Machine Learning for Coherent Structure Identification and Super Resolution in Turbulent flows

Sahil Kommalapati

A thesis
submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

University of Washington

2021

Committee:

Owen J. H. Williams

Steven L. Brunton

Brian L. Polagye

Program Authorized to Offer Degree:
Mechanical Engineering

University of Washington

Abstract

Machine Learning for Coherent Structure Identification and Super Resolution in Turbulent flows

Sahil Kommalapati

Chair of the Supervisory Committee:
Research Assistant Professor Owen J. H. Williams
Aeronautics & Astronautics

Particle image velocimetry (PIV) techniques provide high fidelity measurements of mutliscale turbulent fluid motion. The research presented in this thesis broadly explores the utilization of Machine Learning for processing and analysing non-time resolved PIV measurements of turbulent flows. The primary goal is to utilize bayesian inference to automate turbulent coherent structure identification in boundary layer flows. Automated identification was hitherto hindered by the lack of a priori information about the convecting velocities of the coherent structures. A Rankine vortex model is utilized to implement a Markov chain Monte Carlo based vector matching to overcome this problem. Ultimately, a framework was developed for robust identification of vortices that is capable of visualizing the cumulative distributions of properties of all vortex structures in the flow to provide a rich description of multiscale turbulent coherent structures. Additionally, Bayesian inference is proven to outperform traditional optimization methods based on various loss metrics. The other major goal of this thesis is to implement super resolution in turbulent separated flows using neural network architectures. Machine learning based super resolution using PIV measurements is relatively unexplored, compared to its counterpart with CFD simulation. This is partly due to the unavailability of training datasets with sufficient magnitude. A large PIV dataset of non-time resolved measurements of turbulent separated flow with 50,000 snapshots was col-

lected to overcome this problem. A neural network architecture was trained on this dataset to successfully outperform traditional interpolation based super resolution techniques. Finally, it has been shown that the current approach also accurately reproduced the properties of turbulent derived quantities, like the stream wise variance of velocity fluctuations, with better accuracy in comparison with interpolation based super resolution methods.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

W: Weighting function

A_W: Area of the W

ζ: Weighted mean square difference error

δ: Boundary layer thickness ($0.1275m$)

Γ: Circulation strength

R: Radius of the vortex core

X_C: X-position of the center of the vortex core

Y_C: Y-position of the center of the vortex core

V_c: Convective velocity of the vortex

θ: (Γ, R, X_C, Y_C, V_c)

Γ_2: Vortex localization operator

V_θ: Tangential velocity at a point

σ: Gaussian noise variance scaling factor

ΔX: Grid width, between vectors in PIV measurements

Re_θ: Momentum thickness based Reynolds number

H: Height of the Gaussian speed bump ($0.0766\ m$)

U: Stream-wise velocity field

V: Wall-normal velocity field

Re_L: Reynolds number based on tunnel width

Re_h: Reynolds number based on Gaussian bump height

$\tilde{u}$: Mean velocity profile of all stream-wise velocity fields

u': Stream-wise velocity fluctuations

u'_n: Normalized stream-wise velocity fluctuations

$\|.\|_F$: Frobenius norm

$\|.\|_2$: 2-norm

$\mathcal{L}_{SR}$: Forbenius norm based percentage difference loss estimate

$\mathcal{L}_2$: 2 norm based difference loss estimate

IR^N: N dimensional vector

α: Learning rate for updating the weights in a neural network

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Owen Williams, for guiding me in effectively navigating through the intriguing and challenging world of turbulence for the past two years. It was an exceptional opportunity to learn from Dr. Williams and this work would not have been possible with out his support and patience. I am also grateful to be advised by Dr. Jake VanderPlas, Dr. Steve Brunton and Dr. Brian Polagye, who provided crucial insights towards developing this work.

I am thankful to the current and past lab members at the Williams turbulence lab, for their constant support and feedback that helped me to become a better researcher and collaborator.

I would like to thank Bob Herbold, the mechanical engineering department and the college of engineering at the University of Washington for supporting my research through the Herbold data science fellowship. I would also like to acknowledge the William E. Boeing department of Aeronautics and Astronautics and the Paul G. Allen school of computer science and engineering for supporting me through graduate student assistantships.

Finally, I am incredibly grateful to my family and friends for their unconditional love and support, with out which this thesis could not have been completed.

DEDICATION

In the memory of Pedarla Narasimharao.

Chapter 1

INTRODUCTION

Turbulence is a ubiquitous phenomenon found in all facets of life and supports the development of a plethora of engineering applications. It is crucial to understand robust propulsion and combustion systems, turbines, passive and active mixing devices. Understanding the behavior of turbulent boundary layers provides critical insights into developing more aerodynamic and hence efficient automobiles, spacecraft, and rockets.

Turbulence is of particular interest to the scientific community owing to the time dependant and chaotic nature of its evolution over a wide range of scales. One of the clear signatures of turbulent flows is the effective mixing of all fluid properties. This is not only restricted to mass, but also momentum, energy, temperature, velocity and so on. Turbulence is also characterised by irregularity and chaos, which prompts problems in turbulence to be dealt statistically rather than deterministically. Turbulent flows have non-zero viscosity and are characterized by three dimensional vortex stretching. The multi-scale nature exhibiting vortices or *eddies* at different length scales acts an clear sign in identifying turbulent flows.

Turbulence is governed by the Navier-stokes equations, which have been known for the majority of the past two centuries. These are a set of partial differential equations (PDEs) that govern the motion of fluid particles, based on Newton's second law of motion. However, the closed form solution to this set of equations is yet to be discovered. To be more precise, it is unknown if there exists a smooth[1] and globally defined solution which could solve the aforementioned set of PDEs. This adds to the overall incompleteness in the theoretical understanding of turbulence.

