Master's Thesis
at the
Department of Mathematics and Computer Science

Classification of Lagrangian tracer trajectories in 3D fluid turbulence

Malte Detlefsen
`mdetlefsen@posteo.de`

Advisors:
Prof. Dr. Daniel Göhring
Prof. Dr. Wolf-Christian Müller

December 11, 2016

Declaration of authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.

This paper was not previously presented to another examination board and has not been published.

Berlin, 11th of December 2016

Malte Detlefsen

Abstract

In fluid dynamics, computer simulations have become an invaluable tool to study the effects of turbulence in fluid motion. Especially direct numerical simulations provide the means to observe turbulence in great detail. However, the vast amounts of data produced make statistical analysis difficult, if time and manpower are scarce. To gather relevant information, automatic feature extraction becomes a necessity.

This thesis presents an approach for automatic classification of Lagrangian tracer trajectories in 3D fluid turbulence using discrete wavelet transform as a basis for feature extraction. A typical phenomenon of turbulent fluid motion is the *vortex*, describing a swirling motion around a common axis of rotation. Its oscillating movements become visible in the lateral acceleration of a particle tracer, conceptually causing regions of increased energy in the coefficients of a discrete wavelet transform, which in conjunction with statistical considerations build the basis for an automatic vortex detection.

Subsequently, an unsupervised cluster analysis built on different measures of similarity is performed, trying to uncover significant structure and generative processes within the data.

To my friends and family.

Contents

1 Introduction

1.1 Motivation

Turbulence is part of our everyday life. We observe its effects in the motion of fluids, the distributions of aerosols or in geological phenomena like lahars. Despite its ubiquitous nature, turbulence is still not very well understood. Seemingly simple problems like the dissipation rate of kinetic energy in a stirred up fluid are questions scientists still try to answer to a satisfactory degree. Unfortunately, analytical solutions to turbulence are "notoriously difficult to deal with" [1] and in many fields of study, like astrophysics, natural conditions are difficult to reproduce in a controlled environment. Especially, systems with very strong turbulences are often of extraterrestrial nature, like solar winds or Jupiter's great red spot. It is therefore unsurprising that under these conditions research still needs ground to cover in order to fully understand the processes that drive turbulence.

But also under earthly conditions, turbulence eludes most of our understanding. Unfortunately, it plays an important role in several areas that affect social or personal life. Engineers need to know the air flow around an aircraft in order to ensure safety, physicians need to understand the blood flow in the human heart to detect irregularities caused by certain diseases, and meteorologists need to predict the development of tropical storms in order to issue warnings to the public in case of danger. Since the first experiments by Osborne Reynolds in 1883, physicists and mathematicians have been working on the understanding of turbulence. Unfortunately, turbulence is highly chaotic motion that tends to be unpredictable. Even the most minute changes in the setup of an experiment will cause dramatic differences in the results. But most often, these changes are inevitable in practice making it inherently difficult to study turbulence under reproducable conditions.

Already in the 19th century, Claude-Louis Navier and George Gabriel Stokes laid the theoretical groundwork to describe viscous fluid motion what would later become known as Navier-Stokes equations. They describe a dissipative system that proved invaluable for many fields of study, including but not limited to fluid dynamics, and allowed engineers to approximate the development of chaotic systems sufficiently enough for real-world applications. However, the existence of solutions to the Navier-Stokes equations for every system working in three dimensions is still unproven and deemed to be one of the most important problems in physics and mathematics up to date (see [2]).

To deal with the ongoing difficulties and thanks to advances in computer hardware in the last 30 years, computational fluid dynamics have emerged as

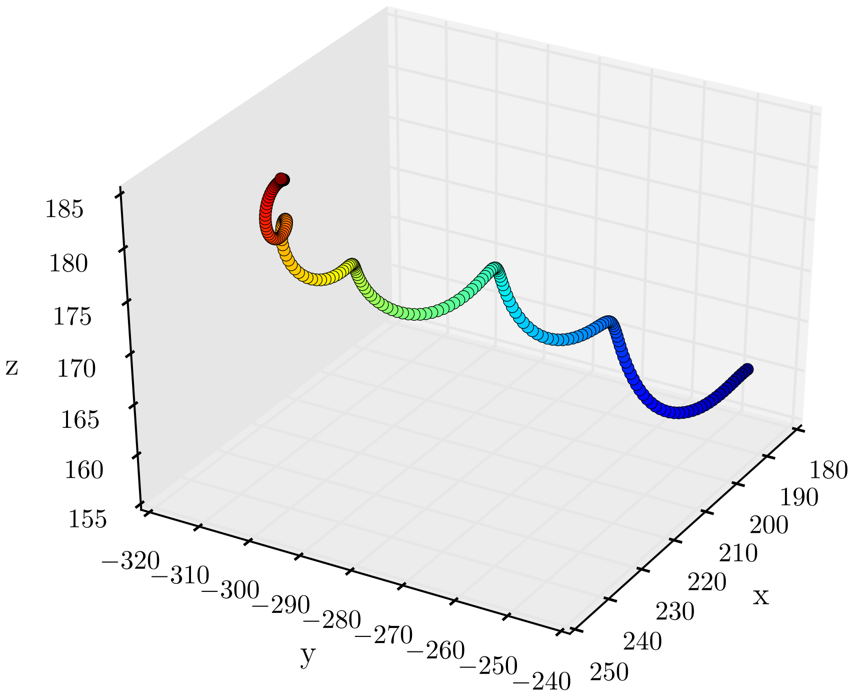


Figure 1: Lagrangian tracer trajectory taken from NVFOU512N3 describing a vortex; colored by time steps from blue to red.

an independent field of research simulating fluid motion using mathematical models in order to reveal new aspects of turbulence and allowing for studies in controlled environments. Today, computers are powerful enough to provide the means for numerical simulations in 3D with feasible runtimes allowing researchers to study the effects of turbulence in realistic scenarios. But such capability comes at the cost of vast amounts of data which make it difficult to extract information *by hand*. Especially, in order to gather statistically significant findings, automated feature extraction and classification is not only of great interest but sometimes inevitable due to lack of time or manpower.

1.2 Problem statement and objective

The research group Plasma Astrophysics at Technische Universität Berlin concerns itself with nonlinear dynamics of plasmaphysical processes. In this context, multiple 3D simulations have been carried out to study the behaviour of Lagrangian tracer trajectories in incompressible fluid turbulence with conducting and nonconducting materials. This thesis discusses the possibilities for automated trajectory classification using feature extraction based on discrete wavelet transform for the nonconducting-material simulation NVFOU512N3.

The simulation encompasses particle trajectories that mostly describe relatively linear movement but also exhibit chaotic and turbulent sections of sometimes notable structure. Exactly these minute structures are particularly interesting as they shed light on different aspects of turbulence. One of these structures is the *vortex* that describes a swirling motion around an axis of rotation (see Fig. 1). As common phenomenon of turbulent motion in nonconducting fluids, it gives information about the dissipation processes that take place during turbulence and thus describes a meaningful event in the course of a particle trajectory.

Given such a trajectory in 3D space, the objective of this thesis is to detect each vortex and classify it accordingly. Decisions will be based on a feature extraction approach using the discrete wavelet transform of the particle's lateral acceleration $\boldsymbol{A}_\perp$. Since a trajectory describes a motion in time, the objective can be formulated as a classification problem for multivariate time series. To provide further information after classification and possibly reveal additional structure or even generative processes of the data, the detected vortices are examined using unsupervised cluster analysis based on different kinds of similarity measures.

2 Incompressible fluid turbulence

2.1 What is turbulence?

Turbulence is a phenomenon observable in almost every aspect of nature. Albeit its omnipresence, no universal *theory of turbulence* has emerged since the first experiments have been undertaken by Reynolds in 1883. Rather a multitude of theories, each covering a different aspect of turbulence, is all that has followed more than ”a century of concerted effort involving engineers, physicists, and mathematicians” [3]. What seems worse is that there is no consensus regarding a formal definition of turbulence and that a definition might even be counterproductive because

> ”[t]urbulence is not a well-defined problem awaiting solution but is a state of motion with innumerable different facets which depend on the context in which it occurs.” [4]

Nonetheless, turbulence may loosely be described as flow with chaotic behaviour caused by sudden changes in pressure and flow velocity. More specific, Davidson [3] characterizes two distinct similarities that turbulences share with one another:

1. random fluctuations of a highly disordered velocity field with a wide range of length scales in time, and
2. unpredictability with respect to the evolution of the velocity field at slight changes of initial conditions.

In 1883, Reynolds first divided fluid motion into two dominant flow regimes, laminar and turbulent flow, and described the transition from one regime to the other [5]. While the former exhibits an obvious sense of stability, the latter is almost always inherently unstable as depicted in Fig. 2. At a specific point, the flow changes from laminar to turbulent and seems irreversably lost in chaotic motion. The moment of this transition depends on the velocity of the flow and the material’s viscosity as well as the flow configuration, e. g. the geometry of the exterior. In general, fluid motion becomes turbulent above certain speeds and at low viscosity. Consequently, laminar flow is dominated by viscous forces, while turbulence is mainly characterized by inertial forces.

A quantitive measure of this behaviour is given by the Reynolds number *Re*. It is defined as the ratio of inertial forces to viscous forces and gives insight into the state of a flow given a specific flow configuration. It also allows for comparison between different flows or to characterize different regimes in the

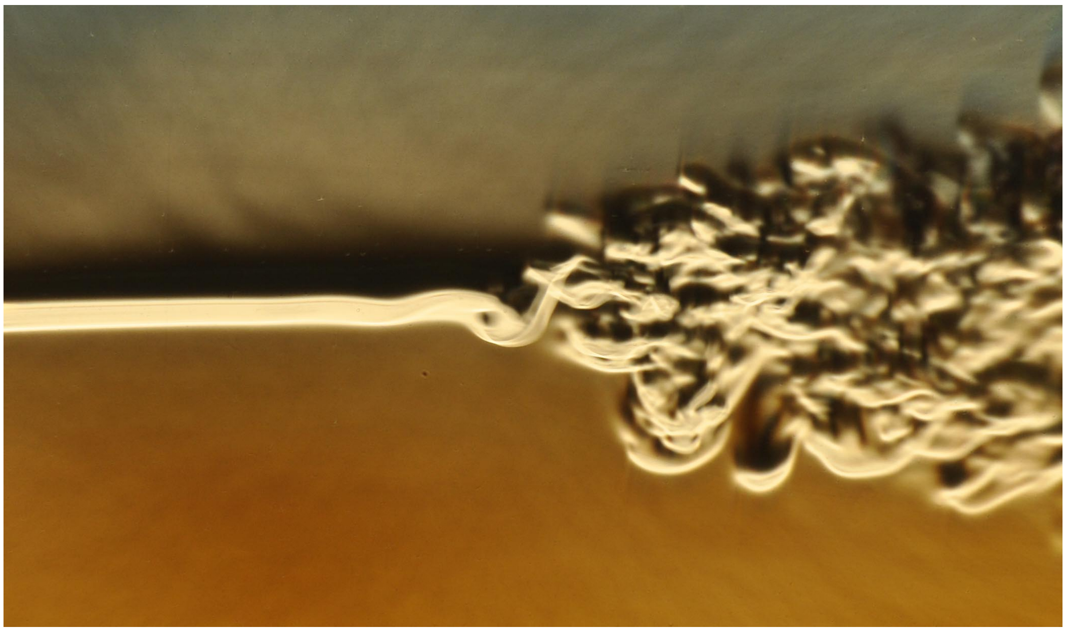


Figure 2: Transition of a candle plume from laminar (left) to turbulent (right) flow. Author: Gary Settles.

same fluid flow. Formally, it is defined by

$$Re = \frac{vl}{\nu}, \tag{1}$$

where v is the velocity of the flow, ν the fluid's kinematic viscosity, and l the configuration's characteristic length. Consequently, low Reynolds numbers correlate with low speeds and high viscosity, while high Reynolds numbers reflect flows of high speeds and low viscosity, respectively. Therefore, turbulence is closely associated with high Reynolds numbers. In this respect, Davidson notes that ”turbulence is the natural state of things” and that ”laminar flow is the exception and not the rule in nature and technology” because ”virtually all fluids have an extremely low viscosity” [3], which even applies to the molten metal in the earth's core.

2.2 Nonlinearity and the statistical nature of turbulence

The motion of viscous fluid is commonly described by the Navier-Stokes equations, which for incompressible fluids are given by

$$\partial_t \boldsymbol{v} + (\boldsymbol{v} \cdot \nabla)\boldsymbol{v} = -\frac{1}{\rho_0}\nabla p + \boldsymbol{g} + \nu \nabla^2 \boldsymbol{v}, \tag{2}$$

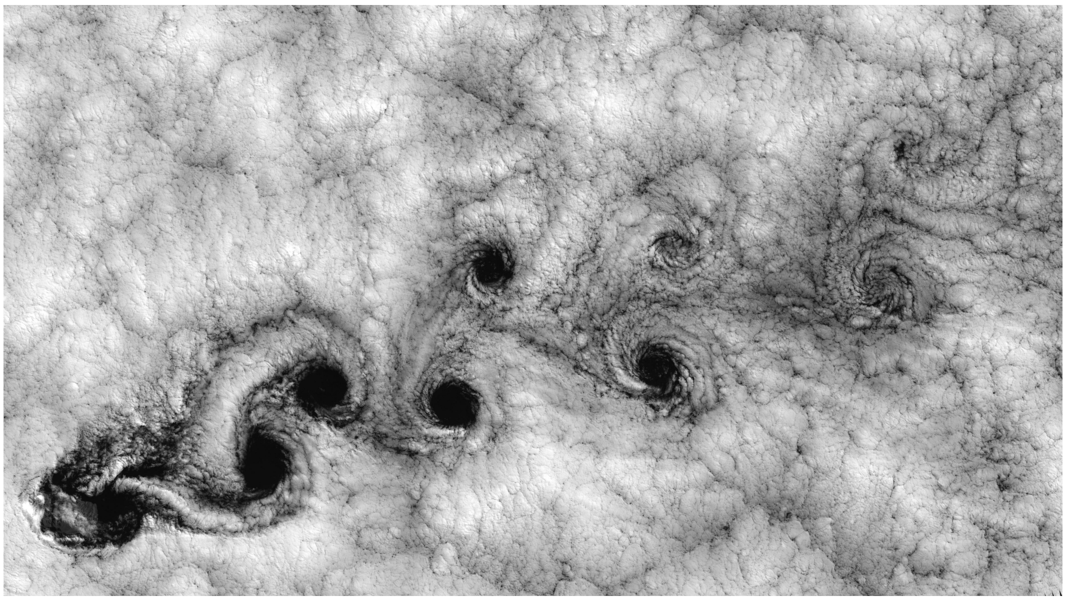


Figure 3: Wind flowing around an island causes turbulent motion with visible eddies of alternating rotation. Author: Bob Calahan.

where $\boldsymbol{v}$ is the flow velocity, ρ_0 the material's uniform density, p pressure, ν viscosity, and $\boldsymbol{g}$ forces acting on the material, like gravity. Although the Navier-Stokes equations can be applied to many physical phenomena of practical relevance, they still pose a mathematical problem due to their nonlinearity originating from the involvement of convective acceleration, i. e. change of velocity over position in time. Unfortunately for research, the nonlinearity term $(\boldsymbol{v} \cdot \nabla)\boldsymbol{v}$ is the biggest contributor to turbulence within the equations.

As for most nonlinear problems, dealing with it can be notoriously difficult. In case of turbulence, this finds expression in the unpredictable development of flow velocity given two slightly different starting conditions. However, if one was to *average* the results, all of a sudden turbulence becomes reproducable. Slight changes in initial conditions may lead to individual realizations that are unpredictably different, but they cause only slight differences in the time averages of those signals. This important observation has paved the way for the statistical analysis of a field that has otherwise eluded most theories up until that point and led to the impression that "any theory of turbulence has to be a statistical one." [3]

2.3 Energy cascade

Although turbulence mostly consists of chaotic motion, vortices and vortex-like structures, so-called *eddies*, are occasionally visible to the naked eye (see Fig.

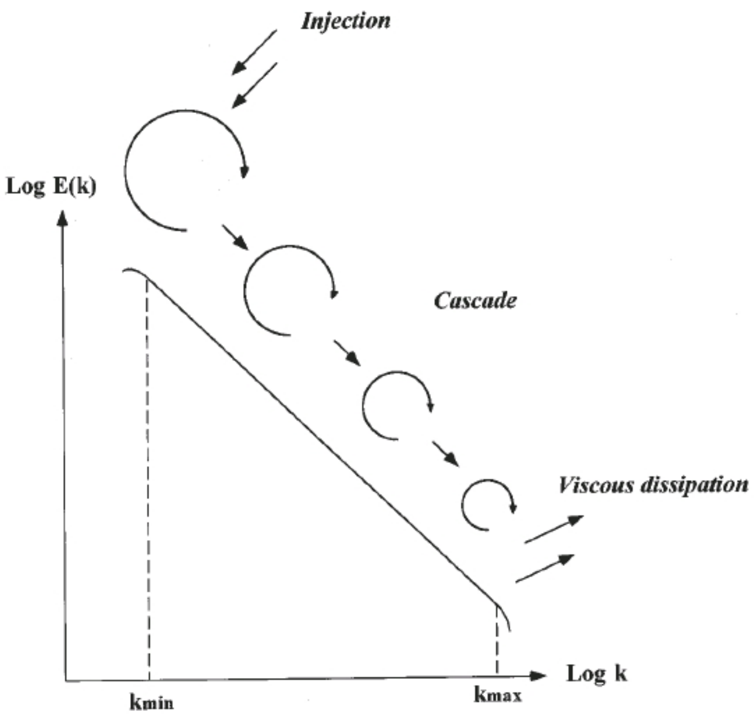


Figure 4: Kolmogorov's 5/3 law.

3). Considering a fluid streaming around a fixed object with high velocity, i.e. *Re* is sufficiently large, an instant moment might reveal little structure, but when time-averaged eddies of different scales form in the object's wake indicating a process first pointed out by Lewis Richardson in 1922 [6], the so-called energy cascade. During this process, eddies of larger scale *fall apart* into eddies of smaller scale, which after a short period of time will in turn break-up themselves into even smaller scaled eddies and so on until all energy has dissipated into thermal energy. At this point, *Re* has become so small that viscous forces begin to dominate over inertial forces and kinematic energy transforms into thermal energy [3].

In 1941, Andrey Kolmogorov first gave a successful quantitive description about the processes present during the energy cascade at Reynolds numbers high enough that for a specific range of scales was independent of parameters specific to flow configurations, like exterior or energy supply. According to Kolmogorov, eddies of the biggest scale L_E still are subject to those. But from a certain scale length $\ell \ll L_E$ down to scale lengths $\eta \ll \ell$, which cause dissipation, properties of turbulence are shown to be statistically isotropic and of universal nature depending only on the energy dissipation rate[1] [7]. This is known as

[1]L_E and η can be estimated using the energy spectrum E, the viscosity ν and the energy

Kolmogorov’s 5/3-law (see Fig. 4) and one of the few notable breakthroughs in turbulence theory that has been confirmed by several empircal studies ever since [7].

2.4 Turbulence simulations

Because of its sensitivity to initial conditions, chaotic behaviour, and unpredictable nature in general, fluid turbulence is especially difficult to handle inside of laboratory experiments. Therefore, computational fluid dynamics have established themselves early on as separate discipline contributing to turbulence research since the 1970’s. Constant progress in computer hardware and algorithms, most notably the Fast Fourier Transform (FFT) by Cooley and Tukey in 1965 [8], led to the possibility to simulate more complex and higher dimensional flow configurations, especially in areas where experimental setups are infeasible, like conducting gas plasmas of extremely high Reynolds numbers, which usually only occur in extraterrestrial environments.

Until today, numerical simulations have become an important tool for turbulence research. Fröhlich [9] characterizes three main approaches to turbulence simulation:

1. Reynolds-averaged Navier-Stokes equations (RANS),
2. large eddy simulations (LES), and
3. direct numerical simulations (DNS).

