and evauates some of the alternative avenues explored. In particular, attention is paid to the rationale behind the decisions made.

7. **Testing and Evaluation**
   This chapter provides details on the testing strategy employed and evaluates the overall implementation.

8. **Conclusion**
   This chapter closes the dissertation with a summary on objectives satisfied, contributions made and future work that can be completed to extend the work presented.

Chapter 2

Literature Review

2.1 Introduction

Areas to be covered

As the dataset has been provided, attention can be focused on post-processing tasks split into two distinct areas, extraction and visualisation of vortices. Both of these have had extensive research, however, as the definition of a vortex in viscous flow is an open problem [13], with a lack of an acceptable definition [14] approaches and techniques, are varied.

2.2 Background

2.2.1 Data

The data set used for the dissertation is from a wind tunnel run provided by Toro Rosso in April 2018. The data contains four hundred samples at a single z position, where the z-axis is the depth along the length of the car, just behind a simple aerodynamic surface to capture produced vortices. Therefore, the data samples are two-dimensional layers correlated by time.

Variables

Per frame within the data set are nineteen variables to describe the measurements obtained, they are listed in table 2.1 with their measurement units.

Format

The data can be formatted as a grid, however, the data has irregular spacing and some values are missing. In addition, although samples are two dimensional the points have different z values, this is because the plane is at an angle and not parallel to the viewing plane.

Table 2.1: Variables present in the provided data

Variables	Description	Units
x	physical coordinate X	[mm]
y	physical coordinate Y	[mm]
z	physical coordinate Z	[mm]
u	velocity coordinate X	[m/s]
v	velocity coordinate Y	[m/s]
w	velocity coordinate Z	[m/s]
velmag	velocity magnitude	[m/s]
flag	status flag 0=Disabled, 1=NoResult, 2=Outlier, 3=ValidData, 4=OtherPeak, 5=Interpolated	
count	number of valid data at node	
valid	validation rate [0-1]	
snr	signal-to-noise ratio	
cor	correlation coefficient	
dudx	velocity gradient du/dx	1/us
dudy	velocity gradient du/dy	1/us
dudz	velocity gradient dv/dx	1/us
vort z	voritcity component Z	1/us
shear z	shear strain component z	1/us
nstrain	normal strain	1/us

2.2.2 Types of Flow

Steady and Unsteady vector fields

The differences between the types of flow are distinguished by Tobias Günther and Holger Theisel [15].

Definition 2.2.1. Steady Vector field. A field that does not change over time and is formally given as a time-independent map $v(x) = v(x, y[, z]) : D \to D$

$$2D : v(x, y) = \begin{pmatrix} u(x, y) \\ v(x, y) \end{pmatrix},$$

$$3D : v(x, y, z) = \begin{pmatrix} u(x, y, z) \\ v(x, y, z) \\ z(x, y, z) \end{pmatrix},$$

where $u, v[, w]$ denote the direction components. Steady flows describe instantaneous fields, such as a magnetic field, which even though may change - our focus is on a single time slice.

Definition 2.2.2. Unsteady Vector field. changes over time and is formally given as a time-dependent map $v(x) = v(x, y[, z], t) : D \times T \to D$

$$2D : v(x, y, t) = \begin{pmatrix} u(x, y, t) \\ v(x, y, t) \end{pmatrix},$$

$$3D : v(x, y, z, t) = \begin{pmatrix} u(x, y, z, t) \\ v(x, y, z, t) \\ z(x, y, z, t) \end{pmatrix}.$$

Streamlines and Pathlines

Both streamlines and pathlines represent the trajectory of a tracer particle in flow, where the difference lies in the type of flow the particle resides in. If the flow is steady, the trajectory is called the streamline otherwise it is called a pathline.

2.2.3 Tensors

A first-order tensor is also known as a vector. Defined by Aris [16] as a nine component entity that can be written as a 3 x 3 matrix.

$$\mathbf{A} = \begin{bmatrix} A_{11}, A_{12}, A_{13} \\ A_{21}, A_{22}, A_{23} \\ A_{31}, A_{32}, A_{33} \end{bmatrix}$$

Symmetric and Antisymmetric tensors

A symmetric tensor is when $A_{ij} = A_{ji}$ and it has only six distinct components. If the transposition equals its negative, $A_{ij} = -A_{ji}$, then it is Skew, Anti- or A-symmetric and its characterised by three quantities as the diagonal are comprised of zeros [16].

2.2.4 Properties of vector fields

Nabla (Gradient) Operator

The Nabla operator ∇ is a vector that contains the partial derivative symbols with respect to the spatial dimensions [15]:

$$\left(\frac{\vartheta}{\vartheta x}, \frac{\vartheta}{\vartheta y} \left[, \frac{\vartheta}{\vartheta z} \right] \right)^T$$

Divergance

Divergence $\nabla \cdot v$ of a steady or unsteady flow $v(x, y, [, z, t]) - (u, v[, w])^T$ is a scalar field that characterises the change in volume of a virtual, finite-sized sphere that is advecting with the flow. An incompressible, or solenoidal, field is one where the divergence is equal to zero everywhere.

Galilean invariance

Galilean invariance states that the laws of motion are the same in all inertial frames [17]. A method that is Galilean invariant can track equal speed translations like the acceleration of a field.

The Velocity Gradient Tensor

The gradient of a vector field is found by applying the gradient operator to each of the three components $[x\ y\ z]$ of the three-dimensional vector field. This results in the Jacobian of the vector field ∇u which can be decomposed into S and Ω which are the symmetric and skew-symmetric parts, where S is known as the rate of strain tensor and Ω the vorticity tensor. They can be used to compute fields like the divergence, curl, helicity, acceleration and curvature as well as the direction of tangent curves of the flow using the eigenvectors and eigenvalues of the matrix [18].

The characteristic equation for ∇u is given by $\lambda^3 + P\lambda^2 + Q\lambda + R = 0$ where P, Q and R are the invariants of the velocity gradient tensor. They can be expressed as follows [19]:

$$P = -tr(\nabla u)$$

$$Q = \frac{1}{2}(tr(\nabla u)^2 - tr(\nabla u^2)) = \frac{1}{2}(\|\Omega^2\| - \|S\|^2)$$

$$R = -det(\nabla u)$$

where, $tr\nabla u$ is the trace of the velocity gradient tensor and $det\nabla u$ is the determinate [20].

Vorticity

Vorticity w, or curl, is Galilean-invariant quantity that indicates how the flow swirls at a certain point and can be computed from Ω, the vorticity tensor, that can be defined in both two and three dimensions. If a virtual finite-sized particle is advected within a flow, it may spin. In two dimensions the sign of the scalar w represents the rotation direction, in the third, the axis it spins around is parallel to the vorticity vector and w is a vector quantity where its magnitude $|w|$ is twice the angular velocity. An irrotational, or conservative, field is one where the vorticity is zero everywhere. A rotational, or nonconservative field cannot be expressed as the gradient of a scalar field.

2.3 Flow Visualisation Methods

Traditionally, flow visualisation has been divided into four categories, direct, geometry, texture and feature-based techniques. Direct methods encode primitives to represent properties, to visualise a large subset of, or the complete, data set. It is successful in two but limited in three dimensions and requires minimal pre-processing. Geometry-based methods use primitives to represent

flow which needs to be first extracted. This encompasses geometric models such as streamlines and time surfaces [1]. Texture-based or Dense-based methods are those where data is encoded in a dense manner using textures and are predominantly used for visualising flow in two dimensions and on surfaces. Feature-based methods partition and only describe the flow by structures in the vector field topology, the methods for extraction and visualisation are dependent on the data set and application. Direct, texture and geometric methods are respectively shown in figure 2.1 from left to right. Each group of methods has been examined in this section, divided into their application in two dimensions, slices and boundaries and three dimensions.

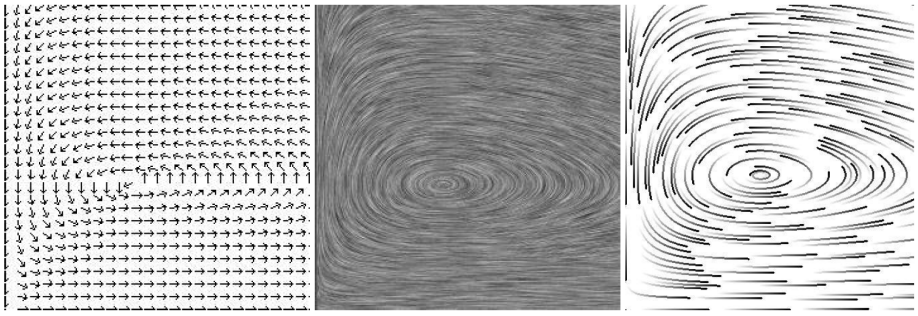


Figure 2.1: A Direct (LHS), Texture (middle) and Geometric plot (RHS) [1]

2.3.1 Direct Visualisation

Visualisation in two dimensions

Colour coding

Common flow attributes can be mapped to colour resulting in intuitive representations of the data with a reasonable colour scale [21]. The technique can be extended to temporal data resulting in animated colour plots depending on the flow properties.

Arrow plots

Flow attributes can be used to orientate arrows on a regularly spaced grid where either the arrow length is fixed or used to indicate the flow velocity. The technique is named a *Hedgehog* visualisation by Klassen and Harrington [22] and Schroedel et al [23] but the term *Quiver plot* is used in python libraries Matplotlib [24], Plotly [25] and Matlab documentation [26]. The technique can also be extended to temporal data, however, the time steps need to be considered for fluid transitions.

Hybrid plots

A hybrid approach was suggested by Kirby et al [27] to combine both arrow plots and colour coding to achieve a more descriptive result.

Visualisation on slices or boundaries

Colour coding

Two-dimensional slices are layered with clipping, which are the transparent sections rendered to prevent occlusion from other slices.

Two dimensional Arrows

Arrows were utilised by Fenlon et al [28] to visualise the complex nature of flow geometry in a three-dimensional space. However, they only represent the components in the orthogonal plane [21].

Visualisation in three dimensions

Occlusion and complexity makes it hard to interpret entire three-dimensional data sets making rendering an important issue.

Volume rendering

Volume rendering is an extension of colour coding in two dimensions that accounts for the occlusion and complexity problems that arise from three-dimension renders. Volume rendering is used in the medical field and some challenges prevail in both:

- Flow data sets are normally smoother than medical data - the absence of sharp and clear boundaries makes mapping harder and less intuitive.
- Flow data is often given on non-Cartesian grids - making the rendering more complex
- Flow data is time-dependent which adds to the rendering process

Algorithms have been presented by Clyne and Dennis [29] and Glau [30] that use graphics hardware to render time-varying vector fields. More recently, Ebert and Rheingans [31] used non-photorealistic rendering techniques such as silhouette enhancement to improve renders.

Three dimensional Arrows

The use of three-dimensional arrows present two problems:

- The position and orientation of the vector can be difficult to understand because of its projection onto a surface. This problem can be reduced with the use of three-dimensional representations of arrows.

- The occlusion problem arises with the use of three-dimensional arrows, or glyphs, which can be reduced by the seeding the positions of the glyphs to avoid overcrowding.

Arrow plots in three dimensions usually undergo *selective seeding*, addressed by Boring and Pang [2] who use interactive light sources to highlight regions of different flow directions combined with other geometry. By changing the light direction, regions with aligned vectors are highlighted as clusters which change shape and location. Adding more lights works to narrow the focus of the flow directions. Their work was applied to a delta wing in a CFD simulation as shown in figure 2.2.

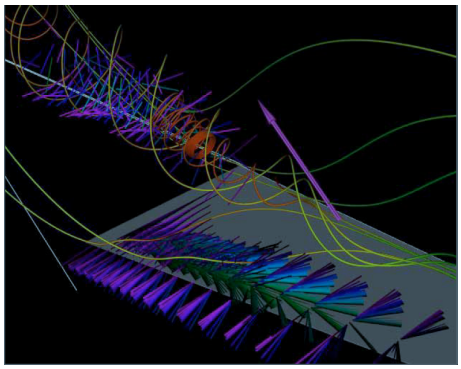


Figure 2.2: 40° illumination on a delta wing from CFD data [2]

2.3.2 Geometry-based methods

Contouring in two dimensions

This is the use of boundaries between two distinct areas, often used on Topographic maps to illustrate elevation. Transition areas are depicted as a change of colour in colour plots. With contour plots, an explicit line is drawn.

Visualisation using integral objects in two dimensions

Streamlets and Streamlines

Integrating flow vectors for a very short time produces streamlets. Although only a short time is used, they indicate the movement of flow. Integrating vectors over a longer period produces streamlines which are an extension to glyph techniques.

Streaklines, timeline and pathlines

When interrogating unsteady flow, several integral objects are used. Pathlines, timelines - join points at a constant time, t and streaklines - trace particles that

have passed through a unique domain point [23]. Streaklines can represent the injection of foreign material into the flow. An adaptive method was proposed by Sanna et al [32] where the seeding of streaklines is a function of local vorticity.

Streamline seeding

The initial conditions are important when visualising vector field streamlines as an evenly spaced distribution of seed points does not result in evenly spaced lines. Turk and Banks [33] and Jobard and Lefer [34] both developed automatic techniques to place seed points to achieve a uniform distribution of streamlines on a two-dimensional vector field. Turk and Banks use an energy function to guide their placement strategy which uses a low-pass filtered version of the image to identify the visual density.

Verma et al [35] presented a topology-based strategy for placing streamlines from flow features with the primary goal of capturing flow patterns near critical points and a secondary goal of ensuring that the placement of streamlines results in insufficient coverage of non-critical regions in an aesthetically pleasing way. Verma et al compare their generated images to Turk and Banks' [33] in figure 2.3.

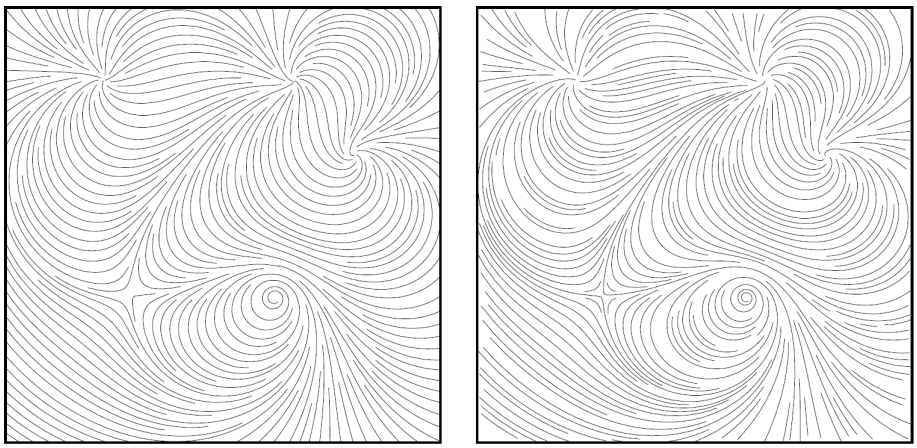


Figure 2.3: Left: Image produced using Turk and Banks's image-guided method. Right: Verma's flow-guided method. Where the streamline seperation was chosen to be 1% of the image width.

Jobard and Lefer [3] build on their previous work to present a multiresolution (MR) method for visualising large two dimensional, steady-state vector fields. Where a viewer can interactively select desired density levels and zoom in and out of complex areas, as shown in figure 2.4.

Time-dependant data is a special challenge when seeding. Jobard and Lefer [36] proposed a method to effectively visualise two-dimensional unsteady vector fields. Their approach correlates the instantaneous vector field at the stream-

Figure 2.4: User zooming into an area of interest using the Multiresolution algorithm proposed by Jobard and Lefer [3].

line level and for each frame, uses a feed-forward algorithm to calculate streamlines as a function of the streamlines generated from the previous frame. Their method allows for control on the image density allowing it to represent sparse or dense images.

Visualisation on slices or boundaries

Visualisation on objects

Geometric objects can also be used for visualising boundaries or slices through three dimensional flow [28] but can be misleading as flow components can be omitted during flow integration [21].

Streamline seeding on boundary surfaces

The work of Turk and Banks [33] was extended to generate evenly distributed streamlines on boundary surfaces within curvilinear grids by Mao et al [4] who present a new energy function guiding the placement of streamlines in the computational space with desired local densities. Their technique is shown in figure 2.5.

Visualisation using three-dimensional geometric objects

Isosurfaces

Contouring can be extended to three dimensions using isosurfaces. Because of the smooth nature of flow data, the selection of the isovalue lacks intuitive interpretation.

Isosurfaces have been applied to local density gradient maxima (shockwaves) and the extraction of crestline and ridges by Tang and Medioni [37] who used Tensor Voting and Extremal algorithms to provide robust and meaningful extraction for efficient visualisation and further processing.

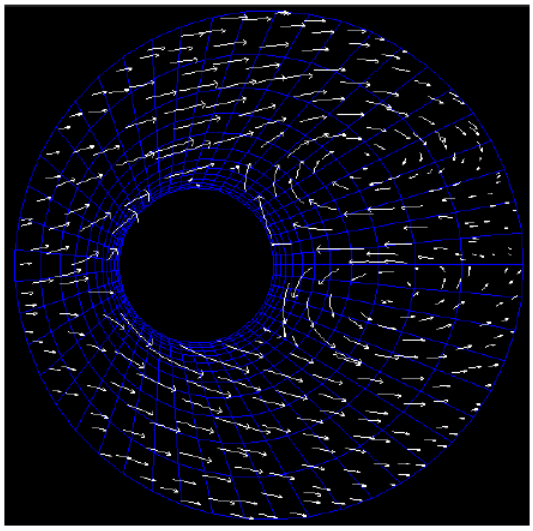


Figure 2.5: An uniform distribution of streamlets generated with Mao et al's technique [4].

Röttger et al [38] use a hardware-accelerated technique and a Projected Tetrahedra (PT) algorithm by Shirley and Tuchman [39] to render multiple shaded isosurfaces without reconstruction. They believe that, as three-dimensional texture mapping becomes more widespread, their method will become more attractive.

Streamlets

Streamlets can be extended to three-dimensions however occlusion and perceptual problems can arise due to distortions when projecting onto a two-dimensional surface. Similarly to streamline seeding in two-dimensions, seeding strategies are important. Löffelmann and Gröller [40] present a technique to visualise dynamic systems using a thread of streamlets. To tackle occlusion, bundles of streamlets are placed near characteristic trajectories instead of entirely populating the space with streamlines.

Streamlines

Careful placement, to avoid clutter, is used at NASA using the Flow Analysis Software Toolkit (FAST) to visualise Computational Fluid Dynamics data [41].

Illuminated Streamlines

To improve the perception of streamlines, Zöcker et al [42], used the Phong [43] lighting model for realistic shading - further improved by making some curves partially transparent. They show that streamline images can be generated using

hardware supported texture mapping and suggest that seeding near features of interest as an avenue of future improvement.

Particle tracing

For real-time applications, tetrahedral decomposition is used to speed up a point location and velocity interpolation in curvilinear grids by Kenwright and Lane [44], who developed the algorithm for interactive visualisation of large, unsteady, aeronautical simulations. Their efficient algorithm allowed for interactive positioning and the adjustment of streakline injectors to aid detection in the unsteady flow.

An efficient and accurate method for computing tangent curves was presented by Nielson and Jung [45] whose exact methods use values based on piece-wise linear variations over a tetrahedrisation and barycentric coordinates to track movement instead of using approximations like Kenwright and Lane. In σ- transformed grids, Sadarjoen [46] details an improved method for six tetrahedra which is robust but introduces large shearing. They also make comparisons between five- and six-decomposition methods.

In a virtual 3D environment, Bryson and Levit [47] seed integral objects using a 'rake' and gesture-based controls. Their approach gives the user interactive control and supports a user placing seed points in advance. The former method uses a tool they define as a 'rake', which allows users to group multiple seed points. Properties can be associated with a rake such as the type of visualisation, the number of seed points in the rake or the length or the streamlines to create the different visual representations.

Stream ribbons and streamtubes

Streamlines can be adapted to store additional information like flow rotation and expansion using stream ribbons and tubes respectively. A stream ribbon is a streamline with a wing-like strip allowing to visualise the rotation, whereas a streamtube is a thick streamline that can be extended to show the expansion, contraction and deformation. This makes them more advantageous over streamlines as they can encode more data [48]. Efficient constructions of streamlines, stream ribbons and streamtubes on unstructured grids were presented by Ueng et al [48] who develop a specialised Runge-Jutta method to speed up the tracing of such lines and calculate them using the canonical coordinate system, reducing computational costs allowing large flow fields to be explored.

Fuhrmann and Gröller [49] use dash tubes an animated, opacity-mapped streamline. They present an algorithm that distributes dash tubes evenly and applies magic lenses and boxes to add an element of interactivity to add detail in areas of interest allowing for an intuitive approach, similarly to Jobard and Lefer [3]. They found that interaction with the magic elements to be valuable but cumbersome with a mouse compared to a virtual environment.

Stream runners

Laramee [50] introduced stream runners to expand on streamtubes in an attempt to reduce the occlusion problem by allowing the user to define the length of streamtubes.

Stream polygons

Schroeder et al [51] extend streamlines to stream polygons which can illustrate the local deformation along a streamline. The polygons can be rotated and undergo shear and normal strain. The polygons also can be shaded offering the further encoding from the data set.

Streamballs

Streamballs were adopted from metaballs and used by Brill et al [52] which allow for the visualisation of complex flow fields due to their property to split and merge and therefore represent convergence and divergence and acceleration and deceleration, respectively.

Stream surfaces

Stream surfaces are another extension to streamlines. They are surfaces that are tangent to a vector field approximated using adjacent pairs of streamlines [53]. Surfaces can use texture to represent other attributes of the flow field. Hultquist [54] presented an interactive technique using stream surfaces, succeed by Wijk [55] who proposed a method to construct implicit stream surfaces which could also be used for other visualisation techniques, such as time surfaces and stream volumes.

Streamarrows were presented by Löffelmann et al [56] [57] which is a technique to enhance streamsurfaces by separating arrow portions from the surfaces. Streamarrows are based on regular tiling which isn't suited for regions of convergence and divergence, therefore a hierarchical extension was proposed to enhance streamsurfaces.

Van Wijk [58] use 'surface particles', a point with a normal, diffuse and specular coefficients for the front and back. As the particles are small and the density per pixel is low, the chance that occlusion occurs is small. They suggest rendering improvements, like depth of field and the use of a z-buffer - which is a standard approach for pixel-based rendering like raytracing, resulting in high-quality images and animations.

Time surfaces

'Time surfaces' are an extension to timelines. They are a scalar field converted from the flow field allowing for the spatial and temporal evolutions to be analysed. Westermann et al [59] applied this principle to level-set surfaces to propose a technique for topology-preserving smoothing of sampled vector fields.

Flow Volumes

A flow volume is the volumetric equivalent of a streamline. Max et al [60] presented a method for producing flow volumes using a transparently rendered tetrahedra. This can be rendered using the method proposed by Shirley and Tuchmann [39] which breaks down a volume into partially transparent triangles.

The concept was extended to unsteady flow by Becker et al [61] where the unsteady flow volumes are the three dimensional analogue to streaklines. As particle paths can become convoluted in time, further considerations and solutions to problems where presented allowing for unsteady flow volumes to be applied to different flow types including moving curvilinear grids.

2.3.3 Texture/Dense -based methods

Visualisation in two dimensions

Spot noise

One of the first texture-based techniques was introduced by Van Wijk [62]. Textures are generated by distributing intensity functions or spots, over the domain, where a spot represents a particle moving over a small step in time resulting in an elliptical stretch causing a streak [63]. Strongly varied velocity cannot be represented making the texture unable to show velocity magnitude information, therefore, De Leeuw and Van Wijk [64] created an enhanced spot noise algorithm. Improving the original by adapting the shape of spots to the local velocity field and filtering to remove low-frequency components.

Separately, De Leeuw proposed an algorithm for interactive spot noise visualisation [65] and visualisation of turbulent flow [66]. De Leeuw and Van Luere [67] compare spot noise to line integral convolution (LIC).

Line integral convolution

LIC was introduced by Cabral and Leedom [68] and is a popular and powerful technique for visualising the dense coverage of vector fields. The original method takes a vector field on a Cartesian grid and a white noise texture of the same size outputting an image where the noise has been smoothed along paths of the streamlines resulting the dense visualisation of the flow field. There is a one-to-one correspondence between the input grid cells and output pixels.