---

[1] differentiable everywhere

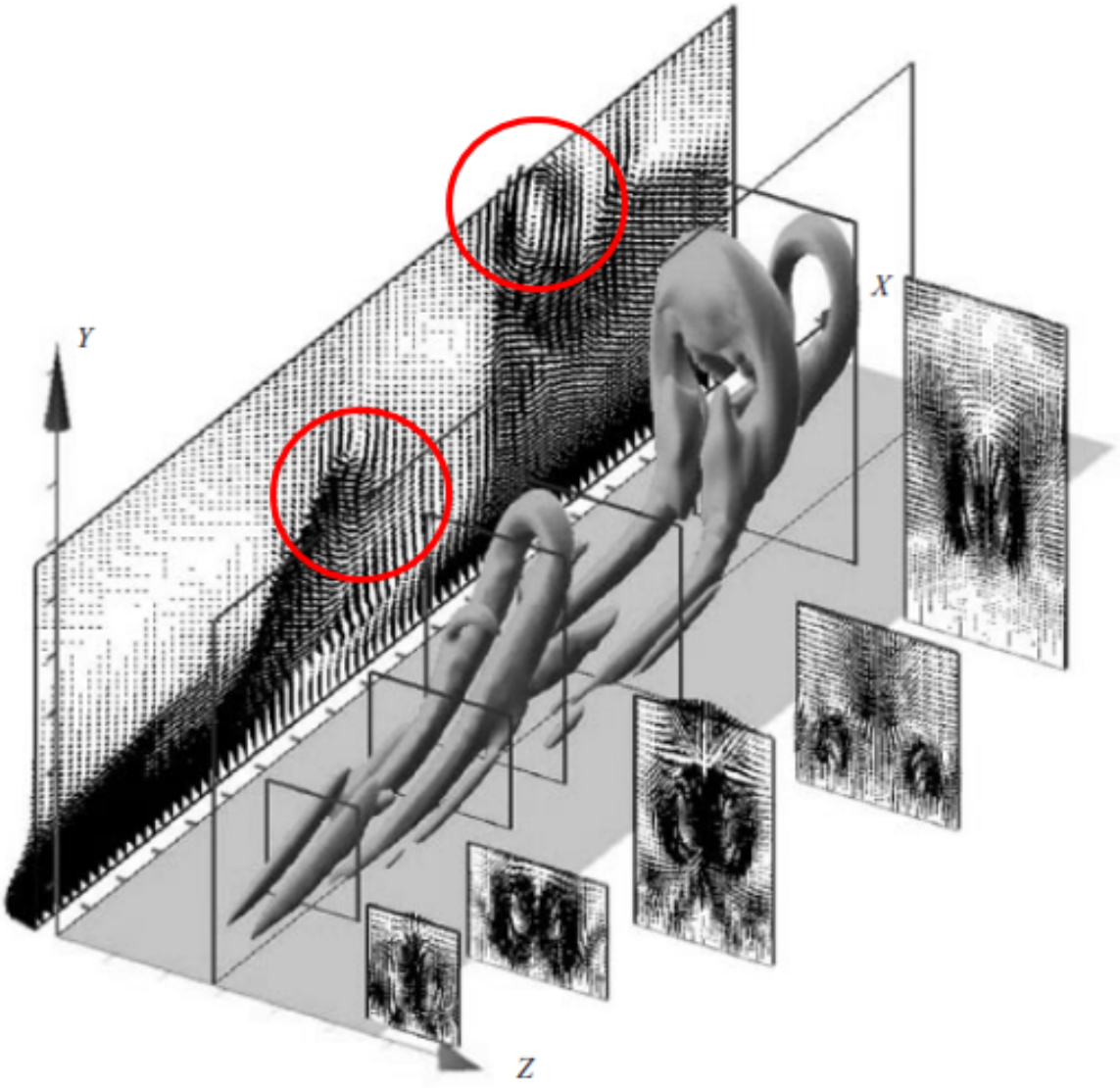


Figure 1.1: Hairpin vortex signatures observed in PIV measurements along stream wise - wall normal plane [1].

One of the fundamental aspects of scientific thinking is breaking down a complicated phenomenon into its elementary units and understanding their behavior to draw conclusions about the overall phenomenon. Analogously, to study turbulence, it is broken down into smaller and fundamental units, upon which further analysis is executed. These fundamental

units are called as eddies or coherent structures. These are structures that maintain individual coherence, or 'togetherness', and through their interactions, they make the underlying physics highly non-linear are difficult to predict. They are also responsible for transporting momentum and generating turbulent kinetic energy[1].

The mass and momentum mixing capabilities of vortices are pivotal to understand the onset of dynamic stall [3], turbulent combustion in diesel engines [4] and passive turbulent fluid mixing devices[5] and many other engineering designs.

Wall bounded turbulent flows are of critical importance because of their ubiquity in a large number of practical applications. In such flows, coherent structures are observed as horseshoe or hairpin-shaped structures typically with their base attached to the wall. In order to qualitatively analyze the properties of a turbulent flow field, it is necessary to understand the properties of the coherent structures that generate turbulence in the first place. The central focus of this current work is to develop computational tools to help in studying turbulent coherent structures. The complexity of flow fields and the high dimensional nature of modelling coherent structures make this a challenging task.

The current work focuses on addressing these challenges and broadly strives to further the theoretical understanding of turbulent flow behavior and its measurement. The primary goal is to accurately identify the properties of turbulent coherent structures from Particle image velocimetry (PIV) measurements. A second broad objective is to understand how high performance computing and deep learning based methods can help in learning the mapping from low resolution measurements to high resolution measurements of turbulent flows using super resolution (SR) techniques. The emphasis is now shifted to more closely look at both of these objectives.

Coherent structure identification is a long-standing problem in fluid mechanics which is crucial to numerous theoretical and practical applications such as turbulence modelling, flow control and understanding the production and dissipation of turbulence[6]. Efforts often focus on the identification of hairpin vortices[2, 1], which are the most ubiquitous structures in turbulent flows.

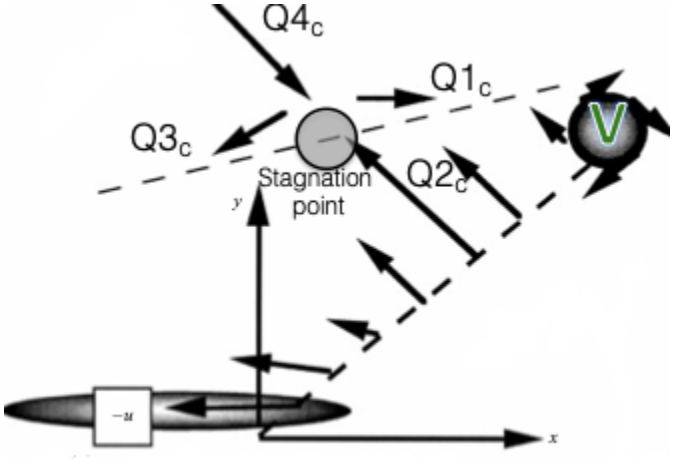


Figure 1.2: Cross section of a Hairpin vortex.[2] The field is composed of a vortex (V) and an inclined stagnation point, which is the signature of a hairpin vortex.

Hairpin vortex structures are high-dimensional coherent structures that organize as packets [2, 1] and are responsible for the production of turbulence in wall bounded turbulent flows. They evolve in three dimensional flows, but their signatures can still be observed in two dimensional slices, for example, using Particle Image Velocimetry (PIV) as seen in Fig. 1.1. Here the signatures are highlighted in red. This signature is a combination of a prograde vortex[2] and a stagnation (saddle) point as seen in Fig. 1.2. Robust vortex identification techniques are required for the preliminary identification of these signatures, which could later be integrated with saddle-point identification.

There have been several approaches that were proposed in literature[7] to robustly identify vortex properties. Adrian [2] proposed a manual procedure for identifying vortices in PIV velocity vector fields by subtracting the convective velocity of the structure and observing circular motions in an manual iterative, visual manner. However, the lack of prior knowledge of convective velocities and the high dimensional nature of the problem has restricted an automated identification of vortices.

[2]A prograde vortex is defined by a clock wise rotation of the hairpin head and a positive swirl field

These drawbacks in identifying vortex properties from non-time resolved PIV measurements[3] have prevailed for the past two decades. Recent works[9] have shown promise in identifying the convective velocity by capturing the surface pressure values over the wall boundary. These values measure the motion (translation) of the global minimum over multiple frames using Machine Learning and estimate the convective velocity of the vortex. However, this method entirely fails when considering multiple vortices that occur in the same frame.

This thesis research overcomes these barriers and additionally proposes to do it using measurements from a single snapshot of PIV velocity vector fields of boundary layer flow. The ultimate goal of is to enable automated identification of vortex properties, including convective velocities from single velocity field snapshots. This serves as the first step in the development of a general framework for automated turbulent coherent structure identification and subsequent extraction of their properties.

Bayesian Inference has proven to be very effective for solving optimization problems that are too expensive to evaluate [10] or for solving problems that are critically sensitive to measurement noise or the number of modelled parameters[11].

As a result, the current work explores a novel framework utilizing Bayesian optimization with Markov chain Monte Carlo as the resource to achieve the goal of automated identification of vortex properties, without any prior knowledge, including the convective velocities of the vortices.