RANS simulations decompose an instantaneous flow quantity, e. g. velocity $\boldsymbol{v}$, into its time average $\overline{\boldsymbol{v}}$ and perturbation part $\boldsymbol{v}'$

$$\boldsymbol{v}(\boldsymbol{x},t) = \overline{\boldsymbol{v}}(\boldsymbol{x},t) + \boldsymbol{v}'(\boldsymbol{x},t) \tag{3}$$

such that only stationary equations need to be solved. Thus, the mean flow velocity is a direct result which makes averaging of individual realizations over a period of time unnecessary. However, this also means that turbulent fluctuations are not computed directly but rather approximated using turbulence models, which is an inherent weakness since in most cases these models need manual refinement and therefore do not generalize well. One way how LES overcome this weakness is by computing the biggest scales directly and to model only the smaller ones. Understandably, this comes at the cost of increased computational demands and can be regarded as a trade-off between accuracy and performance since for a variety of problems it proves sufficient to only compute larger scales.

dissipation rate [7].

However, since every scale length affects the flow, it is of natural interest to look at the contributions of all scales. DNS simulate vector fields fine enough to directly compute every scale length down to the smallest possible vortices before energy dissipates due to viscosity. Naturally, this method produces more information than others. But it is also the most expensive method which is why it has long been deemed infeasible for complex simulations. Especially in case of high Reynolds numbers, scale lengths get increasingly small and therefore require grids of high resolution [9].

2.5 Simulation NVFOU512N3

Today, modern computer hardware is powerful enough to provide the means for DNS in 3D with feasible runtimes, one of which being the direct numerical simulation NVFOU512N3. It encompasses trajectories of Lagrangian particle tracers whose movements have been simulated using Eulerian vector fields which represent numerical solutions to the incompressible Navier-Stokes equations (see Fig. 5). In total, the movements of 500’000 particle tracers have been simulated for 3125 iterations in a volume with a side length of 512 units. For each iteration (or time step) t, the trajectories’ relevant information includes position in cartesian coordinates $\boldsymbol{X}(t)$, velocity $\boldsymbol{V}(t)$, and acceleration $\boldsymbol{A}(t)$. Due to memory issues, the vector fields have not been stored after computation and are therefore not available for analysis.

2.5.1 Solutions to the Navier-Stokes equations: pseudo-spectral methods

Incompressibility has an interesting implication with respect to the implementation of the Navier-Stokes equations. An incompressible flow is given when the material density is constant within an infinitesimal volume of the flow and is formally described by

$$\boldsymbol{\nabla} \cdot \boldsymbol{v} = 0. \tag{4}$$

Thus, the pressure component within the Navier-Stokes equations becomes a passive variable and solely determinable using Poisson’s equation. Therefore, it does not need to be solved numerically. As a consequence, the vorticity equation becomes an alternative representation of the incompressible Navier-Stokes equations since it fully determines the velocity when the pressure component is already known:

$$\partial_t \boldsymbol{\omega} = \boldsymbol{\nabla} \times (\boldsymbol{v} \times \boldsymbol{\nabla}) + \nu \Delta \boldsymbol{\omega}, \tag{5}$$

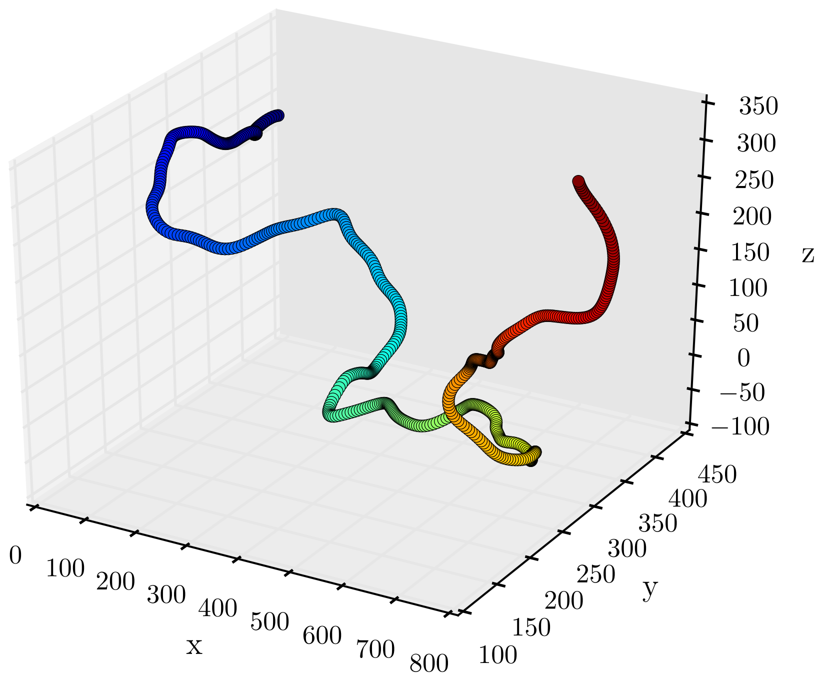


Figure 5: Tracer trajectory taken from NVFOU512N3; colored by time steps from blue to red.

where the vorticity $\boldsymbol{\omega}$ is given by

$$\boldsymbol{\omega} = \nabla \times \boldsymbol{v}. \tag{6}$$

By exploiting Fourier space, one can directly solve the velocity's Fourier transform $\hat{\boldsymbol{v}}$ using the Fourier transform of the vorticity $\hat{\boldsymbol{\omega}}$. That is, given the velocity as a finite Fourier series

$$\boldsymbol{v}(\boldsymbol{x}, t) = \sum_{\boldsymbol{k}} e^{i\boldsymbol{k}\boldsymbol{x}} \hat{\boldsymbol{v}}(\boldsymbol{k}, t), \tag{7}$$

with wave vector $\boldsymbol{k}$, its Fourier transform is given by

$$\hat{\boldsymbol{v}}(\boldsymbol{k}, t) = i \frac{\boldsymbol{k} \times \hat{\boldsymbol{\omega}}(\boldsymbol{k}, t)}{|\boldsymbol{k}|^2} \tag{8}$$

with

$$\partial_t \hat{\boldsymbol{\omega}} = i\boldsymbol{k} \times \left(\widetilde{\boldsymbol{v} \times \boldsymbol{\omega}}\right) - \nu \boldsymbol{k}^2 \hat{\boldsymbol{\omega}}, \tag{9}$$

which yields a solution with high accuracy since gradients reduce to simple multiplications with the wave vectors. Eventually, a second-order leapfrog integration is used to integrate over time [7].

2.5.2 From vector fields to tracer trajectories

After obtaining the velocity field as equally spaced grid for each time step t, a tracer's trajectory is computed by interpolation since its position will most likely not align with any grid points. Trigonometric interpolation methods are usually the most accurate for the given case, but they lack computational efficiency which is the reason they have not been considered for NVFOU512N3 [7]. Instead, tricubic interpolation has been chosen as a reasonable compromise between accuracy and performance giving the following interpolation scheme for cartesian coordinates

$$\begin{aligned} \boldsymbol{X}'(t+\Delta t) &= \boldsymbol{X}(t) + \Delta t \boldsymbol{v}(\boldsymbol{X}, t) \\ \boldsymbol{X}(t+2\Delta t) &= \boldsymbol{X}(t) + \Delta t \left(\boldsymbol{v}(\boldsymbol{X}, t) + \boldsymbol{v}(\boldsymbol{X}', t+\Delta t)\right) \end{aligned} \tag{10}$$

and acceleration

$$\boldsymbol{A}(t) = \frac{d}{dt}\boldsymbol{V}(t) \approx \frac{1}{4\Delta t}(\boldsymbol{V}(t+2\Delta t) - \boldsymbol{V}(t-2\Delta t)). \tag{11}$$

3 Vortex detection in 3D fluid turbulence

3.1 Related work

Despite new potential discoveries, the blessing of increased computational power also has a downside: big data. The amount of data created over weeks of computation is far too vast for being processed manually. Therefore, automatic feature extraction is a necessity, if time and manpower are scarce. A particular hindrance in case of vortices is the lack of a generally accepted definition as traditional vortex definitions have shown to be ambiguous [10, 11]. Despite this seemingly difficult discussion, several methods have been proposed with respect to 3D vortex detection in the past, all of which are based on velocity or vorticity fields (see for example [11–14]). Unfortunately, these have not been stored as part of NVFOU512N3, which renders the aforementioned methods useless. In fact, there seems to be no method in the literature solely based on Lagrangian tracer trajectories that works entirely without Eulerian quantities.

Obviously, in cases like NVFOU512N3 there is need for vortex detection methods that work exclusively on Lagrangian tracer trajectories. However, the absence of a general vortex definition still retains making a clear understanding of the requirements difficult. Although George Haller did propose a Galilean-invariant and "objective definition of a vortex" [15] in 2005, it is based on Eulerian quantities and thus does not help much. In context of this thesis and with respect to Lagrangian tracer trajectories, a vortex will therefore be defined as a distinct spiraling motion exhibiting strong similarity with a helix, not necessarily straight in motion, but potentially bent and twisted, following the notions of Hans Lugt and Stephen Robinson:

> "A vortex is the rotating motion of a multidude of material particles around a common center." [16]
>
> "A vortex exists when instantaneous streamlines mapped onto a plane normal to the vortex core exhibit a roughly circular or spiral pattern, when viewed from a reference frame moving with the center of the vortex core." [17]

An important thing to note is that this expertise refers to trajectory positions $\boldsymbol{X}(t)$.

Naturally, an instant impulse is to follow the geometric approach pointed out by Robinson, i. e. trying to confirm a helical motion by projecting back the direction vector onto the normal plane of the rotation axis (see also [18]). However, these methods usually rely on a bent helix model that is far too smooth and simplistic to cope with actually shaped vortices as depicted in Fig.

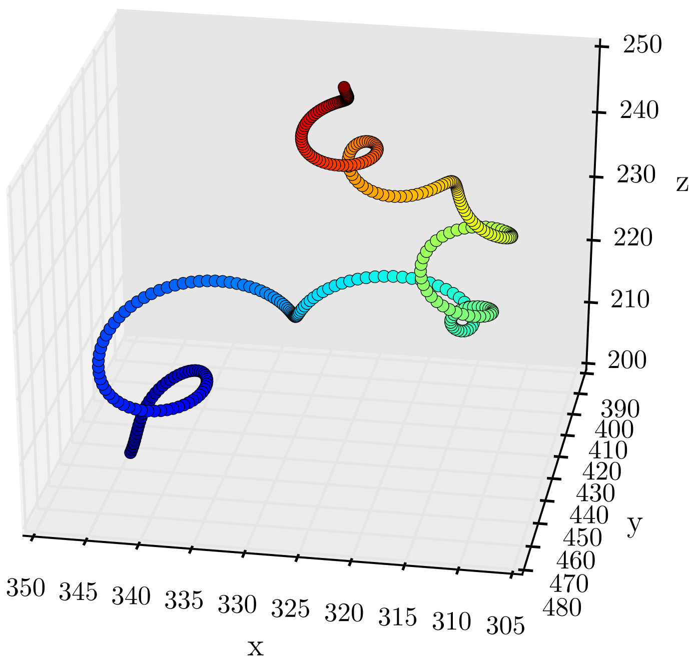


Figure 6: Vortex trajectory with a heavily bent axis of rotation; colored by time steps from blue to red.

6 for which it can be notoriously difficult to reliably fit the rotation axis.

Instead of focusing on motions in cartesian coordinates, this thesis will propose an alternative approach based on the discrete wavelet transform of a trajectory's lateral acceleration $\boldsymbol{A}_\perp$.

3.2 Towards a DWT-based approach

3.2.1 Fourier transform

In 1822, Jean-Baptiste Joseph Fourier introduced the idea of decomposing periodic functions into a linear combination of sine and cosine waves, called Fourier series [19]. Based on the idea by Fourier, every signal in time $f(x)$ can thus be described by an infinite sum of sines and cosines of varying frequencies (see Fig. 7), called Fourier transform (FT) $F(k)$:

$$F(k) = \int_{-\infty}^{\infty} f(x)\ e^{-i2\pi kx}\ dx, k \in \mathbb{R}, \tag{12}$$

where k corresponds to frequency and $i \equiv \sqrt{-1}$. Using Euler's formula

$$e^{ix} = \cos(x) + i\sin(x), \tag{13}$$

one can see that the complex exponential corresponds to a complex number with sinusoids as both real and imaginary parts. Since complex exponentials are orthogonal, the Fourier transform can represent any piecewise continuous function while minimizing the least-squares error between signal and transform [20]. To phrase it differently, the Fourier transform exploits the complex number's magnitude and angle in order to capture both the signal's amplitude A and phase ϕ, respectively. This way, a signal can be transformed from its representation as time series into the frequency domain giving information about individual frequency components present in the signal, as well as their overall contribution.

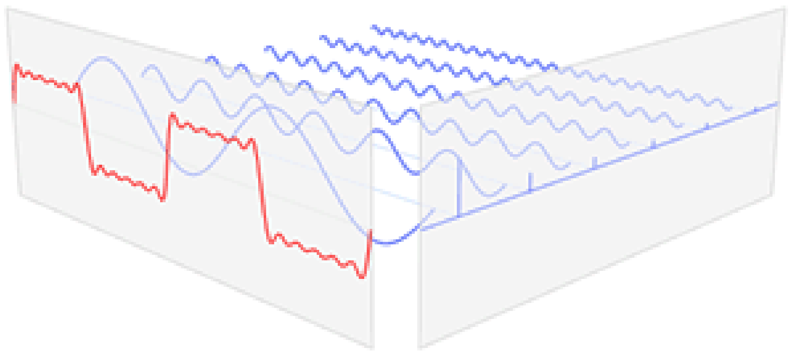


Figure 7: Illustration of splitting a periodic function into its frequency components using the Fourier transform. The original signal in time domain (red) can thus be represented by the sum of sine waves of different frequencies (blue) weighted by their amplitudes, depicted as blue bars on the right side describing each frequency's amplitude. Author: Lucas V. Barbosa.

Lastly, the inverse of the Fourier transform is given by

$$f(x) = \int_{-\infty}^{\infty} F(k)\ e^{i2\pi kx}\ dk,\ \ x \in \mathbb{R}. \tag{14}$$

3.2.2 Discrete Fourier transform

The FT "converts a time-domain signal of inifinite duration into a continuous spectrum composed of an infinite number of sinusoids" [21]. In computational sciences, such spectra cannot be computed since every computer works with limited precision and therefore discrete values. For these cases, the discrete Fourier transform (DFT) computes a frequency representation of a discrete signal $x = \{x_1, x_2, \ldots, x_n\}$ using a finite number of sinusoids. It is given by

$$X_k = \sum_{j=1}^{n} x_j \cdot e^{-2\pi ikj/n},\ k \in \mathbb{Z} \tag{15}$$

with its inverse

$$x_j = \frac{1}{n}\sum_{j=1}^{n} X_k \cdot e^{2\pi ikj/n},\ j \in \mathbb{Z}. \tag{16}$$

It returns a frequency spectrum of length n with frequencies k ranging from $-(n/2-1)$ to $n/2$. In case of real-valued input, this reduces to $n/2$ since the result becomes hermitian and negative frequencies are therefore of redundant information.

Since the formulation of the FFT in the 1960s, DFTs have become a popular tool in engineering due to the reduced complexity of $\mathcal{O}(N\ \log_2(N))$. In particular, using FFT and its inverse it is possible to compute the convolution of two functions $f(x)$ and $g(x)$ in $\mathcal{O}(N\ \log_2(N))$ since it equals piecewise multiplication of their Fourier transforms:

$$g(x) * f(x) = G(k)F(k). \tag{17}$$

3.2.3 Wavelet transform

Using DFT, a signal can be decomposed into its frequency components giving information about the amount each frequency contributes to the overall signal. While this might be sufficient for certain applications, it may not be enough in cases where it is also of interest when or where certain frequencies are present. Unfortunately, Fourier transform falls short in this respect giving rise to the question how to create a representation that covers both time and frequency information.

Short-time Fourier transform offers a solution by taking the Fourier transform of small, fixed-sized, consecutive windows [22]. Thus, each window gives information about the frequencies contained in it and their moment in time. It feels tempting to just set the window size to 1, thus giving perfect time information. However, this effectively equals a Dirac pulse δ, also called Delta function, for which the Fourier transform is known to be

$$F(\delta) = e^{-2\pi ikt},\ k \in \mathbb{R}, \tag{18}$$

scaled by the signal value at time t. Obviously, this doesn't provide any useful information since the DC component ($k = 0$) will always be the dominant part. The question then is how to set an appropriate window size?

Unfortunately, following Heisenberg's uncertainty principle there cannot be a satisfactory answer since time and frequency cannot be sharply localized simultaneously. Widening the window increases frequency resolution at the cost of decreased time resolution. Tightening the window increases time resolution at the cost of decreased frequency resolution. The way wavelet transforms can help to mitigate this dilemma is by shifting windows of varying size repeatedly along the signal creating a multiresolution analysis of the signal such that for varying window scalings the result is a time-frequency representation of different resolution.

3.2.4 Continuous wavelet transform

Given a signal $f(t)$ in time domain, a scale factor $s \in \mathbb{R}$, and a translation $\tau \in \mathbb{R}$, the continuous wavelet transform F_w (CWT) is written as

$$F_w(s,\tau) = \int_{-\infty}^{\infty} f(t)\ \bar{\psi}_{s,\tau}(t)\ dt. \tag{19}$$

Here, s correlates to the notion of window size, where a low scale factor relates to high frequency and vice versa, and τ correlates to the shifting size. Similar to the Fourier transform, the signal is decomposed into a set of basis functions $\bar{\psi}_{s,\tau}(t)$, called wavelets, where $\bar{\psi}$ denotes the complex conjugation. Wavelets themselves are composed from a so-called mother wavelet $\psi(t)$:

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\ \psi\left(\frac{t-\tau}{s}\right). \tag{20}$$

The name *wavelet* originates from the wavelet's main porperties: admissibility and regularity [23]. Admissibility requires the zero-frequency or DC component of the wavelet's Fourier transform to be 0, i. e. it acts like some sort of band-pass filter. From this it follows that the wavelet integrates to zero and must consequently be a wave since it has to oscillate due to its zero DC component [23]. The notion of regularity is best described by the so-called vanishing moments of the wavelet. The p-th moment m_p of a wavelet, $p \in \mathbb{Z}$, is defined by

$$m_p = \int_{-\infty}^{\infty} t^p \psi(t)\ dt. \tag{21}$$

If $m_p = 0$, it is a vanishing moment. The number of vanishing moments can be used to characterize the wavelet's ability to accurately represent the signal. Wavelets with more vanishing moments can represent signals with higher complexity more accurately than wavelets with less vanishing moments. As a downside, this comes at the cost of a longer *support*, i. e. more scale and thus less time resolution. Also, a wavelet with vanishing moments of order p cannot capture any information regarding polynomials of order up to $p-1$. Ultimately, wavelets with more vanishing moments have higher regularity than wavelets with less vanishing moments.

3.2.5 Scaling function ϕ

It is known from Fourier analysis, that reducing the window size by a factor of c results in a frequency spectrum *stretched* and *shifted* by that same factor. Much like time resolution is guaranteed by shifting along a window over a signal of possibly infinite length (*translated* wavelets), the frequency spectrum can be covered by *dilating* wavelets. However, doubling the window size effectively cuts the spectrum's bandwidth in half, which is why every dilation by a factor of 2 covers only half of the remaining spectrum. Following Zeno's famous paradox of Achilles and the tortoise [24], this will lead to an inifinite number of scalings.

To solve this problem, the concept of the wavelet's scaling function ϕ has been introduced by Mallat in 1989 [25], which can be formulated as

$$\phi(t) = \sum_{j,k} F_w(j,k)\ \psi_{j,k}(t). \tag{22}$$

For some scale j, it essentially serves as a *final* low-pass filter eventually setting an end to the potentially infinite scaling process.

3.2.6 Discrete wavelet transform

Until now, wavelets more or less describe a conceptual framework that decomposes a signal into its time-frequency representation by the means of continuous functions. Apart from this, wavelets take no specific form other than what is required by their properties. In fact, there exist an infinite number of potential wavelets and the choice of wavelet depends heavily on the application. But so far, nothing has been said about how to actually implement them.