Research on LIC has been extended in different ways: directional cues, velocity magnitudes, support for non-Cartesian grids, real-time and interactivity, three dimensions and unsteady vector visualisation with time coherency [1].

Shen et al [69] add directional cues to LIC by combining animation and coloured ‘dye’ injection. Kiu and Banks [70] added direction cues and velocity magnitudes by using a multifrequency noise where the frequency of the noise is inversely proportional to the velocity magnitude in the vector field. Khouas et al [71] use furlike to generate images similar to LIC. Their technique allows for

the use of orientation, length, density and colour of furlike texture to represent the direction and magnitude of the vector field.

Computation of LIC for interactive visualisation has undergone lots of research. Stalling and Hege improved the performance by using box filters, minimising the number of streamlines required to be calculated [72] and later expanded it to piecewise polynomial filter kernels [72] [73]. Cabreal and Leedom [74] and Zöcker et al [75] present parallel algorithms for LIC.

Oriented Line integral convolution for 2D FlowVis

Oriented Line integral convolution (OLIC) was introduced by Wegenkittl et al [76] which is achieved by a sparse input texture and a directionally dependant convolution kernel. A faster version, called FROLIC, was later presented by Wegenkittl and Gröller [77] which was used to animate two-dimensional FROLIC images by Berger and Gröller [78]. FROLIC computes an approximate solution, approximating a trace by a set of small disks which individually have a constant intensity [63].

Löffelmann [79] et al use virtual ink droplets to visualise two-dimensional systems. The droplets, like OLIC, allow for the visualisation of direction, velocity and orientation of the flow but is more efficient at producing animations at a speed of around two hundred times.

2D Texture Advection

A data structure called a 'Motion Map', was used by Jobard and Lefer [80] that contains a dense representation and the information to animate flow. A motion map saves memory and computation time compared to LIC as only one image of the flow has to be computed and stored in the data structure instead of all.

Jobard and Lefer [81] also proposed a technique to visualise dense representations of time dependant unsteady vector fields based on a 'Lagrangian-Eulerian Advection (LEA) scheme'. For still images, their technique depicts the instantaneous structure of the flow and undergoing animation shows the motion dense particles would take if released into the flow.

Jobard et al [82] [83] achieved high performance in their unsteady visualisation techniques by utilising graphics hardware. They detail their temporal and spatial coherence, dye advection and feature extraction techniques as well as how they achieved high image quality by careful attention to noise frequency, edge effects and image enhancement.

Visualisation on surfaces or boundaries

Spot noise

De Leeuw et al [84] used a visualisation technique that pre-processes data and parameters for spot noise, to enhance similarity of the output images. The 'visual simulation' is based on the quantities chosen to be visualised and how they

map to the spot noise parameters. This extension to the spot noise algorithm allows for the comparison between flow surfaces to spot noise on surfaces.

Telea and Van Wijk [5] combine texture-based flow visualisation with three-dimensional arrows to expand the visualisation from two-dimensions. They used hierarchical clustering to produce simple but suggestive images where clusters are visualised with arrows to represent characteristics and allowed for user input to vary parameters to emphasis different flow attributes. In three dimensions, only arrows are three-dimensional with a plane of LIC data to visualise the remaining field.

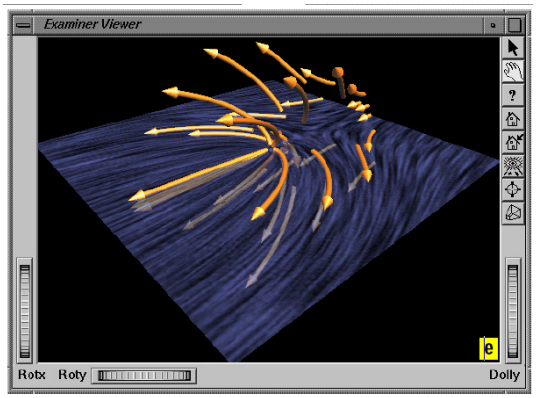


Figure 2.6: Example of 3D flow field simplification by Telea and Van Wijk [5]

LIC for boundary flows

LIC was extended by Forssell [85] and later with Cohen [86] to apply it to parametric surfaces found in curvilinear grids, which are used in CFD simulations. As an animated flow on a parametric surface can appear misleading they also adjust the LIC algorithm to use kernel phase shifts to create a variable speed animations with better results.

Teitzel et al [87] present a method that works on two-dimensional unstructured grids and directly on triangulated surfaces in three-dimensional space making LIC more applicable for real applications that often use unstructured grids. Mao et al [88] present an algorithm for combining white noise on triangle meshes in three-dimensional space and extend LIC to arbitrary three-dimensional surfaces, such as from the output from the Marching Cube algorithm or implicitly by part of a curvilinear or unstructured grid. Battke et al [89] extend LIC and extended algorithms that are limited to curvilinear surfaces by proposing a method that can handle general, multiply connected surfaces. Textures can appear distorted when mapped to a parametric surface therefore Mao et al [90] suggest a method to solve this using a multi-granular noise input image for LIC. Using Poisson sampling the computational space of a curvilinear grid can be resampled into a randomly distributed set of points that allow for the

reconstruction to the physical space of the grid.

Unsteady Flow LIC and Pseudo-LIC

Unsteady Flow LIC (UFLIC) was presented by Shen and Kao [91] [92] who propose a different convolution algorithm that consists of a *time-accurate value depositing scheme* and a *successive feed-forward* method which can produce time-accurate, coherent flow animations. The original LIC method by Cabral and Leedom [68] uses a *value gathering* scheme, instead of gathering, where pixels *deposit* their values along streamlines which is used to compute the convolution. The depositing scheme and feed-forward methods combine to create temporal coherence. Pseudo-LIC (PLIC), proposed by Verma et al [93], can generate flow visualisation for the comparison of streamlines and LIC-like images.

Visualisation in three dimensions

Line integral convolution

Due to the higher computational costs and memory requirements, the importance of interactive performance and occlusion are challenges when implementing LIC in three-dimensions. Interactive performance is tackled by Rezk-Salama et al [94], who used a three-dimensional texture mapping approach with clipping mechanisms to tackle occlusion.

Strategies for portraying three-dimensional flow using LIC were proposed by Interrante and Grosch [95] [96], which include the clarifying of distinct identities and selectively highlighting regions of interest. They also describe a technique for generating and rendering three-dimensional halos that help to indicate the presence of depth discontinuities and clarify the spatial structure of the flow.

2.3.4 Feature-based methods

These methods partition and describe the flow by structures in the vector field topology. To visualise a vortex, it must first be extracted from the data set.

Classification of Vortex Extraction Techniques

A selection of methods have been categorised in the table 2.2 to provide an overview of vortex extraction techniques.

Levy et al's Normalised Helicity

Levy et al [98] developed a method under the assumption that a vortex is located where normalised helicity, H_n, approaches ± 1. They extended previous helicity methods [97] [108] that used a threshold on helicity which were originally presented to resolve vorticity shortcomings, which has the potential to produce false positives in shear flow [109].

Table 2.2: Categorisation of Vortex Extraction Methods

	Region-Based	**Line-Based**	**Geometry/ Integration based**	**Vortex Boundary**
No invariance	Helicity [97] Normalised Helicity [98]	Reduced Velocity (3D) [99]	Curvature Center [100] Winding Angle [101] [100]	x
Gailian invariance	Q-Criterion [102] Δ-Criterion [103] λ_2-Criterion [104] Zhou et al [105] Chakraborty et al [106]	Reduced Velocity (2D) [99]	Banks and Singer [107][1]	Banks and Singer [107][2]

[1] Pressure is required.
[2] with a pressure threshold.

To represent the flow fields both velocity and vorticity fields have to be related. This has to be a scalar quantity so that viewing angle or cross-sections don't complicate visualisation. Helicity was chosen as it can distinguish between primary and secondary vortices and indicate their sense of swirl which is, with magnitude, both meaningful. Levy et al found that the use of helicity filters out the regions of low and high velocity but low velocity where the angle between the velocity and vorticity vectors are large.

Hunt el al's Q-Criterion

The vortex definition in incompressible flow used by Hunt et al [102] is a connected fluid region with a positive second invariant of ∇u [18]. Meaning, the areas where the vorticity magnitude is greater than the strain rate magnitude [19].

Chong et al's Δ-Criterion

The Δ-Criterion [103], however, defines vortices as regions where the eigenvalues of ∇u are complex and the streamline pattern is spiralling or closed in the local reference frame. For incompressible flow, where $P = 0$, the following equation is used to determine complexity

$$\left(\frac{Q}{3}\right)^3 + \left(\frac{R}{2}\right)^2 > 0 \tag{2.1}$$

and spirals are determined if two eigenvalues form a complex conjugate pair [18].

Jeong and Hussian's λ_2-Criterion

Jeong and Hussian [104] believed a pressure minimum wasn't enough as a detection criterion, due to unsteady irrotational straining which can create pressure minimums without a vortex present. By decomposing the gradient of the Navier-Stokes equation, $a_{i,j} = -\frac{1}{\rho}p_{ij} + \nu u_{i,jkk}$, to its symmetric, p_{ij}, and antisymmetric parts - where $a_{i,j}$ is the acceleration gradient, results in the vorticity transport

equation below [19] .

$$\frac{\nabla u S_{ij}}{\nabla u t} - v S_{ij,kk} + \Omega_{ik}\Omega_{ij} + S_{ik}S_{ij} = \frac{1}{\rho} p_{ij} \tag{2.2}$$

Removing the unsteady irrotational straining, $\frac{\nabla u S_{ij}}{\nabla u t}$, and viscous effects, $v S_{ij,kk}$, yields $S^2 + \Omega^2$. Where a vortex is a connected fluid region with two negative eigenvalues of $S^2 + \Omega^2$ [104], as it is symmetric, real eigenvalues only exist and if these values are ordered like: $\lambda_1 \leq \lambda_2 \leq \lambda_3$ then the vortex-identification-criterion is equivalent to $\lambda_2 < 0$ [19].

The λ_2-Criterion was designed for incompressible flow, if compressible, Cucitore et al [110] showed that further terms occur in equation 2.2 on the left-hand side and are required to be disregarded in addition to the unsteady irrotational straining and viscous effects.

Berdhal and Thompson's swirl parameter

Berdhal and Thompson [20] presented a method that introduced the intrinsic swirl parameter τ, defined as the ratio of the convention time to the orbit time, using the connection between complex eigenvalues from ∇u and swirling motion.

$$\tau = \frac{t_{conv}}{t_{orbit}} = \frac{|Im(\lambda_{1,2})|L}{2\pi|v_{conv}|} \tag{2.3}$$

Where, $\lambda_{1,2}$ are assumed to be the complex conjugate eigenvalues and λ_3 is purely real - yielding an exponentially decaying or growing solution along the real axis [20]. Therefore $Im(\lambda_{1,2})$ is the imaginary part of the pair, L is the characteristic length associated with the size convention time and V_{conv} is the convention velocity along with L.

Berdhal and Thompson claim that the parameter appears to be good at locating the core regions of vortices, however, they also highlight that the selection of convection velocity is still unresolved for three-dimensional applications and Jiang et al claims that the selection of a suitable threshold for τ to distinguish individual vortices is difficult [111].

Zhou et al's Swirling-strength Criterion

A similar approach to Berdhal and Thompson's, using the imaginary part of the complex eigenvalue of ∇u, was suggested by Zhou et al [105]. Zhou et al use contours of the second invariant, where the tensor has a complex eigenvalues and base their idea that the velocity gradient tensor in Cartesian coordinates can be decomposed as:

$$\nabla u = [d_{ij}] = [\bar{v_r} \bar{v_{cr}} \bar{v_{ci}}] \begin{bmatrix} \lambda_r & 0 & 0 \\ 0 & \lambda_{cr} & \lambda_{ci} \\ 0 & -\lambda_{ci} & \lambda_{cr} \end{bmatrix} [\bar{v_r} \bar{v_{cr}} \bar{v_{ci}}]^T \tag{2.4}$$

where λ_r is the real eigenvalue with a corresponding eigenvector $\bar{v_r}$ and $\lambda_{cr} \pm \lambda_{ci} i$ are the pair of complex eigenvalues with complex eigenvectors $v_{cr} \pm v_{ci} i$. The

strength of the local swirling motion is quantified by λ_{ci}, the imaginary part of the complex eigenvalue pair, and is referred to as the local *swirling strength* of the vortex. A distinctive feature is that the method also identifies the vortex core region as well as the strength and local plane of swirling. Chakraborty et al [106] clarify that although $\Delta = 0$ and $\lambda_{ci} = 0$ are equivalent, their interpretation is different for non-zero thresholds. The threshold for λ_{ci} should be set to zero but the results are smoother when set to a positive value.

Chakraborty et al's Enhanced Swirling-strength Criterion

The work of Zhou et al is further developed by Chakraborty et al [106], by including a local approximation of the non-local criterion from Cucitore et al [110]. Cucitore el al adds the concept of vortices as *structures* introducing *non-locality*, where all methods based on the velocity gradient tensor are characterised by local analysis and propose a Galilean invariant non-local criterion. They introduce a ratio to quantify the small change in relative distance between particles inside a small vortex structure.

$$R(x,t) = \frac{|\int_0^t u_a(\tau)d\tau - \int_0^t u_b(\tau)d\tau|}{\int_0^t |u_a(\tau) - u_b(\tau)|d\tau} \tag{2.5}$$

Where, u_a and u_b are the velocities of two fluid particles a, b in the flow and x is the position vector of their mid-point location. The numerator measures the evolution of relative distance between the two particles from time 0 to t, whereas the denominator is the difference of the particle trajectories in the same time interval. R is bound between 0 and 1 and it assumes lower values for pairs of particles belonging to a vortical structure [110]. Therefore a vortex is identified to be the flow region less than a threshold value. For some non-vertical uniform flow, the ratio R can be inferred to remain small, therefore, Cucitore et al propose their criterion be used in conjunction with the $\Delta > 0$ criterion [106].

Chakraborty et al propose three criteria for the identification of a vortex core:

1. The identification criterion should be Galiean invariant;
2. The local flow in the frame of reference translating with the vortex should be swirling;
3. The separation between swirling material points inside the vortex core should remain small.

They state that the Δ or swirling strength criteria by themselves satisfy only the first two requirements, where the combination of a threshold value for R satisfies all three. However, the approach is more computationally involved as it is global and requires evaluations of particle trajectories. Therefore the local approximation is introduced. By examining the projected motion of a fluid particle it can be shown that the period for a single revolution is $2\pi/\lambda_{ci}$. Two

points with an initial separation of r_0, after n revolutions, will be separated by r_f. The distances can be expressed in terms of the eigenvalues of ∇u as:

$$\frac{r_0}{r_f} = exp\left(2\pi n \frac{\lambda_{cr}}{\lambda_{ci}}\right). \tag{2.6}$$

The ratio $\frac{\lambda_{cr}}{\lambda_{ci}}$ is the *inverse spiralling compactness*, which measures the spatial extent of the local spiralling motion. It is used to approximate the orbital compactness, where $\frac{\lambda_{cr}}{\lambda_{ci}} = 0$ results in a perfectly circular path. This method has an advantage as it can restrict outward spiralling motions. A vortex is defined as regions where $\lambda_{ci} \geq \epsilon$ and $\lambda_{cr}/\lambda_{ci} \leq \delta$ where ϵ and δ are positive thresholds [19] [106].

Sujudi and Haimes' Reduced Velocity

Sujudi and Haimes [99] proposed a method based on critical point theory, where a critical point is a point in the flow field where all three velocity components are zero and the streamline slope is indeterminate [103], and uses eigenvalues and eigenvectors of the velocity gradient tensor. Their algorithm has been implemented for tetrahedral cells and proceeds, for each cell, as follows:

1. Linearly interpolate the velocity within the cell;
2. Find the eigenvalues of the velocity gradient tensor, ∇u and continues only if the values has one real and a pair of complex conjugate eigenvalues;
3. Project the velocity onto the plane normal to the eigenvector belonging to the real eigenvalue calculating a reduced velocity;
4. Linearly interpolate each component of the reduced velocity to obtain,

$$w_i = a_i + b_i x + c_i y + d_i z \tag{2.7}$$

   where i = 1,2,3;
5. By reducing w in equation 2.7 to zero then the equation can be reduced to the equation of two planes, whose solution is a line;
6. If the line intersects the cell at more than 1 point, the cell contains a centre - which is defined by the line segment formed by the two intersection points of a local swirling flow.

The algorithm, however, is sensitive to the strength of the swirling flow and can result in incoherent structures or, if the gird cells are large has difficulties finding structures [1]. Kenwright and Haimes [112] studied and concluded that the eigenvector method is effective in many applications. However, the algorithm doesn't produce continuous lines and that swirling flow doesn't guarantee the presence of a vortex as they can from in boundary layers.

Banks and Singer's Self-correcting Predictor-corrector Method

Banks and Singer [107] proposed a self-correcting predictor-corrector method to be able to tolerate transitional flow. The method produces an ordered set of points that approximate a vortex skeleton and produces lines similar to vorticity lines, which wander away from vortex cores but instead self-correct and reconverge to the centres. Before growing the skeleton, the core of the vortex is found. The core is indicated if two threshold values for low pressure and a large vorticity magnitude are satisfied. However, the threshold values can be reached without the grid point being part of a vortex core [111]. This is because the existence of a local pressure minimum doesn't guarantee the existence of a vortex and vorticity is not suitable as it cannot distinguish between pure shearing motions and the actual swirling motion of a vortex [1].

Roth and Peikert

Roth and Peikert [113] introduced a higher-order method where they detect the parallel alignment of the velocity and acceleration vectors [6], to better capture slowly rotating curved or bent vortices [1]. They introduce two attributes to core lines, strength of rotation and quality of the solution which allow for a threshold to be introduced to remove weak vortices. Separately Roth and Peikert introduce a *parallel vectors'* [114] operator '||' to mathematically describe a selection of previously developed methods based on zero curvature, ridge and valley lines and extremum lines. Their definition is as follows

Definition 2.3.1. Parallel Operator '||'. For two given n-dimensional vector fields v and w denoted by v || w ("v parallel w") the operator which returns the set S restricted to nondegenerate solutions [114], where S is expressed as:

$$S = \{x : v(x) \times w(x) = 0\}. \tag{2.8}$$

Their definition can also describe Sujudi and Haimes (equation 2.9), Banks and Singer (equation 2.10), and the Helicty criterion (equation 2.11), where u is interpreted as a steady velocity field and p is the minimum in the plane orthogonal to $\nabla \times u$.

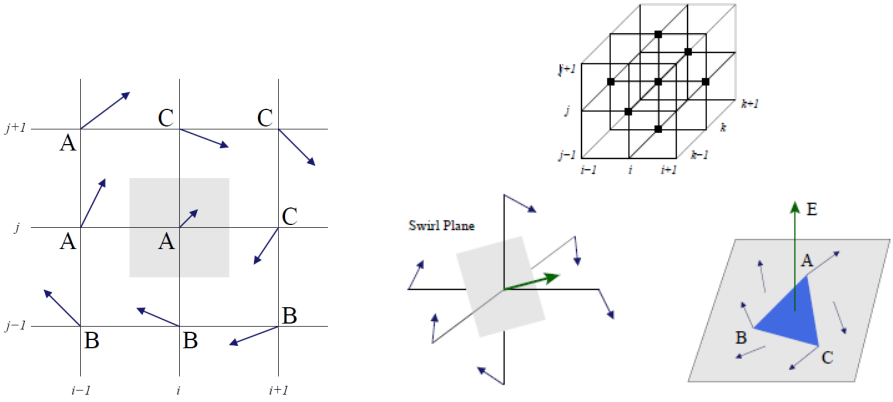
$$u||(\nabla u)u \tag{2.9}$$

$$\nabla \times u||(\nabla u)p \tag{2.10}$$

$$u||\nabla \times u \tag{2.11}$$

Portela

Portela [101] define a vortex as a central core region surrounded by swirling streamlines [6]. The two-dimensional method verifies if the winding angle around a grid point is a multiple of 2π. This method has a simplification suggested by Sadarjoen et al by using the summation of signed angles along the streamline



(a) The neighbours of (i, j) satisfy the *direction-spanning* property

(b) The neighbours of (i, j, k) satisfy the *direction-spanning* property on the swirl plane

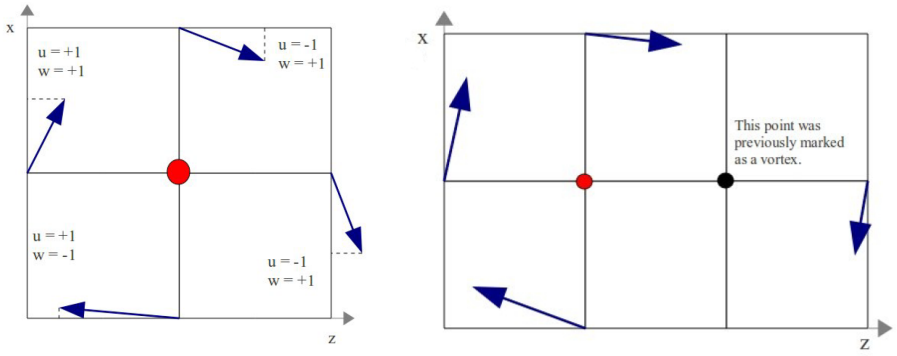
Figure 2.7: Two and three dimensional vortex core detection algorithms [6].

instead. By their definition, the winding angle of a streamline is a measure of the cumulative change of direction of streamline segments [111]. It is important to highlight that winding angles are only meaningful in two dimensions [111] and that spiralling streamlines are obtained for an observer moving with the vortex [18].

Jiang et al's Combinatorial Method

An approach based on ideas from combinatorial topology was presented by Jiang et al [6] who uses a labelling scheme based on *Sperner's Lemma* to identify centres of swirling flows. The scheme is used to classify points belonging to a vortex core by, for each grid point, examining the immediate neighbours' velocity vectors to see if they satisfy the *direction-spanning* property. This is followed by a growth algorithm, which recursively examines all vertex-, edge- and face-connected neighbours. Although the algorithm examines each grid point, it is not a criteria and can use centres of grid cells or interpolated positions - with the added interpolation cost. Since laminar flows do not have switching or swirling flows in a local neighbourhood, the direction-spanning property implies the presence of a vortex core region.

The two-dimensional algorithm involves two passes over the point grid where the first pass applies direction labelling to all the velocity vectors and the second checks for the *direction-spanning* property among immediate neighbours. The algorithm is shown in figure 2.7a for a structured grid, where the core region can be identified through observation. The *direction-spanning* property is a term used to describe a fully labelled triangular cell, occurring when each vector at each vertex of the cell point in a unique direction range.



(a) Identification Stage (b) Growth Stage

Figure 2.8: Two step algorthim as presented by Holmén

As the *direction-spanning* property identifies two types of flow a 'topological clean up' step is performed using eigenanalysis. The clean-up eliminates detected grid points that don't meet swirling flow conditions which, as noted by Berdhal and Thompson [20], Sujudi and Haimes [99] and Perry and Chong [115], imply conjugate pair eigenvalues in the velocity gradient tensor. In two dimensions the algorithm has linear complexity.

In three dimensions vortex cores are harder to identify but the algorithm still has a linear complexity. This is because they compute the core direction before applying their two-dimensional algorithm to velocity vectors that are projected onto the *swirl* plane. In the third dimension, their algorithm only requires a single pass over the data, since the directional labelling is applied at each grid point. Figure 2.7b shows the algorithm for a structured grid in three dimensions.

Vivianne Holmén

Similar to Jiang et al, Vivianne Holmén [19] proposes a two-stage method analysing the velocity field directions of the nearest neighbours. After labelling the velocity components each point's neighbours are analysed and categorised as a vortex core if the sum of the signum values equal 0 (figure2.8a). Like Jiang et al, an extra check also has to be performed to remove false vortices that were labelled. In their growth stage, the algorithm passes over the data again except ignores the previously identified cores, shown in figure 2.8b.

Shock wave extraction

Shock waves are another important feature that can occur in flow data sets, but are unlikely to appear within the data provided by Scuderia Toro Rosso. Shock waves are characterised by surfaces that separate zones of different pressure, density and velocity. Therefore, shockwaves can be extracted using edge detec-

tion and image processing. Detecting shock waves in two dimensions has been examined, however, cannot be directly applied to shocks in three dimensions. Algorithms are compared by Ma et al [116] who also present their technique using isosurfaces.