Current work also focuses on testing the robustness of Bayesian inference against various factors. This includes the addition of external noise in measurements, sensitivity of various optimization parameters, sensitivity due to initial conditions, sensitivity towards the resolution of the measurement and also the overall performance comparison against classical non-Bayesian optimization techniques.

---

[3]It should be noted that finding the localizing vortex structures and their convective velocities is relatively easy if time resolved PIV data is available[8]. However measuring time resolved data is usually prohibitively expensive.

The second major objective of this current work is the super resolution (SR) of turbulent PIV measurements using Machine Learning (ML). Super resolution refers to any method of image processing that increases the resolution of the images. SR is currently employed in the context of turbulent PIV flow measurements. The goal is to provide real time online conversion from computationally inexpensive simulations to high fidelity multi-scale simulations that can guide further analysis.

In the context of turbulence, capturing high fidelity flow measurements either through numerical simulations or through experimental set ups, is prohibitively expensive. So, SR proposes an alternative to capture low fidelity measurements and later enhance them at a subsequent stage [12, 13].

It is certainly difficult to develop SR methods for natural images like photographs. However, it is more challenging while handling fluid simulations. The multi-scale nature of turbulence can be largely attributed to complex nature of generation and interaction between coherent structures of a range of sizes. This multi-scale nature introduces a lot difficulties while analyzing and modifying turbulence data using traditional data science tools.

Classic SR algorithms (a.k.a interpolation methods) are based on linear and polynomial interpolation of neighbouring pixels. The notable ones include linear, bicubic and quintic interpolations. These methods perform poorly for large scale upsampling and do not take multi-scale data into account. As long as the Kolmogorov[4] length scale are not reached, between any two pixel there are rich and intricately evolved sub-scales that simply cannot be captured by smoothing or interpolation based methods without filtering of turbulent energy and fluxes.

In the current work, artificial neural network (ANN) architectures are trained on large PIV datasets[5] for super resolution. This model is finely tuned, trained and the ultimately the results of SR reconstruction are compared with classical linear, bicubic and quintic interpolation methods. One of the goals of SR research is to understand the extent to which

[4]smallest scale in a turbulent flow

[5]50,000 non-time resolved PIV snapshots of turbulent flow; 43 GB.

the magnitude of the training dataset effects the ANN based super resolution reconstructions. In addition to measuring the reconstruction accuracy for instantaneous velocity fields, the ability for SR to reconstruct true turbulent statistics is also examined.

Chapter 2
BACKGROUND & THEORY

The significance of coherent structures in generating turbulence[1, 2, 6] and its importance in various engineering applications[3, 4, 5] have been introduced earlier. This chapter focuses on shedding more light on the nature of coherent structures and the importance of identifying them. An overview of the extensive literature that focuses on this problem, their short comings and finally how this current research overcomes these short comings is narrated. Broadly, the problems with previously existing coherent structure identification and super resolution techniques are addressed and Bayesian inference and deep neural networks are introduced with the goal of proposing these methods as better alternatives.

2.1 Coherent structures in Turbulence

The continuity equation ensures that all fluid motion is spatially coherent to a certain degree. So, coherent structures emphasize and refer to local regions in flow that are not just spatially, but also temporally coherent for significant enough time periods. Coherent structures in turbulent flows can significantly influence the time averaged statistics and play a crucial role in the turbulence generation mechanism[1].

Current work focuses on wall-bounded turbulent flows that are referred to as the most common, the most important and the most complicated form of fluid motion [14].

These hairpin vortex structures were first identified by Theodersen[15] who described a horseshoe or a hairpin shaped vortex. Since his first observations, the experimental and numerical capabilities have come a long way in enabling high fidelity measurements to understand the structure and evolution of these hairpin vortices. This would be comprised of smaller scaled hairpin structures to showcase a hierarchy of scales in the flow. (see Fig. 1.1)

This multi-scale nature of hairpin vortex structures and their recursive vortex generation phenomenon is beautifully summarized in this short poem by Lewis Fry Richardson, while talking about the supply of energy from and to atmospheric eddies[16].

"Big whirls have little whirls, That feed on their velocity; And little whirls have lesser whirls, And so on to viscosity."

Hairpin vortex structures are inherently multi-scale three dimensional structures. However, the two dimensional cross-sectional (Fig. 1.1) view along the plane normal to the span-wise direction provides a lot of information. The cross section of the hairpin vortex along this plane would simply look like a vortex. The entire hairpin vortex, on the other hand, can be modelled as a combination of vortex and a saddle point.

More details about the vortex modelling can be found in Chapter 3. Measurements along this (or any other) cross-sectional plane can be acquired by using Particle Image Velocimetry (PIV) techniques, which have stood the test of time in robust measurement of turbulent fields. Current work utilizes a turbulent boundary layer dataset that has been collected using PIV. Further details about the PIV setup and the flow conditions are detailed in Chapter 4.

Given the critical nature of these structures, it is highly desirable to acquire the properties of hairpin vortex and other coherent structures. These properties provide much more complete and informed view of the flow compared to stand alone instantaneous velocity and pressure fields. However, identifying coherent structure properties is not a trivial task owing to the multi-scale and chaotic nature of turbulent flow fields. Extensive literature can be found that focuses on this objective with varying degrees of success and it is still an active field of research.

It would be a mammoth undertaking to identify the properties of various coherent structures in a generalized manner. This thesis work focuses on developing a framework to identify vortex properties from single velocity fields, which would be the first step in identifying the properties of a hairpin vortex structure. Through out this development, the scalability of this framework was kept in mind, to ensure to extension towards identification of other coherent coherent structures with minimal efforts. However, this is yet to tested to make qualitative

conclusions.

The next section introduces one of the most successful approaches to identify vortex properties from PIV velocity vector fields. Then, the major drawbacks of this methods and other alternatives are also explained.

2.2 Other vortex identification methods.

Extracting vortex properties mandates that we have prior knowledge of at least a vague idea about the location, the radius and most importantly the convective velocity of the vortex core. Once, these parameters are known, the flow field can be localized and more accurate values of these, as well as, other vortex properties can be found by matching the field with some vortex model. The location and the radius of the vortex core can be vaguely extracted using derived turbulent quantities like swirling strength[17], helicity[18], eigen-heliciy[19], vorticity[20] or a plethora of other criterion[7]. However, the evaluation of the convective velocity of the vortex candidate is more difficult and can only be realized through convoluted approaches.

In the landmark paper, Adrian et al.[2], provide invaluable insights into studying turbulent flow fields. The Reynolds decomposition of a turbulent flow fields separates the mean velocity profile from the velocity fluctuations to give a more invariant view of flow. However, Adrian et al. write that it is more beneficial to observe the flow in different Galilean frames of reference that follows a vortex as they provide more physical representations of the flow. They extend this notion to pride a simple rule to identify the vortex core. In a frame of reference moving with the convective velocity of the vortex core, the core structure can easily be identified. In a PIV vector field this would be visible as group of vectors rotating about a point, the most intuitive representation of a vortex. And, by changing the velocity of the reference frames, the structure of the core becomes more, or less clear. In this way, determining the convective velocity of a vortex is iterative and labor intensive.

This is a simple and robust technique to identify the vortex cores in a flowfield. However, this work assumes the prior knowledge of the convective velocity of the structure. If this

is not know, a series of convective velocities are tested out to see which particular choice visually modifies the flow field to look the most like a vortex. This is not an ideal way to extract the vortex proprties, and can become tiresome even if working with a handful of vortices, let alone entire flowfields. For example, in a PIV snapshot of boundary layer flow, there could be a few hundred hairpin vortex candidates, and each would have its own convective velocity. This points out the major drawback of Adrian's [2] work: it is difficult to be automated. One of the primary goals of current work is to overcome this problem, by automating the entire vortex identification and subsequent extraction of vortex properties in a unified framework.