Since the 1970s and 1980s, there has been increased interest in wavelet analysis due to the work of Jean Morlet, Ingrid Daubechies, and others, leading to the first formalization of the CWT based on modifications of the Gabor transform by Morlet in 1984. In 1988, Ingrid Daubechies laid the groundwork for wavelets with finite processing time (*Daubechies waveletes*), making wavelets practical for

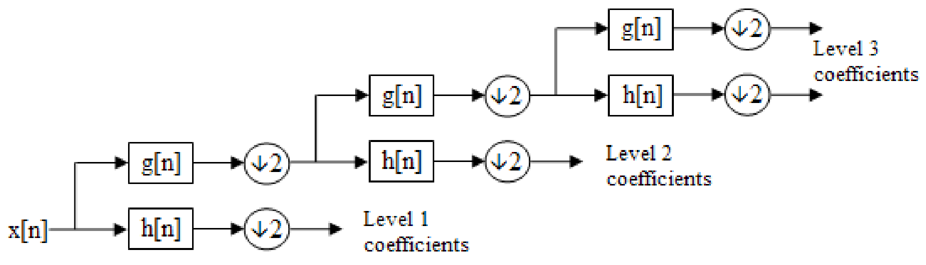


Figure 8: Illustration of the filter bank used as implementation basis for the discrete wavelet transform. Author: Johnteslade.

real-world applications [26]. However, there are generally no analytical solutions to wavelets. Thus, the question becomes how to solve them numerically.

Recalling the scaling function ϕ, the wavelet transform can be regarded as a filter bank with the wavelet as band-pass and ϕ as low-pass filter. Then, both filters produce a set of *approximation* and *detail* coefficients that for each scale represent the signal in the time-frequency domain (see Fig. 8). This is known as the fast wavelet transform algorithm, building on insights from multiresolution analysis: Given the approximation coefficients $s^{(j)}$ for scale j, $s^{(j-1)}$ is obtained recursively by convolving $s^{(j)}$ with a low-pass filter g:

$$s^{(j-1)} = \frac{1}{\sqrt{2}}(\downarrow 2)(g * s^{(j)}), \tag{23}$$

where $(\downarrow 2)$ denotes the downsampling operation. $s^{(j)}$ itself is given by taking the scalar product of the signal $f(t)$ and the wavelet's scaling function:

$$s_n^{(j)} = 2^{(j)}\langle f(t), \phi(2^{(j)}t - n)\rangle. \tag{24}$$

Similar, the detail coefficients of scale $j-1$ are given by convolving $s^{(j)}$ with a high-pass filter b instead of a low-pass filter:

$$d^{(j-1)} = \frac{1}{\sqrt{2}}(\downarrow 2)(b * s^{(j)}). \tag{25}$$

Both sets of coefficients have different semantics. Approximation coefficients are useful in order to reconstruct the signal. In fact, given the set of detail coefficients, only the approximation coefficients of the last scale are necessary to reconstruct the signal. This makes DWT especially interesting for data compression since almost all approximation coefficients can be discarded. Consequently, most of the information is described by the detail coefficients, which are therefore also called energy coefficients. However, a difficulty remains: since ϕ still involves infinite integrals, $s_n^{(j)}$ cannot be computed directly using 24.

Algorithm 1 Implementation of DWT for the Haar wavelet

```
1: procedure HAAR($f_n$)
2:     $s \leftarrow f_n$
3:     $d \leftarrow \varnothing$
4:     while $|s| > 1$ do
5:         $d \cup \{\frac{1}{\sqrt{2}}(s_i - s_{i+1}) \mid i = 0, 2, \ldots, |s| - 1\}$
6:         $s \leftarrow \{\frac{1}{\sqrt{2}}(s_i + s_{i+1}) \mid i = 0, 2, \ldots, |s| - 1\}$
7:     return $\{s, d\}$
```

To solve this problem intuitively, the signal $f(t)$ itself can be interpreted as the approximation coefficients for the highest scale. This simple trick also determines the number of total scalings since the algorithm will come to an automatic halt when the input size is reduced to one. On the other hand, the input size is usually required to be a power of two. This way, DWT can be implemented using a filter bank with a low- and high-pass filter, recursively splitting the signal into two sets of coefficients at each scale (see Fig. 8).

Interestingly enough, in this scenario, explicit specification of both wavelet and scaling function becomes superfluous, as they are replaced by filters. The oldest and simplest wavelet transform is probably the Haar transform (see Alg. 1), whose wavelet has been introduced by Alfréd Haar in 1909 [27], long before the term *wavelet* has been coined. As compactly supported wavelet with one vanishing moment, it represents a special case of the Daubechies wavelets.

3.3 Vortex detection using discrete wavelet transform

3.3.1 Preliminary considerations

As mentioned earlier, conventional vortex detection methods found in the literature work on Eulerian vector fields and are therefore able to determine the entire area of a vortex. In case of passive drift motions of Lagrangian particle tracers, a trajectory can only serve as a vortex's scattered footprint and thus seldomly covers the entire field of influence. Rather, numerous kinds of encounters are possible, ranging from an ideal traversal of the entire vortex to short visits of a few revolutions. In the worst case, the particle hits the vortex head-on, leaving only a short burst in curvature, before leaving the vortex again. Since there is no chance to reliably distinguish such minimal contact from random chaotic changes in direction, the notion of a minimal vortex length needs to be considered.

In context of this thesis, a particle tracer's trajectory is then said to exhibit a

(a) Corkscrew vortex in water. Author: Robert D. Anderson.

(b) Wingtip vortex forming in the wake of an aircraft. Author: NASA.

vortex, if it shows clear signs of consistent oscillatory motion around a straight, bent or twisted, but otherwise imaginary axis of rotation. Looking at the Haar transform (Alg. 1), strong detail coefficients, or *energies*, are caused by strong differences in the signal. Due to the signal's prolonged oscillation in case of a vortex, DWT should produce significantly higher energies for the duration of the vortex than usual, which is the key idea of the proposed approach.

3.3.2 Feature selection

Due to the difficulties regarding spatial information in turbulent environment (see section 3.1), alternative features and types of information need to be considered. E. g., the Frenet-Serret formulas are particularly well suited to represent helical motion. Given a curve in cartesian coordinates $\boldsymbol{r}(t)$, its curvature $\mathcal{K}$ and torsion τ are given by

$$\mathcal{K} = \frac{||\boldsymbol{r}'(t) \times \boldsymbol{r}''(t)||}{||\boldsymbol{r}'(t)^3||} \tag{26}$$

$$\tau = \frac{\boldsymbol{r}'(t) \cdot (\boldsymbol{r}''(t) \times \boldsymbol{r}'''(t))}{||\boldsymbol{r}'(t) \times \boldsymbol{r}''(t)||^2}, \tag{27}$$

providing a natural parameterization for helical motion in particle trajectories. Unfortunately, computation of torsion requires taking the third derivative of $\boldsymbol{r}(t)$ in 27, which in case of NVFOU512N3 could have not been done with satisfactory precision. Torsion is therefore not available. Curvature, however, should theoretically form an even shaped plateau in case of a vortex and might thus give good insights with respect to vortex detection. But, as can be seen

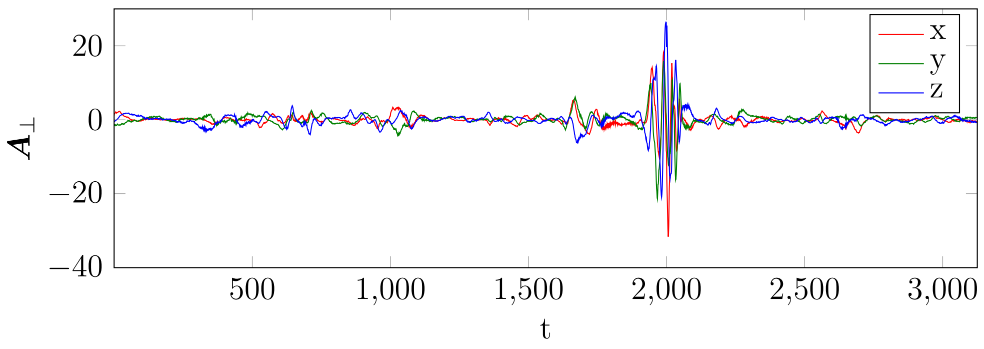


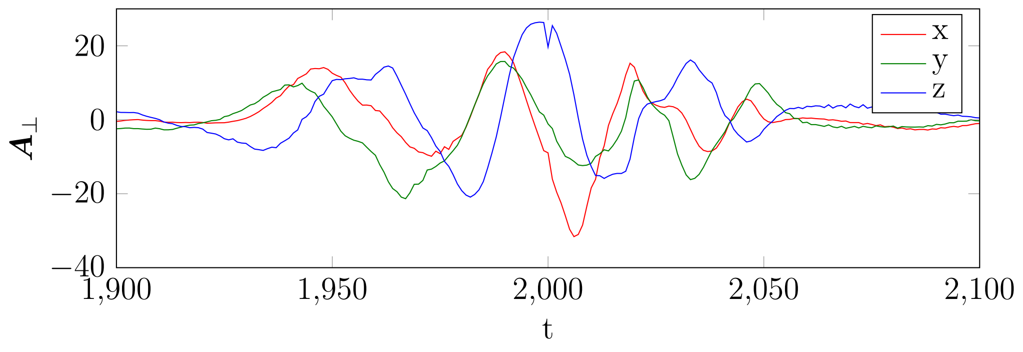
Figure 9: $\boldsymbol{A}_\perp$ of a particle trajectory with a vortex pattern showing around timestep 2000.

in Fig. 6, vortices in turbulent flow are seldomly smooth but induce rather rippled signatures, leading to relatively peaked curvature profiles, which reveal only little detail about the underlying phenomena.

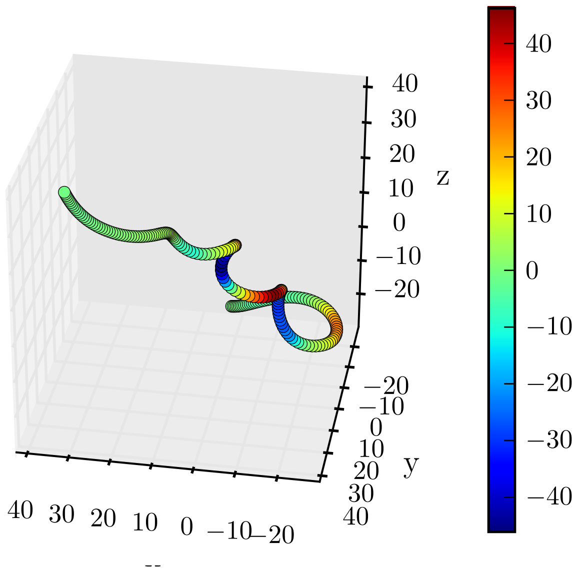
Since acceleration $\boldsymbol{A}(t)$ is inseparably connected to velocity $\boldsymbol{V}(t)$, it also needs to reflect the effects induced by vortices. In fact, considering the acceleration with respect to rotations around a common axis, it is to be expected that oscillating patterns become observable in the *lateral* components of the acceleration since centripetal forces need to act in vertical direction to the direction of motion in order for particle tracers to rotate around an axis. Furthermore, a vortex should involve all three cartesian axes – with the rare exception that an axis coincides with the axis of rotation – and therefore cause clear and structured patterns of oscillation while leaving only small room for random deviations. In other words, the lateral acceleration should reflect the particle's state of captivity within the vortex's field of influence.

Due to the consistent oscillations expected in the lateral acceleration, DWT should produce detail coefficients of high energies. Thus, lateral acceleration represents a feature in which the formulation of expected behaviour with respect to vortices seems comprehensible and whose wavelet transform may only show dense regions of high energy when the particle is affected by a vortex for which the assumption that a vortex inevitably involves all axes of motion is key. Therefore, it will be considered a suitable feature for vortex detection using DWT.

Given $\boldsymbol{A}(t)$ and the normalized velocity vector $\boldsymbol{V}_n(t)$ of a particle trajectory, the parallel and lateral components of the acceleration, $\boldsymbol{A}_\parallel$ and $\boldsymbol{A}_\perp$ respectively,



(a) $\boldsymbol{A}_\perp$ for vortex signature in Fig. 9.



(b) Trajectory of vortex signature in Fig. 9, colored by $\boldsymbol{A}_\perp$ accumulated along the axes.

are calculated by

$$\boldsymbol{A}_\parallel(t) = \langle \boldsymbol{A}(t), \boldsymbol{V}_n(t) \rangle \cdot \boldsymbol{V}_n(t) \tag{28}$$

$$\boldsymbol{A}_\perp(t) = \boldsymbol{A}(t) - \boldsymbol{A}_\parallel(t). \tag{29}$$

Fig. 9 shows $\boldsymbol{A}_\perp$ of a particle with a vortex signature visible at around $t = 2000$. Zooming into this region (see Fig. 10a) exhibits increased motion at roughly the same frequency and period for all axes. Fig. 10b shows the vortex in cartesian space.

Remarks It needs to be noted that the proposed approach solely relies on the assumption that vortices exert significant forces on the lateral acceleration of by-passing particle tracers. There is no affirmative geometric or otherwise appropriate conclusion that could confirm the suggested outcome of the algorithm. Also, vortices that fall short of exerting enough energy will consequently be hidden to the approach. While this in turn does not necessarily mean that there have not been other particles with more obvious signatures caused by the same vortex, it nevertheless describes a lower bound on the sensitivity with regard to the formation of smaller vortices.

3.3.3 Detection algorithm

The outline for the detection algorithm is given by the following steps. For each tracer trajectory,

1. compute $\boldsymbol{A}_\perp$,
2. transform $\boldsymbol{A}_\perp$ using DWT and
3. extract vortices from resulting detail coefficients.

Choice of wavelet As mentioned before, DWT is a general framework with the ability to utilize different wavelets. Commonly, wavelets are grouped in so-called families, such as Daubechies, Biorthogonal, or Symlets, describing different properties for each family. Depending on the application, certain wavelets are more appropriate than others. If the wavelet transform needs to be orthogonal, one may choose orthogonal wavelets. Biorthogonal wavelets have proven useful in case of signal reconstruction and are primarily used in compression techniques, such as JPEG-2000 [28]. Furthermore, symmetric wavelets exhibit benificial characteristics with respect to the analysis of finite signals, if for example a filter needs to have linear phase.

In context of this thesis, good reconstruction capability or orthogonality are not necessarily properties the wavelet needs to have. Rather, good time localization is desired. Therefore, the wavelet should have as few vanishing moments as possible. Thanks to its one vanishing moment, the already introduced Haar wavelet has superior time resolution but is also conceptually simplistic and has advantageous performance characteristics (fast, memory efficient). It will therefore be the wavelet of choice.

Extraction strategy Following the wavelet transform, the algorithm has to decide which part of the transformed signal might represent a vortex signature. In terms of DWT, a vortex has been described as a region in time of *significant* energy and length along at least two axes. Note that in the given setting no

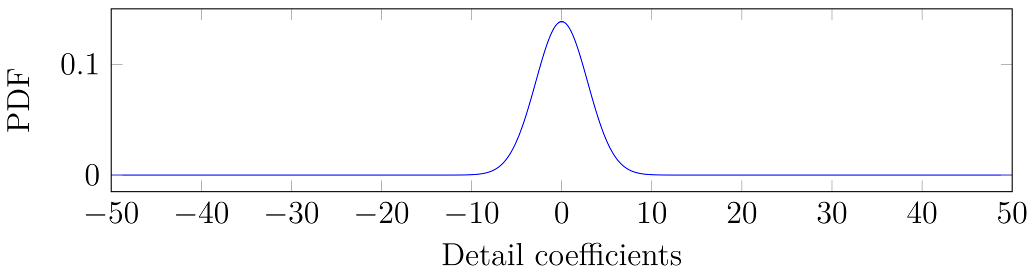


Figure 10: Distribution of detail coefficients obtained from the Haar transform of $\boldsymbol{A}_\perp$ for all particles in NVFOU512N3.

prior information about the distributions of the variables of interest exist, i. e. there is no knowledge about what constitutes a significant length or energy. Furthermore, manual determination of parameters based on data subsets can only be reasonable, if they are guaranteed to meaningfully represent the entire data. Such assumptions cannot be made in this case, because there simply exists no experience in this regard. To keep the semantics of *significant* as objective as possible, it should therefore be defined on the basis of statistical considerations.

Looking at the Haar transform (Alg. 1), one can see that high energies result from sudden changes in the signal. Due to the spiraling motion within a vortex, it is to be expected that $\boldsymbol{A}_\perp$ oscillates stronger and with increased frequency than usual, ultimately creating a dense region of high energy coefficients for the duration of the vortex. In this sense, *high* can be considered a synonym for *significant* and should express a value some degrees above average. Assuming a Gaussian distribution of energies – seemingly confirmed by Fig. 10 –, the standard deviation σ scaled by a factor s_σ is a commonly used statistic that in the absence of more fitting criteria will serve as a threshold parameter, thus determining *high* energies.

Since the detail coefficients are computed for each axis separately, the question becomes how to *merge*, i. e. combine, the energies of all axes to allow for a decision based on a unified set of energies. Again statistically, the element-wise sum, average, or maximum could be taken, but since the assumption has been made that at least two axes need to be involved, the most natural way to look at it would be to take the element-wise median, which in addition avoids side-effects associated with arithmetic operations, e.g. outlier sensitivity, unwanted smoothing, or similar. It also guarantees that the second-least expressive axis (seen element-wise) becomes the decision criterion since the least ”energetic” axis can be discarded due to the possibility of it being the direction of motion.

The notion of *significant* with regard to length depends heavily on the minimum

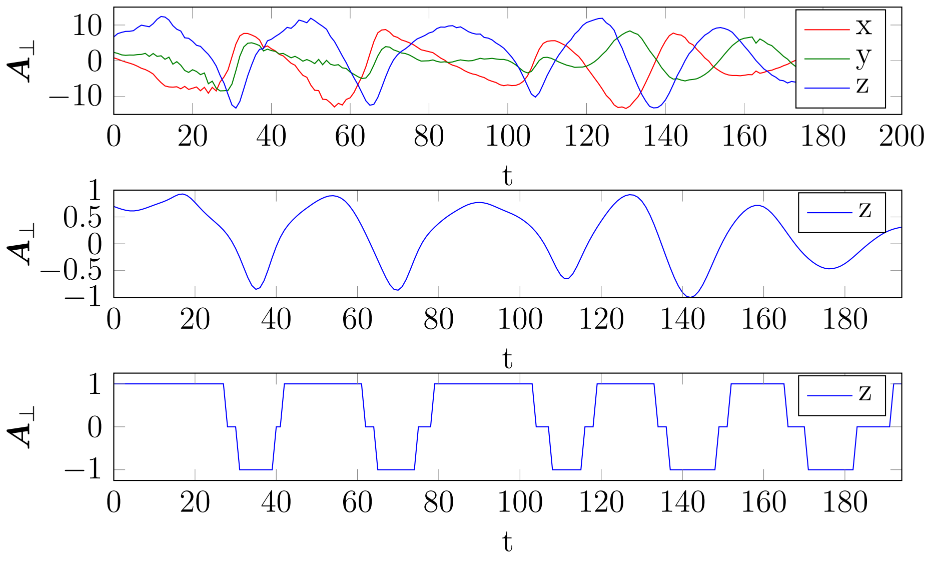


Figure 11: Determination of the number of revolutions. In the given example, a total number of 5 revolutions is proposed.

length of an observable vortex signature and therefore can not be decided satisfyingly by statistics, because there are no statistics to work with, i. e. there is no information on the lengths of smallest vortex signatures.[2] In light of the multitude of potential ways a particle tracer may interact with a vortex, a minimum number of visible revolutions seems to be an intuitive measure that will therefore be considered for vortex detection. Since a single revolution might be a little too vague, a minimum number of two revolutions will be proposed as a requirement.

To determine the number of revolutions of a potential vortex candidate, the notion of zero crossings with regard to $\boldsymbol{A}_\perp$ will be considered. For this, the signal's axis with highest variance is picked, smoothed using convolution with an averaging filter of length 10, normalized into range $[-1, 1]$, and eventually discretized into values $[-1, 0, 1]$ according to a threshold value of 0.25 (see Fig. 11). The number of revolutions is then set to the number of zero crossings divided by 2.