Separation and attachment line extraction

Separation and attachment lines are other features that can occur and are aerodynamically undesired behaviour as they can cause increased drag and reduced lift [117]. Design of aerodynamic parts at Scuderia Toro Rosso focuses on retaining the flow lines and detaching them in desired locations to produce vortices. The measurements taken using PIV do not allow for the exploration of these lines unless the data is extended to three dimensions and taken from a side profile. Helman and Hesselink use vector field topology to visualise flow fields and present topology schematics for a cylinder-like in figure 2.9. Kenwright provides an overview of techniques for visualising separation and attachment lines in addition to two algorithms, an automatic detection algorithm [112] and a parallel vector algorithm [118].

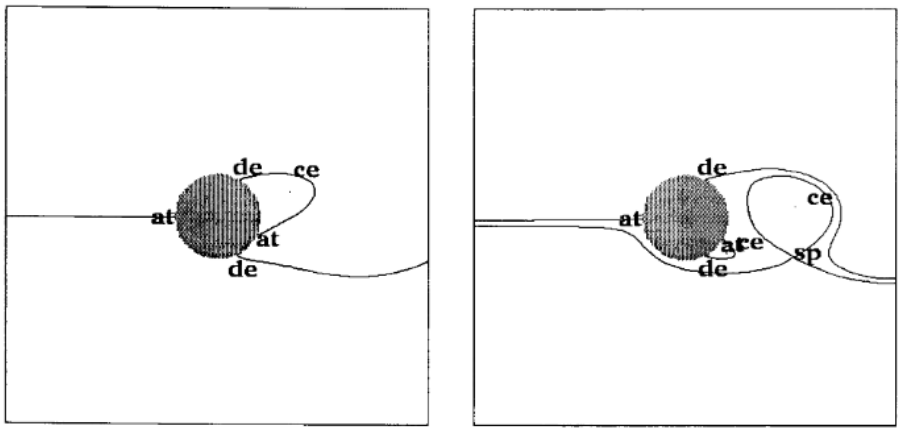


Figure 2.9: Topology schematics for two time steps in the computed flow around a circular cylinder.

Other types of features

Furthermore, flow features can be divided into recirculation zones and boundary areas. Where Hunt et al [102], gives criteria to divide flow into three areas: eddies, streams and convergence zones.

2.4 Feature tracking and event detection

The movement of vortices is a challenging aspect of vortex extraction [15]. The correspondence problem is used to describe the process of determining the correspondence between features in successive time steps. While tracking a feature it can evolve, interacting with other features and undergo shape changes. To be able to describe a feature's evolution - event detection algorithms are employed. There are two approaches to solve the correspondence problem.

2.4.1 Region correspondence

Region correspondence is the use of regions, obtained by feature extraction, for comparisons between time steps. Haak et al [119] use a minimum distance or maximum cross-correlation criterion to find the correspondence. Weigle and Banks [120] use a recursive contour meshing to efficiently extract features construct smooth animations.

Silver and Wang present a volume-based feature tracking algorithm [121] [122] [123] improving on their previous work [124], where features were tracked using their centroids. They found that their previous method was not reliable, resulting in noticeable errors. These problems have been corrected by analysing a feature's volume and performing *template matching*. Silver and Wang use octree's to reduce memory overhead where an overlap test and difference calculations can be computed at the same time by 'merging' trees in different time steps.

2.4.2 Attribute correspondence

The comparison of features are performed depending on a feature's attributes - like position, size, volume and orientation, which can be calculated in the extraction phase. Samtaney et al [124] use attribute values with user tolerances. Their equations for position (2.12) and volume (2.13) are shown below:

$$dist(pos(O_{i+1}), pos(O_i)) \leq T_{dist} \tag{2.12}$$

$$\frac{vol(O_{i+1}), vol(O_i))}{max(vol(O_{i+1}, vol(O_i)} \leq T_{vol} \tag{2.13}$$

where the equations have been adapted slightly by Post et al [1] from their original notation. O_i and $O_i + 1$ represent objects in their respective timesteps i and $i + 1$ and T_{dist} and T_{vol} are the tolerances.

These equations can be used to test for certain events, for example, a bifurcation event can be determined if the total volume before the event approximately equals the sum of the parts afterwards.

Motion analysis has been suggested with the use of markers or tokens by Sethi et al [125] who examine trajectories in property space using a greedy exchange and simulated annealing algorithm, where property space is when properties or attributes are points in the space.

Reinders et al [126] [127] assume that properties evolve consistently for their tracking system, giving features a predictive behaviour. They use a series of correspondence functions, with their thresholds and weights, to detect a correspondence between a prediction in a different time step and a candidate feature.

A candidate can be all or unmatched features. The former method can result in more than one feature being added to a path which is then required to be resolved, the latter method is more efficient but then is dependent on the order features are tested - solved by tightening restrictions and gradually reducing in latter passes. If a feature matches a path, it is added to evolution and the search is continued, this continues until the path ends or the last frame is reached. Multiple paths can share the same feature, where multiple candidates match - or when a candidate has already been added to a path. In this scenario the path with the best confidence index, using equation 2.14's, is used.

$$\text{Conf(path)} = 1 - e^{-\frac{t}{\tau}} \tag{2.14}$$
$$\text{with } t = \sum_{i=1}^{edges} C_i$$

where τ is a growth factor - equal to minimal path length, an 'edge' is a connection between two features and C_i is each a correspondence factor. The minimal path length is used to avoid erroneous or coincidental paths, usually set to four or five frames. Correspondence factors are used, similarly to Samtaney et al [124], which give a positive value if within the user given tolerance. Each correspondence function has an associated weight which is used to create a weighted average across all functions.

2.4.3 Event detection

After feature tracking the next step is to perform event detection. Samtaney et al [124] outlined five events: continuation, bifurcation, amalgamation, creation and dissipation. These are tracked by Reinders et al [126] where they are called continuation, split, merge, birth and death, they also detect entry and exit events where a feature moves into or out of the data set. Like feature tracking, attributes can be used to create functions to determine if an event is occurring. Bifurcation, for example, could be determined if the volume of a feature in time step i is roughly equal to that of two features in time step $i + 1$. Reinders et al test for amalgamation, or death, by reversing the timesteps but performing the same check.

2.5 Visualisation of features and events

There are several techniques to visualise the tracked features. The simplest is to show nodes that match the criterion from the initial extraction phase, another is to use isosurfaces. Selected streamlines could be used to extend feature information, without overloading on extra information and occluding important

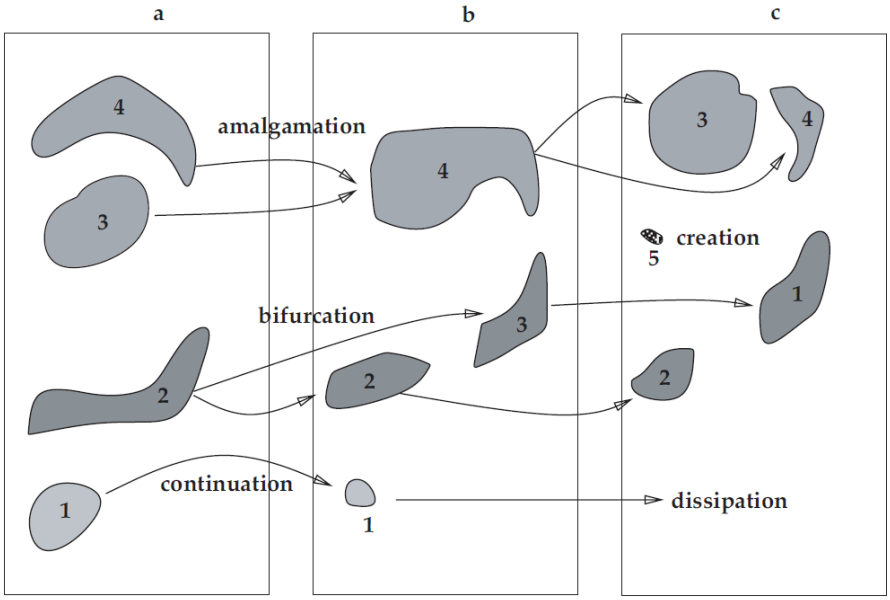


Figure 2.10: Types of events as introduced by Samtaney et al

data. Rainders et al extract two-dimensional vortices before extending them to three dimensions and allow for users to interact with an animation to visualise the time dimension.

2.6 Chapter Summary

There are a large number of techniques that have been presented to tackle the visualisation of flow. Direct, Geometry and Texture -based methods have been examined to visualise flow two dimensions, boundaries and three dimensions. Additionally, feature-based methods can extract vortices from data allowing them to be tracked and their lifespan visualised. Many of the methods researched are user and context dependant, however, the following trends have been observed. The expansion to three dimensions often includes the problem of occlusion and requires special consideration to prevent it. Interactive elements can reduce the amount of occlusion that arises, allowing users to change thresholds and explore data, however, can decrease the performance of the visualisation method. Furthermore, the extension of existing methods to encode more information, like the extension of streamlines to stream ribbons and streamtubes, is deemed useful allowing newer methods to inherit the advantages of its predecessors. The next chapter uses the research and conclusions made to inform a set of requirements that allows for the development of the system to visualise the data provided by Toro Rosso.

Chapter 3

Requirements

This chapter provides an overview of the requirements capture process that formulated the requirements. Requirements were elicited from analysing previous solutions, the literature review and through discussions with engineers at Toro Rosso. The requirements are then separated by type into functional and non-functional categories, where a priority is assigned using the key terms; *Must*, *Should* and *May*.

3.1 Capture/Elicitation

To capture the requirements the state of the art was explored, followed by discussions with aerodynamicists at Toro Rosso. Requirements were orientated around finding a solution to the extraction and visualisation problem, with particular attention, paid towards what worked with current methods in the field as well as those employed at Toro Rosso.

3.1.1 Literature and Technology Review Findings

The research conducted during the literature review allowed for the examination of possible methods to visualise the data set. Analysing the techniques for the two-dimensional representation of data, direct-based techniques allow for the comprehension of velocity information, using arrows in the form of quiver plots. Furthermore, texture-based techniques are suitable for animation in two dimensions, however, are predominantly used for surface flow and struggle in three dimensions. Feature extraction allows for the segregation of features from the flow and less relevant data to be disregarded - reducing file sizes.

3.1.2 Discussions with Toro Rosso

Discussions were held at Scuderia Toro Rosso, for which the transcripts have been collated and added to Appendix B.1. The discussions were informed by the literature review and outlined the requirements of the project. Questions

were posed to the team to gain an understanding of their requirements and how they may fit into the researched techniques including the current system and processes.

At Toro Rosso, there is a particular interest in *known* vortices, for example, the *Y-250* and *Z-0*, which are commonly known by the coordinate they appear at. Generalising this idea to unknown vortices disrupts their current streamline seeding strategy, which requires the position of the vortex in advance, giving way to methods for identifying unknown vortices. With this focus on the visualisation of features, in particular vortex core's, visualisation methods designed to represent the entire domain were found to be unsuited and focus can be shifted to feature extraction techniques. Extracting features from the data could also allow to implement tracking and event detection algorithms, like those presented by Samtaney et al [124] and Reinders et al [126].

Furthermore, the discussion illustrated that performance is important when visualising the extracted data where an aerodynamicist states that the previous method using eigenanalysis was slow. To improve performance in the visualisation tool it may be useful to split the processes up and cache files to allow for the user to revisit the same sample without reprocessing.

3.1.3 Data analysis

The data is infrequently generated and saved in a custom format making the tight coupling with the project a weakness. Although unlikely, the format the data is not guaranteed to remain a constant. Every effort to segregate the effect of changes made to the data on the visualisation must be made to ensure viability. These processes could be run after the data is generated as a post-processing step, prior to the use for visualisation. These changes broaden the system's scope allowing other methods, like CFD, to be used by the system.

3.1.4 Elicitation Summary

Taking the literature review, discussions and data into account the following conclusions can be drawn. Firstly, the election process is primarily focused on the application of existing research and within the constraints set by the data. It is noted that the existing technique employed visualises streamlines on an internal web-based system. Although there was no mention of possible extensions to their system during the discussions, it was felt that the result of this research should also be web-based, to allow for the possibility in the future.

Secondly, it shows that the current streamline strategy could be extended by using an automatic technique like one proposed by Turk and Banks [33] or Jobard and Lefer [34], referenced in section 2.3.2, who extended streamtubes allowing for more information to be encoded. The discussions emphasise the visualisation vortex cores from the data where geometric, direct (section 2.3.1) and textural (section 2.3.3) techniques are more useful at representing regions that require the analyst to interpret the data. Furthermore, these comments gave focus to the identification and visualisation of unknown vortices and events

that can occur. The emphasis on vortex cores puts the focus on feature extraction techniques, in particular those that are topological. Although there are a lot of different feature-based techniques they often require certain parameters, such as pressure which is unavailable in the data, whereas topological solutions use particle velocity information, which is available, to identify vortex cores.

As discussed in section 3.1.2, a performance gain could be obtained by segregating processes - preventing repeating work. Given the number of frames possible within a data sample, the associated processes for vortex identification could be separated from visualisation into a front and back -end. Our use of the term 'back-end' is atypical where it is traditionally used to describe server-side operations that support the primary operations. Here, our definition is atypical as it creates a standard format for the system, stripping redundant data and reformatting it in a manner to make interrogation more useful. The processes described can be considered to be either, post-processing or pre-processing. Due to the belief that there is a distinction between the terms, where post-processing implies a tighter coupling with the external system, we classify the back-end as a pre-processing step.

3.2 Key Requirements

The conclusions drawn after gathering the requirements can now be formulated into requirements. This section is split by type of requirement and by the system area. Along with each, a key term has been used to indicate a priority level:

- *Must* - The requirement has been deemed essential and is expected to be completed at the end of the project;
- *Should* - The requirement has been deemed beneficial and its completion would support and round the project in its scope;
- *May* - The requirement has been deemed useful and could provide benefit if completed at the end of the project.

3.2.1 Functional Requirments

Back-end

F-B1. The system *must* use a data interface;

F-B2. The system's functionality *must* be split into a front-end and back-end;

F-B3. The system *should* allow for parameters to be changed with ease;

F-B4. The visualsation system *must* be able to use the raw 'dat' format;

F-B5. Coupling with the given data format *should* be minimised;

(a) The data *should* be reformat for the tool after it has been collected;

F-B6. Nodes in the data *must* be able to be extracted from the data source;

(a) A vortex *must* be able to be identified from data;

(b) A vortex *should* be tracked;

F-B7. Vortex information *must* be able to be saved for the front-end;

Front-end

F-F1. The visualisation system *should* allow for interactive control;

F-F2. The visualisation system *may* use animation;

(a) If animation is used, the user *should* have control to start it manually

(b) If animation is used, the user *must* have control over what frame to visualise

F-F3. The visualisation system *may* use three dimensional representation;

F-F4. The visualsation system *should* using a web based interface;

F-F5. User *should* be able to alter parameters to achieve different results.

3.2.2 Non-functional Requirments

Back-end

NF-B1. The system *must* be maintainable;

NF-B2. The system *must* save the data in a manner that minimises the size;

NF-B3. The data fromatted *must* be able to be saved in intermediary steps;

NF-B4. The data fromatted *must* be able to be loaded at intermediary steps.

Front-end

NF-F1. The system *must* provide a good user experience;

NF-F2. A user *must* be able to interpret the data;

NF-F3. The system *must* be easy to use;

NF-F4. The system *must* scale for different window sizes; .

3.3 Chapter Summary

The requirement election process was orientated to give a framework around the scope for the system, where the discussions provided insight and narrowed the focus to form a set of requirements focused on the effective visualisation of the data. The requirements are orientated around producing an efficient visualisation of vortex cores from the data using a web environment to host the end product. The web product only serves to illustrate the research undertaken to chose a visualisation method that is appropriate for this data, requirements around user interaction with this environment have a lower priority in order to reduce the time spent on auxiliary tasks.

Although some discussions mentioned possible expansion and application to three dimensions, due to COVID-19, Toro Rosso entered an FIA enforced shutdown. During this period restrictions prevented the second data set from being obtained. This has limited the scope of the project to the investigation of time dependant, two-dimensional methods.

Chapter 4

Design

This chapter examines how the requirements outlined in chapter 3 have been developed into design elements. The chapter begins by discussing the system architecture which provides a strong foundation for the rest of the design. Attention is paid to the user interface which supports the main visualisation element. Particular scrutiny is given to visualisation design to allow for the efficient interpretation of data. Each design choice is discussed and weighed in detail to explain the rationale behind the final design choices.

4.1 System Architecture

As described in the requirements (Section 3.1.4), the processes should be split into a front and back -end, allowing for pre-processing to be performed on demand by the user. The architectural sections have therefore been split into their respective sections to simplify the design process. Considerations for each part will be outlined in this section before combining them to form the complete architectural design. As the front-end motivated the back-end it is covered first in this chapter. In later chapters, the order will be reversed.

4.1.1 Front-End Architecture

Considerations

Requirement **F-F4** that the system should be a web-based environment influenced the design of the front-end architecture. As the requirement was inferred from discussions and not declared, the design of the front-end was delegated, sacrificing control, to Juypter notebook (Section 6.1.2). Jupyter notebook, as opposed to a traditional web development technology stack, removes the ability to customise the web environment but allows for visual code and interactive graphing - satisfying requirements **NF-F1**, **NF-F2 NF-F3**. Therefore, more focus was spent on the visualisation design and development.

Jupyter notebook is a web-based interactive environment that allows for the division of code into cells - combining the use of code, rich text, images, animations and plots. Jupyter notebook allows the use of Python but can also extend to other languages, such as C++, Java 9 and Matlab. Furthermore, a notebook can be exported as HTML. By using Python within a Jupyter notebook, access to a large variety of data science and visualisation libraries are available and can be wrapped for simple representation within a web environment. Several Python graphing libraries were examined: Matplotlib and Plotly which are libraries that allow for visualisation in python. Matplotlib [128] is a popular python visualisation library that allows for the creation of interactive visualisations. Plotly [25] was created for animated interactive plots that can be hosted online, reasons for choosing Plotly are discussed in section 6.1.2 in detail. The Plotly library handles stylistic choices of user interface elements and allow for the interaction with the visualisation of data.

Summary

The front end will be comprised of a single web page generated by Juypter notebook to HTML. Juypter notebook has been chosen to allow for dynamic code interaction and will be able to load serialised data from the back-end for visualisation using the library Plotly. Figure 4.1 shows a representation of the front-end architecture components discussed.

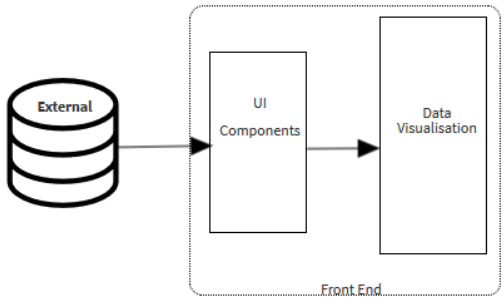


Figure 4.1: Front-End System Architecture

4.1.2 Back-End Architecture

Considerations

The back-end performs as a pre-processing step to prepare the data for visualisation. Made independent of the external system, the importing and exporting data via files is a manual process. This removed the need for an API to interact with the network to dynamically ingest data and simplified the development.

The back-end is required to perform a series of processes to extract the vortices in preparation to be visualised, saving the output in the form of a file.

As the front-end needs to read this file, the design is informed by the choices made for the front-end, therefore, the back-end will also use Python to allow for the files to be shared across the two parts using native object serialisation.

Summary

The back-end described uses an atypical definition where it consists of a series of processes to prepare the data before visualisation. The back-end loads raw data in the form of text files and outputs processed files as serialised python objects for the front-end. The back-end is represented in figure 4.2

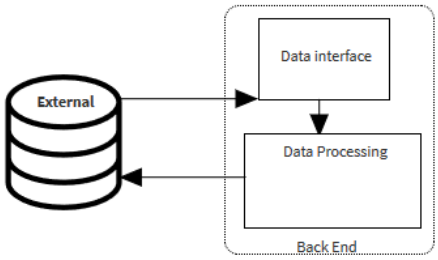


Figure 4.2: Back-End System Architecture

4.1.3 Architecture Summary

The system architecture, designed to fulfil the requirements gathered, has been designed and combined in figure 4.3. The back-end is responsible for preparing the data for ingestion by the front-end, it imports the raw text files and exports the serialised binary files to a shared network drive allowing for the front-end to read these files and visualise them, using Plotly.

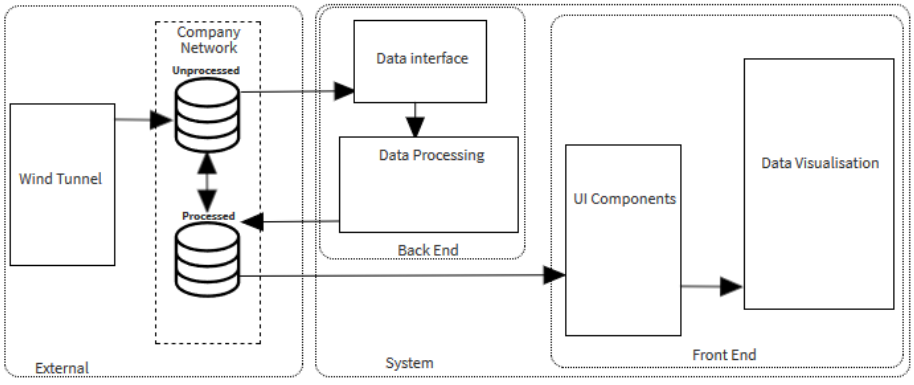
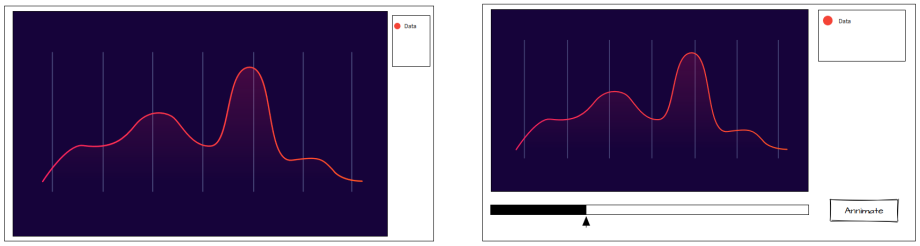


Figure 4.3: Detailed System Architecture



(a) Low-Fidelity Prototype A (b) Low-Fidelity Prototype B

Figure 4.4: Low-Fidelity Prototypes A and B

4.2 User Interface Design

The user interface is coupled with the visualisation library where the user interface should allow the user to interact without distractions and increase the total experience. The overall interface is controlled by the Python library Plotly and is responsible for the generation of charts and auxiliary interface elements. To get an idea of the layout a prototype is designed and analysed to determine viability, before the user interface is examined in detail.

4.2.1 Prototyping

A low-fidelity prototype was used as a starting point in the form of a wireframe. The prototype was designed with the requirements in mind as described Chapter 3 and later discussed in chapter 6. The low-fidelity prototype is represented in figure 4.4 and was made using Mockflow [129]. The figure 4.4a is an initial mockup design. If animation is used then the design has been updated with animation related user interface elements, shown in figure 4.4b. In the figure, the line plot represents where the visualisation method, with the legend to the right and a timeline for the animation below. The arrow under this timeline is an arrow which represents the current frame displayed. The frames can be played using the animate button on the right of the timeline.

The requirements in mind during the design of this prototype are outlined below, where: *F*: Functional, *NF*: Non-Functional, *F*: Front-End.

- **F-F2a**: The potential for animation is considered and interface elements have been added to the modification of the basic wireframe;
- **F-F4**: The outline of the wireframes represent the border of a webpage;
- **NF-F2**: The graph is made to be the center of attention;
- **NF-F3**: The simplicity of design to make the tool easy to use.

Analysing the design choices

The addition of fine animation controls should be added to allow for more control over the frames viewed, these will be added to 'Prototype B' underneath the timeline on the left-hand side. There is the potential that the user would want to re-examine select frames or control the frame rate of their animation - which is made possible by these controls. This improvement is shown in figure 4.5.