Before delving deeper into the particular tools and methods used to achieve the set objectives, it would be ideal to mention the current state of extracting the convective velocity of a vortex in an automated fashion. One particularly intuitive approach is to track the vortex core over time and by looking at the spatial displacement and the elapsed time, one could simply calculate the velocity at which the vortex core is travelling. If done using PIV techniques, the measurements would need to be time-resolved to accurately track the motion of the vortex core. This method would also require tools to track the vortex in each frame. Machine learning based approaches have shown promising results in this direction[9]. However, it might not always be possible to measure time resolved data of all processes.

In the current work, a new approach is developed that evaluates the convective velocity from a single PIV snapshot, in an automated manner. This is accomplished using a Bayesian Inference based framework which is the focus of the next section.

2.3 Why Bayesian Inference?

This section discusses the different methods used in developing the vortex Identification framework. The actual framework itself is presented in Chapter 3. In later parts of the thesis, it will become more apparent as to how the process of extracting vortex properties is framed into an optimization problem. The immediate goal, however, is to justify using a Bayesian Inference based approach to tackle this optimization problem and convince the

reader of the advantages in comparison to traditional optimization methods.

There are a plethora of advantages with using a Baysian Inference based optimisation in comparison to traditional approaches. In fact, Baysian optimization methods are a super set of traditional approaches, in the sense that Bayesian optimization can be modified to maximum likelihood estimation through simple modifications. This means that Bayesian optimization has all the advantages of traditional optimization and some additional features, which are exclusive[21].

Most notably, Bayesian Inference provides measures of uncertainty, which are invaluable to judge the quality of the final solutions. It also gives the extent of convergence or divergence for a given run of optimization. This is notoriously in the dark for traditional/ frequentist approach[1]. This is crucial to rectify the bounds of the initial problem set up, which will be shown to be relevant for the vortex identification problem at hand. In the event of multi-modal solutions, they become clearly visible in Bayesian Inference and the choice of solution empowers us to not get stuck in a local minimum.

The most beneficial feature of using Baysian Inference is the ability to embed prior knowledge about the problem set up to nudge the optimization towards global extrema. This feature has unforeseen consequences and needs to be handled carefully. More details on priors are given in Chapter 3. The benefits of using a Bayesian Inference are extensively analysed for both synthetic and real PIV data containing vortices and these results are included in Chapter 3 and 4.

The next section explains a specific form of Bayesian Inference called Markov Chain Monte Carlo (MCMC), that will be the core of the current vortex identification framework.

---

[1]There is long-running historical debate between the Bayesian and Frequentist approaches to reasoning. The fundamental difference is the way in which both schools of thought define probability. For Frequentists, it is the limiting case of repeated measurements and closely related to the frequency of an event (hence the name, *frequentism*). For bayesians, probability covers the degree of certainty about the occurrence of an event based on their own knowledge (which could be updated). This subtle definition in the definition of probabilities leads to significant differences in approaching problems. More often than not, bayesian thinking stays closer to experimental results for answering questions, which motivates the current usage of bayesian optimization. A more detailed account on the difference between these schools of thought can be found in this paper by VanderPlas[22].

2.3.1 Bayesian Inference using Markov chain Monte Carlo

The process of analysing data to capture the underlying probability distribution of a process is called statistical inference. Bayesian inference is one of such methods that utilizes the Bayes' rule to learn the probability distribution of a process. It updates the probability for a prior hypothesis based on newly available data to make better judgements. A key take away from Bayesian inference is that the output of any optimization process is a probability distribution function (PDF) of the variable that is being optimized and a not just scalar value. This PDF in turn, can be used to form an assumption regarding the most likely solution to the optimization problem (say mean, or median of the PDF) with some confidence Intervals. For example, the solution to the function $f(x) = (x-1)^2$, using a Bayesian Inference based methods might produce a PDF for the variable x that is centered around 1, with some variance. More details on analysing the results of Bayesian Inference will be presented in the next subsection. But before that, the focus is brought towards the particular approach used in this work to implement Bayesian Inference is explored, i.e. Markov chain Monte Carlo (MCMC).

A simple introduction to MCMC can be found in the work by Jake Vanderplas [22]. For an exhaustive study, the celebrated textbook on Bayesian Analysis by Gelman[21] and the hand book by Brooks et al.[23] are excellent resources. A non-mathematical introduction to Bayesian philosophy can be found in the popular science book by Lewis[24] .

Markov chain Monte Carlo (MCMC)[21] refers to a class of algorithms used to sample the points of an unknown distribution. A Markov chain is a memory-less stochastic model that describes a sequence of possible events. Monte Carlo methods are random walk methods that are used to for randomly sampling. The idea in MCMC is that, after a large number of points are randomly sampled, the original distribution can be reconstructed by plotting the PDF. MCMC creates Markov chains with the goal that their equivalent distribution is the desired distribution of the underlying process. The differences between various MCMC methods boils down to the mechanism of moving from one Markov state to the next Markov

state in the chain. These are known as Monte Carlo sampling techniques. For example, some of the the most popular Monte Carlo sampling techniques are the Affine Invariant Ensemble MCMC [25, 26], Metropolis-Hastings [27], Gibbs sampling [28] and Hamiltonian MCMC using No-U turn sampling [29]. This current work utilizes the Affine Invariant Ensemble MCMC deployed using the Emcee python module [25]. Compared to its counterparts, Emcee is simpler to implement. The most important input to this module is simply the python function representing the logarithm of the posterior probability. For comparison, PyMC [30] module would require that all the variables are to be defined using PyMC's classes and decorators and PyStan [31] module would require the probability functions to be written in STAN[32] language. To conclude, Emcee was employed for the ease of implementation.

In MCMC, it is a common practise to initialize multiple chains with initial conditions that are sufficiently linearly independent and ultimately taking the ensemble properties of all the states in all the chains. Since, it could take quite a few iterations of the chain to begin converging on the underlying distribution, it also a good practice to ignore the first few steps. This is referred to as the burn-in, which describes the number of initial steps that are to be discarded. The precise number of burn-in steps can be evaluated by manually looking at the chains themselves or also automated by looking at the auto-correlation time and generating the burn-in as some multiple of this time. The auto-correlation time is a good measure to convince oneself regarding the convergence of the chains and is a crucial part for the completeness of any MCMC analysis.

2.4 The super resolution problem in turbulence

Super resolution research involves developing tools and techniques to enhance the resolution of an image or image like systems. Super resolution is of particular interest to the fluid mechanics community, given the extremely high computational expenditure that is required to produce high resolution fluid simulations, both numerically and experimentally. Super resolution based alternatives [13, 12] propose capturing or simulating low-resolution fluid flow motion (say using PIV) and then upscale the sample resolution, thereby reducing the

computational and experimental expenditure.

Experimentally (e.g. with Particle Image Velocimetry, PIV), it would reduce the costs of the camera (lower resolution) and laser (less powerful) set up, and the overall cost of storing and processing the PIV data. This would also reduce the time spent in the pre-processing stage of gathering the PIV information to accelerate the pace of the work. Numerically, with direct numerical simulations, super resolution techniques could reduce the time to resolve all scales of flow in high fidelity simulations or alternatively improve upon the low fidelity simulations. In successful alternatives to DNS, for example with Large Eddy simulations, it is important to model the influence of the smaller scales of flow on the larger scales. Super resolution approaches could possibly provide highly efficient methods to compute the influences of smaller scales for LES with better sub-grid scale models.