Finally, Fig. 12 depicts the detection process. For each particle and every scale, the detail coefficients are computed and normalized by $s_\sigma \sigma$. Then, every region that satisfies an energy sum equal to the number of its time steps multiplied by the number of scales is regarded as *dense* and ultimately picked as a vortex candidate. If its number of revolutions is ≥ 2, it is pronounced a

[2]Note that in this respect *length* refers to a duration in iterations or time steps respectively, and is to be distinguished from the notion of scale length.

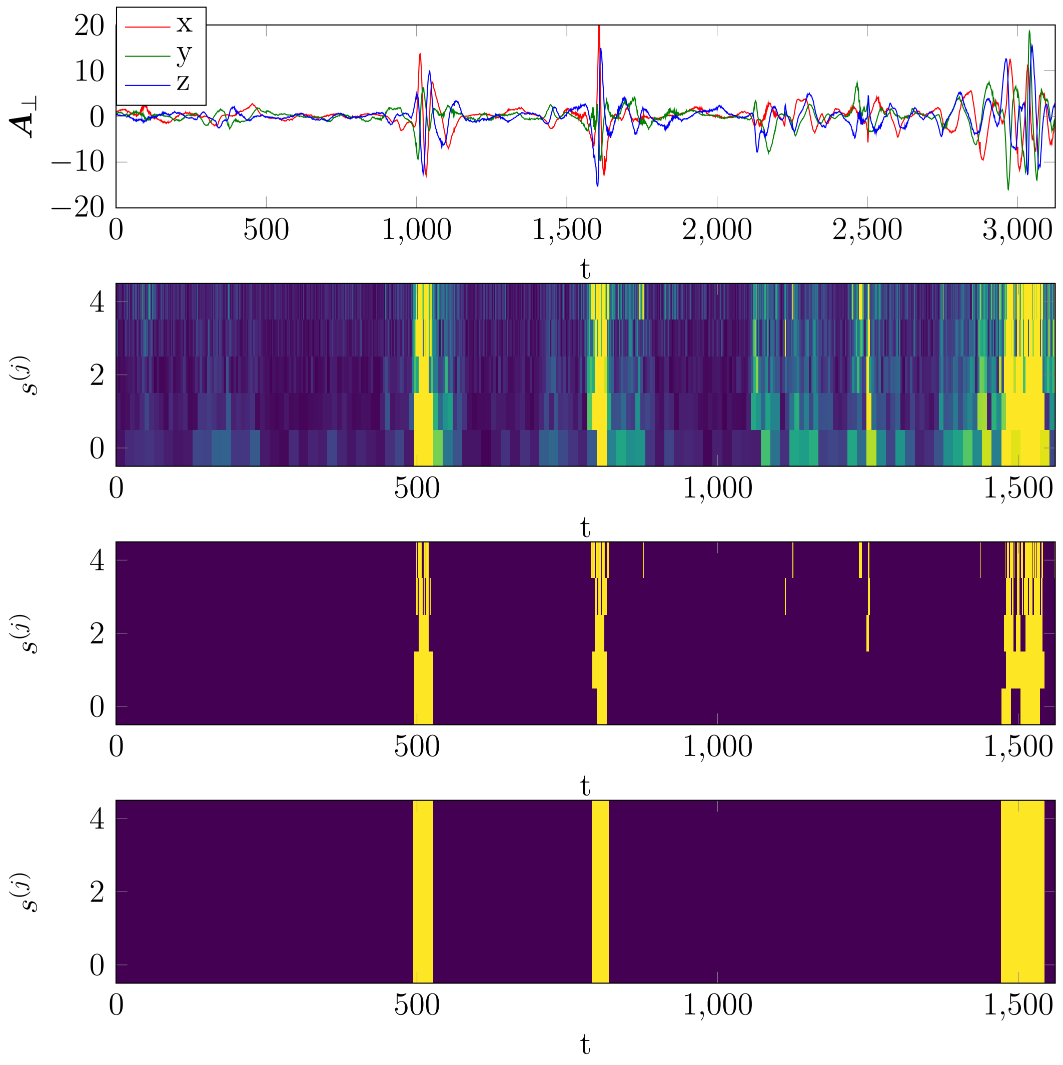


Figure 12: Vortex detection pipeline: After computing $\boldsymbol{A}_\perp$, the median energies are normalized by $s_\sigma \sigma$ (2nd row) and discarded if < 1 (3rd row). If the remaining regions are dense, they are considered vortex candidates, depicted as yellow regions in the bottom row. Detail coefficients are colored from purple to yellow in the range of $[0, 1]$.

vortex signature.

4 Unsupervised data analysis

After extracting vortex signatures using the approach described in the previous section, it is particularly interesting to see if the data displays some kind of structure that may lead to the notion of different types or classes of vortices. Recall that labeled information is not available. Thus, the question becomes how to retrieve structurally relevant information without having any indications about the true nature of the data. Such a scenario is commonly known as unsupervised data analysis. Unsupervised methods try to retrieve information without having *external* knowledge to rely on. Conceptually, they uncover hidden structure by infering a function solely based on the observations, i. e. data points, of the data. Ideally, this function can give insight into the generative model of the data or represent a transformation mapping the data into another space that suppresses noisy phenomena, which contribute only little to the true generative processes but otherwise tend to overshadow them.

Regarding classification, cluster analysis is one of the standard procedures in case of unlabeled data. Based on some notion of distance, it dissects the data into separate groups, which act as individual classes. Here, distance can be regarded as a measure of similarity, decreasing with increased similarity, i. e. two data points are *closer* to one another the more similarity they share. Defined in a metric space, a distance d is called a metric, if for all data points x, y, z it satisfies the following conditions:

1. Positive-definite: $d(x, y) \geq 0$ and $d(x, y) = 0 \Leftrightarrow x = y$
2. Symmetry: $d(x, y) = d(y, x)$
3. Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$

However, the right choice of distance function is not always easy, especially when only little is known about the data. There exists a whole plethora of different distance functions and often representing the data differently will yield much better results given the same distance measure.

For this thesis, several different approaches will be used as analysis methods in order to discover potential structure within the data. In particular, the following two main approaches will form the basis for a subsequent cluster analysis:

1. Shape-based distance functions
    - Dynamic time warping
    - Longest common subsequence
2. Similarity based on data transformation

Algorithm 2 Dynamic time warping.

```
1: procedure DTW($x$, $y$)
2:     $n \leftarrow |x|$, $m \leftarrow |y|$
3:     $M[1..n, 1..m] \leftarrow \infty$
4:     $M[1, 1] \leftarrow 0$
5:     for $i = 2$ to $n$ do
6:         for $j = 2$ to $m$ do
7:             $d \leftarrow \mathrm{dist}(x_i, y_j)$
8:             $M[i, j] \leftarrow d + \min\{M[i-1, j], M[i, j-1], M[i-1, j-1]\}$
9:     return $M[n, m]$
```

- Frequency domain: Fourier transform, (stationary) wavelet transform
- Convolutional sparse autoencoder

4.1 Shape-based distance functions

4.1.1 Dynamic time warping

Dynamic time warping (DTW) is a distance measure originally developed for speech recognition by Sakoe and Chiba in 1978 [29]. In contrast to measuring the distance between two time series $x = \{x_1, x_2, \dots, x_n\}$, $y = \{y_1, y_2, \dots, y_m\}$ per time step, like L^p-norms

$$L^p(x, y) = \left(\sum_i |x_i - y_i|^p \right)^{\frac{1}{p}}, \tag{30}$$

DTW compares two time series by a one-to-many relationship, allowing an *elastic* mapping between two data points. Thus, it is able to find an optimal match between two time series representing a minimum distance and making DTW shift-invariant (see Fig. 13). As a convenient consequence, DTW is able to handle time series of different lengths since the series are *warped* nonlinearly in time. Despite its age, DTW is said to be very competitive still and sometimes even outperforms other state-of-the-art methods with respect to time series data mining applications [30].

DTW makes use of the notion of a local distance function $\mathrm{dist}(x, y)$, e. g. Euclidean or Mahalanobis distance, which allows for more elaborate comparison of multivariate data. As optimization problem, DTW can be implemented using dynamic programming (see Alg. 2), yielding a quadratic runtime. This might seem bad in comparison with the linear L^p-norm but is significantly faster

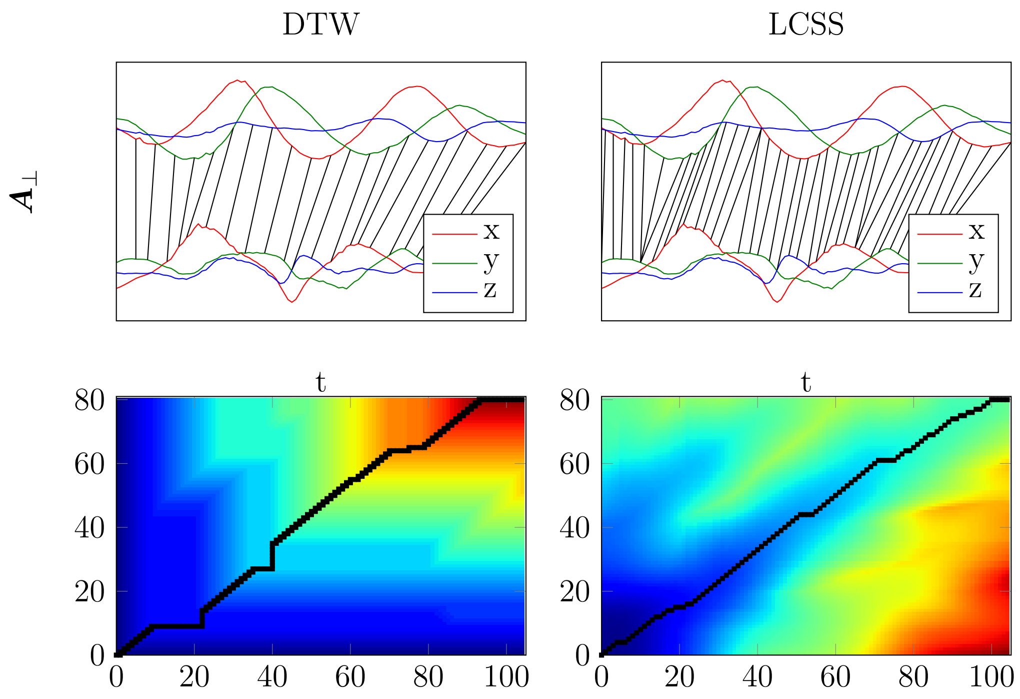


Figure 13: Alignment of individual data points (black lines) and distance matrix for DTW and LCSS ($\epsilon = 10$). The optimal path is marked in black (bottom row).

than the naive approach testing every combination, which runs in exponential time.

DTW finds a nonlinear warping path $P = \{p_1, p_2, \dots, p_l\}$ with minimal cost by matching every data point from one time series to its closest data point of the other time series (see Fig. 13), satisfying the following conditions: [31]

(i) Boundary condition: $p_1 = (i_1, j_1)$ and $p_l = (i_l, j_l)$

(ii) Monotonicity: $i_1 \leq i_2 \leq \dots \leq i_l$ and $j_1 \leq j_2 \leq \dots \leq j_l$

(iii) Step size condition: $p_{l+1} - p_l \in \{(1,0),(0,1),(1,1)\}$

To avoid extreme matchings that map all points in x to a single point y_j, a locality constraint can be introduced such that x_i can only be aligned with y_j, if $|i - j| \leq w$, where w is a locality parameter. Two popular choices for such a locality constraint are the so-called Itakura parallelogram and the Sakoe-Chiba band.

The overall cost $c_p(x, y)$ of the warping path is given by

$$c_p(x, y) = \sum_{t=1}^{l} \mathrm{dist}(x_{i_t}, y_{j_t}). \tag{31}$$

Although it is tempting to regard DTW as a metric, it is in general not positive-definite, even if this holds for $\mathrm{dist}(x, y)$. Also, it does not guarantee the triangle-inequality, i. e.

$$\mathrm{dtw}(x, z) \leq \mathrm{dtw}(x, y) + \mathrm{dtw}(y, z), \tag{32}$$

for time series x, y, z [31]. This might be acceptable in cases that only need some rough measure of similarity but is inappropriate with respect to cluster analysis, where data needs to be separated reliably. However, it can be shown that when using a locality constraint w, DTW satisfies a weak triangle inequality [32]:

$$\mathrm{dtw}(x, y) + \mathrm{dtw}(y, z) \geq \frac{\mathrm{dtw}(x, z)}{\min\{2w + 1, n\}^{1/p}}, \tag{33}$$

where n is the length of the time series and p designates the L^p-norm used as local distance function.

4.1.2 Longest common subsequence

Longest common subsequence (LCSS) is a similarity measure describing the longest subsequence common to both time series. It is based on the edit distance and conceptually similar to DTW. It allows elastic mapping, deals with signals of different lengths, and can be implemented using dynamic programming (Alg. 3). However, since LCSS describes a sequence, it is technically no distance metric by itself. But one can define a metric in the mathematical sense based on LCSS quite easily [33]:

$$\mathrm{lcss}(x, y) = 1 - \frac{|lcss(x, y)|}{\max\{|x|, |y|\}}, \tag{34}$$

where $lcss(x, y)$ is the longest common subsequence of x and y. Intuitively, it measures to what degree the shorter signal differs from the longer. The metric is shown to be symmetric, positive-definite and guaranteeing the triangle inequality [33].

Furthermore, LCSS not only allows one-to-many-mappings but also one-to-none, meaning that the longest subsequence does not necessarily need to be contiguous but can be composed from multiple scattered matching sequences. This is

Algorithm 3 Longest common subsequence

```
1: procedure LCSS($x$, $y$)
2:     $n \leftarrow |x|$, $m \leftarrow |y|$
3:     $M[0..n, 0..m] \leftarrow 0$
4:     for $i = 1$ to $n$ do
5:         for $j = 1$ to $m$ do
6:             if $x_i = y_j$ then
7:                 $M[i,j] = M[i-1,j-1] + 1$
8:             else
9:                 $M[i,j] = \max\{M[i,j-1], M[i-1,j]\}$
10:    return $M[n,m]$
```

especially interesting with respect to signals that miss data, like incomplete reconstructions from fossils or DNA strings.

The longest common subsequence $lcss(x,y)$ can be defined recursively in an intuitive fashion:

$$lcss(x_i, y_j) = \begin{cases} \emptyset, & \text{if } i = 0 \text{ or } j = 0 \\ lcss(x_{i-1}, y_{j-1}) \cup x_i, & \text{if } x_i = y_j \\ \text{longest}(lcss(x_i, y_{j-1}), lcss(x_{i-1}, y_j)), & \text{if } x_i \neq y_j \end{cases} \tag{35}$$

Originally, LCSS was designed for discrete data, like character or DNA strings. Therefore, the condition $x_i = y_j$ in line 6 of Alg. 3 will only seldomly be true for real-valued signals. Instead, a threshold parameter ϵ needs to be specified in this case, yielding the adjusted condition

$$\text{dist}(x_i, y_j) \leq \epsilon. \tag{36}$$

4.2 Similarity based on data transformation

4.2.1 Frequency domain

Due to its oscillations, it should be a reasonable assumption that a vortex's frequency spectrum yields information helping in the clustering process. The idea is to exploit the frequency domain as an intuitive feature space, allowing to extract the data's most significant constituents. Since the Fourier transform is an orthogonal change of basis, no information is lost during the process. Also, it is shift- and rotation-invariant, making for a promising representation that is robust against known issues related to time series similarity.

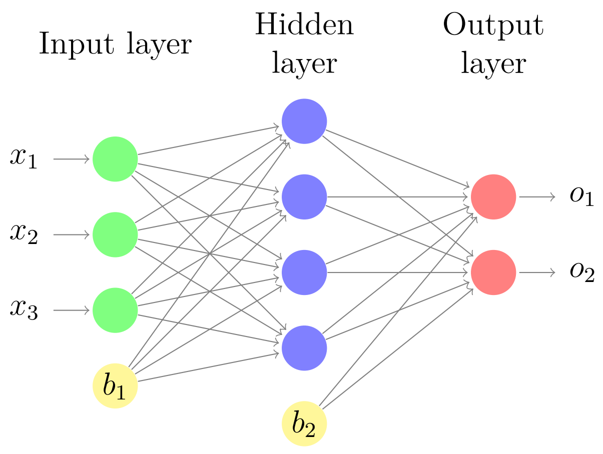


Figure 14: A fully-connected artificial neural network with one hidden layer, three input neurons, and two output neurons.

As discussed in section 3.2.3, wavelet transforms provide another access to the frequency domain. Therefore, detail coefficients represent another possible feature space that can be used as a basis for similarity measures. Unfortunately, time localization now becomes a burden since it prevents the coefficients to be shift-invariant. To overcome this weakness, the highly-redundant stationary wavelet transform (SWT) has been proposed, guaranteeing shift-invariance by substituting the down-sampling process for the up-sampling of detail coefficients instead [34].

4.3 Convolutional sparse autoencoders

4.3.1 Feature learning using artificial neural networks

In artificial intelligence, a common tool to find novel solutions to certain problems is to imitate strategies known from nature. Machine learning is one of the most prominent examples in recent history, building on knowledge about the way the human brain is learning. In particular, the artificial neural network (ANN) is an algorithm trying to mimic a net of interconnected neurons by forming a directed graph of artificial neurons that are grouped in layers capable of learning different activation patterns.

In order to learn the output for a given input, ANNs use specific loss or error functions building the basis for a gradient descent, called backpropagation, that iteratively updates the ANNs' parameters to converge to a minimum. The parameters of ANNs are represented by weighted connections between individual neurons of different layers, which carry an additional bias. Typically,

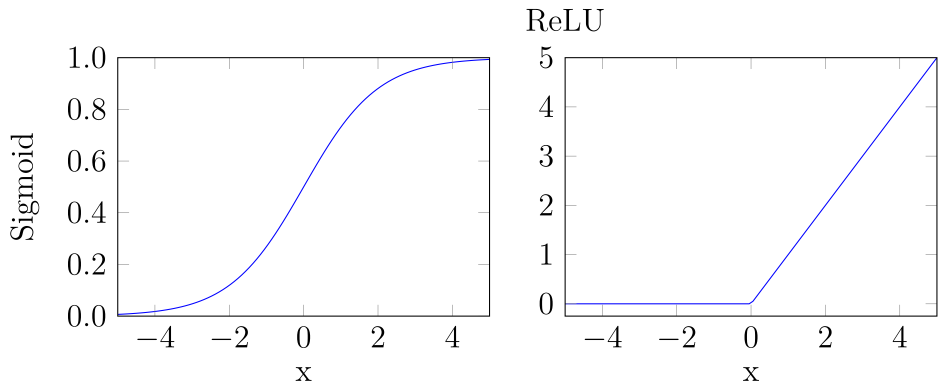


Figure 15: Sigmoid function and rectified linear unit.

ANNs are built from an input layer, one or multiple hidden layers, and an output layer (see Fig. 14). Each layer is composed of a different number of neurons, whose output is passed to the next layer, called a forward-pass. Although layers are connected with each other, there are no connections between neurons of the same layer. Normally, layers are *fully-connected*, meaning that the output of each neuron of one layer is passed to each neuron in the next layer.

Artificial neurons can be regarded as computational units, whose output o is given by a weighted sum of the input $x = \{x_1, x_2, \dots, x_n\}$ passed through a (usually nonlinear) activation function λ:

$$o = \lambda\left(b + \sum_{i=1}^{n} w_i x_i\right), \tag{37}$$

where b is the bias and w_i the i-th weight corresponding to x_i. Depending on the application, different choices for λ may be reasonable. Popular choices are the sigmoid function or rectified linear units (see Fig. 15).

The neural network's overall output is computed by performing a full forward-pass on all layers. Using matrix-notation, each layer l is associated with a weight matrix $W^{(l)}$ and its bias $b^{(l)}$. The output of the l-th layer $a^{(l)}$, also called activation, is then given by

$$\begin{aligned} z^{(l)} &= b^{(l)} + W^{(l)} a^{(l-1)} \\ a^{(l)} &= \lambda(z^{(l)}), \end{aligned} \tag{38}$$

where λ is extended to accept vectorial input. Then, the network's output is the activation of the output layer, denoted $h_{W,b}(x)$.