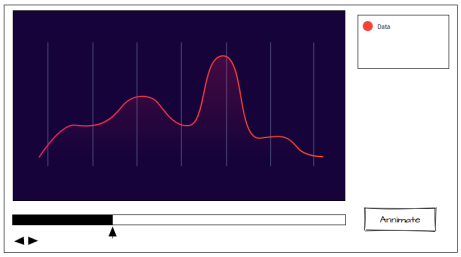


Figure 4.5: Low-Fidelity Prototype B

4.2.2 User Interface

User interface elements will be controlled by the library, this allows for the focus to be distributed to the visualisation method however, this reduces total control over the resulting interface.

4.3 Visualisation Design

Special consideration is given on resolving weaknesses identified from existing methods in a manner that satisfies the requirements previously established, such as occlusion, which frequently arises when extending techniques to three dimensions.

4.3.1 Using a three-dimensional representation

The temporal dimension of the data allows for the representation as a three-dimensional volume. It was hypothesised that a volume could be constructed where its depth represents the path of a vortex in time and its cross-section at each frame represents the size of the vortex in that frame, inspired by Reinders [130] shown figure 4.6. However, by using the depth to represent time, the occlusion problem is made more important and further increases with the number of frames visualised. To solve this, the axis used to represent time could be changed to try to minimise path occlusion, however, this becomes a balancing act between dimensionality of your data. The problem could be partially solved by introducing an element of interactivity allowing the user to dynamically rotate the plot, changing the axis angles relative to the user.

Figure 4.6: Three dimensional path representation of a vortex

4.3.2 Annimation

Animation allows for the representation of data to remain in two dimensions where each frame represents a different time step. Animation only serves as an extension to existing visualisation methods, to be able to take advantage of the benefits, the data still needs to be visualised.

4.3.3 Parrellel Contextual Annimations

Via animation, the temporal data can be collapsed into two dimensions. An animated scatter plot of vortex cores allows for the visualisation of the vortex position and size in a single plane and for the movement to be inferred from watching the animation. To increase the coherency between frames, two distinct visualisation techniques are presented.

Parallel Annimations

The first visualisation method was inspired by colour coding on slices (Section 2.3.1), which uses clipping to prevent the occlusion problem. The proposed method depicts the frames as a stack, where the transparency of each frame increases the deeper into the stack. As the animation proceeds into the stack, the top is 'popped' off and a new frame deeper in the stack is shown. This modifies a typical serial animation to a parallel form, where each thread starts at an offset from the initial frame and displays their results at each timestep. A prototype has been constructed to illustrate this mechanic visualising the stack in figure 4.7 and the individual frames in figure 4.8.

In the prototypes, different layers are differentiated by opacity and colour. When watching the parallel animation, the transparent layers give the illusion that the frames are moving as they become more opaque and naturally change size, allowing the user to manually track the evolution of the vortex cores. However, the method has the potential of overcrowded data frames with large amounts of data being represented in a single frame.

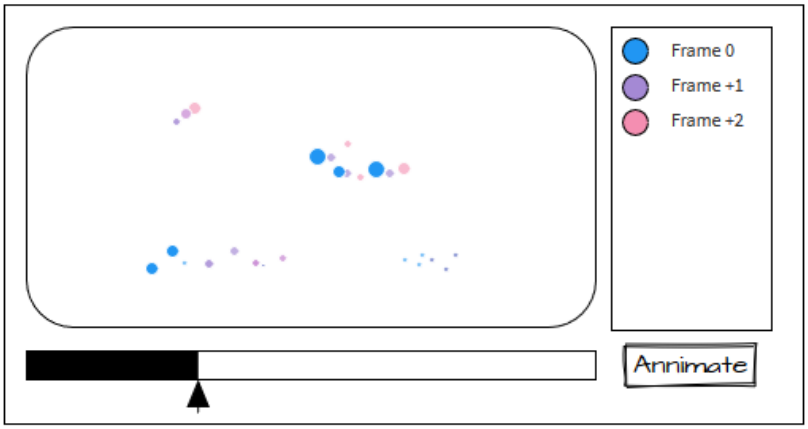
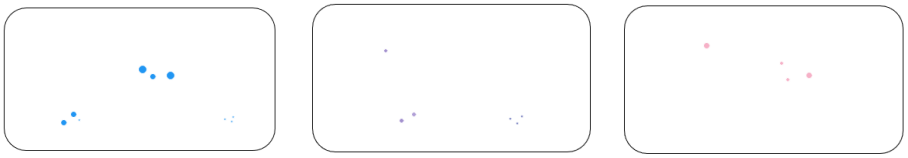


Figure 4.7: Parallel Animation Prototype



(a) Frame 0 (b) Frame 1 (c) Frame 2

Figure 4.8: Individual Frames

Contextual plots

The second visualisation method extends a normal 'serial' animation with the use of glyphs to give a visual representation of the surrounding movement. Research undertaken during the literature review has shown that the extensions that intuitively encode information is beneficial. This is shown by Nielson et al [45] for the extension of stream ribbons into the streamtubes. Therefore a scatter plot has been extended with the use of arrows to encode information about the surrounding movement.

If the scatter plot is combined with a quiver plot it would result in a large number of changes per frame and would be hard to take in information. An automatic selection algorithm is therefore presented to select nodes of interest to represent as arrows. The result of successfully incorporating arrows to the scatter plot could look like the prototype in figure 4.9. This would reduce the clutter of the plot and still allow the glyphs to illustrate the movement of the flow.

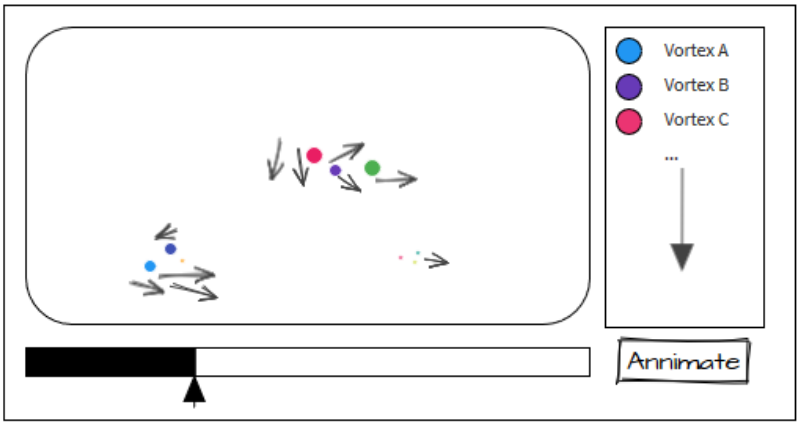


Figure 4.9: Contextual Animation Prototype

Parrallel Contextual Annimations

A third method, named a parallel orientated contextual animation, is also proposed by combining parallel animations and contextual plots. By taking advantage of interactive elements a combined technique can be implemented where the user can choose to only use either sub-component. This allows for the current frame and the next couple to be rendered with the addition of quivers to build the contextual information around the points. To prevent overcrowding, quivers will only be shown for the current frame. Classifying quivers to their originating frame is challenging due to the variables you can change to signify its uniqueness.

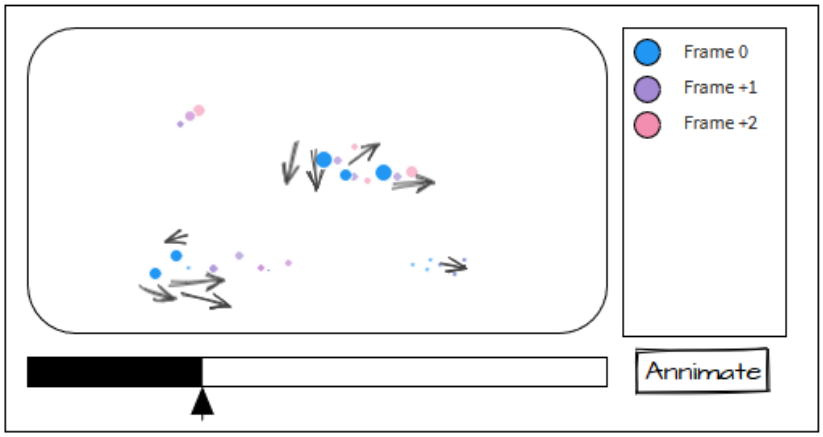


Figure 4.10: Parallel Contextual Animation Prototype

4.3.4 Section Summary

This section introduces two techniques that can also be combined to visualise complex spatio-temporal data. The first technique, parallel animations, render multiple frames with modifiers such as opacity or size on the latter frames to give precedence to the primary frame. The second technique, contextual plots, improves serial animations with the use of quivers to provide intuitive contextual information about the surrounding movement. Combining these two techniques allows for the comprehension of the surrounding movement and its effect on future frames within a single frame of data.

4.4 Chapter Summary

This chapter details the design of the system to visualise the complex spatio-temporal data. The system architecture has been split into two parts, front-end and back-end, to be designed, that work in tandem. These parts are run by the user to process and visualise the data on-demand and share information via saved serialised python objects. The implementation for the user interface is discussed and possible visualisation elements are designed including the use of *parallel animations* and *contextual plots*. The proposed visualisation methods allow for fine control over the frames viewed and allows for more data to be interpreted per frame, without the changes made the amount of data would be overwhelming and reduce the comprehensibility.

Chapter 5

Implementation

This chapter will discuss the implementation of the system as structured by the decisions made during the design chapter. The implementation has been split into the sections back-end and front-end for clarity and in order of development undertaken. The back-end is further dissected into the subsections: the data interface, extraction and tracking. The front-end details the Plotly implementation. Within this chapter, references are made to pseudocode algorithms to support details made in this chapter. They are labelled 'Algorithm i' and are contained within Appendix C.2.

5.1 Back-End Implementation

The back-end was implemented first and quiver plots were used to illustrate the back-end while developing. A frame is represented in a quiver plot in figure 5.1 using the library Plotly. Quiver plots have been used as a direct method to visualise the velocity information as the arrowed glyphs are intuitive to visualise. Furthermore, this section uses biological terminology, taking advantage of pre-existing conceptions, to provide the reader with further understanding of the context. Furthermore some aerodynamic terminology is used and has been included in the Glossary (Appendix D) for reference.

5.1.1 Data Types

To bundle data and related functionality several classes were defined for the back-end. The classes define the following types which are enumerated below:

1. Node;
2. NodeCollection;
3. Grid;
4. Family;

5. Path;

6. Prediction.

Node

A node contains the information at a position within the frame. A row of data within a raw data frame equates to a single node, where all the variables become fields. Custom fields are also initialised at this stage, these include: `u_signed`, `v_signed`, `isVortex`, `Parent` and `children` - which is a `NodeCollection`. A node supports the basic functionality extensions: `addParent()`, `adopt()` and `key()`, which are repsonsible setting a nodes parent, adding other nodes as children and generating a key for accessing the node within a `NodeCollection` respectivly.

NodeCollection

Where a node represents a row of data with associated functions, a `NodeCollection` wraps a dictionary of nodes where the template is:

```
{tuple([node.x, node.y]) : node}
```

Code snippet 5.1: Accessing nodes within a `NodeCollection`

With the key being a tuple of a node's x and y values it allows for an O(1) efficiency when accessing a particular node. A *NodeCollection* supports basic functions such as adding and removing nodes, in addition to access and set functions.

Grid

Searching for neighbours in a `NodeCollection` is challenging due to uneven increments between x and y values, a `Grid` data structure combats these issues. A `Grid` implements `NodeCollection`s to create a specialised data structure, which contains auxiliary information as lists of unique x and y values that exist and their corresponding indices. The `Grid` data type doesn't change the physical structure of the collective but allows for queries. Nodes can be accessed by their indices and neighbouring nodes obtained, by incrementing or decrementing the relevant index, as shown in Algorithm 1.

Family

Vortices that are identified consist of a parent node and their children. This pairing of parent and children is encapsulated within a Family class. When initialised, the Family class calculates a new set of attributes which describes the cluster of related nodes. Further attributes such as `surface_area` are also calculated for tracking algorithms to identify `Path`s.

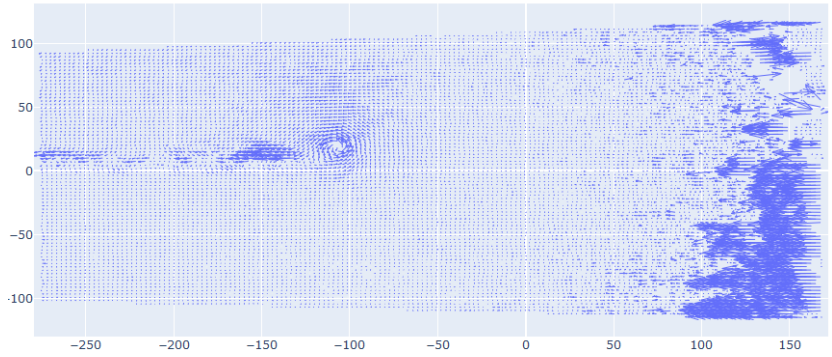


Figure 5.1: Frame 14: Outboard noise

Path

A Path contains the information required to track a vortex in time. It includes information such as start, end frame and confidence level and exposes an `addFeature` method. If a feature is added to the path the `prediction` for the next frame is updated, the prediction is stored as a specialised object.

Prediction

A Prediction object is used to keep track of the prediction for the features in the next frame (frame_{i+1}). When a feature is added to a path it calls the prediction's update method which calculates the difference between the two end vortices on the path and updates its arguments. If a vortex is within the tolerances in frame_{i+1} compared to the prediction arguments it can be added to the path.

5.1.2 Data Interface

A data interface fulfils requirement: **F-B1** which segregates the latter stages from the provided data format. If the format undergoes drastic changes, the amount of required updates to the code must be minimised. Due to this and the large number of frames, a data interface was created to handle the import of data to a commonly used format. Changes to the format should only affect the interface which will still output the required data for the front-end. This step also removes redundant nodes.

Definition 5.1.1. Redundant Node. A node that falls into one of two categories:

1. If a node's x, y, z coordinates and its velocity magnitude equals to 0;
2. The node's x coordinate is ≥ 100.

The first condition removes nodes that are in the raw initial file but have no associated information with it. The second point cleans up the output, due to

the *Outboard Freestream* air. These points are shown in figure 5.1, where there is a lack of data above the frame and freestream air. As this is a consistent element in all the frames and can be removed using this prior knowledge. The output from this step reduces the number of total nodes and produces a cleaner result, as shown in figure 5.2.

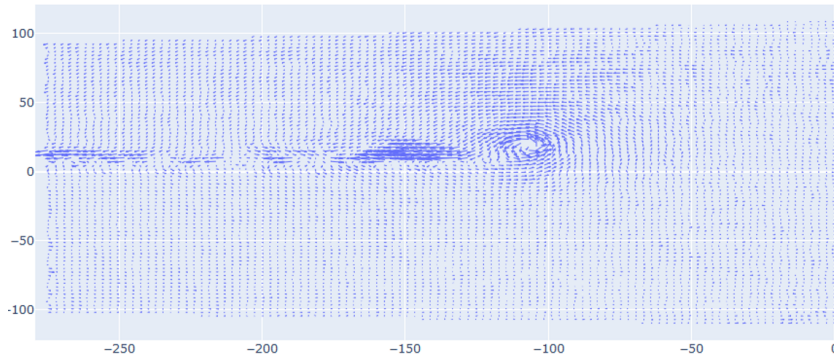


Figure 5.2: First Frame as a quiver

5.1.3 Vortex Identification and Extraction

For the reasons outlined in the requirements elicitation summary (section 3.1.4), feature-based methods have been chosen to identify unknown vortices within the data. Jiang et al and Holmen use similar two-step algorithms. In the first step, identification, Holmen uses a signum operation to find the smallest scale of rotation whereas Jiang et al check their *direction spanning* property before completing a topological clean-up.

The results from Holmén's proposed identification algorithm proposed were superimposed on to the quiver plot as points on a scatter graph in figure 5.3. In the figure, each coloured point represents a uniquely identified vortex core which can be grown in a secondary step. Algorithm 3 provides an overview of our implementation which contains four subfunctions. Before vortices are identified the data is labelled and converted to a `Grid` data structure. The *label_data* subfunction performs a sign function for all nodes' u and v components and stores the results in *u_signed* and *v_signed* respectively. Using the unique x, y values that are exposed by the `Grid` type, each node is iterated to check if they exist and if they can be classified as a vortex. Due to the shape of data, not every x, y combination is guaranteed to exist. If a node does not exist a placeholder is generated and added to the data, therefore filling empty positions with data. Each point is checked if it a vortex by performing the calculations as determined by Homén (Equations 5.1 and 5.2).

$$sign(v_{left}) + sign(v_{right}) + sign(u_{up}) + sign(u_{down}) = 0 \tag{5.1}$$

$$sign(v_{left}) + sign(u_{up})! = 0 \tag{5.2}$$

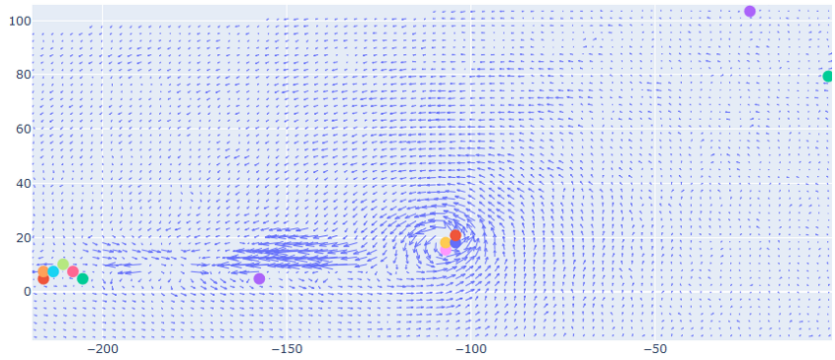


Figure 5.3: First frame as a quiver with vortex cores identified

Performing the second step of Homén's algorithm, the growth algorithm, grows the nodes iterating over the entire domain for a predetermined amount of times or until no new vortex cores are added. Given the size of a frame, a modification is proposed in an attempt to improve its efficiency.

Modified Growth Algorithm

By maintaining the positions of the vortex cores identified in a list from the first stage, the modified algorithm performs Breadth-First Search on each Parent node. Algorithm 4 outlines the modification in the form of three subfunctions. The first function calls the search function on each parent. The search function performs a breadth-first search, checking each parent's nearest neighbours, to see if new vortex positions can be labelled. A node is labelled as a vortex, per the original algorithm or if it is a pre-existing vortex from the identification stage. If a vortex is labelled, it is added to the frontier and as a child to the parent node. If the child has children then the parent node adopts those children. This mechanic is visualised as a family tree in figure 5.4, where an orange node represents the original parent node, a blue node is one with children and green is childless. As the maximum number of generations within the family will always be one, the adoption algorithm is trivial.

The result of Homén's identification algorithm with the modified growth algorithm is shown in figure 5.5, where the individual vortex cores have been grown to form families each representing a feature from the inital figure 5.3 to 5.5.

5.1.4 Tracking

The correspondence problem was investigated to be able to track them as they develop. As discussed in the Literature Review Chapter - in section 2.4, there

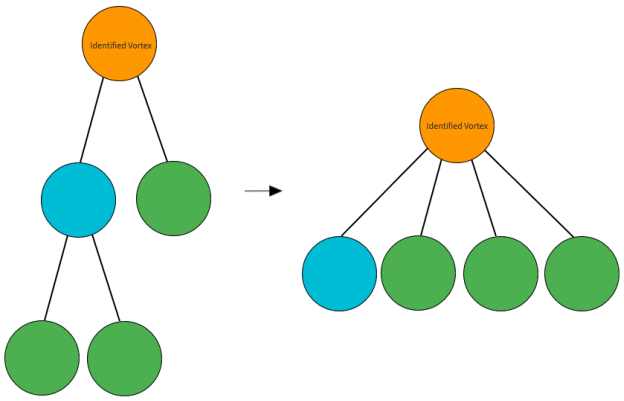


Figure 5.4: Family tree representation showing the adoption of a node with pre-existing children

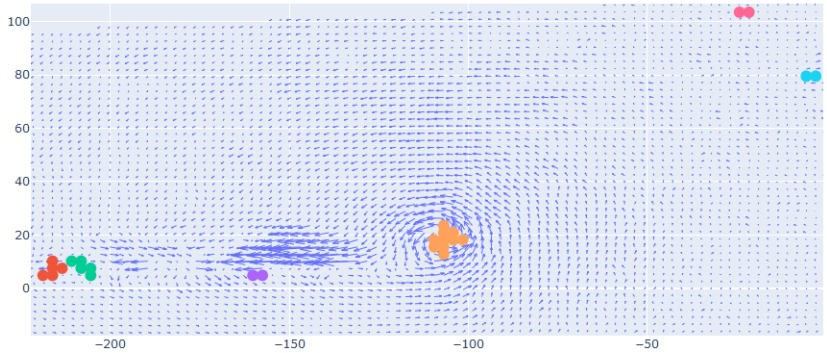


Figure 5.5: First frame post growth algorthim

are two methods for solving the correspondence problem, region and attribute. Reasons why the methods proposed by Silver and Wang and Samanetey et al were deemed unsuitable are in the discussion, section 6.2.4.

Reinders et al's [126] attribute correspondence method, was used to identify the paths of vortices took in the flow field. Their algorithm creates paths by using a prediction based on two features from frame$_i$ and frame$_{i+1}$ which attempts to match a corresponding feature in frame$_{i+2}$. Assuming a valid connection between the feature and candidate on the second frame, the correspondence is calculated. If the correspondence is greater than or equal to zero, a path object is initialised and recursived using depth-first search until the path ends or the last frame is reached. To create a prediction the consistent growth, speed and rotational speed are used in the form of correspondence functions. Reinders et al's correspondance functions are shown below and are the same as Samtaney's

(equations 2.12 and 2.13):

$$\frac{||V_p - Vc||}{max(V_p, V_c)} \leq T_{vol} \tag{5.3}$$

$$dist(p, c) \leq T_{pos} \tag{5.4}$$

5.1.5 Providing Context

As mentioned in the Design Chapter (Section 4.3.3), an automatic selection algorithm is required to avoid clutter. With the vortices extracted, their positions are an obvious attribute to orientate the positions of the key nodes. The algorithm 6 outlines the implementation, where it was decided to choose stochastic nodes based on the output from a multivariate normal distribution, from the NumPy library. If the size of the vortex is one, no additional arrows are added Otherwise, the distribution is centred on the vortex with a standard deviation and number of arrows chosen, dependant on its size. The positions are not guaranteed to equate to node positions so a post-processing step is completed to find the nearest available nodes from the selection algorithm. Rational and further detail behind these decisions are in section 6.2.6.

5.1.6 Back-End Summary

The back-end processes consist of several stages to allow for vortices to be extracted and tracked along with further information to aid user comprehension. The results of the processes are saved in the form of a dataframe, from the pandas library, which integrates with the chosen front-end library Plotly.

5.2 Front-End Implementation

The front end was designed to abstract complexities surrounding figure generation in importable modules. This module allows for complex figure generation algorithms to be accessed, including the Parallel Contextual animation.

The library Plotly offers a high-level API *express* for figure generation which returns figure objects with pre-populated data and layouts according to provided arguments. Similarly, Plotly incorporates a *figure factory* that wraps functions to create unique chart types. The APIs are provided as part of the library and are importable using the code snippet 5.2.

```
import plotly.express as px
import plotly.figure_factory as ff
```

Code snippet 5.2: Importing Plotly modules

5.2.1 Extended Figure Factory

The Parallel Contextual plot is made available in its own *figure factory* module fulfilling the same purpose and acts to extend the existing factory offered by Plotly. The extended figure factory exposes the three advanced plots proposed in the design section and supports animated traces on the same figure, which was inspired by an online plot by the user 'empet' [131], who demonstrates the functionality on stochastically generated line graphs. The code snippet 5.3 illustrates how to access the parallel contextual chart within the module.

```
import figure_factory as ext_ff
fig = ext_ff.parallel_contextual_annimation(df)
```

Code snippet 5.3: Importing custom modules

5.2.2 The Parallel Contextual Annimation

The charts that can be generated in the extended figure factory require the input to be in the form of a dataframe, provided by the pandas library and outputs a full figure like the original factory. The detail on the internals, outlined in Algorithm 9, are discussed in this subsection.