The multi-scale coherent structures in turbulent flows add a certain level of complexity towards the super resolution process. This also serves to reinforce the manner in which the overarching theme of studying turbulent coherent structures is maintained throughout this thesis research.

The next section explores the historical and contemporary methods to achieve super resolution. There are few drawbacks associated with the previously existing methods. The goals of the current research are laid out in the final sections of the chapter.

2.5 Previous attempts towards Super resolution for Turbulent flows

Super resolution research emerged from the image processing and computer vision community and has currently spread into a plethora of scientific fields. Apart from SR in fluids and turbulence, extensive literature can be found in applications targeting medical imaging like microscopy, X-Rays and MRIs and also in astronomy and computer graphics.

The earlier works on SR used nearest neighbours and interpolation based techniques. However, today's research is mostly dominated by machine learning models based on neural network architectures. This trend is consistent with the concomitant developments surrounding fluid mechanics[33, 34, 35] and turbulence[36] based applications. In this current

research, artificial neural networks (ANNs) are utilized to explore super resolution in fully turbulent flows.

Fukami et al. [13] used a combination of downsampled skip-connection (ResNet) and multi-scale model to implement SR in turbulent flows. Erichson et al. [12] showed SR using shallow neural networks which have simple architectures and are easier to train. Generative adversarial networks (GANs) and convolutional neural networks (CNNs) [37] have also shown great promise in implementing SR on turbulent flows. More recently, it has also been shown that SR research immensely benefits from integrating flow physics into thae ML training process[38]. These are incredible advances that show great promise in utilizing ML for SR in turbulent flows. However, application of these techniques towards fully turbulent flow measurements is almost entirely unexplored. This can mainly be attributed to the absence of sufficiently large datasets that enable fruitful ML training. Particle image velocimetry (PIV)[39] techniques are great resources to overcome this problem by enabling a high fidelity measurement of fully turbulent flows. But it should be noted that SR in PIV has been a challenging problem for quite a number of decades and it prompted very exciting research in the past[40, 41], that used non ML techniques.

Despite being a great tool in measuring fully turbulent flows, PIV pose a few drawbacks as they are inherently incomplete descriptions of the flow. Given the extremely three dimensional nature of turbulence, the entire physics of the flow is unlikely to be inferred (by ML) from 2 dimensional slices of the 3 dimensional flow phenomenon. The alternative to capture large datasets of fully turbulent flows is with CFD simulations (2D or 3D). They are captured in a much more complete sense as the resulting data is constrained to follow the governing conservation laws. However, it can be prohibitively expensive and not pragmatic to simulate large datasets of fully turbulent flow simulations that could enable efficient super resolution mapping with neural networks.

This motivates the use of PIV measurements to achieve the primary objectives for the current research: What is the complexity of the ANN model that could enable fully turbulent flow super resolution? What is the order of magnitude of the turbulent flow dataset that

is required to train an ANN model for performing super resolution at that scale? How accurate is the super resolution reconstruction when compared against traditional (like linear or polynomial) interpolation based methods and how it scale in terms of reconstructing turbulent derived quantities?

This research also acts as a proof of concept to extend (at least partially), the excellent SR results observed with ML tools on CFD data, when it becomes pragmatic to capture large datasets with high fidelity.

It should be noted that the objective concerning the magnitude of the dataset used for ML training can be motivated by the fact that it takes a tremendous effort to set up PIV. It is relatively simple to collect bigger data sets with statistically independent samples beyond the initial set up. The next section explores the particular ML architecture chosen for current research: Artificial neural networks (ANNs).

2.6 Why artificial neural networks?

ANNs are the simplest neural network based models with relatively interpretable architectures [42].

Since, the goal of current work is to present a proof of concept that super resolution of fully turbulent flows is possible, choosing ANNs hopes to paint an optimistic picture of future deployments with much more complicated ML architectures and that the results can only be improved from this point.

ANNs are the fundamental backbone, based on which many complicated architectures (like CNNs, GANs, RNNs etc.) have evolved. They were originally inspired by how cats process visual stimulus using their primary visual cortex[43]. The earlier NN models were bolstered by the Universal approximation theorem[44] which states that NNs with sufficiently many hidden neurons are capable of learning an arbitrarily complicated non-linear function. Since then NNs have made impressive progress in many fields, often outperforming humans[45]. This is only made possible by the availability of massive datasets and extremely powerful computational resources.

Using any ML model occurs in different phases. The first phase is the pre-processing phase where care is taken to handle the outliers in the data and ensure appropriate data scaling (standardize or normalize) based on the application. This pre-processed data is then utilized in the training phase, where the model learns the relationship/mapping between the input and output instances. Based on the training and validation metrics, the convergence in the training phase is measured and used to stop training. After the training is complete, the model is ready to be deployed, where it can be used to make new predictions on input data based on the mapping that the model has learnt. In the current research, the goal is to learn the mapping from a low resolution instance to a high resolution instance using an ANN.

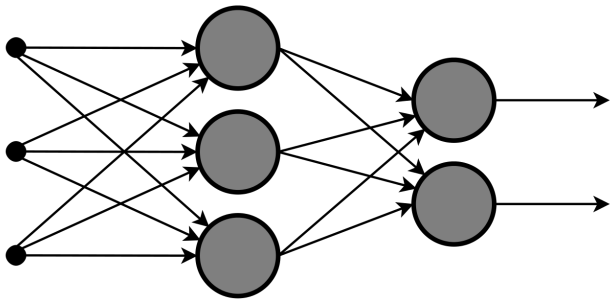


Figure 2.1: an example structure of an artificial neural network with an input layer, one hidden layer and one output layer. The input layer has 3 inputs, the hidden layer has 3 units and the output layer has 2 outputs.

An ANN is a collection of hierarchically arranged nodes that map the inputs to the outputs as shown in Fig. 2.1. Each node in a hierarchical layer is connected to every node in the next layer by a *weight*. The weights between these nodes are learned by using an optimization algorithm (like back propagation by Rumelhart et al.[46]) that minimizes a given loss function between the produced output and the actual output. This is a brief summary of the *training* phase in an ANN. For further information on the functionality of ANNs, the Deep Learning textbooks by Goodfellow et al.[47] and LeCun et al.[48] are great

resource.

In the current work, ANNs are deployed to learn the non-linear mapping between the low resolution snapshots and the high resolution snapshots for accomplishing super resolution. The set up for the framework, the pre-processing, training, post-processing and the subsequent analysis of results are elaborately described in the Chapters 5.

Chapter 3

BAYESIAN INFERENCE FRAMEWORK FOR COHERENT STRUCTURE IDENTIFICATION

In the current work, the properties of convecting vortices in a turbulent boundary layer are identified using Markov chain Monte Carlo-based optimization (MCMC). While there are many methods to localize a vortex, this approach should generalize to more complex structures for which there do not exist other methods of identification. At the same time, estimation of vortex convective velocity based solely on a *single* velocity field remains challenging. One crucial outcome of the current work is a novel approach to determine convective velocities for PIV turbulent flow fields in an automated fashion.