4.3.2 Supervised learning

As it can be seen, the output of an ANN depends on the parameters of the network (W, b). Thus, the question becomes how to set the parameters such that given a specific input the network produces the desired output, which is also known as the *learning problem.* In case of classification, the output should be a correct class estimate or in case of a regression problem the approximation of a function. As mentioned earlier, the idea to solve the learning problem is to formulate a loss function and minimize it with respect to (W, b). In order for the network to know what it is supposed to learn, it must be provided with a labeled training set $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$, correlating the desired output y_i, also called target, with its associated input x_i. Then, the loss function $E(h_{W,b}(x), y)$ is minimized with respect to the network's parameters:

$$\arg\min_{W,b} E(h_{W,b}(x), y), \ (x, y) \in S. \tag{39}$$

This procedure is called supervised learning since the network is provided with manually labeled information it can rely on to fit its parameters. The actual definition of the loss function depends primarily on the application. For classification problems, cross entropy is a popular choice, whereas squared error functions are oftentimes used for regression problems [35].

Backpropagation In most cases, analytically minimizing the loss function is only feasible for special network topologies [35]. In general, it is solved numerically using a gradient descent algorithm called backpropagation. It propagates back the error through the network while simultaneously updating the layers' parameters using gradient descent. For this purpose, both loss and activation functions need to be differentiable. Then, the parameters of the l-th layer are updated by subtracting their partial derivatives:

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial E}{\partial W_{ij}^{(l)}} \tag{40}$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial E}{\partial b_i^{(l)}}, \tag{41}$$

where α is a small learning rate, e. g. 0.01.

In order to determine the partial derivatives, one needs to know the amount $\delta_i^{(l)}$ each i-th neuron of layer l has contributed to the output's overall error. But since each layer's output is based on the former layer's output, $\delta_i^{(l)}$ needs to be retrieved recursively:

$$\delta_i^{(l)} = \begin{cases} \lambda'(z_i^{(l)})(y_i - o_i), & \text{if layer } l \text{ is output layer} \\ \lambda'(z_i^{(l)}) \sum_j W_{ji}^{(l)} \delta^{(l+1)}, & \text{otherwise.} \end{cases} \tag{42}$$

The partial derivatives are then given by [35]

$$\frac{\partial E}{\partial W_{ij}^{(l)}} = \nabla W_{ij}^{(l)} = a_j^{(l)} \delta_i^{(l+1)} \tag{43}$$

$$\frac{\partial E}{\partial b_i^{(l)}} = \nabla b_i^{(l)} = \delta_i^{(l+1)}. \tag{44}$$

Backpropagation is an iterative approach to minimize the loss function. Therefore, multiple strategies to train a neural network have emerged, most notably batch training and mini-batch training. Using batch training, the parameters are updated after the gradients for the entire training set have been accumulated. In contrast, mini-batch training does a parameter update after repeatedly forward-passing small subsets of the training data. Alg. 4 shows the backpropagation algorithm using batch training.

Algorithm 4 Backpropagation algorithm for a given training set S.

```
1: procedure BatchBackpropagation($S$)
2:     Initialize $W^{(l)}, b^{(l)}$ randomly for all $l$
3:     $\Delta W^{(l)} \leftarrow 0$, $\Delta b^{(l)} \leftarrow 0$ for all $l$
4:     for $(x, y)$ in $S$ do
5:         Compute $\nabla W^{(l)}$, $\nabla b^{(l)}$ using backpropagation
6:         $\Delta W^{(l)} \leftarrow \Delta W^{(l)} + \nabla W^{(l)}$
7:         $\Delta b^{(l)} \leftarrow \Delta b^{(l)} + \nabla b^{(l)}$
8:     $W^{(l)} \leftarrow W^{(l)} - \alpha \left( \frac{1}{|S|} \Delta W^{(l)} \right)$
9:     $b^{(l)} \leftarrow b^{(l)} - \alpha \left( \frac{1}{|S|} \Delta b^{(l)} \right)$
```

4.3.3 Unsupervised learning using autoencoders

In the past it has been shown that supervised learning can be a powerful tool capable of solving problems with high complexity[36]. However, the necessity of a labeled training set is something that cannot always be fulfilled. Fortunately, even in cases of unlabeled data, neural networks can be of considerate help thanks to a process fittingly called unsupervised learning. One such example is the *autoencoder*, which can be considered as a function $f : \mathbb{R}^n \to \mathbb{R}^n$ with

$$f(x) = x, \tag{45}$$

i. e. it is trying to learn its input. Now, this might seem unintuitive at first, but in case of autoencoders one is usually not interested in the output so much as in the *encoding* since it can reflect the generative processes describing the input.

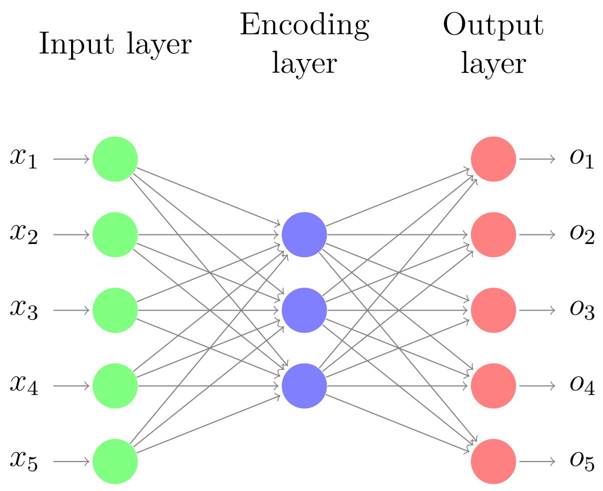


Figure 16: Typical autoencoder. After training, the output layer is usually no longer needed, since the encoding layer represents a transformation that ideally condenses all important constitutional properties of the data.

Typically, autoencoders are fully-connected neural networks with equally-sized input and output. The hidden layer(s) are called encoding layer(s) and decoding layer(s), respectively. In order to train autoencoders using backpropagation, the loss function becomes

$$E(h_{W,b}(x), x), \; x \in S, \tag{46}$$

where S is usually not a subset anymore but rather the entire data. This way, an autoencoder is able to learn different features embedded within the data that would have otherwise been hidden to the observer. In this context, the autoencoder can be regarded as a transformation from one basis into another, leading to a representation that might reveal information about the generative model of the data.

In order for the encoding layer to be meaningful, it usually contains fewer neurons than the input, effectively resembling a dimensionality reduction (see Fig. 16). In contrast to PCA, which assumes a linear model, autoencoders are however able to represent nonlinear mappings. But, if the encoding layer is designed unappropriately with respect to the true complexity of the data, the autoencoder will nevertheless fail to learn anything useful [36], a phenomenon not exclusively known to autoencoder but to all neural networks.

4.3.4 Over- and underfitting

Unfortunately, there is no formal rule to the design or architecture of an ANN. Especially, number of layers and neurons are often subject to experimentation. Due to the multitude of possibilities, it is inherently difficult to decide on an adequate design, specifically if the data exhibits a complexity that eludes analytical approaches. Unfortunately, these are the cases which are most prominent to machine learning.

Nevertheless, one of the main reasons ANNs have enjoyed increasing success in recent years is their ability to approximate, i.e. learn, any function that maps from a finite discrete space to another, also known as the *universal approximation theorem* [37]. However, in practice, much of the success depends not only on the correct design choice but also the quality of the training set. A neural network supposed to learn addition in $\mathbb{N}$ will fail miserably, if it is only shown results for numbers $\in [1, 10]$. Ideally, the network's design, as well as the selectiveness of the training set should reflect the complexity of the data's generative model with perfect accuracy. In practice, this is oftentimes infeasible, which is why ANNs are prone to two statistical phenomena called underfitting and overfitting.

Underfitting describes the inability of a model being too simple to express the data's true complexity. On the other hand, overfitting describes a model being too complex, which therefore represents random noise rather than underlying structure. A relatively easy way to detect either phenomenon is to evaluate the error of the training set and a separate evaluation set not used for training purposes. In case of underfitting, both training and evaluation error will perform poorly. Overfitting mainly shows very good performance on the training set accompanied by a poor performance on the evaluation set.

In order to avoid underfitting, one can simply increase the expressiveness of the network, i. e. add more layers and/or neurons, and observe the effects on the training error. Since overfitting occurs mostly when the model is too complex, e. g. the network has too many layers or neurons, the naive idea to avoid overfitting is to decrease the complexity. However, regularization is a concept capable of dealing with overfitting in a way that helps preserving the complexity of the model despite it possibly being too complex while still learning useful and meaningful features. One such method is *sparsity*.

On a side note: the quality of the training set also affects the convexity of the loss function. If the training set does not cover the entire spectrum of the data's structure, the loss function may become non-convex leading to local minima, which the ANN may converge to. In contrast to under- and overfitting, convexity must be ensured by the training set and cannot be helped by the

ANN itself.

4.3.5 Sparse autoencoders

Sparsity can be considered as an approximation to the maximum likelihood training of a generative model having visible variables $\boldsymbol{x}$ and latent variables $\boldsymbol{l}$. Here, $\boldsymbol{l}$ corresponds to the potentially hidden features, which explain the output. One can then maximize the log-likelihood of the model's joint probability $p(\boldsymbol{x}, \boldsymbol{l})$ given by

$$\log p(\boldsymbol{x}, \boldsymbol{l}) = \log p(\boldsymbol{l}) + \log p(\boldsymbol{x} \mid \boldsymbol{l}). \tag{47}$$

During training, sparsity is implemented by applying a penalty $\Omega(\boldsymbol{l})$ on the encoding layer. As usual, there are multiple ways to formalize $\Omega(\boldsymbol{l})$. Conceptually, one tries to keep the average activation of neurons near zero, i.e. only a small subset of neurons should activate during a forward-pass, thus the term *sparsity*. A popular choice for $\Omega(\boldsymbol{l})$ is the Kullback-Leibler (KL) divergence, which is a standard distance function for probability distributions and given by

$$\Omega(\boldsymbol{l}) = \sum_{j=1}^{n} \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j}, \tag{48}$$

where n is the number of the encoding layer's neurons, ρ a user-specified sparsity parameter, and $\hat{\rho}$ the average activation of the j-th neuron constrained to be $\hat{\rho} = \rho$. Lastly, the minimizing objective for sparse autoencoders becomes

$$\arg\min_{W,b} E(h_{W,b}(x), x) + \Omega(\boldsymbol{l}). \tag{49}$$

By constraining most of the neurons to have zero-activation, the autoencoder is required to learn parameters that represent the statistically most generative constituents of the data. Furthermore, the sparsity term can be regarded as "a consequence of the model's distribution over its latent variables" [36], which is why sparse autoencoders learn meaningful features by theory since they describe the latent variables that make up the output.

4.3.6 Convolutional neural networks

Convolutional neural networks (CNNs) expand the traditional ANN-paradigm by two concepts called local connectivity and parameter sharing. Local connectivity restricts neurons to be connected to only a local region of the input instead of the entire input vector, i. e. the neuron's activation is based solely on a (contiguous) subset of the input. Parameter sharing makes the assumption that a feature that meaningfully describes a local region of the input is also meaningful with respect to another region, e. g. stationary points in periodic signals or edges in images. Traditionally, CNNs have been designed for 2D inputs like images but can be applied to 1D input as well. The following descriptions are therefore carried out with respect to 1D inputs.

Convolutional layer A particular strength of CNNs is to cope with high dimensional input since individual features are obtained locally and can in turn be *reused* in case of occurences within other input regions. CNNs implement this behaviour by introducing convolutional layers, which work differently from fully-connected layers. In particular, the output is computed by convolving the input with so-called filters, hence the term *convolutional* layer. These filters can be considered as direct manifestations of the extracted features since they effectively replace the weight matrices $W^{(l)}$ as parameters of the layer and are subject to the backpropagation training.

The filters are of fixed length k, usually $k \ll n$, where n is the length of input $x = \{x_1, x_2, \ldots, x_n\}$. For each filter g, the i-th output $o_i^{(g)}$ is computed by convolving the filter with a portion of the input,

$$o_i^{(g)} = (g * x)[i] = \sum_{j=1}^{k} g(j)x(is + j), \tag{50}$$

then shifted along by a so-called stride parameter s to compute $o_{i+1}^{(g)}$ until the entire input has been traversed. Thus, each filter output $o^{(g)}$, also called filter map, contains $(n - k)/s + 1$ elements. Depending on s and k, the filters might be convolved with overlapping input portions. As example, consider a convolutional layer with 5 filters of length 3, input size of 7, and a stride of 1, then the output will be a 5×5-matrix, while a stride of 3 would lead to a 5×2-matrix. [3] Similar to the number of neurons in a traditional layer, the number of filters of a convolutional layer is user-defined and application dependent.

As seen, convolutional layers implement the concept of neurons by shifting multiple filters along the input combining both local connectivity and parameter sharing and gradually producing the output. Similar to traditional ANNs, the filters are initialized randomly and updated using backpropagation for training.

[3]However, with a stride of 2, the filters could not be shifted along the input properly since they would overlap with the input at the last window. There are several techniques to avoid these problems like zero-padding, where the input is *padded* to comply with the parameters.

Architecture and design of CNNs Besides convolutional layers, CNNs are most notably composed of activation and pooling layers. Activation layers take over the role of nonlinearity similar to the activation function in traditional ANNs. They take the input and pass it element-wise through an activation function, such as ReLUs or sigmoid. Thus, the output size is equal to the input size. The purpose of pooling layers is mainly to reduce the input size while it is passed forward through the network. They work on non-overlapping regions of the input and *pool* the values based on different criteria like the maximum or mean value. Consider max-pooling with a size of 2, then the output contains the maximum of each neihgboring pair of values:

$$[1\ 4\ 5\ 7\ 3\ 2\ 8\ 9] \rightarrow [4\ 7\ 3\ 9].$$

Thus, depending on the pooling size, the input is considerably reduced in resolution, while maintaining most of the structure embedded within the input.

Typically, convolutional layers are followed by an activation and a pooling layer. But again, there is no formal rule to the design of CNNs. A convolutional layer may as well directly follow another convolutional layer, or an activation layer may be skipped inbetween a combination of two convolutional and pooling layers. However, CNNs always end in a traditional fully-connected ANN. To convert the output of a convolutional layer into the input for a fully-connected layer, one simply *flattens* the filter maps into a vector. The output of the ANN then is the overall output of the CNN.

4.3.7 Convolutional autoencoders

The same way autoencoders resemble mirrored ANNs, convolutional autoencoders can be constructed from mirroring a standard CNN (see Fig. 17). Unlike ANNs, operations like pooling or convolution considerably alter the form of the input. In order to *mirror* a CNN, question becomes how to reverse these steps appropriately, also called deconvolution and unpooling. Commonly, unpooling is implemented by simply upsampling the input to the desired dimensions. Surprisingly as simple, deconvolution is nothing other than another convolution, which is why deconvolution layers can be implemented using normal convolutional layers.

5 Cluster analysis

In unsupervised settings, where labeled information is not available, cluster analysis is one of the standard approaches for finding seemingly inaccessible

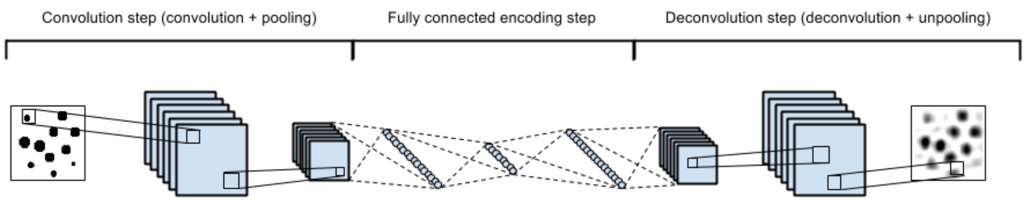

Convolution step (convolution + pooling)
Fully connected encoding step
Deconvolution step (deconvolution + unpooling)

Figure 17: 2D convolutional autoencoder [38].

structure. Given a data set X of multivariate data points $x = \{x_1, x_2, \dots, x_n\}$, clustering algorithms subdivide X into groups (clusters) of individual points based on specific criteria, often a pair-wise distance measure, i. e. points in close proximity to each other are grouped together, whereas points far away are separated by different clusters. As indicated before, in this respect distance is a way to express (dis)similarity, not just necessarily in a metric sense. Points sharing certain similarities, e. g. a generative model, should exhibit close proximity and unsimilar points, of course, a rather large distance to each other. The nature of the distance measure therefore determines as to what aspects the data is looked upon.

Cluster analysis has its tradition in the long history of scientific classification, be it taxonomy of biological organisms or descriptions of different types of art. In most cases, however, scientific data is too complex and rich in features to manually group them into clusters. Therefore, a wealth of clustering algorithms has been developed to aid this need of statistical analysis. In general, two main approaches exist: flat clustering and hierarchical clustering.

5.1 Flat clustering

Flat clustering describes methods that assign clusters with a flat hierarchy and no ”explicit structure that would relate clusters to each other” [39]. Intuitively, every cluster represents an equally valid group within the data’s distribution. Due to its simplicity and efficiency, k-means is one of the most popular flat clustering algorithm (see Fig. 18). Each x is assigned its nearest *mean* measurement m_i, $1 \leq i \leq k$, which is usually $\notin X$ but a randomly initiated point representing one of the centroids of the k clusters. Points are assigned based on their Euclidean distance, which mathematically equals a Voronoi-partitioning and informally assumes clusters of roughly the same size. This way, k-means can be formulated as optimization problem with respect to the set of clusters $C =$

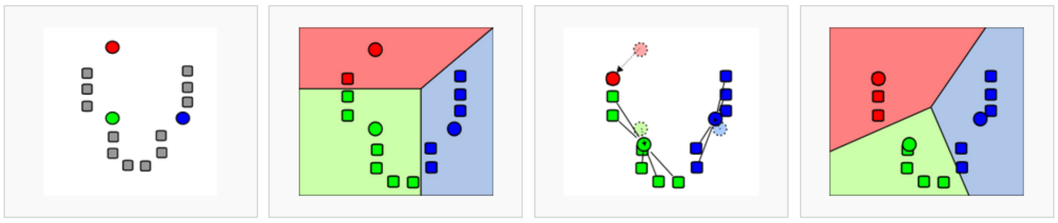


Figure 18: Estimation-maximization steps of the k-means algorithm. k initial means are seeded randomly (left). During estimation, each data point is assigned its nearest mean based on euclidean distance (mid-left). During maximization, each mean is updated to be the centroid of the respective cluster (mid-right). Each data point is then re-assigned (right). The last two steps are iterated until convergence. Author: Weston.pace.

$\{C_1, C_2, \ldots, C_k\}$ minimizing the squared distances within each cluster:

$$\arg\min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} ||x - m_i||^2. \tag{51}$$

To avoid exponential runtime by enumerating all possible clusterings, k-means is usually implemented using an iterative algorithm organized in two steps: estimation and maximization. During the t-th estimation step, each cluster $C_i^{(t)}$ with mean $m_i^{(t)}$, $1 \leq i \leq k$, is determined the following way:

$$C_i^{(t)} = \{x \mid ||x - m_i^{(t)}||^2 \leq ||x - m_j^{(t)}||^2, 1 \leq j \leq k\},\ x \in X. \tag{52}$$

Afterwards, each $m_i^{(t)}$ is updated with the centroid of $C_i^{(t)}$ defined by

$$m_i^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{x \in C_i^{(t)}} x. \tag{53}$$

Initially, all m_i are set randomly. k-means then runs the expectation-maximization steps repeatedly either until convergence, i. e. all points are assigned the same centroid, or for a fixed number of iterations. Consequently, k-means is nondeterministic.

Unfortunately, k-means suffers from known weaknesses. In general, the correct number of clusters k is unknown, giving rise to a whole branch of research trying to address this problem (see [40] [39]). Often enough, one can only guess from experience about an adequate number of clusters. Furthermore, the iterative approach does not guarantee a global minimum, especially if X contains many outliers, i. e. points that do not fit well into any cluster, because they are inherently different from other data points. In practice, one therefore runs the algorithm multiple times with different starting configurations and then picks the clustering with minimum cost.