The animation accepts the output from the back-end which consists of three dataframes. The figure is initialised using the information for the first frame in the function 'create_initFrame' which creates the figure using the graphic object, *go*, module. The figure allows for layers to be added in the form of traces. The order of the traces are important, the traces are added to the figure in reverse order, with the information required for the later frames added first.

Creating the first frame

The scatter is a native plot supported by the Plotpy library more control over the markers are allowed. A marker is a point that represents a single node. To avoid confusion the markers for each scatter layer are different, this is illustrated along with other available attributes for the frame_{+1} in the following code snippet.

```
dict(
 name = 'Vortices +1',
 opacity = 0.8,
 marker = dict(
  size = 3,
  color = list(dataset_by_nextframe['Fam_idx'])
 ),
 hovertext = ['Vortex ID: ' + str(label[0]) +
 '<br>Size: ' + str(label[1]) +
 '<br>Magnitude: ' + str(round(label[4],2))
 ]
)
```

Code snippet 5.4: Scatter arguements

The attributes in the code snippet describe the output trace. The 'name' attribute for this layer is displayed to the user in the legend. Nodes from frame_{+1} are made smaller and more transparent compared to their counterparts on the first frame, where the opacity and size is implicitly one and five respectively. As the grouped nodes are part of a single vortex they are coloured by their family index, calculated in the back-end. Lastly, a string is generated using HTML syntax for when a user hovers over a node, the information displayed is the unique vortex id, the number of nodes it is made up of and the total magnitude.

As quivers are generated using Plotly's figure factory, which outputs complete figures they are generated and then stripped to expose their data to be added as a trace. If shown at this stage, the collection of traces in the figure would only resemble an interactive figure. The figure has to be extended with further frames within the figure's frame argument to be animated.

Creating the subsequent frames

To create an animation each frame's data is initialised and collated in the form of an array. In a similar manner to the first frame, the traces are added in reverse order, stored in an array for a 'Frame' object. The resulting array is set to the figure's frames attribute.

Creating the user interface

The user interface elements designed are created using dictionaries before being assigned to the layout of the figure. The animate button and corresponding timeline in the form of a slider have similar arguments which are shown in the following code snippet.

```
 dict(mode = 'immediate',
 frame = dict(duration=frame_duration, redraw=True),
 transition = dict(duration= 0)
)
```

Code snippet 5.5: User Interface Dictionary Arguments

The `frame_duration` argument is set at the beginning of the function but can be changed by the user, this allows to slow down the animation to be able to absorb the changes in each frame. The animation could benefit from using a transition between frames to smooth the changes and allow for paths to be tracked by the user. However, transitions are undeveloped and deemed unsuitable, the value is therefore set to zero to be synchronous.

Animation Summary

The animation can be generated using the extended figure factory. Furthermore, the plot supports optional user arguments such as the default axis limits, the physical size of the figure and frame duration. This allows users control over the end animation to focus on specific areas and to slow or speed up the total time

the animation plays for. The use of an animation relies on the user's ability to track elements, in an attempt to aid this unique colour mapping is applied depending on the vortex's unique identification number.

5.2.3 Front-End Summary

The result of the front-end is shown in figure 5.6, which also supports the decomposed methods proposed in the design section 4.3. This additional functionality allows for any of the three animations to be generated, even though a user could interact and remove components to achieve the same effect.

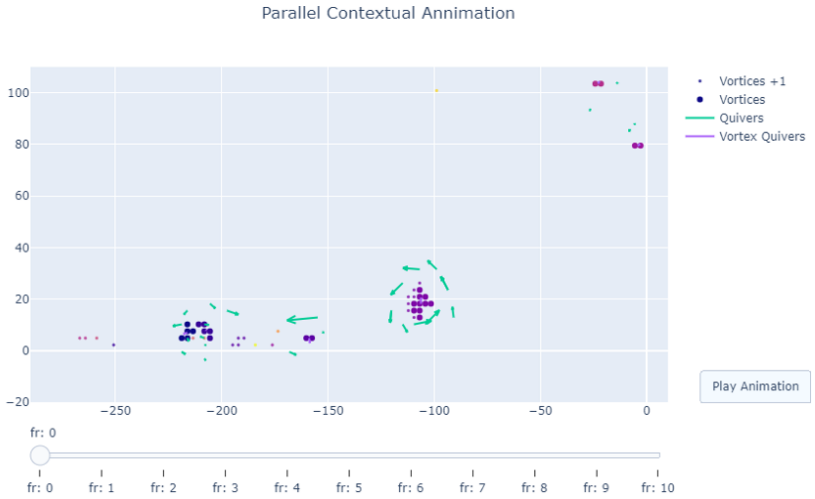


Figure 5.6: Result of the implemented Front-End

5.3 Chapter Summary

The chapter implements the two sections of the system to provide a web-based interface for users to interact with. The back-end uses Holmén's modified algorithm and tracks the features using the algorithm proposed by Reinders et al. To produce the parallel contextual animation, an auto-selection algorithm employed and layered using the library Plotly. The next chapter provides further detail behind the choices made during the implementation process.

Chapter 6

Discussion

This chapter discusses the decisions made during the design and implementation of the system. The chapter follows the structure of the Implementation Chapter, where it will discuss the technologies used before breaking down implementation decisions by back-end and front-end. In the back-end section, a post-processing step is presented for Holmén's original algorithm, alternate tracking algorithms are discussed and the methodology for identifying appropriate default values for both Reinders et al's and the automatic selection algorithm are explicated. Lastly, the front-end considers alternate methods for providing visual cues to encode data, including symbolic use scatter markers and transitions between frames.

6.1 Language and library Technology Choices

Recall that an unconventional technology stack was chosen to develop the front and back -end architectural components. Both ends have been implemented using Python 3.8 and to allow for visualisation a web-based environment Jupyter notebook is used. Python 3.8 is a high-level language that is easy to read which is it important is it allows the exposure of code fulfilling requirement **NF-F4**. Furthermore, The decision to use Python was motivated by previous experience and the extensive library support. The language R was also considered as it is orientated around statistical computing, graphics and can be deployed online - but with limited prior experience and knowledge in the environment it was not used.

6.1.1 Back-end

The back-end uses several auxiliary libraries such as NumPy and Pandas. NumPy is a library that adds support for multi-dimensional arrays in addition to auxiliary functions, which are used thought the processes for the back-end. To pass data to the front-end a Pandas dataframe is used. Pandas is a flexible data

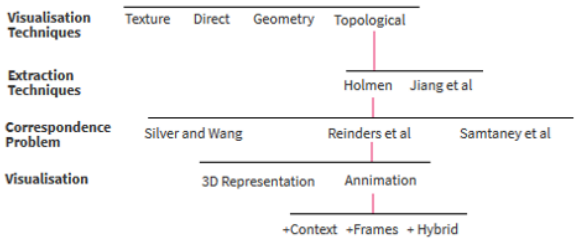


Figure 6.1: Investatory work completed

manipulation library and supports a data structure called `DataFrame`, which is a two-dimensional tabular data format and supports queries for the front end to use. The library is supported by charting libraries HoloViews and Plotly.

6.1.2 Front-end

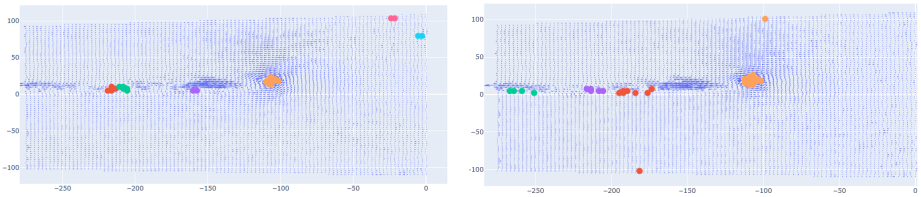
Originally considered was Matplotlib [128], initially released in 2003 it provides a MATLAB-like interface but is offers poor support for web and interactive graphs. As web-based interaction is a high priority requirement an alternative Bokeh was examined. Bokeh [132] provides data interactivity for web browsers and can host interfaces via a Bokeh server which can be used in tandem with HoloViews. HoloViews aims to minimise code by wrapping the back-end plotting libraries: Matplotlib, Bokeh and Plotly. These libraries to provide which allows them to plot using HTML5 canvas and WebGL. HoloViews with a Bokeh back-end was used to provide the initial plots when developing but was found to be too restricting. Plotly, written in JavaScript, offers a fully interactive graphing capabilities with interfaces for Python, JavaScript and R.

6.2 Back-end

This section breaks down some of the avenues explored in addition to the decisions made to extract the vortices and prepare the information to be plot. Figure 6.1 shows some of the work completed, where the path in red was the final implementation that is discussed in Chapter 5. This section revisits Holmén's Growth algorithm - to propose a post-processing step, and the Correspondence Problem - to provide a rationale for the algorithm chosen.

6.2.1 Extending Holmén's Growth Algorithm

It was observed that Holmén's Growth Algorithm could grow nodes as separate features that could be classified as a single feature due to their proximity - a post-process classification extension is therefore proposed. Analysing scatter



(a) Frame 001 (b) Frame 002

Figure 6.2: Frame snapshots taken after performing classification

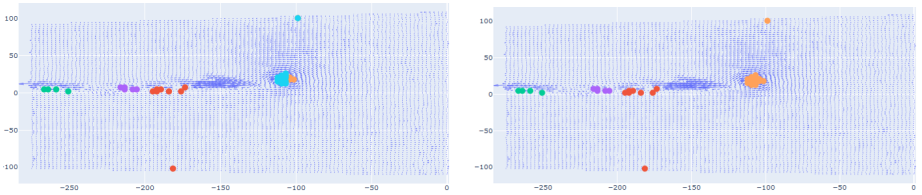
plots of the identified vortex cores, a post-processing step was hypothesised in an attempt to resolve an identified problem. Holmén's algorithm grows vortex cores in parallel by iterating over the entire domain. Therefore if two neighbouring parent nodes grow they would be classified as two separate families, although it is more likely that they belong to the same family. To resolve this problem of classifying points, post-growth stage to families, a K-Nearest Neighbour algorithm (Algorithm 10), was investigated. A K-Nearest Neighbour algorithm was imported from the library sklearn.neighbors and classifies nodes based on the closest neighbour. Two issues were encountered when implementing the classification algorithm using KNeighborsClassifier from the library sklearn.neighbors;

1. Points would be classified as themselves;
2. Testing on multiple samples failed.

The first fault was due to the classifier weighting points by distance and nodes were self-classifying, therefore the weighting was removed and the classifier was used to return the nearest neighbours and their distances and manual examination was used to classify the point. The change was successful and allowed for the first frame to be classified, however, failed when tested on the second frame. The two frames can be compared in figure 6.13 where in subfigure b, two points (-180, -100) and (-100, 100) have been classified incorrectly as part as the nearest family - resulting in the need for a distance threshold, d. To determine the optimal distance threshold and nearest neighbour (k) values, frames were reclassified for different values. The optimal values found were $k = 13$ and $d = 10$.

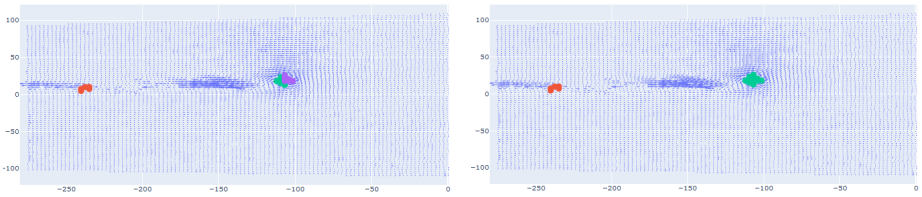
Testing the possible extension to Holmén's Extraction algorithm

The values $k = 13$ and $d = 10$ were obtained by running a series of tests with further plots are in Appendix E.1. Initially the value for k was incremented from 5 until 8, where the output seemed reasonable. Defining a reasonable output implies a subjective categorisation, however, when applying the algorithm to the second frame it was obvious when the output improved. The boundary is shown in figure 6.3, made clear by the successful classification of the large vortex at (-100, 20). However, the two features at (-25, 80) and (-25, 100) respectively



(a) $k = 7$ (b) $k = 8$

Figure 6.3: Testing k values



(a) $k = 12$, $d = 10$ (b) $k = 13$, $d = 10$

Figure 6.4: Frame 18: Testing k and d values

have been classified incorrectly as part as the nearest cluster - resulting in the need for a distance threshold.

The values of k and d were tested, looking for when the features at (-100, 20) , (-25, 80) and (-25, 100) are classified correctly, where it was found that for $k > 8$ and $d \geq 10$ would classify correctly. These values were then tested on other frames including frames 6 and 18 (figure 6.4) where the values failed and were therefore modified to $k = 13$ and $d = 10$.

6.2.2 Classification summary

The process to achieve the values highlights the weakness behind the classification method. Although several frames were used to refine that values those values did not apply to all the frames. Frame 275 is shown in figure 6.5 for which the values $k = 13$ and $d = 10$ did not produce a satisfactory results. In the particular case of frame 275, the green coloured 'feature' on the Inboard side is too large and distributed.

6.2.3 Serialising Holmén's Growth Algorithm

The proposed extension attempts to label close features but is deemed to be to parameter dependant. This motivated the proposed growth modification algorithm which uses a serial approach, where previously identified nodes can be adopted, allowing for features to be merged without reclassifying based on a user-determined value. Serialising with an adoption strategy produced consistent results compared to the classification extension. This form of the algorithm

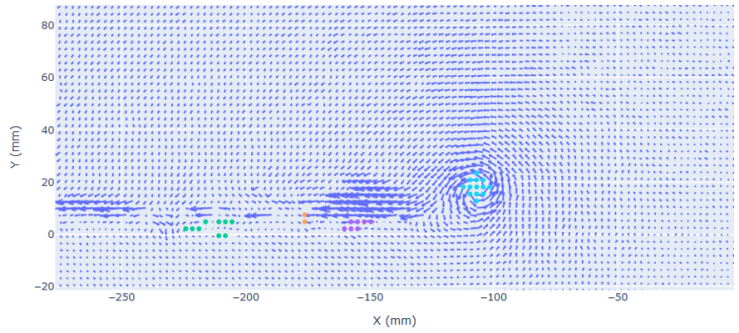


Figure 6.5: Frame 275: Shows the outstretched *inboard* feature.

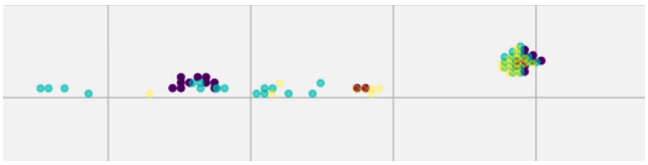


Figure 6.6: The first three frames overlayed.

is less invasive and allows for trust in the data to persist.

6.2.4 Solving the Correspondence Problem

To solve the correspondence problem, methods proposed by Silver and Wang, Samanety et al and Reinders et al were explored. It became evident that their tracking algorithms were weakened by the data, where both Silver and Wang and Samanety et al use spatial overlap as part of their algorithms. Figure 6.6, uses colour to encode frame information, illustrates that the amount of overlap is limited. The sampling rate is unknown but it can be assumed that an increased sampling rate would allow for the small fast-moving features to be tracked.

Choosing the tracking algorithm

Silver and Wang used volume tracking and perform a matching test for three-dimensional datasets. Correspondence is indicated by spatial overlap under the assumption that sufficient sampling is performed. Although a region-based method, they prevent false positives using a volume calculation, which is similar to Samtanty et al - who use overlap as an attribute, inheriting the same assumption of an adequate sampling rate.

Samtanety and Reinders et al [126] [127] separately use attribute correspondence methods, where both methods use user-defined tolerances to identify correspondence. Reinders algorithm was chosen because they claim that their algorithm allows the tracking of fast-moving small objects that do not overlap.

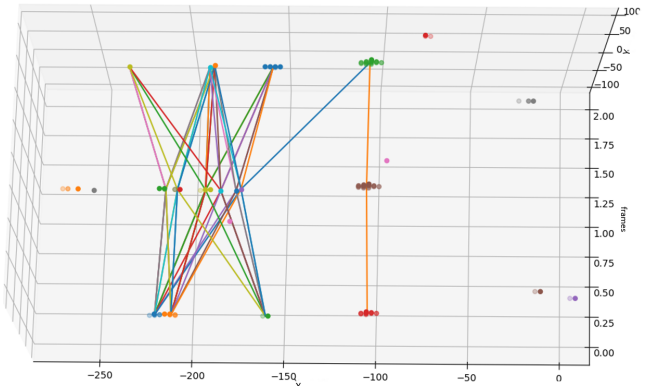


Figure 6.7: All paths visualised in three dimensional space for frames 1 to 3

Due to the limitations of the data, rotational speed was unable as an attribute and the function for consistent growth was modified taking inspiration from Samanety et al. Samanety et al [124] state that the number of nodes is an adequate approximation for volume, therefore, in this two-dimensional domain the number of nodes, is used for surface area.

Reinders et al state that the number of possible correspondence functions depends on the attributes available for a feature. Therefore additional correspondence functions were added, in an attempt to solidify confidence in the paths formed in Algorithm 5. The proposed correspondence functions use the signed velocity components and velocity magnitude, where they return a higher correspondence if the candidate feature has an expected position and constant velocity magnitude respectively. The two additional functions differ where the first compares the candidate's position to the prediction's u and v components and the later compares the velocity magnitudes. Applying the maximum confidence criterion in conjunction with a minimum path of three, as prescribed by Reinders et al, greatly reduces the number of paths. This reduction could be attributed to a poor sample rate, derived from the data capture method PIV.

All paths identified for frames one to three are shown in figure 6.7. The identified paths are dependent on the correspondence values which are, in turn, calculated using specific threshold and weight values.

Correspondence Functions and their Thresholds and Weights

Reinders et al's tracking algorithm was implemented using four correspondence functions. The positioning correspondence function returns a match if the position for candidate feature, relative to the original feature is correct. Therefore adding more bias to candidates whose position is to the right side of paths moving from right to left - assuming a constant speed, and vice versa. Constant speed and not acceleration is considered as the prediction calculation uses the difference between the last two previous features added to the path and assumes

small time difference between them. The velocity magnitude is the weighted sum of the nodes in the feature which is assumed to be constant between features in different time frames.

By default the functions hold the following thresholds of 10, 2, 2, 5 and weights of 1, 1, 0.4, 0.3 respectively. The functions are used to identify whether candidates deviate from the prediction where $0.0 \leq C \leq 1.0$ if the candidate feature is within threshold. The values were developed without the minimum path criteria to observe the produced paths. Due to uncertainty both additional functions have a lower weighting where the constant velocity magnitude function weight is lower due to the weighted calculation performed for each feature. Because these values are volatile, they have been exposed to the user with default values, which were resolved during the testing phase in section 7.1.2.

6.2.5 Three dimensional representation

Access to paths and edge coordinates in each frame gives the possibility of generating a volume using a marching cube algorithm to produce an object who's depth represents time. Edge coordinates have been obtained by using a convex hull algorithm on features. However, by using the depth to represent the time the occlusion problem is made more important, this is increased by the number of frames that may be rendered and would require clipping to visualise a section of frames at a time. By introducing an element of interaction, this problem may be reduced. The basic implementation in figure 6.8 can be rotated to view other angles, however effectively illustrates that the combination of Scatter plots and path objects are unintuitive without making points more inconspicuous.

6.2.6 Providing Context

The Implementation Chapter claims that stochastic nodes are chosen from a multivariate normal distribution with a post-processing step. The multivariate normal distribution was chosen as it can be used to output values with a higher weighting around points closer to the vortex core. During development, the algorithm was updated to be in line with algorithm 6. Whilst the philosophy remains the same, lines 7 and 9 were updated and did not call the gift wrapping algorithm (7) on line 8. Initially the algorithm chose a subsample of points produced by the multivariate normal distribution and converted the points to nodes within the data. The algorithm was improved to produce a sample of *boundary* points, using a convex hull algorithm, from the multivariate normal distribution instead. Variables to determine the covariance and number of arrows used to represent the movement around a vortex were put under scrutiny.

Calculating the bounding surface

A convex hull algorithm can be accessed using Scipy which outputs a hull object which contains attributes such as area and volume. However, it was decided

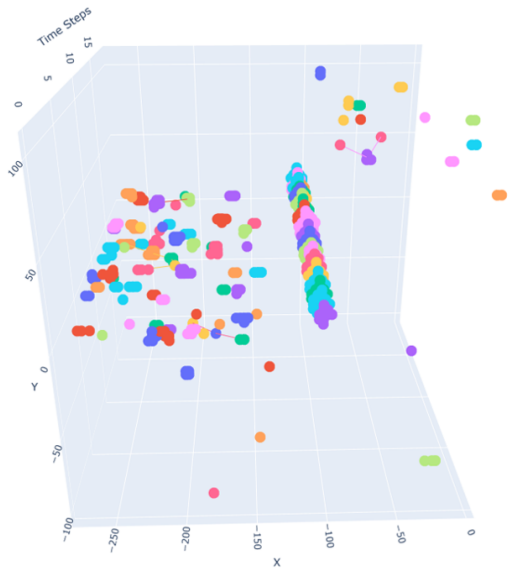


Figure 6.8: Three dimensional mock up

to not use their implementation due to efficiency which can be potentially attributed to the extra, unrequired attributes. Therefore a convex hull module was created to compare different algorithms. The Gift wrapping algorithm is a brute force method that takes $O(nh)$ time as it is output dependant. It can be considered to be faster than the Monotone chain (Andrew's) algorithm (8) when $h < logn$. This alternative algorithm is a variant of Graham scan, which also takes $O(nlogn)$ time. The points are instead sorted, by the x coordinate instead of angle. The hull is computed in two steps, upper and lower and then combined. The Gift wrapping algorithm was chosen as the efficiency can be compared to the monotone chain algorithm by comparing the time taken to compute the hull for a series of points where, on average, where it was ten times faster.

Calculating the width of the distribution

The equation to determine the possible width of the distribution was in the form $y = mx + c$. To calculate the width of the distribution equation 6.2 was used, where the width was made dependant on the vortex size so that arrows for larger vortices would be further away. The minimum distances desired was ten millimetres, for which a circle with that radius takes up 0.5% of the plot. Equations 6.1, 6.2 and 6.2 were considered, with the result shown in figure 6.9.

$$width = 1(vortex.size) + 8 \tag{6.1}$$

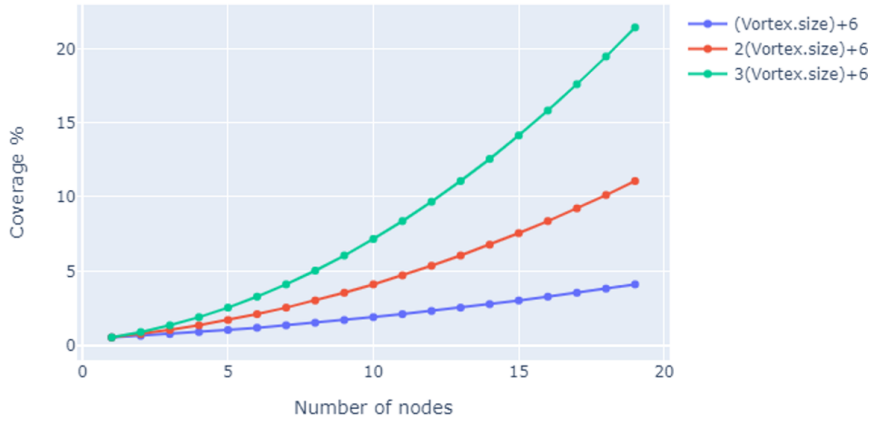


Figure 6.9: Size of a vortex and the maximum potential coverage (%) over the plot.

$$width = 2(vortex.size) + 6 \tag{6.2}$$

$$width = 3(vortex.size) + 4 \tag{6.3}$$

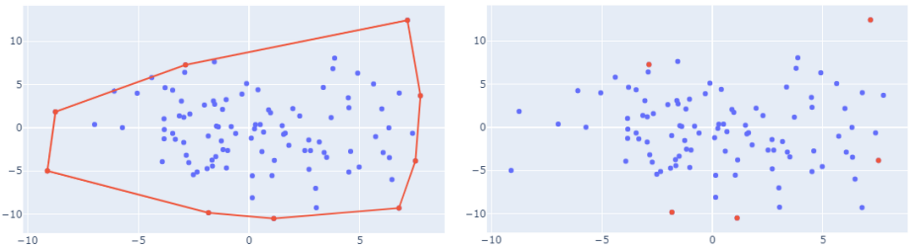
Although the equations (6.1, 6.2 and 6.2) are linear they cause surface area to grow exponentially. Examining the first ten frames there is a constant, large, vortex at (100, 20) with an average size of fifteen which equates to an area worth 3%, 7% and 14% using the three equations respectively. As these arrows are to provide support and to avoid clutter it was decided to limit their coverage and use equation 6.2 to be the default. As the distribution of arrows can be subjective they are exposed to the user to alter if they desire.