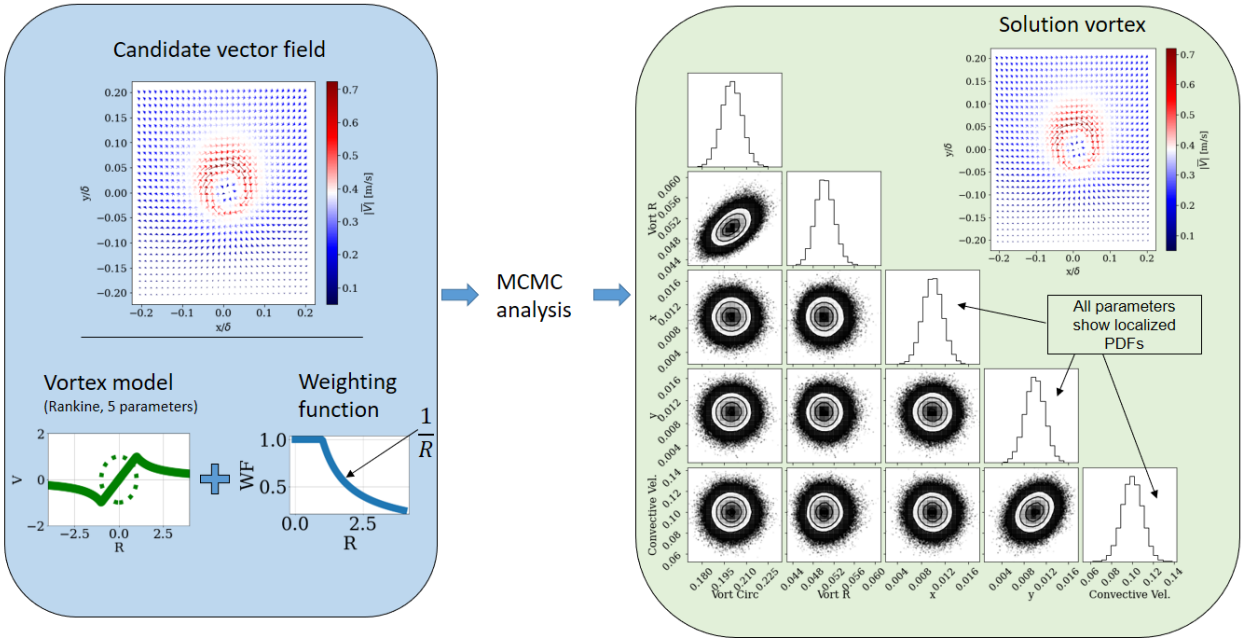


Figure 3.1: Overview of the optimization process

Section 3.1 introduces the framework which enables the extraction of vortex properties from velocity vector fields. The overall process is a vortex model-velocity field matching process. The specific vortex model that is used and how this matching process results in the extraction of vortex properties is explained in Sections 3.1.1 and 3.1.2. The role that the weighting function plays in the optimization process is also discussed. The implementation of this optimization using a Markov chain Monte Carlo approach is compared with traditional minimization.

Section 3.2 of the chapter focuses on testing out this algorithm on artificially generated vortices and the effect of the resolution of the vortex, the presence of noise and initial conditions of the optimization procedure are also explored. The analysis on turbulent velocity field data is narrated in the next Chapter.

3.1 Problem setup

The goal is to extract vortex properties from velocity vector fields using Bayesian inference. The Rankine vortex model takes the values of vortex parameters as its inputs to generate a modelled vortex velocity fields. The goal of the optimization is to find out the specific vortex properties that would reduce the difference between the velocity fields generated by the model and the true (or target) velocity field. In the process, the true vortex properties are learnt by the optimization.

An overview of the optimization process can be seen in Fig.3.1. The figure shows how MCMC analysis uses a Rankine vortex model and a weighting function on a candidate vector field to generate a corner plot from which the vortex properties can be estimated. The reconstructed vortex from these properties is also shown in the top right corner of Fig. 3.1 to observe how it compares against the original vector field. The particular details about the vortex model, the weighting function and the building the optimization framework are explored next.

3.1.1 Rankine vortex model (RVM)

For current analysis, the Rankine vortex model (RVM) is the chosen, given its simplicity and proven effectiveness in literature [49, 50]. In this model, the tangential velocity, is the only non-zero velocity component (i.e. $\overline{V} = V_\theta \hat{e}_\theta$) and it is given by the Eq. 3.1.

$$V_\theta = \begin{cases} \frac{r\Gamma}{2\pi R^2} \text{ for } r \leq R \\ \frac{\Gamma}{2\pi r} \text{ for } r > R \end{cases} \tag{3.1}$$

Here R is the radius of the vortex and Γ is its circulation. The tangential velocity (V) profile along the radius is shown in Fig. 3.2b. A reference vortex core with unit radius is drawn in dotted lines to visualize the switch noticed in Eq. 3.1 for V_θ outside and inside the vortex core. The radius r, is the distance from the center of the vortex at X_c, Y_c. To the resulting vector field, a constant convective velocity, V_c is added. A detailed explanation of the Rankine vortex model can be found in the works by Giaiotti et al[49]. An RVM can be understood as a fixed core with a free rotating vortex and it is ideal to model a hairpin vortex cross-section.

The take away is that the RVM takes in five parameters i.e. The Circulation Strength (Γ), the Convective velocity (V_C), the location (X_c, Y_c) and the radius (r) of the vortex core to generate the stream-wise ($\bar{U}$) and wall-normal ($\bar{V}$) velocity components:

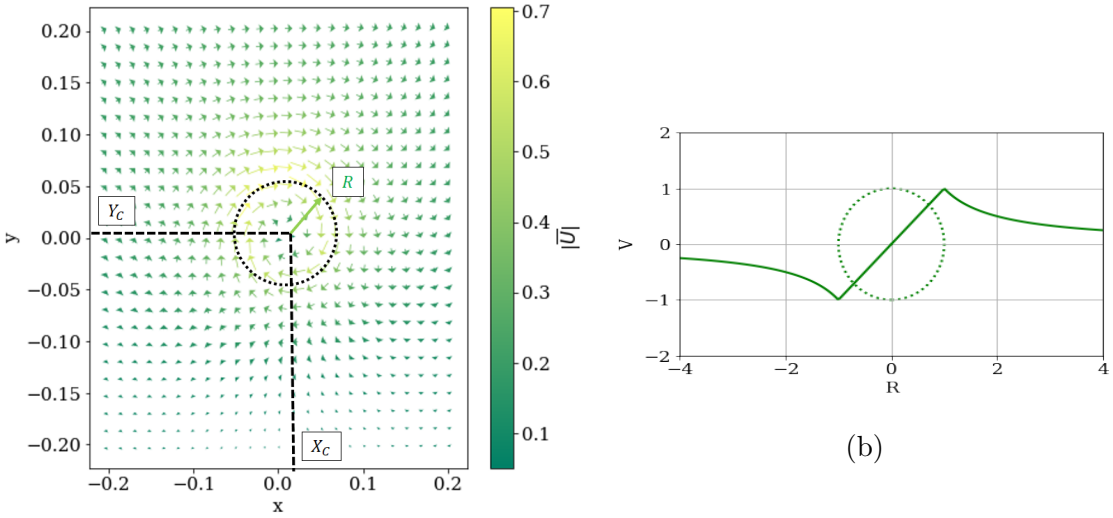
$$U = V_\theta sin(\theta) + V_C \tag{3.2}$$

$$V = -V_\theta cos(\theta) \tag{3.3}$$

Here θ is evaluated as $tan^{-1}(\frac{X-X_c}{Y-Y_c})$. An example vortex generated using the Rankine Vortex model is shown in Fig. 3.2a.