5.2 Hierarchical clustering

In contrast to k-means, hierarchical clustering algorithms do not require a predefined number of clusters and are generally deterministic. In addition, they provide a hierarchically organized set of clusters as output, which can be significantly more informative. The idea behind hierarchical clustering is to gradually merge the closest clusters on each hierarchy step until all clusters have been merged. Initially, each data point is its own cluster. Then, the two closest data points/clusters are merged into one cluster with the rest remaining. In the next step, the two closest clusters again are merged and so on until only one cluster remains. During this process, each merge represents a new level in hierarchy. This strategy is called bottom-up.

Mainly, there are three ways to determine the distance between clusters:

1. single-linkage,
2. complete-linkage, and
3. average-linkage.

Single-linkage is based on the most similar pair of data points from two clusters, whereas complete-linkage uses the most dissimilar pair of all data points from two clusters. Lastly, average-linkage considers the average distance of every pair. If interpreted geometrically, single-linkage factors in only local aspects of the closest proximity between two clusters while neglecting aspects from other areas. This can favor clusters, which span very long, over more compact clusters. In contrast, complete-linkage considers the overall structure of both clusters, taking into account global information, which leads to relatively compact clusters since the pair with smallest diameter is merged. However, this makes complete-linkage sensitive to outliers. Finally, average-linkage is a compromise between single- and complete-linkage that tries to dampen both weaknesses.

Besides bottom-up strategies, hierarchical clustering can also be performed using a top-down approach. The idea behind it is to use a flat clustering algorithm as a subroutine to split each cluster at each hierarchy level recursively, starting with all data points as single cluster at the top. This makes top-down hierarchical clustering conceptually more complex than bottom-up approaches but can ironically yield performance advantages when used in combination with an efficient algorithm. Although flat clustering is exponential in runtime, expectation-maximization methods are widely more efficient since often only few iterations are sufficient to give good results. If run for a fixed number of hierarchy levels, the top-down approach runs linearly in the number of data points and clusters using k-means, whereas bottom-up methods have at least quadratic runtime [39].

Furthermore, top-down approaches take into account the data's global distribution when making top-level partitioning decisions, whereas bottom-up approaches cannot undo early decisions based solely on local criteria. There is evidence, that top-down hierarchies benefit the accuracy of results [39]. However, top-down approaches usually inherit the weaknesses of their flat clustering algorithms.

5.3 Hierarchical density based clustering: HDBSCAN*

The aforementioned methods all share the underlying principle of minimum variance by miminizing some measure of variance within clusters. One of the biggest problems these approaches have in common is the inability to find clusters of arbitrary shapes [41]. Density-based clustering provides a very intuitive solution to this problem by regarding data points as samples drawn from an unknown probability density function in which multiple modes correspond to different clusters. In 1975, Hartigan formalized this idea by describing the concept of density-contour clusters, which for given data points X and some density-threshold ε are maximal subsets C such that $p(x) \geq \varepsilon$ holds for every $x \in C$ with $p(x)$ being a continuous density function defined on all points. This way, clusters of arbitrary shape are nothing other than several Cs connected by a continuous density level $\geq \varepsilon$.

Besides arbitrarily shaped clusters, Hartigan's model also entails the notion of *noise* defined as objects lying in regions with a density $< \varepsilon$. This prevents the rather unrealistic assumption that each data point must belong to a distinct cluster, incorporating the idea of *extreme* observations that do not necessarily contribute to the structure of the data. This is of special interest with respect to modeling unknown phenomena (or anomalies) since the results are less biased by noise.

HDBSCAN* [41] is a density-based clustering algorithm that combines merits both from hierarchical and density-based clustering methods. Based on a single parameter m_{min}, describing the minimum number of data points constituting a cluster, it returns a hierarchical clustering structured as dendrogram based on two notions called

1. core distance, and
2. mutual reachability.

The core distance d_c of a data point x with respect to m_{min} is defined as its m_{min}-nearest neighbor, including x itself. Formally, it defines the minimum radius for which x satisfies the core condition with respect to m_{min}, i. e. for which x is not a noise object anymore. The mutual reachability distance d_{mr} between two data points x_1 and x_2 is defined as

$$d_{mr}(x_1, x_2) = \max\{d_c(x_1),\ d_c(x_2),\ d(x_1, x_2)\}, \tag{54}$$

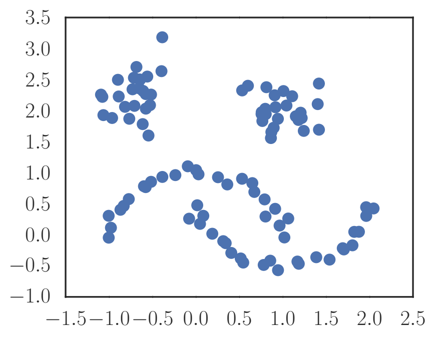
where $d(x_1, x_2)$ is a local distance function, and serves as basis for the so-called mutual reachability graph G_{mr}, which is an undirected complete weighted graph with the data points as nodes and $d_{mr}(x_1, x_2)$ as edge weight between nodes x_1 and x_2. Intuitively, G_{mr} describes a proximity graph of X with respect to the notion of core distance. From G_{mr}, a minimal spanning tree is extracted and expanded by "self edges" leading from every data point to itself weighted by its core distance. After this, a dendrogram can be built in top-down fashion by iteratively removing all edges from the expanded minimum spanning tree in decreasing order of weights. At each removal, the weight of the removed edge becomes the *scale* value for the current hierarchy level. The scale value describes the height of the dendrogram, interpreted as tree structure, at which a cluster is subdivided into multiple clusters when traversed from top to bottom.

To extract a flat clustering from the dendrogram, one can *cut* it at a given scale value v. The separate branches are considered clusters, if the number of their children $\geq m_{min}$. The data points contained in a cluster are then called core objects with respect to v, whereas data points outside of clusters are interpreted to be noise objects (see Fig. 19).

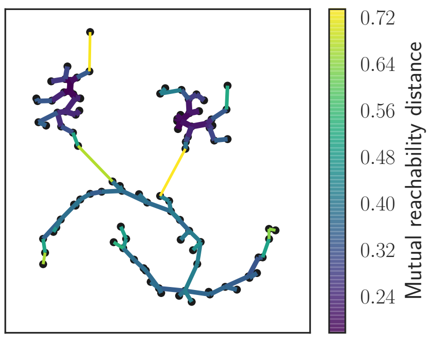
5.4 Evaluation of hierarchical density-based clustering

The particular strengths of HDBSCAN* are its ability to automatically determine the number of clusters, find clusters of arbitrary shape and provide a full hierarchy of the clustering from which different flat clusterings can be extracted. However, this bears a new problem: how to cut the dendrogram to obtain a representative clustering? For this purpose, Campello et al. introduce the notion of cluster stability [43].

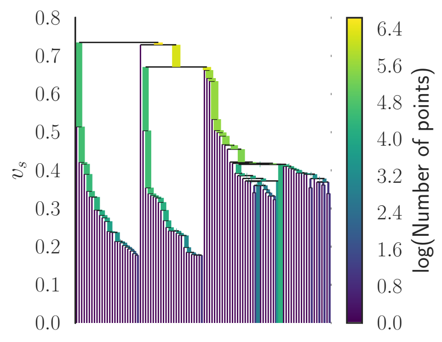
When traversing a dendrogram from top to bottom, a cluster C's lifetime is determined by the scale value v_s, where the cluster splits off from its parent branch, and the value v_e, where it is split into smaller clusters. Additionally, each point x in that cluster is assigned the scale value v_x, $v_e \leq v_x < v_s$,



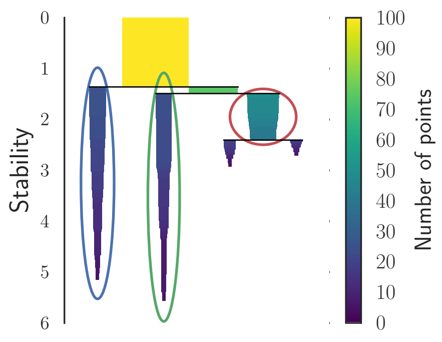
(a) Data points to be clustered with HDBSCAN*.



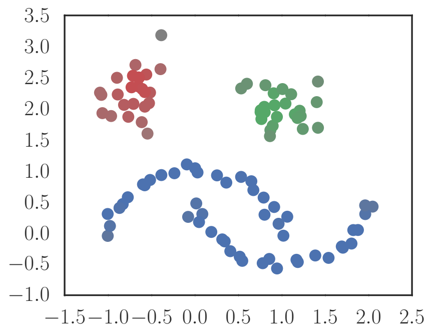
(b) Minimum spanning tree of G_{mr}.



(c) Dendrogram of clustering hierarchy.



(d) *Condensed* dendrogram with selected clusters.



(e) Clustered data points.

Figure 19: HDBSCAN* custering pipeline for a data set with three true clusters [42].

describing the moment x falls out of the cluster. The cluster's stability s_C is then defined as

$$s_C = \sum_{x \in C} \left(\frac{1}{v_x} - \frac{1}{v_s} \right). \tag{55}$$

To determine the most stable clusters of every branch, each data point is initially considered a cluster. Moving bottom up, if the cluster's stability is smaller than the sum of its childrens' stabilities $s_{children}$, set $s_C = s_{children}$. Else, pick the cluster as the current most stable cluster of the branch. At the root, return each current cluster as flat clustering result (see Fig. 19).

6 Evaluation

6.1 DWT-based vortex detection

To evaluate the effectiveness of the proposed algorithm based on discrete wavelet transform, a subset of 200 particle trajectories has been manually labeled – dubbed ground truth – with a total outcome of 70 vortices spanning 14530 steps in time. Recall that particle trajectories might only cover small portions of a vortex before leaving the vortex's field of influence. It is therefore important to define the notion of a minimal vortex signature, which in context of this thesis is required to contain at least two revolutions (see Fig. 20).

	Time steps												
Ground truth	0	0	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	0	0	0	$\underline{1}$	$\underline{1}$	0	0
Algorithm	0	0	0	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	0	$\underline{1}$	$\underline{1}$	0	0	0
Evaluation	t_n	t_n	f_n	t_p	t_p	t_p	f_p	t_n	f_p	t_p	f_n	t_n	t_n

Table 1: Evaluation scheme, where $\underline{1}$ denotes a vortex time step, 0 a non-vortex time step, t_n a true negative, t_p a true positive, f_p a false positive, and f_n a false negative.

As parameters, the proposed algorithm expects a scaling factor s_σ governing the energy threshold for the detail coefficients and the vortex's minimum number of revolutions r. To measure the accuracy of the algorithm with respect to the label span, the trajectories have been labeled per time step (see Table 1). For statistical interpretation, the true positive rate TPR and true negative rate TNR are defined as

$$TPR = \frac{n_{tp}}{n_{lp}} \tag{56}$$

$$TNR = \frac{n_{tn}}{n_{ln}}, \tag{57}$$

where n_{tp}/n_{tn} are the number of true positives/negatives, i. e. correctly classified time steps labeled as vortex/non-vortex, and n_{lp}/n_{ln} the number of time steps labeled as vortex/non-vortex. Furthermore, since non-vortex labels dominate the data by more than 97%, their statistics will be strongly biased. Focus will therefore be directed on statistics with respect to vortex labels, which promise more expressive results. In particular, the positive predictive value PPV and its counterpart, the false discovery rate FDR are defined as

$$PPV = \frac{n_{tp}}{n_{tp} + n_{fp}} \tag{58}$$

$$FDR = \frac{n_{fp}}{n_{tp} + n_{fp}}, \tag{59}$$

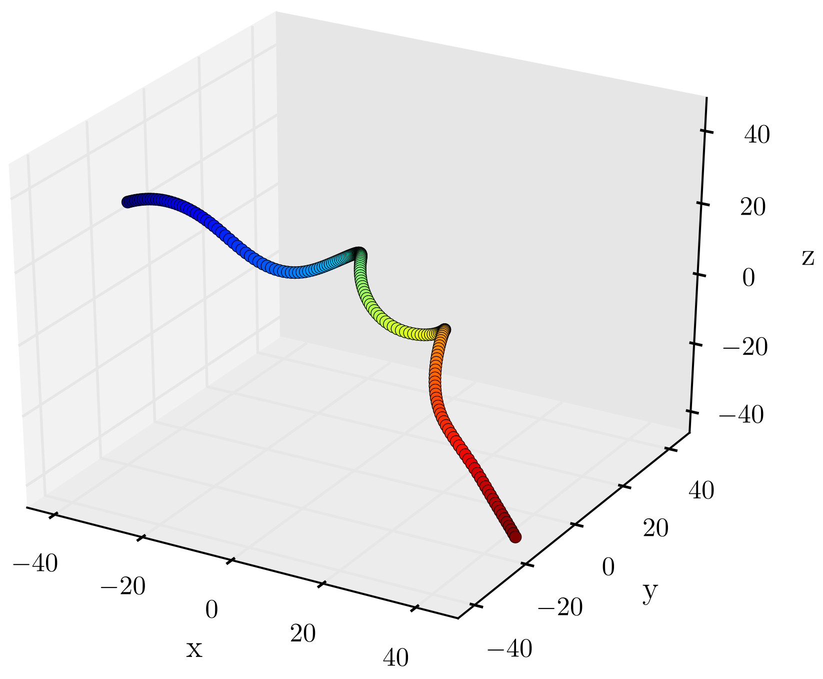


Figure 20: Example of a particle trajectory satisfying a minimum requirement of two revolutions; colored by time steps from blue to red.

where n_{fp} is the number of false positives.

Evaluation results are given in Table 2. Unsurprisingly, TNR is exceptionally good for all parameters since it is heavily biased by a ratio of 97.62% non-vortex labels. This would make it easy to simply reject every time step as non-vortex and still score extremely well. On the other hand, results for TPR are almost disappointing. Only parameters ($s_\sigma = 2, r = 2$) score above 50%, which at the same time comes at the cost of highest FDR. One reason for this is most likely sensitivity to label borders, which even for the trained eye are not easy to select, because a vortex signature's beginning and ending are rather fuzzy concepts (see Fig. 21). In fact, if one was to label only the vortex signatures as simple occurences, i. e. without paying attention to actual time steps, results for TPR universally improve (see Table 3). However, TPRs of 60% at best are still rather unsatisfactory.

Question then becomes where the algorithm struggles the most. Looking at Fig. 21, the increased false detection rates for $s_\sigma = 2$ seem to be caused mainly by short bursts of lightly increased $\boldsymbol{A}_\perp$, which to a smaller degree also holds for $s_\sigma = 3$. Again, this is understandable since the lower threshold will allow

(s_σ, r)	TPR	TNR	PPV	FDR
(2, 2)	0.5282	0.9865	0.3976	0.6024
(2, 3)	0.3624	0.9948	0.5404	0.4596
(2, 4)	0.1297	0.9995	0.8053	0.1947
(3, 2)	0.4316	0.9959	0.6413	0.3587
(3, 3)	0.2343	0.9984	0.7168	0.2832
(3, 4)	0.0991	0.9998	0.8707	0.1293
(4, 2)	0.3530	0.9978	0.7310	0.2690
(4, 3)	0.1758	0.9997	0.9008	0.0992
(4, 4)	0.0991	0.9998	0.9110	0.0890

Table 2: Evaluation results for the proposed vortex detection algorithm based on DWT.

(s_σ, r)	TPR	PPV	FDR
(2, 2)	0.6571	0.5000	0.5000
(2, 3)	0.3714	0.7222	0.2778
(2, 4)	0.0857	1.0000	0.0000
(3, 2)	0.6000	0.8077	0.1923
(3, 3)	0.2571	0.9000	0.1000
(3, 4)	0.0857	1.0000	0.0000
(4, 2)	0.4571	0.8000	0.2000
(4, 3)	0.1714	1.0000	0.0000
(4, 4)	0.0857	1.0000	0.0000

Table 3: Evaluation results, if one was to neglect time step accuracy. Since in this case the notion of non-vortices is not applicable, TNR is omitted.

more positives. However, the vortex label in Fig. 21 (red box) indicates that the assumption of the detection algorithm does not always hold. The subtle and almost unremarkable region for $\boldsymbol{A}_\perp$ shows a distinct vortex signature when plotted for $\boldsymbol{X}$ (see Fig. 22). This creates a problem because such cases are essentially hidden to the algorithm. In cases, where $\boldsymbol{A}_\perp$ exhibits a clear and distinct vortex pattern, almost all parameters perform reasonably well.

6.2 Cluster analysis

Since parameters ($s_\sigma = 3$, $r = 2$) describe the most reasonable compromise between TPR and FDR, all 500'000 particles have been processed with ($s_\sigma = 3$, $r = 2$) as parameters, leading to a total of 127'815 vortex candidates with an average and maximum length of 124 and 360 time steps, respectively. A histogram of different numbers of revolutions is shown in Fig. 24. Fig. 23

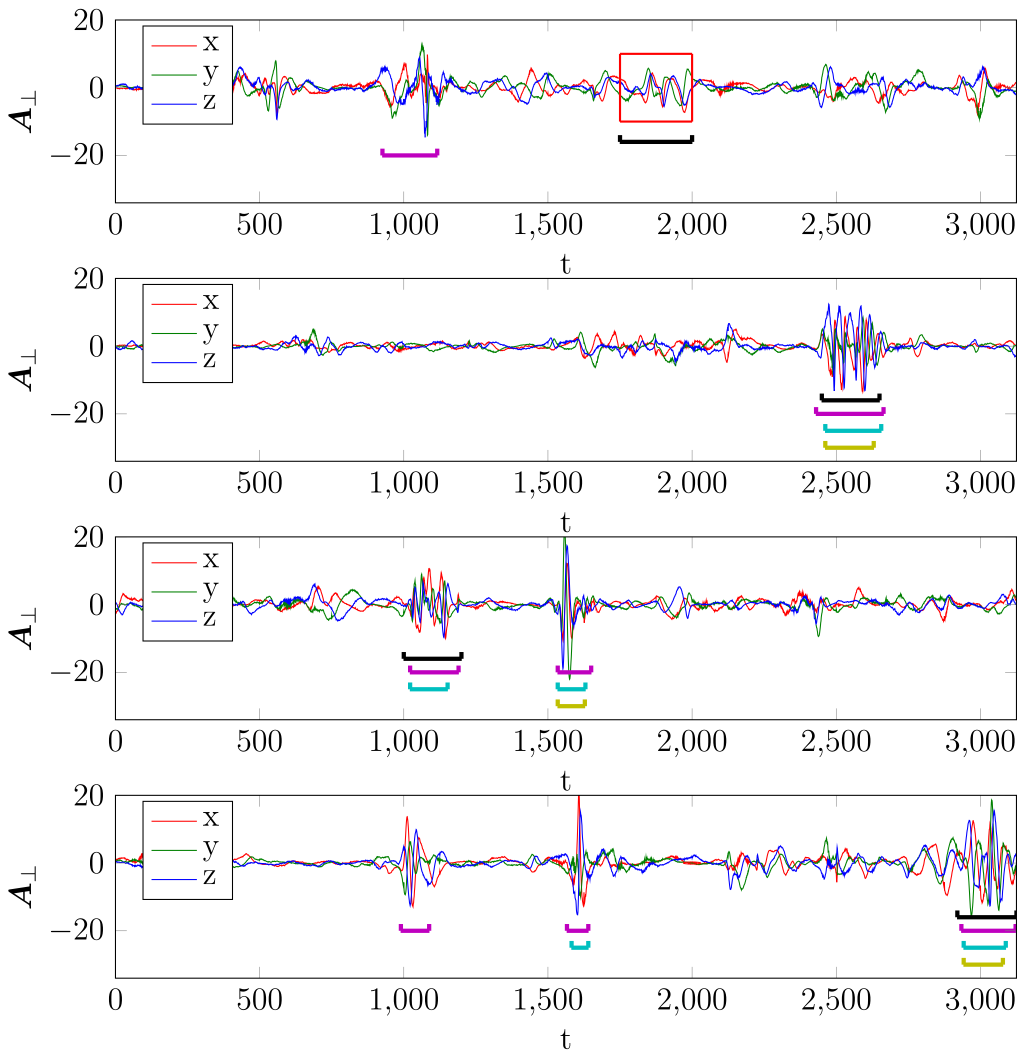


Figure 21: $\boldsymbol{A}_\perp$ for multiple tracer trajectories. Labels are marked in black, vortex detection estimates in magenta ($s_\sigma = 2$), cyan ($s_\sigma = 3$), and yellow ($s_\sigma = 4$) with $r = 2$. The red box depicts the undetected vortex signature shown in Fig. 22.

shows the spatial distribution of 300 detected vortex signatures within the time span $t \in [1000, 1500]$.