Calculating the amount of arrows associated with a vortex

The size of the vortex is considered to determine the number of arrows to draw. Considering the minimum case, it was decided that a vortex of two nodes should ideally have three surrounding arrows, excluding its own. Jiang et al base their topology algorithm on the duality between equally spaced direction ranges and the direction labelling of a triangular cell. Therefore, three as a minimum number was deemed adequate and the following equation was used.

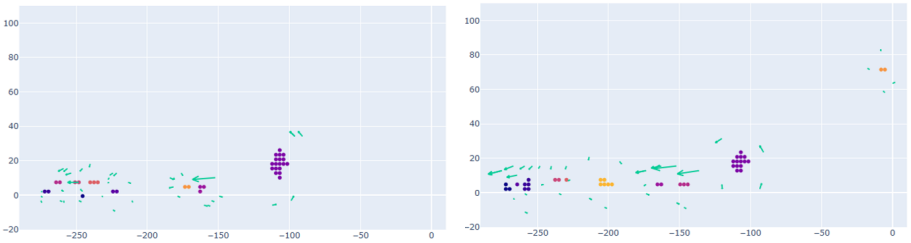
$$nArrows = vortex.size + 1 \tag{6.4}$$

Figure 6.10 shows how the algorithm chooses the contextual arrows for a vortex of size four at (0,0), where blue nodes represent the nodes chosen from the multivariate normal distribution. In figure 6.10a the convex hull algorithm has identified the hull, shown in red, the hull is then subsampled to the following



(a) Convex hull from the distribution (b) Subsample of points on the hull

Figure 6.10: Auto selection algorithm - Broken down



(a) Frame 005 (b) Frame 007

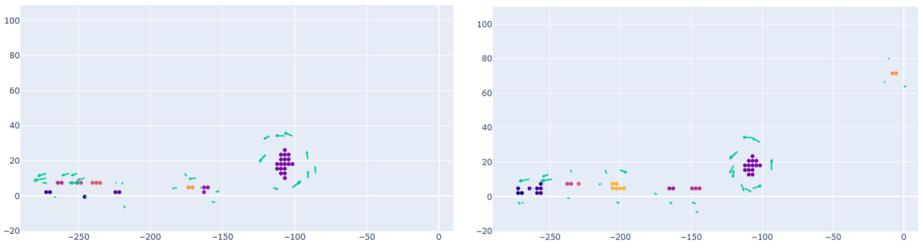
Figure 6.11: Frame snapshots taken with a five arrows per vortex

red marked nodes in figure 6.10b. The identified nodes are then plot as arrows, using their associated velocity components u and v.

The need for a dynamic number was reinforced by testing with a static number, five. Figure 6.11 uses frames five and seven to show that the visibility is degraded, other information from other frames is hidden. These frames can be compared to 6.12 where equation is used. Analysing the images it is clear that there is a reduction in complexity *Inboard* and a clear picture of movement for the largest feature. Adding contextual arrows in the design section were placed in surrounding regions.

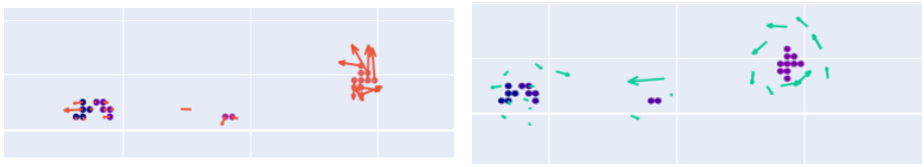
6.3 Front-end

The topic of visualisation can span human-computer interaction and psychology in regards to providing the best platform for comprehensibility of spatio-temporal data. This will remain an area of future work for this dissertation. Instead, the focus was on the visualisation of the data to satisfy the requirements established. However, the Gestalt laws - in particular similarity, were considered in an attempt to separate the features with their counterparts. The use of transitions and three-dimensional objects are also discussed.



(a) Frame 005 (b) Frame 007

Figure 6.12: Frame snapshots taken using a dynamic number of arrows



(a) Frame 001 (b) Frame 001

Figure 6.13: Arrows on nodes and displaced

6.3.1 Identifying Features

Gestalt psychology proposes laws to explain why humans group certain objects together, such as Proximity and Similarity - where properties such as shape, colour, shading and size are used. Ambiguity arises when nodes of two different features are in close proximity. Consequently, colour is used to describe vortices, originally assigned depending on their position within the frame. This lead to the incorrect implication of connectivity between certain features. Initially, the system was reworked where the colourmap was constant for the entire animation, resolving the problem of the matching colour. However, if a large number of vortices are identified, it can become hard to identify distinct colours on the continuous colour map. Therefore, the final version uses a static colour map that is initialised using the maximum number of features that appears within a frame in the animation. The use of colour to imply connectivity was used to enforce vortices where features belonging to the same vortex are the same colour.

6.3.2 Segregating Frames

With colour used to identify features, the challenge of grouping features per frame was emphasised. It was decided to change the size and opacity for features in the next frame. If the features were treated as a whole, changing their size could impact the perceived accuracy when comparing features across frames. By reducing the size of the nodes the feature can still be envisioned. By reducing

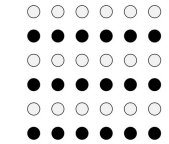


Figure 6.14: The Gestalt law of Similarity [7]

opacity, the colour can still be relied on to inform connectivity of vortices but the strength is reduced. Ideally, the use of colour could be combined with a secondary characteristic to illustrate a connection between vortices.

6.3.3 Transitions

Ideally, the connectivity between vortices should be stronger than a colour relationship. This has the potential to be explored with the use of transitions which would interpolate between the discrete frames. Plotly supports transitions for figures, exposing core attributes for 'standard' scatter plots. The term 'standard' is used to describe scatter plots in which the number of nodes in each frame are consistent. This is important as in our implementation, the number of points in each frame differs. Therefore the transition connects unconnected nodes animates the arrows which is unrequired. The use of transitions would be beneficial but is unable to be implemented due to functionality supported by the library - if it were to implemented it would be an optional attribute as the transition could falsely allude to incorrect movement of vortices between frames.

6.3.4 User Interface

Plotly allows for chart interaction, charts produced have the following function buttons: Zoom, Pan, Select, Auto Scale; in addition to tooltips: show nearest data and compare data on hover. Charts can show a legend when can be interacted with by clicking, where single clicking toggles the trace and double-clicking only shows that trace. These controls that pre-exist allow for data interrogation with ease.

6.4 Chapter Summary

This chapter reviews the rationale behind the language and associated library choices. Revealing the surrounding work completed, the alternative back-end processes are examined, including a classification step, a region tracking algorithm and use of three dimensions to represent two-dimensional temporal data. For the processes that form the back-end, further detail and rationale is provided, before reviewing the front-end.

Chapter 7

Testing and Evaluation

This chapter discusses the implementation from the perspective of testing and evaluation of the key features. It breaks down the testing strategy used, the plan and problems faced, before evaluating the system.

7.1 Testing

The resulting implementation pipeline favours a simple testing strategy. The linearity of the implementation process allows for unit tests to be created to target specific areas, which were chosen based on their impact on the remaining pipeline. These areas were identified to be the output from the data interface, identification and modified growth and tracking algorithms. In addition to targeting key areas, standard unit tests were added to ensure auxiliary algorithms were working as expected.

7.1.1 Test plan

To test the functionality unit tests were derived. The initial stages of development, orientated around the data interface and extraction, were driven by the tests written beforehand. As the format of the output data is known beforehand, tests could be written to ensure the interface and associated defensive programming techniques were working. This was deemed an important area due to the exposure and limited control over the input to the process. In later stages test-driven development was not followed, this was because the output of the processes were more complex to anticipate, where the final visualisation stage used a user study to analyse the final result. Furthermore, to be able to do derive tests for the given data, synthetic data was created. The use of synthetic data allowed to mock up base cases with expected results. The base cases were designed to test the identification and growth algorithm was working as expected in varying scenarios such as missing data and for edge cases.

Problems

The use of synthetic data allowed specific scenarios to be tested. The creation of the frames was a manual process where each node was placed with the algorithm in mind, however, this allows for biases to be ingrained. A flow generation algorithm would remove these biases from the data, producing randomised flow fields for the identification algorithm. Recall that Holmén's identification algorithm selects the smallest scale of rotation in a cell. Therefore a randomised flow field can be abstracted down to rotations around cells, or specifically, the generation of randomised base cases which are already used.

A randomised flow field is more applicable to the growth stage after the vortex cores have been identified, but the results are hard to verify. Ideally, the flow field is not randomised but produced by CFD which simulates flow fields. CFD is performed by calculations and cannot be relied on as a perfect source. Therefore data produced would have to be manually compared to determine accuracy requiring a visualisation method and subjective interpretation. To compare data a quiver plot produced by the non-invasive technique PIV in simulated conditions in a wind tunnel can be used. This would allow the grown vortices to approximated and interpreted, where the concept of manual interrogation and comparison withstands.

7.1.2 Back-End

Testing the Data Interface

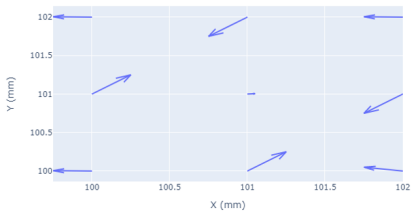
The format of the ingested data has been assumed to remain constant, the data interface has been designed with requirement **NF-B1** in mind where the code provides a flexible platform if changes are made to the format. The interface employs defensive programming to ensure that the rows of data are converted to nodes and saved as a `NodeCollection`. The use of synthetic data combined with a front-end that supported tooltips allowed to manual interrogation to take place.

Testing Holmén's identification algorithm

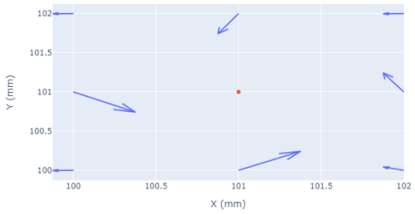
The use of synthetic data allowed specific cases to be tested before applying the algorithm. Figure 7.1 shows four of these cases which were devised to test particular behaviour. From analysing the data the algorithm was made to work with missing nodes, the results for the tests are shown in figures 7.1b, 7.1c and 7.1d. In figure 7.1d an extra node was added compared to figure 7.1c. This tests that the algorithm responsible for obtaining the nearest, non-vortex nodes works as intended.

Testing Holmén's Extraction algorithm

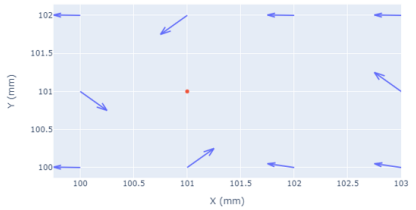
As part of Homén's algorithm, the domain is iterated though until a threshold or no new vortices are classified. Assuming the latter is used, their algorithm is



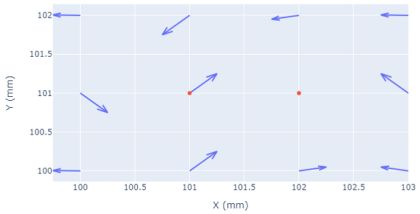
(a) Testing with incorrect data



(b) Testing with a missing center node



(c) Testing with a missing neighbour node



(d) Testing the growth algorithm

Figure 7.1: Testing Base Cases

checking:

$$x \times y \times s \times nframes \tag{7.1}$$

Where x, y are the number of horizontal and vertical nodes and s is the maximum size per frame. The value of s can be approximated by taking the average vortex size over ten frames which is 15.6. The equation can therefore be calculated to equate to $104 \times 84 \times 15.6 \times nframes$.

The modified algorithm is independant of amount of nodes in the data. The number of times the Breadth-first search is performed is dependent on equation 7.2, which can be approximiated by equation 7.3.

$$nodes_{perframe} \times nframes \tag{7.2}$$

$$avg\_nodes_{perframe} \times nframes \tag{7.3}$$

The average number of nodes identified before the identification stage per frame was 17. However, the number of features after the modified growth algorithm is 6.5, with an average size of 4. Therefore, equation 7.3 can be approximated as $6.5 \times nframes$. This can be extended with the knowledge that the average feature size is 4, which can be observed in figure 7.2, and that each feature has at most 4 adjacent edges. Breadth-first search can, therefore, be represented by its complexity which equals the number of edges, $(V-1) \times E = 3 \times 4$, where the actual average number of edges per node would be closer to three. The final approximation for the number of nodes checked is $6.5 \times 12 \times nframes$ which is considerably lower compared to the estimate for the original algorithm.

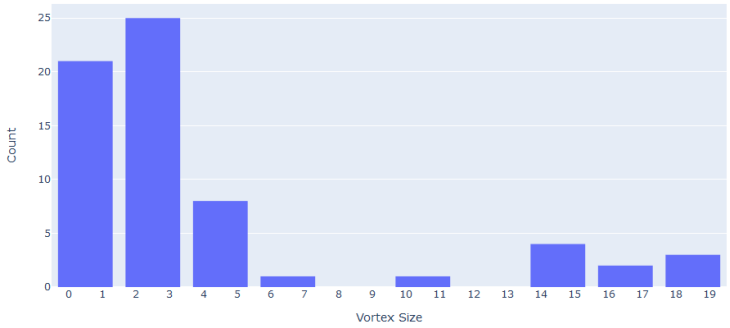


Figure 7.2: Distribution of feature sizes post modified growth stage

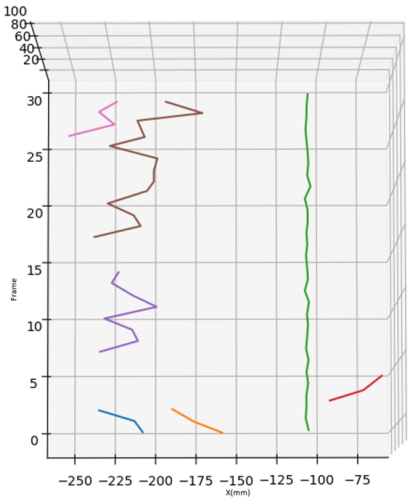


Figure 7.3: Paths identified from a 'top-down' angle

Testing the Tracking algorithm

The verification of Reinders et al's tracking algorithm was challenging. Tests were made to ensure functionality, however, it was hard to subjectively define if the tracking algorithm was producing the correct paths. The change between frames is quite large and makes it hard to justify some of the paths produced by the algorithm. Figure 7.3 shows the output paths for frames 1 to 30 in two dimensions where the dimensionality has been reduced to remove the y component and the minimum path length of 3 has been revoked. Seven paths have been identified, moving upwards and are identified by their colour. Of these paths the green path seems to be correct with minimal horizontal noise between frames, this can be directly contrasted against the brown and purple paths. In a second phase of development, the tracking was revisited. Using the contextual

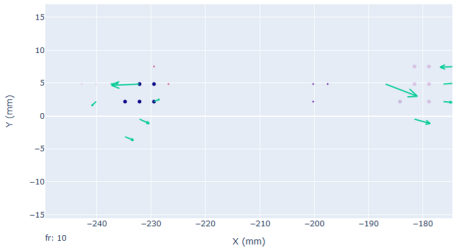


Figure 7.4: Figure 10 visualised using the contextual plot method

animation, frame ten was further examined. In figure 7.4 the relevant features have been selected allowing it to be compared against the purple path in figure 7.3. In figure 7.4 it can be observed that they are moving apart and therefore unlikely to be part of the same path. This motivated further correspondence functions to limit the resulting paths.

Determining the default values was a challenging task, where different configurations would produce different results. As there was no method of deterministically declaring a correct result the paths had to be interpreted to determine their correctness. The values were chosen so that only the green path from figure 7.3 remained. Due to the low confidence in the results produced, tracking of features is left for the user to interpret - however features that share a path share the same unique identification number that is accessible via a tooltip.

7.1.3 Front-End

Testing the front-end was a manual process, due to the use of the Plotly library the testing was done on interactive elements and available supported interaction elements to ensure they worked.

Testing the UI

The user interface is provided by Plotly, whose Python interface is on GitHub. No issues regarding the user interface were reported when it was checked, however, visual testing on the functionality provided was carried out for an initialised plot.

7.2 Evaluation

To evaluate the system a user study was carried out, in an attempt to identify areas of weakness and reinforce claims made within the dissertation.Furthermore, this section evaluates the system designed as a whole, in regard to the conclusions drawn from testing.

7.2.1 User Study

A user study was performed on students at the University of Bath, to evaluate the work completed. Ideally, this would have been with aerodynamicists from Toro Rosso whose existing expertise in the area would aid their comprehension enabling valuable feedback. A summary of the results obtained is in Appendix F.

Aim and Structure of the study

The study had three goals. The study attempts at evaluating the ability to track like objects within an animation, compare a Quiver plot to a parallel orientated animation and enforce our belief that interactive elements help the comprehension of static charts. The study was split into sections that allowed the goals to be realised.

After obtaining consent to partake in the study and a briefing, where they were informed about Holmén's smallest scale of rotation criterion, the participants were presented with an online questionnaire.

They were then presented with a quiver representation of a frame as an image and asked if they could identify vortices and count them, before being asked about their confidence in their answer. This allowed further sets of questions, using an interactive quiver plot and the parallel contextual animation, to be tested. This was followed by a series of open-ended questions which allowed participants to give feedback and offer suggestions on the different visualisation methods presented.

Summary of the results

Participants were able to identify at least one vortex stating high confidence for their answer, their confidence increased when attempting the same task on the interactive plot. Participants were then asked to describe movement within the frame and were able to realise that a vortex was spinning, in addition to inboard movement where 60% of the participants stated that the introduction of contextual arrows helped understand the plot.

When describing the static Quiver plot, synonyms for confusion and difficulty were mentioned five times - with it described as ambiguous and complex. The interactive Quiver plot's zoom functionality was described as useful but participants made comments on the comprehension of information. Finally, participants were presented with the animated contextual plot which where information was described as intuitive and easier to work with.

Evaluation of the study

The study that was held was limited due to the participants that were able to complete the study. Ideally, further user studies and questionnaires should have been completed with aerodynamicists from Toro Rosso, however, due to COVID-19 they never occurred. However it was able to reaffirm some of the

design descions made and gave insight to the performance when used by users with limited knowledge. The study can only 'reaffirm' decisions due to its small scale and inadequte participant base, to be be considered viable the study would have to be completed on a larger, more experienced group of participants.

7.2.2 Evaluation of the techniques employed

Important components in the visualisation pipeline have been tested, therefore the components such as the positions of vortex cores and contextual arrows can be relied on. The algorithm that is unreliable due to its subjective nature is Reinder's tracking algorithm. As alluded to in section 7.1.1 the use of synthetic data is limited to the creation of predetermined scenarios. This creates a vulnerability for all processes that require subjective input to determine their 'correctness'.

To solidify confidence in the results, multisource analysis would be required. This can be achieved with CFD simulations which could allow for the verification of tracking algorithms and refinement of parameters. This would require knowledge about the wing geometry that was present in the original data.

7.3 Chapter Summary

This chapter explains the tests completed to ensure that the system works as expected where, testing driven development was originally used, followed by visual testing. Problems with testing with synthetic data is discussed and a priliminary user study is performed. The study gains insight into the possible advantages of using an interactive, environment that is supported with extra contextual information.

Chapter 8

Conclusion

8.1 Introduction

This chapter examines the dissertation as a whole and concludes the work. The project investigated the state of the art in flow visualisation, where existing work is often completed on CFD data, not from PIV measurements. Using these measurements the number of available techniques were limited which resulted in the formulation of the Parallel contextual animation.

8.2 Satisfying aims and objectives

Recall the aims and objectives that were stated in the Introduction chapter. This section assesses how much they have been satisfied by the work completed, the aims and objectives are listed below.

1. To research existing algorithms in two and three dimensions within the flow visualisation field;
2. To extend and apply existing work to be able to identify a vortices from the data;
3. To investigate the time-dependent vector data produced by PIV;
4. To produce a system capable of extracting and visualising vortices from the data; and to provide:
5. A summary of research into flow visualisation;
6. A system that makes large, complex, time-dependent data comprehensible;
7. An assessment of the system.

The literature review outlines the state of the art and draws conclusions that are applied to two-dimensional input data. Useful algorithms, identified from this analysis, have been implemented and extended to be applied on the vector data - producing system capable of extracting and visualising vortices. The system is then examined in the Testing and Evaluation chapter, section 7.2.

8.3 Summary of Contributions

This dissertation proposes an improvement to Holmén's vortex extraction algorithm which can be used to visualise temporal data by using a parallelised animation. This can be extended by selective visualisation using our automatic selection algorithm to show quivers in selected areas that provide contextual information. Furthermore, the developed system uses PIV data as opposed to the traditional method of using simulations. PIV is not as common as CFD but allows for the flow measurements obtained to be visualised.

8.4 Limitations of the Project

Despite the contributions outlined in the section beforehand, the system is limited by the type of data it can ingest and lack of confidence in the tracking algorithm employed. Furthermore, as previously mentioned, the testing and evaluation techniques predominantly rely on subjective intuition and justification. This makes it hard to definitely declare the techniques as correct, even though they are producing visualisations.

8.5 Future work and Research

This section analyses the limitations previously identified to present possible areas to extend the work completed.

8.5.1 Future work

The data interface can be extended to allow for other input formats to work within the system such as CFD. This would allow for the comparative visualisation to provide comparative data analysis.

The system effectively prototypes its abilities and can be improved by reducing dependencies on external software, such as Jupyter notebook and implementing a front-end using a typical web stack. Furthermore, a low-level API can be implemented, allowing for the animation to be created and wrapped for other languages. This makes the animation more accessible and open to use in multi-chart dashboards.

The system only handles time-dependent two-dimensional data and can be improved by updating algorithms used to their three-dimensional counterparts.

The automatic selection algorithm would be required to be smarter in an effort to reduce occlusion in higher dimensions.

8.5.2 Further Research

The system only presents and analyses one possible combination of a feature and extraction method that was intuitive to explore. There are many other direct and geometric methods that could integrate better and provide more intuitive results.

Furthermore, only positive matches are shown and there could be benefits included in representing uncertainty. For example in subjective algorithms, such as Reinders et al's tracking algorithm, multiple paths could be visualised that correspond to different input values. The flow visualisation field is diverse and covers a large number of specialist techniques, future research in this area will lean towards the efficient analysis of large, time-dependent three-dimensional domains - putting emphasis on selection and simplification techniques.

Lastly, a user study in essence of the one undertaken should be performed by participants with a suitable background to confirm the validity of the work.

8.6 Closing Remarks

The dissertation presents a technique that allows for the visualisation and exploration of time-dependent flow fields consisting of vector data. The technique combines an improved feature extraction method proposed by Holmén with the direct methods, Quiver and Scatter plots. The modified extraction algorithm is less computationally intensive and reduces the size of the data to output clear images that are suited to the analysis of time-dependant data in the form of animations. Combing the methods with a quiver plot provides more context, clarifying the relation between the raw data and the extracted features.

Bibliography

[1] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch, "The state of the art in flow visualisation: Feature extraction and tracking," in *Computer Graphics Forum*, vol. 22, pp. 775–792, Wiley Online Library, 2003.

[2] E. Boring and A. Pang, "Directional flow visualization of vector fields," in *Proceedings of Seventh Annual IEEE Visualization'96*, pp. 389–392, IEEE, 1996.

[3] B. Jobard and W. Lefer, "Multiresolution flow visualization," 2001.

[4] X. Mao, Y. Hatanaka, H. Higashida, and A. Imamiya, "Image-guided streamline placement on curvilinear grid surfaces," in *Proceedings Visualization'98 (Cat. No. 98CB36276)*, pp. 135–142, IEEE, 1998.

[5] A. Telea and J. J. Van Wijk, *Simplified representation of vector fields*. IEEE, 1999.

[6] M. Jiang, R. Machiraju, and D. Thompson, "A novel approach to vortex core region detection," *Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2002)*.