3.1.2 Vortex identification as an optimization problem

Given a certain set of vortex parameters, the vector fields of a vortex can be generated using the Rankine vortex model (RVM). While extracting the vortex properties of a true field, for



(a) (b)

Figure 3.2: (a) Quiver plot of an RVM vortex. This vortex is generated using $\Gamma = 0.2, R = 0.05, X_c = 0.01, Y_c = 0.01, V_C = 0.1$. (b) V_θ of an RVM

every set of vortex parameters, the difference between the field generated by the RVM and the true field can be evaluated. This difference can be used to guide the solution to the global minimum i.e. the true properties of the vortex candidate. In order to set an optimization framework, it is important to quantify this difference. The difference between the true and the RVM generated fields is still a vector field. In order to convert it to a scalar, the weighted average difference (ζ) of the field is taken.

$$\zeta = \sum \frac{W[(U_T - U_M)^2 + (V_T - V_M)^2]}{A_W} \tag{3.4}$$

where U_T, U_M refer to the true field and the field generated by the RVM model, respectively. The summation is over all the elements present in the 2-D vector field. The W refers to a the weighting function and A_W is the corresponding area under the weighting function. The weighting function would help to take a weighted difference of the true and generated fields.This weighting function is employed as a result of the expectation that velocity vectors

far from the vortex center may be less related to the local vortex induced field, as would be the case for complex turbulent flows with many vortices present. An example weighting function is shown in Fig. 3.3. This results in ζ taking the absolute difference of the fields and as puts less weight on the vectors that are further away from the radius core. The weighting function is chosen to decay following the $\frac{1}{R}$ function outside the vortex core and have a value of one within it (see Fig. 3.3). In true velocity fields captured through experimental setup, this ensures that the vortices which are too close to one another influence each other less during the optimization process. This detail is crucial to ensure the overall convergence of the optimization for turbulent flow vortices, which will evident in the future sections.

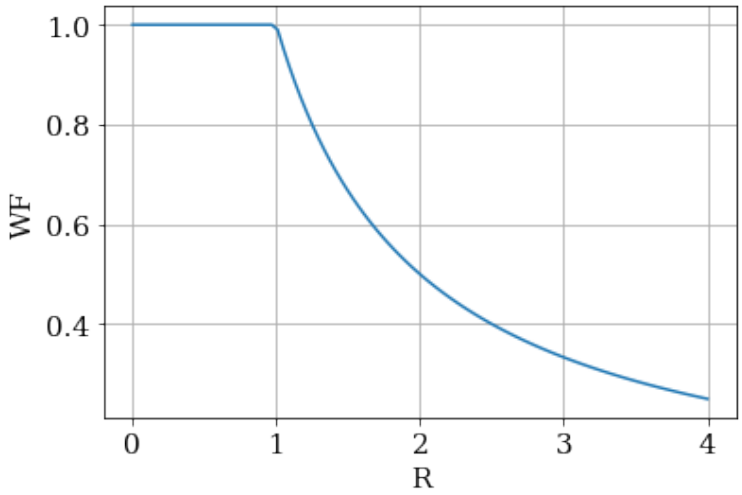


Figure 3.3: Weighting function

3.1.3 Optimization with MCMC using the Emcee python module

Going back to the Markov chain Monte Carlo analysis, the optimization is driven by forming Markov chains in a Monte Carlo fashion such that the equivalent distribution (formed from sampling) learns the underlying process. The formation of effective chains depends on the posterior probability of the optimization process. Posterior probability ($P(\theta|x)$) is defined according to the Baye's rule as shown in Eq. 3.5.

$$P(\theta|x) = \frac{P(x|\theta)}{P(x)} P(\theta) \tag{3.5}$$

Here, x refers the observations made and the θ refers to the parameters. In our case θ would be the vector $(\Gamma, R, X_c, Y_c, V_C)$. The $P(x|\theta)$, $P(\theta)$ are the likelihood and the prior functions respectively. The $P(x)$ is a normalization function which can be dropped as it is arbitrary. It is also common to take the log of the posterior and the corresponding sum of the log-likelihood and the log-prior based on the nature of the arguments. Taking logarithms work well with the current problem set up. For continuous random variables, the posterior probability function is a joint probability density function from which the individual PDF of each parameter can be inferred. Fig. 3.1 can be seen for reference. More details on the corner plots are explained in Section. 3.1.4.

MCMC learns to sample from a posterior probability function. By setting this function as the objective function for optimization process itself, the underlying parameters (θ) that maximize the objective function would be the most likely occurring values in the PDFs generated from the posterior probability. When then use a Gaussian fitting algorithm to extract the peak of the distribution, its most-likely value f. To summarize, the log-posterior function has to be closely related to the objective function in order to mimic and learn the global extremums from the corner plot of MCMC. A step further would be realizing that the objective function can simply be substituted with the log-likelihood function.

In our case,

$$log(P(x|\theta)) = -\zeta(\theta, x) \tag{3.6}$$

$$log(P(\theta)) = \begin{cases} 0; \theta_{min} \leq \theta \leq \theta_{max} \\ -\infty; \theta_{min} \geq \theta; \theta_{max} \geq \theta \end{cases} \tag{3.7}$$

The $\zeta(\theta, x)$ in Eq. 3.6 refers to the mean weighted square difference (Eq. 3.4). It takes the parameter values (θ) to generate the U_M, V_M and the x refers to the observations i.e., the U_T, V_T. The negative sign rectifies the fact that MCMC is maximizing the difference instead of the desired minimization.

The value of $P(\theta)$ can be inferred from Eq. 3.7. $P(\theta)$ becomes 1 and 0 when the parameters (θ) are within and outside the bounds respectively. They become 0 and $-\infty$ after applying a logarithm. This is called a having an un-informative or a flat prior. Current analysis assumes no prior biases towards the parameters, and this MCMC-based optimization is theoretically equivalent to maximum likelihood based optimization, which can be accomplished using a traditional optimization technique (shown in the next section). It is also worth pointing out the fact that strong prior knowledge influencing the $P(\theta)$ has shown to significantly improve the optimisation process. Developing alternatives to the current flat prior function is out of scope for this current work and has the possibility of biasing results, if not carefully designed.

The Emcee Python module[25] is utilized to implement Affine Invariant Markov chain Monte Carlo sampling for optimizing the posterior function. A total of 50 chains (random walkers) were employed for 20,000 time steps. Based on the integrated auto-correlation time[1], it was decided that the initial 5,000 time steps are to be discarded as burn-in. The number of walkers, timesteps and burn-in were all chosen to be larger than required to achieve a consistent output distribution. These hyper parameters were maintained throughout the current analysis.

The Emcee sampler takes in the velocity vector fields as an input and returns the corner plot from which the vortex properties can be extracted by using the custom Gaussian fitting algorithm for the pdf peaks, completing the extraction of vortex properties simply from a single velocity vector field (Fig. 3.1).

The additional inputs to the Emcee sampler include prescribing the initial conditions and bounds of each parameter. As 50 Markov chains are initialized, it is necessary to give the initial conditions for each of them. It is important to note that the Initial conditions have to sufficiently linearly independent from one another, giving a better chance of covering the

[1]Goodman Weare [26] note that the integrated auto-correlation time to quantify can be used to understand the effects of Monte Carlo sampling error on the results, by making sure that the chains are made up of sufficiently independent samples.

entire solution space. A simple technique to generate linearly independent initial conditions would be to have a fixed vector (θ_0)) and some noise while assigning the initial condition as $\theta_{IC} = \theta_0 + \theta_0 * numpy.random.random([50, 5])) * 0.01$ where, $\theta_0 = [\Gamma = 0.08, R = 0.05, X_c = 0.01, Y_c = 0.01, V_C = 0.02.]$.