Due to limitations in RAM, a subset of 10'000 vortices has been chosen as data basis for the following analysis aiming at uncovering unknown structure using unsupervised methods. Based on the results in Table 3, it should be noted that approximately 20% of the data are supposably no vortices. Therefore, it is interesting to see if this will reflect in the clustering results.

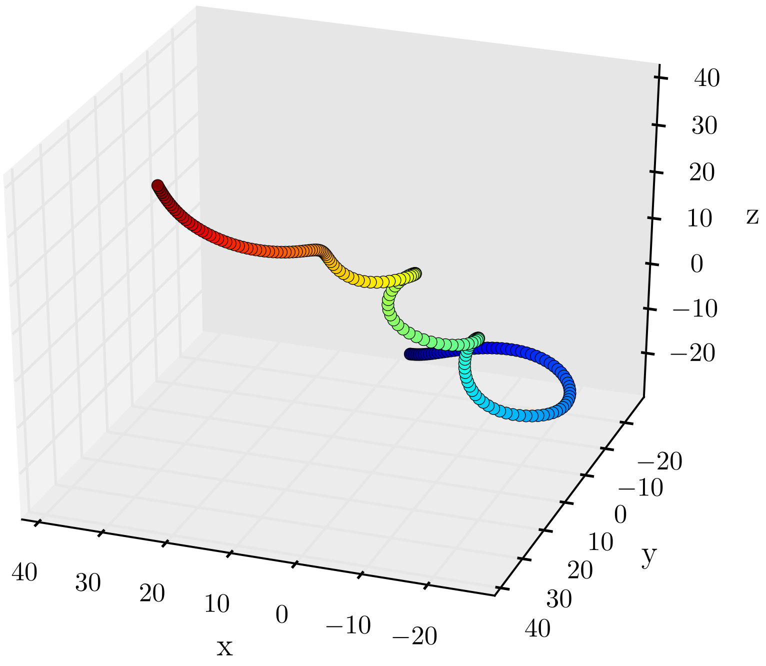


Figure 22: Vortex signature with unremarkable lateral acceleration, depicted as red square in top row of Fig. 21; colored by time steps from blue to red.

Clustering parameterization A great advantage of HDBSCAN* over other clustering methods is its single parameter – minimum cluster size m_{min} –, which is relatively intuitive to handle. For data sets with several thousand samples, a cluster size of several hundred samples should be reasonable. However, since in an unsupervised setting one has simply not enough information to be *sure*, the possibility of clusters with significantly smaller sizes cannot be ruled out completely. For this evaluation, all clusterings will therefore be parameterized with $m_{min} = 100$.

6.2.1 Prerequisites

Curse of dimensionality One of the most prominent problems for multivariate time series is the *curse of dimensionality* describing the statistical phenomenon of increasing sparsity for high-dimensional data.[4] Interpreted geometrically,

[4]In this respect, sparsity is different from the notion of sparsity in terms of regularization methods.

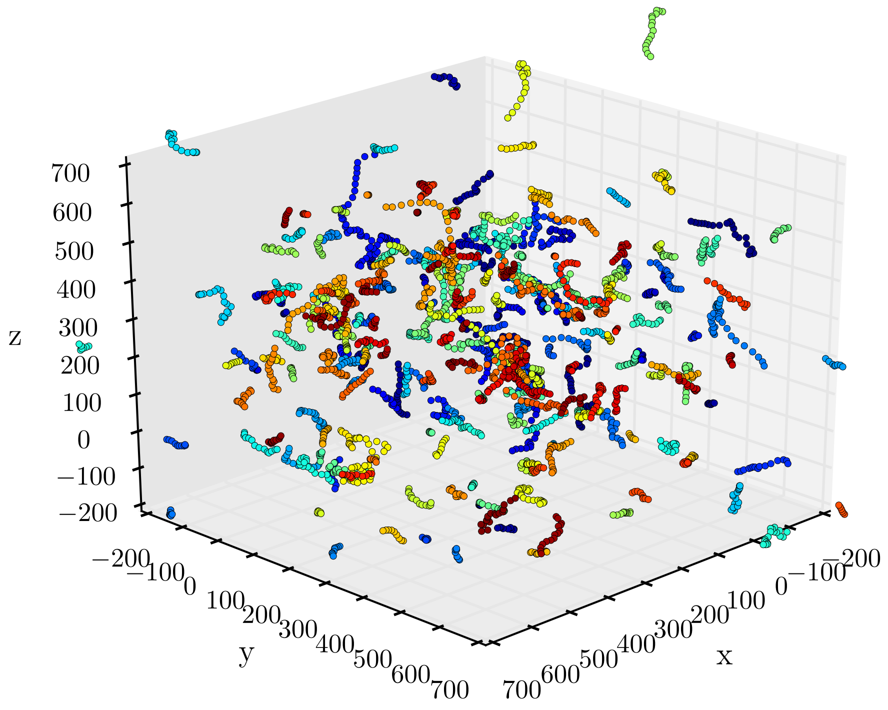


Figure 23: 300 vortex signatures for $t \in [1000, 1500]$ showing a relatively smooth distribution throughout the simulation volume.

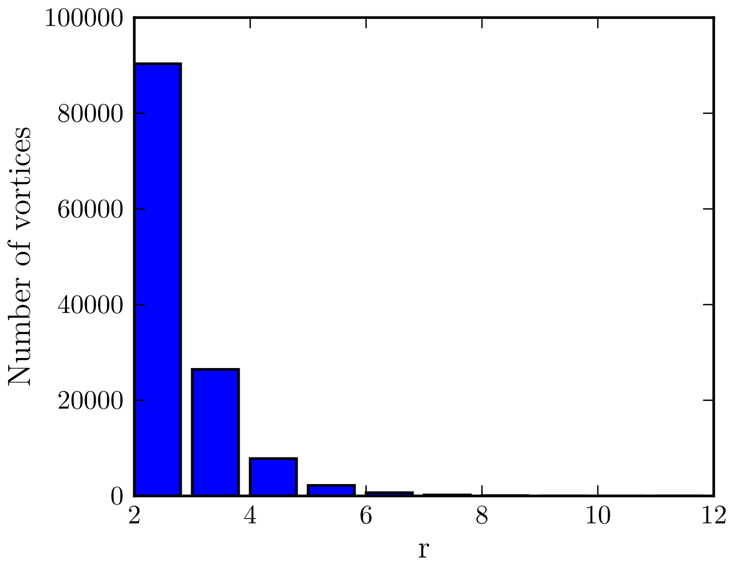


Figure 24: Histogram of revolutions per detected vortex signature.

the feature space's volume increases with every dimension such that the data ultimately becomes sparse [44]. Unfortunately, this poses a problem for methods requiring statistical significance, such as distance measures and – consequently – cluster analysis, since the amount of data needed for statistically sound results often increases exponentially. Since time series "are essentially *high dimensional* data" [30], one needs to account for this problem. Unfortunately, there is little consensus about the number of features for which data becomes high-dimensional. Aggarwal et al. suggest a number of 20 features and greater [44].

Principal component analysis (PCA) is a standard approach for dimensionality reduction and thus mitigates the effects of high-dimensionality. Unfortunately, it assumes a linear model, which might not always be appropriate. For non-linear models, kernel-PCA is a parameterized extension. However, the choice of a kernel and its parameters is not an easy task and requires a sound understanding of the mechanics in the kernel space. It will therefore not be considered. Fortunately, autoencoders represent an alternative to kernel-PCA by constraining the encoding layer to be smaller than the input layer.

Dimensionality reduction is most commonly accompanied by loss of information. To preserve important data components, it is necessary to monitor how much information is lossed during the process. The *coefficient of determination* R^2 indicates how much of the data's variance can be *explained* by the reduced model and is commonly defined as

$$R^2 = 1 - \frac{\sum_i (x_i - f_i)^2}{\sum_i (x_i - \mu)^2}, \tag{60}$$

where f_i are the model's predictions and μ is the mean of the data points x_i. Then, one can stop dimensionality reduction when R^2 falls below a specified threshold.

Another solution to the curse of dimensionality are (local) distance functions that take into account variance distributions, like weighted Euclidean or Mahalanobis distance. Interestingly, L^p-norms with $p \leq 1$ have shown to work significantly better for high-dimensional data than $p > 1$ [44].

Rotation and shift variance In case of 3D vortices, the rotation axes may naturally have different orientations, altering the patterns of the lateral acceleration while potentially representing the same model (see Fig. 25). Also, the detected beginning of a vortex may not always be exact, making the data prone to shift variance. Fortunately, most of the proposed analysis methods and transformations are already shift-invariant. However, DWT is not. Instead, the stationary wavelet transform (SWT) will be used to obtain shift-invariant

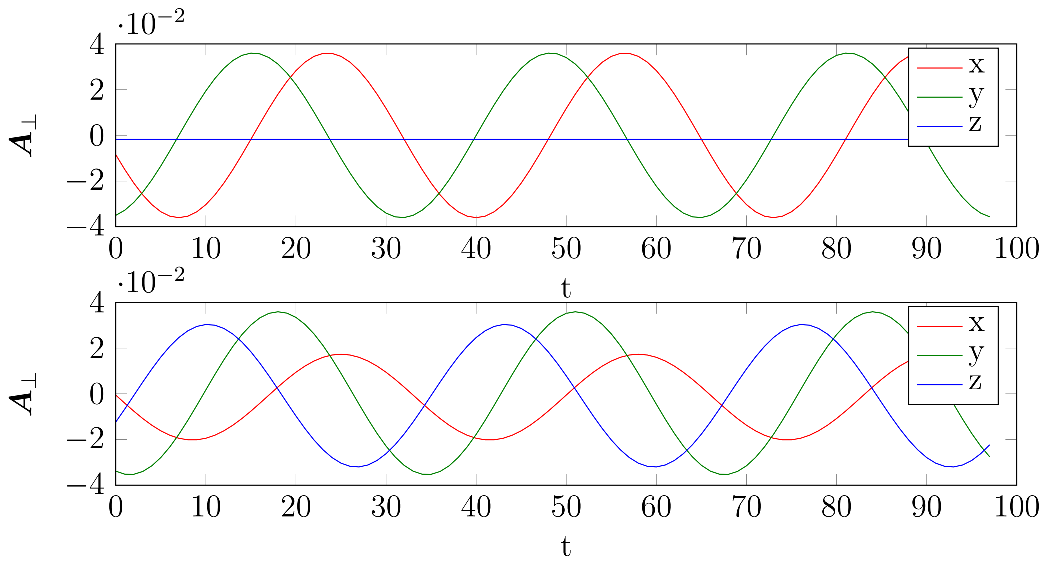


Figure 25: Lateral acceleration patterns of a 3D-helix with the rotation axis centered at the z-axis (top) and then rotated by 45° around each axis (bottom). Although the helix is unaltered with respect to its shape in spatial dimensions, its orientation has a strong impact on the acceleration signal.

detail coefficients. To defend against rotation variance, the data is rotated using singular value decomposition (SVD) before analysis. Since SVD is an orthogonal change of basis, all information is preserved.

Different input lengths Recall that particle trajectories may cover only portions of a vortex. Consequently, the detected vortex signatures differ in length, which is problematic for methods that assume uniform input sizes, e. g. neural networks or Fourier transform. Padding is a common way to circumvent this problem by *padding* the input with additional values like zeros (zero-padding) or other reasonable data. For certain methods, like Fourier transform, SWT or PCA, zero-padding is of no consquence. In case of neural networks, however, every type of padding will influence the parameters. It is therefore important to monitor its effects on the output.

6.2.2 DTW and LCSS

Dynamic time warping Since DTW satisfies a weak triangle inequality when using a locality constraint, it has been used in conjunction with the Sakoe-Chiba

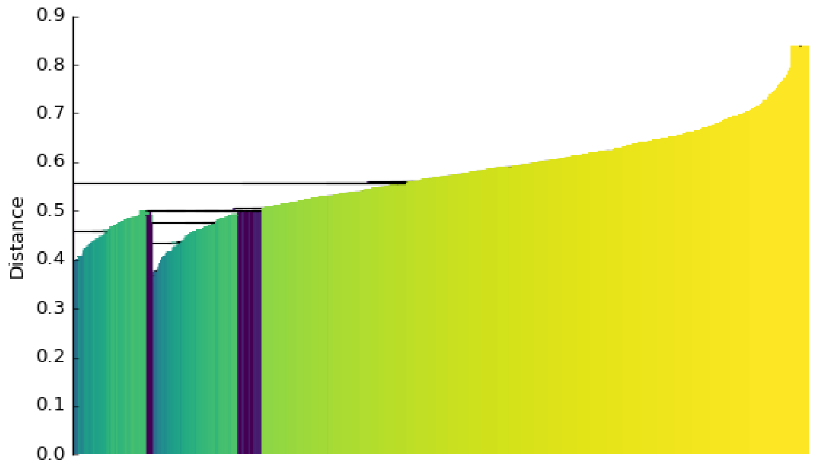


Figure 26: Dendrogram for clustering using LCSS with $\epsilon = 15$ and $m_{min} = 2$.

band and L^1-norm as local distance function. Although DTW accepts different-sized input, length has an effect on the comparability of the results. Consider input pairs (x_1, y_1), (x_2, y_2) with x_1 and y_1 being considerably longer than x_2 and y_2. Let both pairs be equally dissimilar with an average local distance c. Since the warping path for (x_2, y_2) will be shorter, it holds that

$$\mathrm{dtw}(x_1, y_1) > \mathrm{dtw}(x_2, y_2). \tag{61}$$

Conceptually, both pairs are equally dissimilar, meaning that x_1 should be equally distant from y_1 than x_2 is from y_2, but this is not the case. To ensure comparability, each distance has therefore been normalized with the length of the optimal warping path.

Longest common subsequence Recall that LCSS has been devoloped for discrete data and therefore expects a distance threshold ϵ for real-valued input. As any parameter, it is not easy to estimate an appropriate value for ϵ. Furthermore, the data is multivariate, making an intuition even more unlikely. Therefore, ϵ has been tested for values $10, 15$ and 20.

For both DTW and LCSS, the evaluation comprised of the following steps:

1. Rotate each vortex signature using SVD.
2. Build distance matrix D using DTW with Sakoe-Chiba band or LCSS with $\epsilon \in \{10, 15, 20\}$, respectively, and L^1-norm as local distance function.
3. Use D as input to HDBSCAN* with $m_{min} = 100$.

		Found clusters		
Similarity	m_{min}	noise	0	1
DTW	100	10000		
LCSS ($\epsilon = 10$)	100	10000		
LCSS ($\epsilon = 15$)	100	10000		
LCSS ($\epsilon = 20$)	100	10000		
DTW	20	10000		
LCSS ($\epsilon = 10$)	20	4617	5241	142
LCSS ($\epsilon = 15$)	20	3833	5933	234
LCSS ($\epsilon = 20$)	20	8920	504	576

Table 4: Clustering results for DTW and LCSS.

Unfortunately, neither case led to significant results since every sample has throughout been classified as outlier. Gradually decreasing m_{min} to 20 reveales a few clusters for LCSS. But the notion of a single dominant cluster with the rest being mostly outliers poses no real improvement over the former clusterings, which seems to be confirmed by the dendrogram for LCSS with $\epsilon = 15$ that suggests no real clusters (see Fig. 26 and Table 4).

6.2.3 Frequency-based similarity measures

In contrast to DTW and LCSS, the frequency domain offers a representation that abstracts from the actual signal and can therefore give more insight into what constitutes the data from a generative point of view without suffering from rotation or shift variance.

Fourier transform For each sample, the frequency spectrum has been computed independently along all 3 axes. Since FFT expects as input size a power of two, each sample has been zero-padded to 512 time steps. Although the FT does not suffer from shift or rotation variance, it is only ideal for signals with full periodicity, meaning that the sudden jumps at the end points of the signal cause artificial discontinuities that may exceed the Nyquist frequency and thus affect the spectrum negatively, called aliasing. To mitigate this effect, it is common practice to pass the signal through a *window* function weighting the signal such that the end points become continuous. For this thesis, the Hanning window has been chosen (see Fig. 27).

For similarity, the Fourier spectrums have been compared using the L^1-norm. In addition, the fractional $L^{0.5}$-norm has been used. Since an input size of 512 results in a spectrum with 257 values and thus $3 \times 257 = 771$ values in

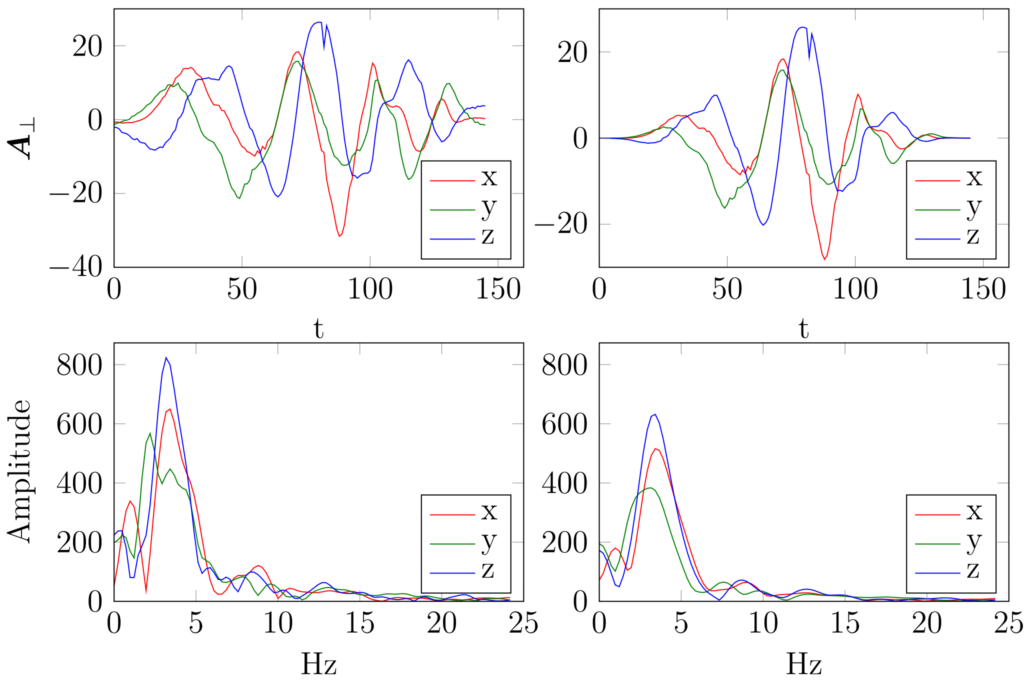


Figure 27: Anti-aliasing: enforcing periodicity on a signal using the Hanning window function. The original signal and frequency spectrum are on the left.

total, the spectrums are very high-dimensional and have therefore been passed through a PCA before clustering. Constraining the PCA with $R^2 \geq 0.90$ led to a dimensionality of 12 features that in total explained $R^2 = 0.91$ of the variance. This seems surprisingly few but can most likely be explained by the rather large amount of frequencies that do not contain much information (see. Fig. 27).

Stationary wavelet transform As mentioned earlier, DWT is not shift-invariant. In order to use energy coefficients as basis for similarity, the stationary wavelet transform (SWT) will be used instead. It tries to handle shift variance by eliminating decimation steps during the filter bank-process and use upsampling of energy coefficients instead (see [34] and Fig. 28). As downside, the energy coefficients become highly redundant and will therefore be passed through a PCA first before measuring similarity with L^1- and $L^{0.5}$-norm.

Because 1D wavelet transform works for 1D signals only, the energies of each axis are summed up to give a single set of energy coefficients. Again, PCA is used with an information preservation of at least 90% leading to 25 dimensions with $R^2 = 0.90$. This increased number of dimensions is due to an increased input size of $9 \times 512 = 4608$ values.