[7] M. Soegaard, "The law of similarity - gestalt principles (1)," Feb 2020.

[8] M. Jenkins and S. Floyd, "Trajectories in the evolution of technology: A multi-level study of competition in formula 1 racing," *Organization studies*, vol. 22, no. 6, pp. 945–969, 2001.

[9] M. Diepraam, "Poachers turned gamekeepers: how the foca became the new fia part 1: Introduction and timeline," 2007.

[10] F. O. W. C. Limited, "The official home of formula 1," 2019.

[11] W. Toet, "Aerodynamics and aerodynamic research in formula 1," *The Aerodynamical Journal*, vol. 177, Janurary 2013.

[12] R. G. Dominy, "Aerodynamics of grand prix cars," *Proc Instn Mech Engrs Vol 206*, pp. 267–274, 1992.

[13] J.-Z. Wu, "Vortex definition and "vortex criteria"," *Science China Physics, Mechanics & Astronomy*, vol. 61, p. 024731, Dec 2017.

[14] V. Kolář, "Vortex identification: New requirements and limitations," *International Journal of Heat and Fluid Flow*, vol. 28, no. 4, pp. 638 – 652, 2007. Including Special Issue of Conference on Modelling Fluid Flow (CMFF'06), Budapest.

[15] T. Günther and H. Theisel, "The state of the art in vortex extraction," in *Computer Graphics Forum*, vol. 37, pp. 149–173, Wiley Online Library, 2018.

[16] R. Aris, "2.13 second order tensors," 1962.

[17] W. D. McComb, "Dynamics and relativity.," *Dynamics and relativity., by McComb, WD. Oxford University Press, Oxford (UK), 1999, XIX+ 372 p., ISBN 0-19-850112-9,*, 1999.

[18] V. Kolář, "Vortex identification: New requirements and limitations," *International journal of heat and fluid flow*, vol. 28, no. 4, pp. 638–652, 2007.

[19] V. Holmén, "Methods for vortex identification," *Master's Theses in Mathematical Sciences*, 2012.

[20] C. Berdahl and D. Thompson, "Eduction of swirling structure using the velocity gradient tensor," *AIAA journal*, vol. 31, no. 1, pp. 97–103, 1993.

[21] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch, "Feature extraction and visualisation of flow fields.," in *Eurographics (STARs)*, 2002.

[22] R. V. Klassen and S. J. Harrington, "Shadowed hedgehogs: A technique for visualizing 2d slices of 3d vector fields," in *Proceedings of the 2nd Conference on Visualization '91*, VIS '91, (Washington, DC, USA), p. 148–153, IEEE Computer Society Press, 1991.

[23] W. Schroeder, K. Martin, and B. Lorensen, "Vtk textbook," 2006.

[24] Matplotlib, "matplotlib.pyplot.quiver."

[25] Plotly, "Quiver plots."

[26] MathWorks, "quiver3."

[27] R. M. Kirby, H. Marmanis, and D. H. Laidlaw, "Visualizing multivalued data from 2d incompressible flows using concepts from painting," in *Proceedings Visualization'99 (Cat. No. 99CB37067)*, pp. 333–540, IEEE, 1999.

[28] A. Fenlon, T. David, and J. Walton, "An integrated visualization and design toolkit for flexible prosthetic heart valves," in *Proceedings Visualization 2000. VIS 2000 (Cat. No. 00CH37145)*, pp. 453–456, IEEE, 2000.

[29] J. Clyne and J. M. Dennis, "Interactive direct volume rendering of time-varying data," in *Data Visualization'99*, pp. 109–120, Springer, 1999.

[30] T. Glau, "Exploring instationary fluid flows by interactive volume movies," in *Data Visualization'99*, pp. 277–283, Springer, 1999.

[31] P. Rheingans and D. Ebert, "Volume illustration: Nonphotorealistic rendering of volume models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 3, pp. 253–264, 2001.

[32] A. Sanna, B. Montrucchio, and R. Arina, "Visualizing unsteady flows by adaptive streaklines," 2000.

[33] G. Turk and D. Banks, "Image-guided streamline placement," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 453–460, 1996.

[34] B. Jobard and W. Lefer, "Creating evenly-spaced streamlines of arbitrary density," in *Visualization in Scientific Computing'97*, pp. 43–55, Springer, 1997.

[35] V. Verma, D. Kao, and A. Pang, "A flow-guided streamline seeding strategy," in *Proceedings Visualization 2000. VIS 2000 (Cat. No. 00CH37145)*, pp. 163–170, IEEE, 2000.

[36] B. Jobard and W. Lefer, "Unsteady flow visualization by animating evenly-spaced streamlines," in *Computer Graphics Forum*, vol. 19, pp. 31–39, Wiley Online Library, 2000.

[37] C.-K. Tang and G. Medioni, "Extremal feature extraction from 3-d vector and noisy scalar fields," in *Proceedings Visualization'98 (Cat. No. 98CB36276)*, pp. 95–102, IEEE, 1998.

[38] S. Rottger, M. Kraus, and T. Ertl, "Hardware-accelerated volume and isosurface rendering based on cell-projection," in *Proceedings Visualization 2000. VIS 2000 (Cat. No. 00CH37145)*, pp. 109–116, IEEE, 2000.

[39] P. Shirley and A. Tuchman, "A polygonal approximation to direct scalar volume rendering," *ACM SIGGRAPH Computer Graphics*, vol. 24, no. 5, pp. 63–70, 1990.

[40] H. Löffelmann and E. Gröller, "Enhancing the visualization of characteristic structures in dynamical systems," in *Visualization in Scientific Computing'98*, pp. 59–68, Springer, 1998.

[41] G. Bancroft, F. Merritt, T. Plessel, P. Kelaita, K. McCabe, and A. Globus, "Fast: A multi-processed environment for visualization of computational fluid dynamics.," pp. 14–27, 461, 11 1990.

[42] M. Zöckler, D. Stalling, and H.-C. Hege, "Interactive visualiztion of 3d-vector fields using illuminated streamlines.," in *IEEE Visualization*, vol. 96, pp. 107–113, 1996.

[43] B. T. Phong, "Illumination for computer generated pictures," *Communications of the ACM*, vol. 18, no. 6, pp. 311–317, 1975.

[44] D. N. Kenwright and D. A. Lane, "Optimization of time-dependent particle tracing using tetrahedral decomposition," in *Proceedings Visualization '95*, pp. 321–328, Oct 1995.

[45] G. M. Nielson and I.-H. Jung, "Tools for computing tangent curves for linearly varying vector fields over tetrahedral domains," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 360–372, 1999.

[46] I. A. Sadarjoen, A. J. de Boer, F. H. Post, and A. E. Mynett, "Particle tracing in σ-transformed grids using tetrahedral 6-decomposition," in *Visualization in Scientific Computing'98*, pp. 71–80, Springer, 1998.

[47] S. Bryson and C. Levit, "The virtual wind tunnel," *IEEE Computer graphics and Applications*, no. 4, pp. 25–34, 1992.

[48] S.-K. Ueng, C. Sikorski, and K.-L. Ma, "Efficient streamline, streamribbon, and streamtube constructions on unstructured grids," *IEEE Transactions on Visualization and Computer Graphics*, vol. 2, no. 2, pp. 100–110, 1996.

[49] A. Fuhrmann and E. Groller, "Real-time techniques for 3d flow visualization," in *Proceedings Visualization'98 (Cat. No. 98CB36276)*, pp. 305–312, IEEE, 1998.

[50] R. S. Lramee, "Interactive 3d flow visualization using a streamrunner," in *CHI'02 Extended Abstracts on Human Factors in Computing Systems*, pp. 804–805, 2002.

[51] W. J. Schroeder, C. R. Volpe, and W. E. Lorensen, "The stream polygon: A technique for 3d vector field visualization," in *Proceedings of the 2nd conference on Visualization'91*, pp. 126–132, IEEE Computer Society Press, 1991.

[52] M. Brill, H. Hagen, H.-C. Rodrian, W. Djatschin, and S. V. Klimenko, "Streamball techniques for flow visualization," in *Proceedings Visualization'94*, pp. 225–231, IEEE, 1994.

[53] J. P. Hultquist, "Constructing stream surfaces in steady 3d vector fields," in *Proceedings of the 3rd conference on Visualization'92*, pp. 171–178, IEEE Computer Society Press, 1992.

[54] J. P. Hultquist, "Interactive numerical flow visualization using stream surfaces," *Computing Systems in Engineering*, vol. 1, no. 2-4, pp. 349–353, 1990.

[55] J. J. Van Wijk, "Implicit stream surfaces," in *Proceedings Visualization'93*, pp. 245–252, IEEE, 1993.

[56] H. Löffelmann, L. Mroz, and E. Gröller, "Hierarchical streamarrows for the visualization of dynamical systems," in *Visualization in Scientific Computing'97*, pp. 155–163, Springer, 1997.

[57] H. Löffelmann, L. Mroz, E. Grller, and W. Purgathofer, "Stream arrows: enhancing the use of stream surfaces for the visualization of dynamical systems," *The Visual Computer*, vol. 8, no. 13, pp. 359–369, 1997.

[58] J. J. Van WijK, "Flow visualization with surface particles," *IEEE Computer Graphics and Applications*, vol. 13, no. 4, pp. 18–24, 1993.

[59] R. Westermann, C. Johnson, and T. Ertl, "Topology-preserving smoothing of vector fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 3, pp. 222–229, 2001.

[60] N. Max, B. Becker, and R. Crawfis, "Flow volumes for interactive vector field visualization," in *Proceedings Visualization'93*, pp. 19–24, IEEE, 1993.

[61] B. G. Becker, D. A. Lane, and N. L. Max, "Unsteady flow volumes," in *Proceedings Visualization'95*, pp. 329–335, IEEE, 1995.

[62] J. J. Van Wijk, "Spot noise texture synthesis for data visualization," in *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pp. 309–318, 1991.

[63] A. Sanna and P. Montuschi, "A survey on visualization of vector fields by texture-based methods," in *Devel. Pattern Rec*, Citeseer, 2000.

[64] W. C. de Leeuw and J. J. Van Wijk, "Enhanced spot noise for vector field visualization," in *Proceedings Visualization'95*, pp. 233–239, IEEE, 1995.

[65] W. De Leeuw and R. Van Liere, "Divide and conquer spot noise," in *Proceedings of the 1997 ACM/IEEE conference on Supercomputing*, pp. 1–13, 1997.

[66] W. C. de Leeuw, F. H. Post, and R. W. Vaatstra, "Visualization of turbulent flow by spot noise," in *Virtual Environments and Scientific Visualization'96*, pp. 286–295, Springer, 1996.

[67] W. De Leeuw and R. Van Liere, *Comparing lic and spot noise*. IEEE, 1998.

[68] B. Cabral and L. C. Leedom, "Imaging vector fields using line integral convolution," in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pp. 263–270, 1993.

[69] H.-W. Shen, C. R. Johnson, and K.-L. Ma, "Visualizing vector fields using line integral convolution and dye advection," in *Proceedings of 1996 Symposium on Volume Visualization*, pp. 63–70, IEEE, 1996.

[70] M.-H. Kiu and D. C. Banks, "Multi-frequency noise for lic," in *Proceedings of Seventh Annual IEEE Visualization'96*, pp. 121–126, IEEE, 1996.

[71] L. Khouas, C. Odet, and D. Friboulet, "2d vector field visualization using furlike texture," in *Data Visualization'99*, pp. 35–44, Springer, 1999.

[72] D. Stalling and H.-C. Hege, "Fast and resolution independent line integral convolution," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 249–256, 1995.

[73] H.-C. Hege and D. Stalling, "Fast lic with piecewise polynomial filter kernels," in *Mathematical Visualization*, pp. 295–314, Springer, 1998.

[74] B. Cabral and C. Leedom, "Highly parallel vector visualization using line integral convolution," tech. rep., Society for Industrial and Applied Mathematics, Philadelphia, PA (United States), 1995.

[75] D. Stalling, "Parallel line integral convolution," *Parallel Computing*, vol. 23, no. 7, pp. 975–989, 1997.

[76] R. Wegenkittl, E. Groller, and W. Purgathofer, "Animating flow fields: rendering of oriented line integral convolution," in *Proceedings. Computer Animation'97 (Cat. No. 97TB100120)*, pp. 15–21, IEEE, 1997.

[77] R. Wegenkittl and E. Groller, "Fast oriented line integral convolution for vector field visualization via the internet," in *Proceedings. Visualization'97 (Cat. No. 97CB36155)*, pp. 309–316, IEEE, 1997.

[78] S. Berger and E. Gröller, "Color-table animation of fast oriented line integral convolution for vector field visualization," 2000.

[79] H. Löffelmann, A. König, and E. Gröller, "Fast visualization of 2d dynamical systems by the use of virtual ink droplets," in *13th Spring Conference on Computer Graphics*, pp. 111–118, 1997.

[80] B. Jobard and W. Lefer, "The motion map: efficient computation of steady flow animations," in *Proceedings. Visualization'97 (Cat. No. 97CB36155)*, pp. 323–328, IEEE, 1997.

[81] B. Jobard, G. Erlebacher, and M. Y. Hussaini, "Lagrangian-eulerian advection for unsteady flow visualization," in *Proceedings Visualization, 2001. VIS'01.*, pp. 53–541, IEEE, 2001.

[82] B. Jobard, G. Erlebacher, and M. Y. Hussaini, "Tiled hardware-accelerated texture advection for unsteady flow visualization," *Proceedings of Graphicon 2000*, pp. 189–196, 2000.

[83] B. Jobard, G. Erlebacher, and M. Y. Hussaini, "Hardware-accelerated texture advection for unsteady flow visualization," in *Proceedings Visualization 2000. VIS 2000 (Cat. No. 00CH37145)*, pp. 155–162, IEEE, 2000.

[84] W. C. De Leeuw, H.-G. Pagendarm, F. H. Post, and B. Walter, "Visual simulation of experimental oil-flow visualization by spot noise images from numerical flow simulation," in *Visualization in Scientific Computing'95*, pp. 135–148, Springer, 1995.

[85] L. K. Forssell, "Visualizing flow over curvilinear grid surfaces using line integral convolution," in *Proceedings Visualization'94*, pp. 240–247, IEEE, 1994.

[86] L. K. Forssell and S. D. Cohen, "Using line integral convolution for flow visualization: Curvilinear grids, variable-speed animation, and unsteady flows," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 133–141, 1995.

[87] C. Teitzel, R. Grosso, and T. Ertl, "Line integral convolution on triangulated surfaces," in *WSCG 1997 Conference Proceedings*, pp. 572–581, Citeseer, 1997.

[88] X. Mao, M. Kikukawa, N. Fujita, and A. Imamiya, "Line integral convolution for 3d surfaces," in *Visualization in Scientific Computing'97*, pp. 57–69, Springer, 1997.

[89] H. Battke, D. Stalling, and H.-C. Hege, "Fast line integral convolution for arbitrary surfaces in 3d," in *Visualization and Mathematics*, pp. 181–195, Springer, 1997.

[90] X. Mao, L. Hong, A. Kaufman, N. Fujita, M. Kikukawa, and A. Imamiya, "Multi-granularity noise for curvilinear grid lic," in *Graphics Interface*, vol. 98, pp. 193–200, 1998.

[91] H.-W. Shen and D. L. Kao, "Uflic: a line integral convolution algorithm for visualizing unsteady flows," in *Proceedings. Visualization'97 (Cat. No. 97CB36155)*, pp. 317–322, IEEE, 1997.

[92] H.-W. Shen and D. L. Kao, "A new line integral convolution algorithm for visualizing time-varying flow fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 2, pp. 98–108, 1998.

[93] V. Verma, D. Kao, and A. Pang, "Plic: Bridging the gap between streamlines and lic," in *Proceedings Visualization'99 (Cat. No. 99CB37067)*, pp. 341–541, IEEE, 1999.

[94] C. Rezk-Salama, P. Hastreiter, C. Teitzel, and T. Ertl, "Interactive exploration of volume line integral convolution based on 3d-texture mapping," in *Proceedings Visualization'99 (Cat. No. 99CB37067)*, pp. 233–528, IEEE, 1999.

[95] V. Interrante and C. Grosch, *Strategies for effectively visualizing 3D flow with volume LIC*. IEEE, 1997.

[96] V. Interrante and C. Grosch, "Visualizing 3d flow," *IEEE Computer Graphics and Applications*, vol. 18, no. 4, pp. 49–53, 1998.

[97] H. K. Moffatt, "The degree of knottedness of tangled vortex lines," *Journal of Fluid Mechanics*, vol. 35, no. 1, pp. 117–129, 1969.

[98] D. Degani, Y. Levy, and A. Seginer, "Graphical visualization of vortical flows by means of helicity," *AIAA Journal*, vol. 28, no. 8, pp. 1347 – 1352, 1990.

[99] D. Sujudi and R. Haimes, "Identification of swirling flow in 3-d vector fields," in *12th Computational Fluid Dynamics Conference*, p. 1715, 1995.

[100] I. A. Sadarjoen and F. H. Post, "Geometric methods for vortex extraction," in *Data Visualization'99*, pp. 53–62, Springer, 1999.

[101] B. Ganapathisubramani, E. K. Longmire, and I. Marusic, "Characteristics of vortex packets in turbulent boundary layers," *Journal of Fluid Mechanics*, vol. 478, pp. 35–46, 2003.

[102] J. C. Hunt, A. A. Wray, and P. Moin, "Eddies, streams, and convergence zones in turbulent flows," 1988.

[103] M. S. Chong, A. E. Perry, and B. J. Cantwell, "A general classification of three-dimensional flow fields," *Physics of Fluids A: Fluid Dynamics*, vol. 2, no. 5, pp. 765–777, 1990.

[104] J. Jeong and F. Hussain, "On the identification of a vortex," *Journal of fluid mechanics*, vol. 285, pp. 69–94, 1995.

[105] J. Zhou, R. J. Adrian, S. Balachandar, and T. M. Kendall, "Mechanisms for generating coherent packets of hairpin vortices in channel flow," *Journal of Fluid Mechanics*, vol. 387, p. 353–396, 1999.

[106] P. Charkraborty, S. Balachandar, and R. J. Adrian, "On the relationships between local vortex identification schemes," *Journal of Fluid Mechanics*, vol. 535, p. 189–214, 2005.

[107] D. C. Banks and B. A. Singer, "Vortex tubes in turbulent flows: Identification, representation, reconstruction," in *Proceedings of the Conference on Visualization '94*, VIS '94, (Washington, DC, USA), p. 132–139, IEEE Computer Society Press, 1994.

[108] A. F. Hussain, "Coherent structures and turbulence," *Journal of Fluid Mechanics*, vol. 173, pp. 303–356, 1986.

[109] H. J. Lugt, "The dilemma of defining a vortex," in *Recent developments in theoretical and experimental fluid mechanics*, pp. 309–321, Springer, 1979.

[110] R. Cucitore, M. Quadrio, and A. Baron, "On the effectiveness and limitations of local criteria for the identification of a vortex," *European Journal of Mechanics-B/Fluids*, vol. 18, no. 2, pp. 261–282, 1999.

[111] M. Jiang, R. Machiraju, and D. Thompson, "Detection and visualization of vortices,"

[112] D. N. Kenwright and R. Haimes, "Automatic vortex core detection," *IEEE Computer Graphics and Applications*, vol. 18, no. 4, pp. 70–74, 1998.

[113] M. Roth and R. Peikert, "A higher-order method for finding vortex core lines," in *Proceedings Visualization'98 (Cat. No. 98CB36276)*, pp. 143–150, IEEE, 1998.

[114] R. Peikert and M. Roth, "The 'parallel vectors' operator-a vector field visualization primitive," in *Proceedings Visualization'99 (Cat. No. 99CB37067)*, pp. 263–532, IEEE, 1999.

[115] A. E. Perry and M. S. Chong, "A description of eddying motions and flow patterns using critical-point concepts," *Annual Review of Fluid Mechanics*, vol. 19, no. 1, pp. 125–155, 1987.

[116] K.-L. Ma, J. Van Rosendale, and W. Vermeer, "3d shock wave visualization on unstructured grids," in *Proceedings of 1996 Symposium on Volume Visualization*, pp. 87–94, IEEE, 1996.

[117] M. Roth, *Automatic extraction of vortex core lines and other line type features for scientific visualization*, vol. 2. Hartung-Gorre, 2000.

[118] D. N. Kenwright, C. Henze, and C. Levit, "Feature extraction of separation and attachment lines," *IEEE transactions on Visualization and Computer Graphics*, vol. 5, no. 2, pp. 135–144, 1999.

[119] M. In den Haak, H. Spoelder, and F. Groen, "Matching of images by using automatically selected landmarks," *Proceedings of Computer Science in the Netherlands*, pp. 27–40, 1992.

[120] C. Weigle and D. C. Banks, "Extracting iso-valued features in 4-dimensional scalar fields," in *IEEE Symposium on Volume Visualization (Cat. No. 989EX300)*, pp. 103–110, IEEE, 1998.

[121] D. Silver and X. Wang, "Volume tracking," in *Proceedings of Seventh Annual IEEE Visualization'96*, pp. 157–164, IEEE, 1996.

[122] D. Silver and X. Wang, "Tracking and visualizing turbulent 3d features," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 2, pp. 129–141, 1997.

[123] D. Silver and X. Wang, "Visualizing evolving scalar phenomena," *Future Generation Computer Systems*, vol. 15, no. 1, pp. 99–108, 1999.

[124] R. Samtaney, D. Silver, N. Zabusky, and J. Cao, "Visualizing features and tracking their evolution," *Computer*, vol. 27, no. 7, pp. 20–27, 1994.

[125] I. K. Sethi, N. V. Patel, and J. H. Yoo, "A general approach for token correspondence," *Pattern Recognition*, vol. 27, no. 12, pp. 1775–1786, 1994.

[126] F. Reinders, F. H. Post, and H. J. Spoelder, "Attribute-based feature tracking," in *Data visualization'99*, pp. 63–72, Springer, 1999.

[127] F. Reinders, F. H. Post, and H. J. Spoelder, "Visualization of time-dependent data with feature tracking and event detection," *The Visual Computer*, vol. 17, no. 1, pp. 55–71, 2001.

[128] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[129] Produle, "Wireframe tools, prototyping tools, ui mockups, ux suite, remote designing."

[130] K. F. Reinders, K. Frederik, and J. Reinders, "Feature-based visualization of time-dependent data," 2001.

[131] empet, "Animate succesively two traces in the same figure." https://chart-studio.plotly.com/ẽmpet/15557/animate-succesively-two-traces-in-the-sa/ [Last accessed: 01/05/2020].

[132] Bokeh Development Team, *Bokeh: Python library for interactive visualization*, 2018.

Appendix A

Ethics Checklist

This form must be attached to the dissertation as an appendix.

Department of Computer Science
12-Point Ethics Checklist for UG and MSc Projects

Student Oliver Eisenberg

Academic Year or Project Title 2020

Supervisor Yongliang Yang

Does your project involve people for the collection of data other than you and your supervisor(s)? **YES** / ~~**NO**~~

If the answer to the previous question is YES, you need to answer the following questions, otherwise you can ignore them.

This document describes the 12 issues that need to be considered carefully before students or staff involve other people ('participants' or 'volunteers') for the collection of information as part of their project or research. Each answer must be affirmative. Replace the text beneath each question with a statement of how you address the issue in your project.

1. *Have you prepared a briefing script for volunteers?* **YES** / ~~**NO**~~
   A briefing has been written in advance so that participants can understand what is involved and why.

2. *Will the participants be informed that they could withdraw at any time?* **YES** / ~~**NO**~~
   All participants have the right to withdraw at any time during the investigation, and to withdraw their data up to the point at which it is anonymised. They are told this in the briefing script.

3. *Is there any intentional deception of the participants?* ~~**YES**~~ / **NO**
   Withholding information or misleading participants is not done during the study.

4. *Will participants be de-briefed?* **YES** / ~~**NO**~~
   Participants will be informed in the briefing that debriefing is available to understand the nature of the investigation. This phase is held until after the study is completed as it is necessary to protect the integrity of the study.

5. *Will participants voluntarily give informed consent?* **YES** / ~~**NO**~~
   Participants are required to provide their consent before taking part in the study, informed by the briefing sheet. Participants give their consent explicitly.