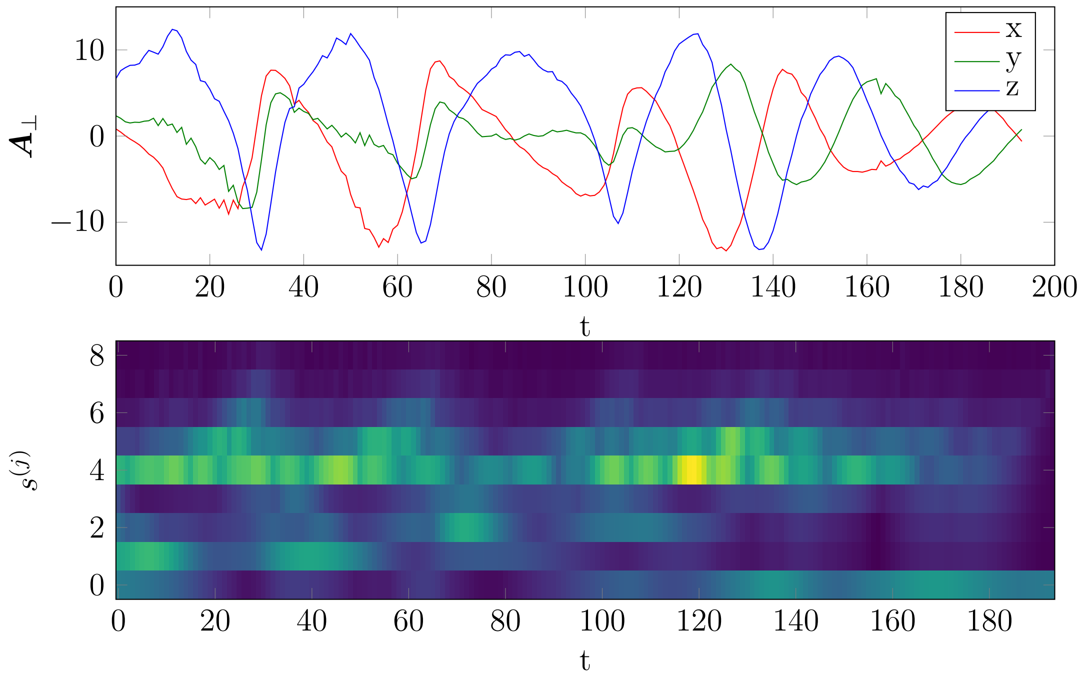


Figure 28: Detail coefficients of a vortex sample using SWT.

			Found clusters		
Similarity	L^1 / $L^{0.5}$	m_{min}	noise	0	1
FT	L^1	100	10000		
SWT	L^1	100	10000		
FT	$L^{0.5}$	100	10000		
SWT	$L^{0.5}$	100	10000		
FT	L^1	4	110	9881	9
SWT	L^1	4	3419	6569	12
FT	$L^{0.5}$	4	114	9878	8
SWT	$L^{0.5}$	4	5539	4450	8

Table 5: Clustering results for Fourier and stationary wavelet transforms.

However, both clustering results look identical to the previous outcome (see Table 5). In case of a very low $m_{min} = 4$, FT exhibits a very dominant cluster with almost no outliers, which nevertheless should be regarded as no cluster at all. On the other hand, SWT shows very similar results to LCSS. Either way, these clusterings are more or less not to be taken seriously.

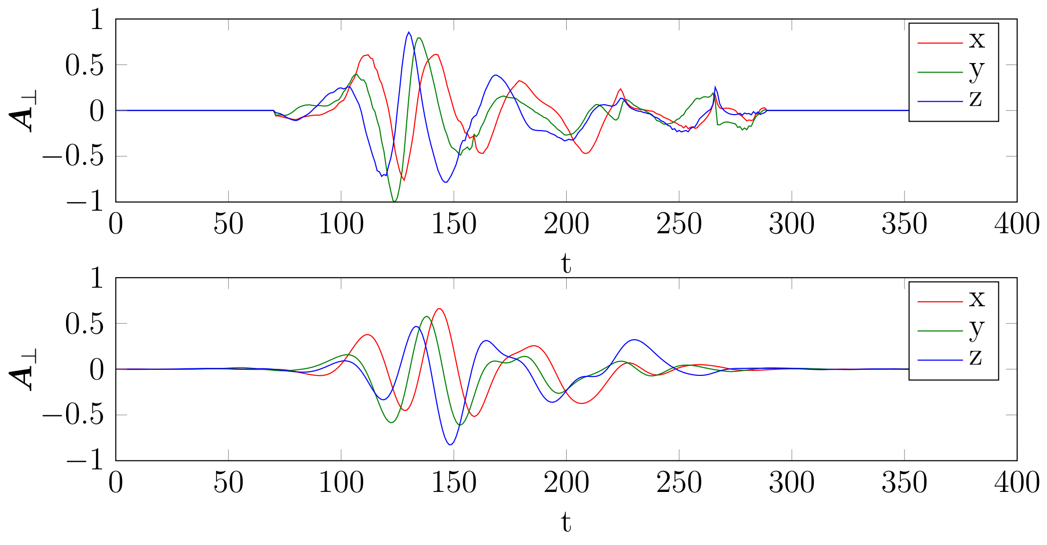


Figure 29: Reconstruction (bottom row) of a vortex signature (top row) using the convolutional sparse autoencoder.

6.2.4 Convolutional sparse autoencoder

The ability to learn a generative model of the data is a great advantage of sparse autoencoders and especially interesting for unsupervised data analysis since it delivers a description of the processes that explain the data. As a convenient side effect, it also allows for data compression along the way, thus mitigating the effects of the curse of dimensionality. For this to be effective, an appropriate design is important. However, as with most neural networks, design choices are mostly subject to experimentation. Fortunately, the task at hand already gives certain design directives that help with the decision process. For instance, it is important to find a model that is low in dimension, i. e. has as few neurons as possible, but at the same time is as expressive as possible. Unfortunately, it is difficult to have both, making a trade-off inevitable. In other words, the design objective is to reduce the size of the encoding layer until R^2 falls below a certain threshold.

The final architecture of the network used for this thesis is comprised of an encoding step consisting of two pairings of a 1D-convolutional layer with 32 filters of size 7 followed by a max-pooling layer with pooling size 2, i. e. the input size is halved after each convolutional layer. It follows with a rather deep fully-connected network consisting of a total of 5 dense layers and an encoding layer with 32 neurons. The network is then mirrored for the decoding step. To ensure sparsity, the activations (hyperbolic tangent) are penalized with the KL-divergence of the encoding step. Since training did not show any

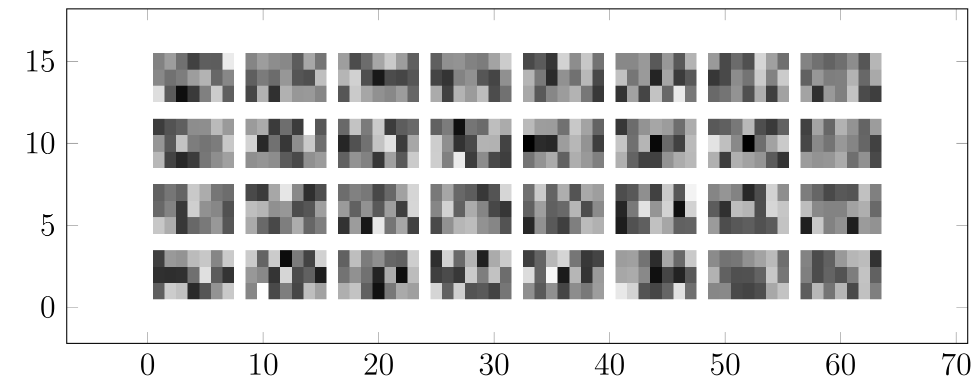


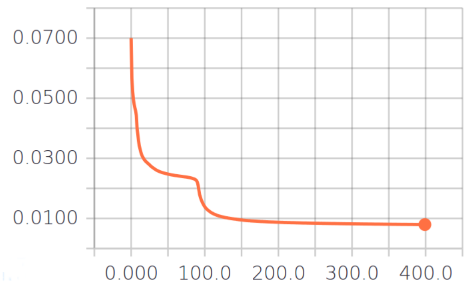
Figure 30: Visualization of the top convolutional layer's weights, normalized into range $[0, 255]$.

signs of overfitting (see Fig. **??**), other regularization methods have not been considered.

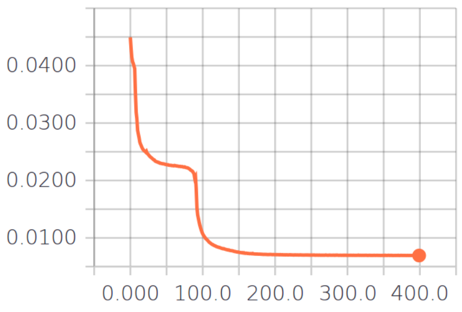
For training, each sample is zero-padded to a size of 360 time steps and normalized to $[-1, 1]$ to comply with tanh-activations. At an encoding size of 32 neurons, this corresponds to a compression of around 97% since each input consists of $3 \times 360 = 1080$ values. All the more surprising, after training the network for 400 epochs with a mini-batch-size of 100 samples, roughly 89% of the information has been preserved by the encoding layer. Visual inspection shows a reasonable reconstruction with some smoothing effects at the borders, which are most likely artefacts from zero-padding (see Fig. 29). Fig. 30 depicts all 32 filters of the top convolutional layer.

For cluster analysis, the data is transformed using the encoding layer's output. As distance measure, the L^1-norm is used. Since 32 neurons are still fairly high-dimensional, the fractional $L^{0.5}$-norm is applied as well. Albeit promising training results, clustering again does not show significant structures for both distance metrics, as depicted in Table 6.

As an alternative to the convolutional approach, a traditional fully-connected sparse autoencoder has been used as well to see if significant differences might show in the results despite theoretical considerations. Since fully-connected layers do not support multivariate data, an autoencoder for each spatial axis has been used with 7 layers for encoding and 6 layers for decoding. Each encoding layer has 16 neurons, which gives 48 neurons in total for all 3 axes. As before, sparsity is ensured by KL-divergence as penalty enforced on the activations during the encoding step. Thanks to the reduction in axis and extra capacity, information has been preserved by up to 93%. However, the increase in accuracy has no consquences for the cluster analysis since results look identical (see Table 6).



(a) Training error of the training set.



(b) Training error of the validation set.

			Found clusters			
Similarity	L^1 / $L^{0.5}$	m_{min}	noise	0	1	2
CSAE	L^1	100	10000			
AE	L^1	100	10000			
CSAE	$L^{0.5}$	100	10000			
AE	$L^{0.5}$	100	10000			
CSAE	L^1	10	9893	578	29	
AE	L^1	10	8893	1063	44	
CSAE	$L^{0.5}$	10	9507	469	24	
AE	$L^{0.5}$	10	9264	33	681	22

Table 6: Clustering results for (convolutional) sparse autoencoders.

7 Conclusion and discussion

Considering the true negative rate, the proposed detection algorithm based on discrete wavelet transform seems to work remarkably fine with rates of more than 99% regarding the manually labeled test set. Unfortunately, the algorithm performs considerably worse with respect to true positive and false detection rates. This notion is particularly important for this thesis since non-vortex labels make up more than 97% of the test data. With TPRs below 50%, results are but unsatisfactory. This does not automatically mean one was better off guessing since the algorithm still does well with respect to the true negative rate. But it clearly points out that the approach has distinct weaknesses. As mentioned before, border sensitivity does play a small role since results improve to more desirable FDRs around 20% and TPRs of 60% and the approach works best for strong and relatively high-frequency signals that span a certain amount in time. However, if the signals are very subtle or rather low in frequency, the algorithm more or less fails reliably. Furthermore, the algorithm struggles with short bursts that do not show themselves as vortices in spatial data but are often misclassified as such.

Only in cases for which the theoretical assumptions hold that have been laid-out with respect to the anticipated acceleration patterns does the algorithm manage to perform well. The main problem seems to remain that it does not concern itself with the trajectory's true geometry but rather relies on assumptions which do not seem to generalize well. This way, small vortices with subtle acceleration patterns are mostly hidden to the algorithm.

Looking at the clustering results, a similar situation presents itself. None of the proposed methods were able to reveal significant structure within the data, which can mean two things: either there truly is no distinct structure to be revealed since the examined data does already describe a single phenomenon, or the methods fail to capture meaningful characteristics and thus prove inappropriate. However, given the estimate that roughly 20% of the classifications are expected to be true positives one may have legitemately hoped for at least a clear distinction between clusters and outliers. Since this did consistently not show in the results, it is to be assumed that in conjunction with the proposed normalization the methods lack the necessary expressiveness with respect to vortex data.

All in all, a key problem seems to be the abstraction from geometrical space, i. e. the attempt to describe a phenomenon observable mainly in geometrical space by looking at theoretically anticipated patterns in another representation. Evidently, this attempt only succeeds in situations favoured by the theoretical assumptions themselves but significantly lacks the ability to generalize to other situations. There is simply no mathematically sound, i. e. geometrical way to verify if a vortex has truly been observed. In other words, the hypothesis does not make its decision based on the criterion used by the expert to distinctly describe the phenomenon itself.

7.1 Future work

Given the results, it seems obvious that in order to reliably detect structures of significant interest in 3D fluid turbulence other means seem to be necessary. One possibility to tackle the problem could be to use conventional supervised methods to train classifiers based on labeled training data. Naturally, this involves painstakingly labeling a subset of the data by hand, which can be lengthy and tedious. But given the rich pool of supervised classification algorithms, e. g. support vector machines, neural networks, and others it could be well worth the effort, especially in consideration of the broad success supervised methods have had in the past decade. Another possibility could be to go back to geometrical space and test novel solutions with respect to robustly fitting the axis of rotation using e. g. splines.

On the other hand, maybe there is justified reason to why the proposed methods did not produce the results one was tempted to hope for. There exists a plethora of different techniques and algorithms dedicated to signal processing and pattern recognition and only few have been considered for this thesis. Therefore, the possibility cannot be ruled out that other methods will perform much better.

Bibliography

[1] Dieter Biskamp. *Magnetohydrodynamic Turbulence*. Cambridge University Press, 2003.

[2] Clay Mathematics Institute. *Navier-Stokes Equation*. 2016. URL: `http://www.claymath.org/millennium-problems/navier-stokes-equation`.

[3] P.A. Davidson. *Turbulence: An Introduction for Scientists and Engineers*. OUP Oxford, 2007.

[4] G. Batchelor. *Opening address at the First European Turbulence Conference*. 1986.

[5] O. Reynolds. "An experimental investigation of the circumstances which determine whether the motion of water shall be direct or sinuous, and the law of resistance in parallel channels". In: *Philosophical Transactions of the Royal Society* 174 (1883), pp. 935–982.

[6] L. Richardson. *Weather prediction by numerical process*. Cambridge University Press, 1922.

[7] A. Busse. "Lagrangesche statistische Eigenschaften hydrodynamischer und magnetohydrodynamischer Turbulenz". PhD thesis. University of Bayreuth, 2009.

[8] J.W. Cooley and J.W. Tukey. "An algorithm for the machine calculation of complex Fourier series". In: *Mathematics of Computation* 19 (1965), pp. 297–301.

[9] J. Fröhlich. *Large Eddy Simulation turbulenter Strömungen*. Teubner, 2006.

[10] J. Jeong and F. Hussain. "On the identification of a vortex". In: *Journal of Fluid Mechanics* 285 (1995), pp. 69–94.

[11] M. Jiang and D. Thompson. "Detection and Visualization of Vortices". In: *Visualization Handbook*. 2005, pp. 295–309.

[12] G. Haller and F.J. Beron-Vera. "Coherent Lagrangian vortices: The black holes of turbulence". In: *Journal of Fluid Mechanics* (2013).

[13] L. Zhang. "Boosting Techniques for Physics-Based Vortex Detection". In: *Eurographics Conference on Visualization* 32 (2013).

[14] R. Cucitore, M. Quadrio, and A. Baron. "On the effectiveness and limitations of local criteria for the identification of a vortex". In: *European Journal of Mechanics - BFluids* 18 (1999), pp. 261–282.

[15] G. Haller. "An objective definition of a vortex". In: *Journal of Fluid Mechanics* 525 (2005), pp. 1–26.

[16] H.J. Lugt. *Vortex Flow in Nature and Technology*. Wiley, 1972.

[17] S.K. Robinson. "Coherent Motions in the Turbulent Boundary Layer". In: *Annual Review of Fluid Mechanics* 23 (1991), pp. 601–639.

[18] M. Jiang and D. Thompson. “Geometric verification of swirling features in flow fields”. In: *Proceedings of the conference on Visualization* (2002), pp. 307–314.
[19] J. Fourier. *Théorie analytique de la chaleur.* 1822.
[20] R.N. Bracewell. *The Fourier Transform and its Applications.* Stanford, 1986.
[21] National Radio Astronomy Observatory. *Fourier Transforms.* 2016. URL: `https://www.cv.nrao.edu/course/astr534/FourierTransforms.html`.
[22] E. Jacobsen and R. Lyons. “The sliding DFT”. In: *Signal Processing Magazine* 20 (2003), pp. 74–80.
[23] C. Valens. “A Really Friendly Guid to Wavelets”. In: (1999).
[24] N. Hugget. *Zeno’s Paradoxes: 3.2 Achilles and the Tortoise.* 2010. URL: `https://plato.stanford.edu/entries/paradox-zeno/#AchTor`.
[25] S. Mallat. *A wavelet tour of signal processing.* Academic Press, 1999.
[26] I. Daubechies. *Ten Lectures on Wavelets.* SIAM, 1992.
[27] A. Haar. “Zur Theorie der orthogonalen Funktionensysteme”. In: *Mathematische Annalen* 69 (1910), pp. 331–371.
[28] JPEG. *JPEG 2000.* 2016. URL: `https://jpeg.org/jpeg2000/`.
[29] H. Sakoe and Chiba. “Dynamic programming algorithm optimization for spoken word recognition”. In: *IEEE Transactions on Acoustics, Speech and Signal Processing* 26 (1978), pp. 43–49.
[30] X. Wang et al. “Experimental Comparison of Representation Methods and Distance Measures for Time Series Data”. In: *Data Mining and Knowledge Discovery* 26 (2013), pp. 275–309.
[31] M. Müller. *Information retrieval for music and motion.* Springer, 2007.
[32] Daniel Lemire. *Faster Sequential Search with a Two-Pass Dynamic-Time-Warping Lower Bound.* 2008. URL: `https://arxiv.org/abs/0807.1734`.
[33] D. Bakkelund. “An LCS-based string metric”. In: (2009).
[34] M.J. Shensa. “The Discrete Wavelet Transform: Wedding the A Trous and Mallat Algorithms”. In: *IEEE Transaction on Signal Processing* 40 (1992).
[35] R. Rójas. *Neural Networks - A Systematic Introduction.* Springer, 1996.
[36] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. “Deep Learning”. Book in preparation for MIT Press. 2016. URL: `http://www.deeplearningbook.org`.
[37] P. Baldi and K. Hornik. “Neural networks and principal component analysis: Learning from examples without local minima”. In: *Neural Networks* 2 (1989), pp. 53–58.
[38] swarbrickjones. *Convolutional autoencoders in python/theano/lasagne.* 2016. URL: `https://swarbrickjones.wordpress.com/2015/04/29/convolutional-autoencoders-in-pythontheanolasagne/`.

[39] C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[40] R. Tibshirani, G. Walther, and T. Hastie. “Estimating the number of clusters in a data set via the gap statistic”. In: *Journal of the Royal Statistical Society: Series B* 63 (2001), pp. 411–423.

[41] R. Campello et al. “Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection”. In: *ACM Transactions on Knowledge Discovery from Data* 10 (2015).

[42] L. Mcinnes, J. Healy, and S. Astels. *How HDBSCAN works*. 2016. URL: `https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html`.

[43] R. Campello et al. “A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies”. In: *Data Mining and Knowledge Discovery* 27 (2013).

[44] C. Aggarwal, A. Hinneburg, and D. Keim. “On the Surprising Behavior of Distance Metrics in High Dimensional Space”. In: *Database Theory*. Vol. 1973. 2001, pp. 420–434.