6. *Will the participants be exposed to any risks greater than those encountered in their normal work life (e.g., through the use of non-standard equipment)?* ~~**YES**~~ / **NO**
   The risk of harm is no greater than in ordinary life.

7. *Are you offering any incentive to the participants?* ~~**YES**~~ / **NO**
   The payment of participants is not used to avoid biasing the sample of participants.

8. *Are you in a position of authority or influence over any of your participants?* ~~**YES**~~ / **NO**
   Participants are asked about their opinions of data, without the presence of the interviewer.

9. *Are any of your participants under the age of 16?* ~~**YES**~~ / **NO**
   Participants will not be under the age of 16.

10. *Do any of your participants have an impairment that will limit Their understanding or communication?* **YES** / ~~**NO**~~
    Participants with impairments will not be sought after to conduct the experiment, however, adjustments will be made to aid those with impairments who partake in the study.

11. *Will the participants be informed of your contact details?* **YES** / ~~**NO**~~
    Participants will have the email address of the interviewer in case they have further question after the study is completed.

12. *Do you have a data management plan for all recorded data?* **YES** / ~~**NO**~~
    All participant data is autonomised and stored online automatically from user input.

Appendix B

Requirements

B.1 Scuderia Toro Rosso Meeting notes

Group Discussion at Scuderia Toro Rosso

Representatives:
Aero Performance Engineer, Aero Performance Intern, Aero Systems Leader, Aerodynamicist Team Lead, Wind Tunnel Engineer

Q: What is currently done to visualise vortices?
A: Data from CFD use streamlines seeding from a dropdown list of vortices. We don't tack vortices from the wind tunnel but can produce animations from samples collected prior to POD (Proper Orthogonal Decomposition) and averaging. Furthermore, a method is being worked on to allow for thresholding to produce ISO surfaces around the path of a streamline but this would only work on CFD data.

Q: What do you allow users to track?
A: Users can select from a predefined list which includes the Y-250, Z-0, BB1 (Barge Board 1), BB2 (Barge Board 2), FBB (Front Barge Board) and a couple more.

Q: What features would you like to see your current system expanded to?
A: Strength and Tracking of the Y-250, or just general strength of decays and births. Could identify structures and correlated them over time. It would be useful to use two dimensional analysis on instantaneous data to track and plot a cores location and represents that in a digestible manner.

Q: Have you used other tools in the past?
A: Yes, a tool based on eigenanalysis was used that did a BL swirl or cloud on point technique. It was quite slow to use though.

Q: How are results generated and viewed at the moment?
A: Currently streamlines are only generated for CFD results which are able to be accessed using an internal tool. PIV data is a bit different, it is kept separate - stored in files on a network drive on the company's network.

Q: This existing tool, may I ask how that is hosted?
A: The tool is web based and run on our intranet.

Q: Who is likely to use the tool?
A: I (Wind Tunnel Engineer) will or Jerome (Aerodynamicist Team Lead) and maybe a the odd person but it would not be for a large audience.

Appendix C

Code

C.1 Access

The work can be accessed on: https://people.bath.ac.uk/ohe21/Eisenberg_FYP.html

C.2 Algorithms

Algorithm 1 Get Neighbouring Node Algorithm

```
1: function GET_RIGHTNODE(self, node)
2:     self.shift_x_axis(node, -1)                    ▷ -1 equals right shift
3: end function
1: function SHIFT_X_AXIS(self, node, shift)
2:     x_locs ← node.x_idx + shift
3:     if  x_locs < 0 then
4:         raise InvalidNodeException                 ▷ non-existant node
5:     end if
6:     new_x ← self.x[x_locs]          ▷ gets the x co-ordinate of the new node
7:     return self.getNodeAt(new_x, node.y)
8: end function
```

Algorithm 2 Get Neighbouring non-Vortex Node Algorithm

```
1: function GET_NODE(data, node, x_increment, y_increment)
2:     while not node.isVortex do
3:         node ← data.shift_x_axis(node, x_increment)
4:         node ← data.shift_y_axis(node, y_increment)
5:     end while
6:     return node
7: end function
```

Algorithm 3 Holmén's Identification Algorithm

```
1: function IDENTIFACTION_ALGORITHM(data)
2:     labelled_data ← label_data(data)        ▷ labels u, v using a sign function
3:     labelled_data ← reformat_data(data)   ▷ from NodeCollection to Grid
4:     return identify_vortices(labelled_data)
5: end function
1: function IDENTIFY_VORTICES(data)
2:     for all y in data.y do
3:         for all x in data.x do
4:             node ← data.getNodeAt(x, y)
5:             if not data.contains(node) then
6:                 data.addNode(node)                          ▷ add empty node
7:             end if
8:             if isVortex(data, node) then                    ▷ add vortex node
9:                 data.vortices.addNode(node)
10:            end if
11:        end for
12:        return data
13:    end for
14: end function
1: function ISVORTEX(data, node)      ▷ Check node's non vortex neighbours
2:     left_node ← get_Node(data, vortex, 1, 0)
3:     right_node ← get_Node(data, vortex, -1, 0)
4:     upper_node ← get_Node(data, vortex, 0, -1)
5:     lower_node ← get_Node(data, vortex, 1, 0)
6:     if left_node.v_signed + right_node.v_signed == 0 and
7:         upper_node.u_signed + lower_node.u_signed == 0 then
8:         if not left_node.v_signed + upper_node.u_signed == 0 then
9:             return True
10:        end if
11:    end if
12:    return False
13: end function
```

Algorithm 4 Modified Growth Algorithm

```
1: function GWTH_ALGO(data)
2:     visited ← List()
3:     for all vertex in vertices do
4:         visited.addNode(vertex)
5:         BFS(data, vertex, visited)                ▷ 'data' is of type Grid
6:     end for
7: end function
1: function BFS(data, init_vortex, visited)
2:     frontier = Queue()
3:     frontier.put(init_vortex)
4:     while not frontier.empty() do   ▷ Check node's non vortex neighbours
5:         vortex ← frontier.get()
6:         left_node ← data.get_leftnode(vortex)
7:         right_node ← data.get_rightnode(vortex)
8:         upper_node ← data.get_uppernode(vortex)
9:         lower_node ← data.get_lowernode(vortex)
10:        checkNode(data, left_node, visited, init_vortex)
11:        checkNode(data, right_node, visited, init_vortex)
12:        checkNode(data, upper_node, visited, init_vortex)
13:        checkNode(data, lower_node, visited, init_vortex)
14:    end while
15:    return data, visited
16: end function
1: function CHECKNODE(data, node, visited, init_vortex)
2:     if not node in visited then
3:         if isVortex(data, node) or node.isVortex then          ▷ new or old
4:             data.vortices.addNode(node)
5:             frontier.put(node)
6:         end if
7:         visited.addNode(node)
8:     end if
9:     return data, frontier, visited
10: end function
```

Algorithm 5 Reinder's Initalisation Tracking Algorithm

```
1:  function StartPaths(data)
2:      for all Frame in data do
3:          for all Feature in Frame do
4:              for all Feature in Frame_{+1} do
5:                  prediction ← Prediction(feature,
6:                  candidate_{+1})                          ▷ assume connection
7:                  for all candidates in Frame_{+2} do
8:                      c ← Correspondence(prediction, candidate_{+2})
9:                      if c ≥ 0 then
10:                         path ← Path(prediction, c)          ▷ create new path
11:                         path ← ContinuePath(path, frame_{+3})
12:                     end if
13:                     if length(path) > minimum_path_length then
14:                         path.end()                                ▷ add Path
15:                     end if
16:                 end for
17:             end for
18:         end for
19:     end for
20: end function
1:  function ContinuePath(path, frame)                            ▷ DFS
2:      if self.all_frames ≥ length(path)  then
3:          return path                                       ▷ end of frames
4:      end if
5:      if length(potential_paths) == 0 then
6:          return path                                        ▷ end of path
7:      end if
8:      for all candidates in potential_paths do
9:          new_path.addFeature(candidate)
10:         possible_branches.append(continue_path(new_path, Frame_{+1}))
11:     end for
12:     return max_confidence(possible_branches)              ▷ best branch only
13: end function
```

Algorithm 6 Automatic Selection Algorithm

```
1: function AUTOMATIC_SELECTION(family)
2:     if family.size == 1 then
3:         return [ ]                                   ▷ add no arrows
4:     end if
5:     mean ← [family.x, family.y]                      ▷ centers the distribution
6:     cov ← [[6+family.size*2,0], [0,6+family.size*2]]
7:     nodes ← random.multivariate_normal(mean,cov,100)
8:     nodes ← convex_hull_algos.gift_wrapping(nodes)
9:     nodes ← random.choices(nodes, 1+family.size)
10:    return nodes
11: end function
```

Algorithm 7 Gift wrapping algorithm

```
1: function GIFT_WRAPPING(set_of_nodes)
2:     if length(set_of_nodes) ≤ 3 then
3:         return set_of_nodes
4:     end if
5:     node_on_hull ← get_left_node(set_of_nodes)
6:     nodes_on_convex_hull ← [ ]
7:     endpoint ← node_on_hull
8:     q ← 0
9:     while True do
10:        nodes_on_convex_hull.append(set_of_nodes[endpoint])
11:        q ← (endpoint+1) % len(set_of_nodes[0])
12:        for all i in range(len(set_of_nodes) do
13:            if orientation(endpoint, set_of_nodes[i], set_of_nodes[q]) == 2
   then
14:                q ← i
15:            end if
16:            endpoint ← q                    ▷ greater left turn found, update
17:        end for
18:        if set_of_nodes[endpoint] == nodes_on_convex_hull[0] then
19:            break                                    ▷ wrapped around
20:        end if
21:    end while
22:    return set_of_nodes
23: end function
```

Algorithm 8 Monotone chain algorithm

```
1: function MONOTONE_CHAIN(set_of_nodes)
2:     sort(set_of_nodes)                                   ▷ sort by x then by y
3:     if length(set_of_nodes) ≤ 1 then
4:         return set_of_nodes
5:     end if
6:     upper ← list()
7:     for all node in reversed(set_of_nodes) do                    ▷ Upper hull
8:         while length(upper) ≥ 2 and
9:             orientation(upper[-2], upper[-1], node) == 2 or 0) do
10:            upper.pop()                                ▷ Remove if to the left
11:        end while
12:        upper.append(node)                          ▷ Assume node is correct
13:    end for
14:    lower ← list()
15:    for all node in set_of_nodes do  ▷ Lower hull  ▷ same as for upper hull
16:    end for
17:    return set_of_nodes
18: end function
```

Algorithm 9 Parallel Contextual Annimation

```
1: function PARALLEL_CONTEXTUAL_ANNIMATION(dfs)
2:     df ← dfs[0]
3:     main_quivers ← dfs[1]
4:     all_quivers ← dfs[2]
5:     dataset_by_frame ← df_0
6:     dataset_by_nextframe ← df_1
7:     quivers ← all_quivers_0
8:     main_quivers ← main_quivers_0
9:     fig ← create_initFrame(dataset_by_frame, dataset_by_nextframe, quivers,
   main_quivers)
10:    frames ← [ ]
11:    for all iFrame in n_Frames do
12:        dataset_by_frame ← df_iFrame
13:        dataset_by_nextframe ← df_iFrame+1
14:        quivers ← all_quivers_iFrame
15:        main_quivers ← main_quivers_iFrame
16:        frames ← create_frames(frames, dataset_by_frame,
17:        dataset_by_nextframe, quivers, main_quivers
18:    end for
19:    animate_button ← dict(...)
20:    sliders ← dict(...)
21:    fig.update_layout(..., sliders=sliders, updatemenus=[animate_button])
22:    fig.frames ← frames
23:    return fig
24: end function
1: function CREATE_INITFRAME(dataset_by_frame, dataset_by_nextframe,
   quivers main_quivers)
2:     fig ← go.Figure()
3:     fig.add_scatter(dataset_by_nextframe)
4:     fig.add_scatter(dataset_by_frame)
5:     fig.add_traces(ff.create_quiver(quivers).data[0])
6:     fig.add_traces(ff.create_quiver(main_quivers).data[0])
7:     return fig
8: end function
1: function CREATE_FRAMES(frames, dataset_by_frame,
   dataset_by_nextframe,quivers, main_quivers)
2:     frames.append(go.Frame(data=[
3:         go.scatter(dataset_by_nextframe),
4:         go.scatter(dataset_by_frame),
5:         ff.create_quiver(quivers).data[0],
6:         ff.create_quiver(main_quivers).data[0]],
7:     traces←[0,1,2,3],
8:     name←'frame:'frame))
9:     return frames
10: end function
```

Algorithm 10 Knn algorithm

```
1: function KNN_CLASSIFIER(vortices, all_nodes)
2:     family_labels ←generate_lables(vortices)
3:     neigh ← KNeighborsClassifier(n_neighbors, weights)
4:     neigh.fit(vortices, family_labels)
5:     distance, indicies ← neigh.kneighouts(vortices)
6:     family_idxs ← make_predictions(distance, indices, family_labels)
7:     return adopt_nodes(vortices, all_nodes, family_idxs)
8: end function
1: function MAKE_PREDICTION(all_distances, indices, familiy_labels)
2:     family_values ← familiy_labels[indices]          ▷ list of family indicies
3:     for all node_distances in all_distances do
4:         for all distance in node_distances do
5:             if distance ≥ threshold then
6:                 family_values.rmNode()               ▷ Removes neighbour
7:             end if
8:         end for
9:         predictions ← calcMean(family_values)        ▷ rounded down
10:    end for
11:    return predictions
12: end function
```

Appendix D

Glossary

Freestream Air before it encounters an aerodynamic body. 52

Inboard An aerodynamic term, used to describe an area located near the center of the racecar. In the data, inboard is more negative on the x-axis.. 68

Outboard An aerodynamic term, used to describe an area located farther from the center of the racecar. In the data, outboard is more positive on the x-axis.. 52

Y-250 An acronym for a particular vortex found at $y = 250$ which is generated from a front wing of the racecar. 36

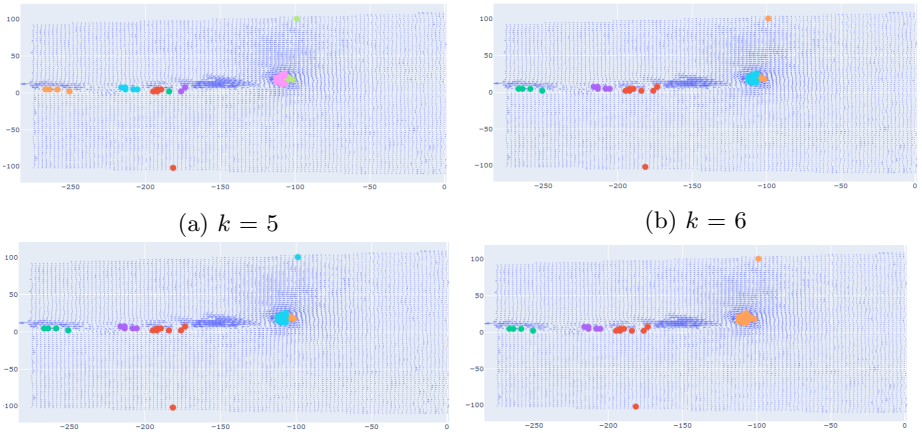
Z-0 An acronym for a particular vortex found at $z = 0$. 36

Appendix E

Testing

E.1 K-Nearest Neighbours

This chapter contains the output when testing the values for the K-Nearest neighbours algorithm. In these plots - made by Plotly, colour is used to differentiate between features.



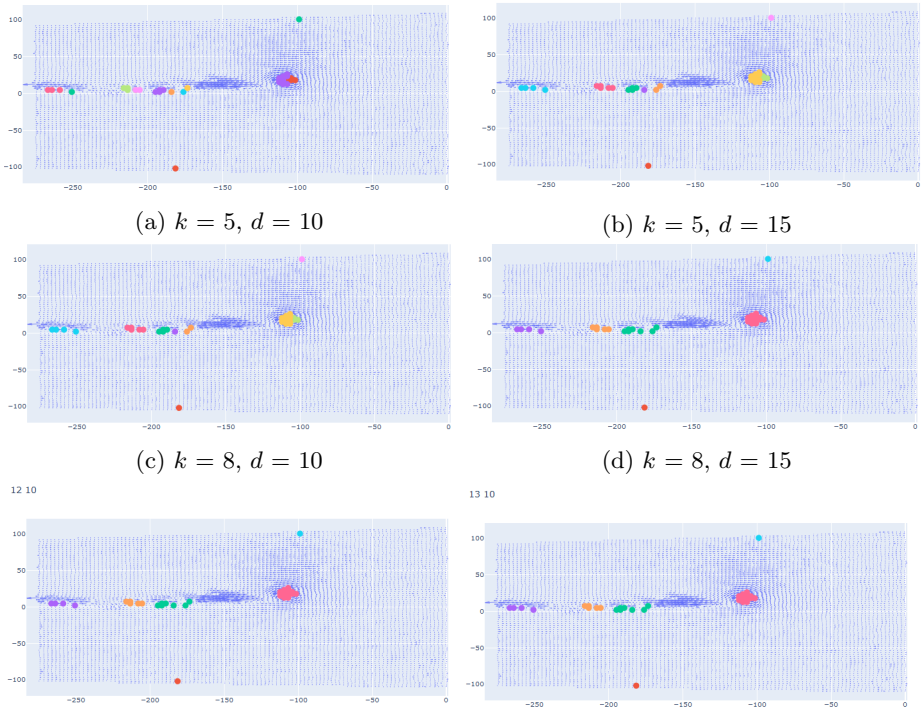
(a) $k = 5$

(b) $k = 6$

(c) $k = 7$

(d) $k = 8$

Figure E.1: Frame 2: K-Nearest Neighbour k testing



(a) $k = 5$, $d = 10$

(b) $k = 5$, $d = 15$

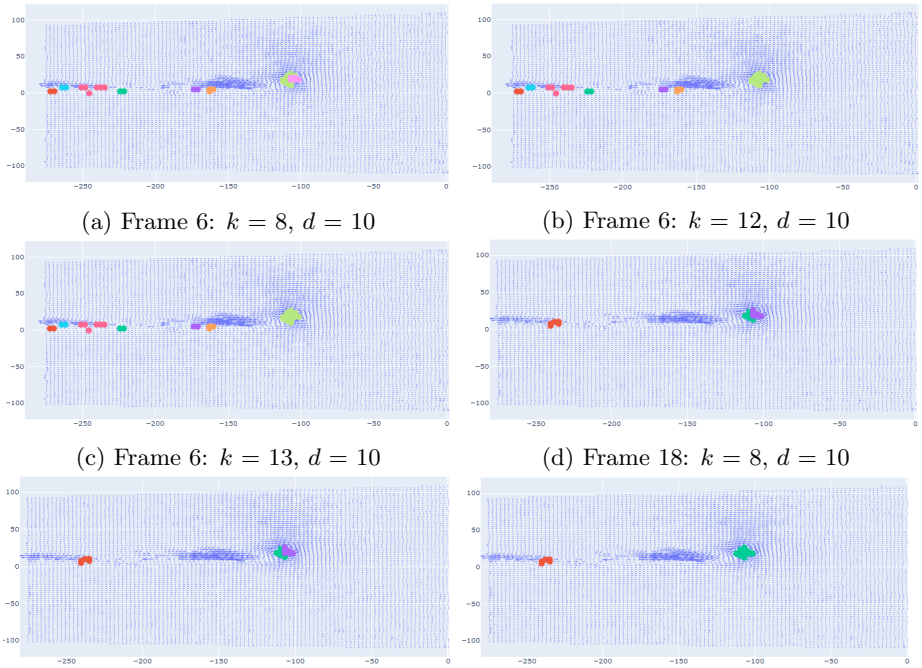
(c) $k = 8$, $d = 10$

(d) $k = 8$, $d = 15$

(e) $k = 12$, $d = 10$

(f) $k = 13$, $d = 10$

Figure E.2: Frame 2: K-Nearest Neighbour k and distance threshold testing



(a) Frame 6: $k = 8$, $d = 10$

(b) Frame 6: $k = 12$, $d = 10$

(c) Frame 6: $k = 13$, $d = 10$

(d) Frame 18: $k = 8$, $d = 10$

(e) Frame 18: $k = 12$, $d = 10$

(f) Frame 18: $k = 13$, $d = 10$

Figure E.3: KNN k and distance threshold refining

Appendix F

User Study

A user study was performed to evaluate the system. The study aheard to the ethics guidline as required and took place virtually usiing Google forms. The comments from the study have been transcribed below.

User Study Summary

1) How many vortex cores are you able to identify in the following image?

1: 70% 3: 20%

2) How easy was it to complete the task ?

1: 20% 2:0% 3:20% 4:40% 5:20%

2) Was your answer correct?

No: 100%

3) What would make it clearer ?

Larger arrows (therefore a larger graph). Different colours where there are vortices (separate colours if multiple are close together)
Maybe a use of a different colour to highlight vortices?
I have no idea
Labelling the 6 vortices
Identifying the cores as well as the velocities

4) How confident are you with the plot presented?

1: 60% 2:20% 3:20% 4:0% 5:0%

5) Are you able to identify the six vortices?

No: 100%

6) How easy was it to complete the task ?

1: 0% 2:0% 3:40% 4:0% 5:60%

7) Did being able to interact help? Please explain your answer.

I was definitely closer to being able to identify them, but it was still fairly hard.
Yes, I could zoom in so I could see in more detail what direction the arrows were pointing (although I still had no idea what it really meant)
Yes, being able to zoom in was helpful to see the individual arrows
Not really. I didn't understand how I was interacting with it and what the results were showing
Yes as i could zoom in and move around

8) What would make it clearer ?

Clear instructions on how to use the interactive graph. I felt like I had to discover how to use it by myself
Clearer explanation of what a vortex should look like? Im not really sure what Im looking for
Maybe colours? I really struggle to see where these vortcies are
a 3D view. Finding it hard to visualise what the 2D space is showing
color gradients when areas get more dense

9) How confident are you with the plot presented?

1: 0% 2:40% 3:20% 4:40% 5:0%

10) Are you able to identify the six vortices?

No: 100%

11) How easy was it to complete the task ?

1: 40% 2:40% 3:0% 4:20% 5:0%

12) How many vortices can you identify in frame 3? (count from 0)

2:40% 3:60%

13) How easy was it to complete the task ?

1: 40% 2:40% 3:0% 4:20% 5:0%

14) Was your answer correct?

Yes: 80% No: 20%

15) What would make it clearer ?

Can't think of anything, this was super clear
I dont know what a vortex should look like, just took a guess because there were two main clusters
That one was really clear.

16) Are you able to track any movement - What is moving?

Yes, perhaps the vorticies?
Yes, the arrows are going off to the left, the vortex is spinning?
There seems to be some movement from right to left in the bottom left
Looks like vortex at -100, 20 is the only vortex I can see moving. The others are new on each frame.
vortex in the middle is spinning

17) Can you track any movement? - what is moving?

The dots moving in direction of the arrows
yes movement but not sure what it means
Same again
Yes. The vortices in the lower half of the image
Can't tell

18) Has the addition of the arrows helped?

Yes: 60% Maybe:40% No:0%

19) What would make it clearer ?

A description of what the arrows mean, near the plot
clicking on Quiver removed all arrows so I thought I was meant to be looking at the graph without arrows until I read the second question. I had been looking at the arrows for all the previous questions.
Colour consistency between frames

20) How satisfied are you with your answers for part A?

1:40% 2:20% 3:20% 4:20% 5:0%

21) How satisfied are you with your answers for part B?

1:20% 2:0% 3:20% 4:60% 5:0%

22) How would you describe the interface for graph A (static image)

Ambiguous
Quite difficult to understand but maybe an expert would understand whats going on?
Quite complex if you don't know what you're looking for
Very confusing but I think if there was some more explanation in the intro or a more detailed example that resembled the questions more closely then it would be fine.
Requires thought to understand

23) How would you describe the interface for graph A (interactive)

Less ambiguous, but hard to know how to interact with it
Still very difficult to understand but appreciated being able to zoom
The same, but ability to zoom helped
Didn't know what I was doing.
Easier to navigate, but still hard to understand

24) How would you describe the interface for graph B

Much clearer and more intuitively interactive
Still had no idea what was going on but looked pretty - liked the colours and labels
Much clearer information, but less detail
Easier to work with
Better than graph A