The bounds for the optimization are shown in Table 3.1 corresponding to the parameters Γ, R, X_c, Y_c and V_C respectively in the RVM model. The x and y values refer to the location of the grid points on the velocity vector fields in pixels. Selecting proper bounds for optimization can easily be overseen, but is of crucial importance to the overall optimization and requires utmost attention. For Γ, it is known only that this is positive value. The R can be as small as the smallest grid resolution or could take up the entire frame at hand. The X_C, Y_C values are allowed to be within 5 pixels from the center of the snapshot on either sides. Hence, it is important to have a good initial estimate of the vortex candidate, otherwise it is ideal to increase the span of this range. Finally, it is known that the convective velocity cannot be larger than 25% of the stream-wise mean velocity[2]. These details played a role in deciding on the bounds detailed above.

Parameter (θ^i)	θ^i_{max}	θ^i_{min}
Γ	0	∞
R	$x[1] - x[0]$	$max(x)$
X_C	$-5(x[1] - x[0])$	$5(x[1] - x[0])$
Y_C	$-5(y[1] - y[0])$	$5(y[1] - y[0])$
V_c	-0.25	0.25

Table 3.1: Optimization bounds for Markov chain Monte Carlo

3.1.4 Corner plots

The outputs from a traditional optimization (like truncated Newton's) would be a scalar value. However, the output from an MCMC based optimization would be a Markov chain.

In order to get the optimal value itself, the equivalent distribution of this Markov chain is plotted and from this PDF, the most likely value is extracted. In a optimization problem with N variables, the output of the MCMC process would be (at least) N Markov chains corresponding to each variable. Once these chains are obtained, the PDFs corresponding to each one could be plotted, and later the complete solution set can be extracted.

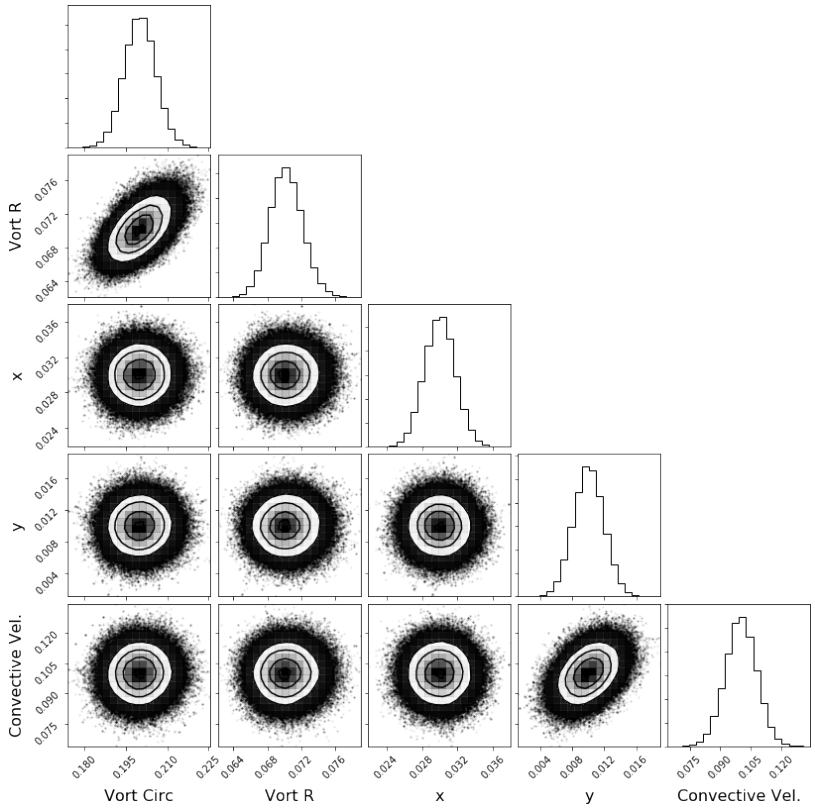


Figure 3.4: An example corner plot for MCMC fitting of a Rankine vortex.

The Corner[51] module in Python is used to showcase a projection of a data-set in a multi-dimensional spaces. This is particularly useful to visualize the MCMC results, as they are an N dimensional datasets containing Markov chains.

Fig. 3.4 shows an example corner plot resulting from an MCMC process that is optimizing an objective function with five parameters. It shows the bottom left half of an entire grid.

The missing half provides no additional information as it would be symmetric with the plotted figures. The horizontal and vertical labels explain the ownership of the plots. The plots themselves reveal the covariance between corresponding random variables. It is a good check to see that the variables being optimized are linearly independent from one another. A scatter plot with a single cluster in the covariance field is desired and it indicates a uni-model solution that is not correlated with other parameters. The plots on the diagonal can be analysed as the PDFs for the corresponding variables. More details about corner plots can be found in the works of Foreman-Mackey[52].

From these PDFs, the most likely solution can be estimated by, say, for example taking the mean or the median. However, these PDFs are often noisy due to the large number of points from which they are comprised. Also, in the case of working with true PIV vortex data, these PDFs are often bi or tri modal distributions due to bad initial estimates[2] of the interrogation window. These issues mandates that a robust algorithm is to be developed which can process any arising PDFs to produce the most likely solution of that parameter. The intuitive solution is to simply take the argument maximizing the PDF. In a Gaussian-like distribution, this would return the mean. This is an good initial guess, but can be significantly improved. The room for improvement comes from the fact that the signal is too noisy and that the choice of bin size is crucial in determining the uncertainty of the acquired quantity.

To overcome the prior issue a custom Gaussian fitting algorithm is developed and here are the steps followed:

1. Isolate the PDF of each variable from the MCMC sampler.

2. Identify the tallest peak of the signal. A peak prominence threshold of 10%* is used to find the tallest peak. This is iteratively reduced to handle noisy signals.

3. Calculate the relative height of the peak and make note of the extreme points (e.g. the

[2]where multiple vortices might be present in the considered bounding box.

location where the red line stops in Fig. 3.5c).

4. Calculate 50%* of the distance from the peak to both sides of the peak and change the overall region of consideration. (Fig. 3.5d)

5. Check if percentage of points considered are at least 15%* of the total number of points in the signal (PDF).

6. If yes, do a Gaussian fit with non-linear least squares method. Note the mean of the Gaussian fit as the proxy for the most probable value of this variable. If no, increase the region of consideration from 50%* to 100%*. (Fig. 3.5e)

7. Repeat the process for the remaining variables in the MCMC sampler.

Here, the * on the top of a number indicates that it is a tunable hyper-parameter value and the indicated value is best suited for the current work[3]. Figs. 3.5a-3.5e show the successive steps taken in implementing this algorithm on a bi-modal distribution.

In this example from Fig. 3.5e, the algorithm returned a Gaussian fit whose mean was 3.00046 which is close to true mean of 3. While analysing corner plots, the PDF of each variable is chosen and the Gaussian fit is evaluated using this method. Hence, each corner plots essentially boils down to an N dimensional vector, based on the number of parameters in the optimization.

This simple step is crucial to ensure that multiple peaks in a PDF are handled well, especially in a noisy signal and makes the overall process significantly more robust and also reduces the uncertainty of the acquired solution. The properties corresponding to the other peak could be interesting to explore, but are deemed out of scope for the current work. This algorithms can also be used for other traditional signal processing application and a python implementation is available here [4].

[3]Considering too many points would deviate the extracted mean from true value and on the other considering too little would make the fitting more sensitive to noise

[4]https://github.com/kommalapatisahil/CoherentStructures/blob/master/gaussianFit.py

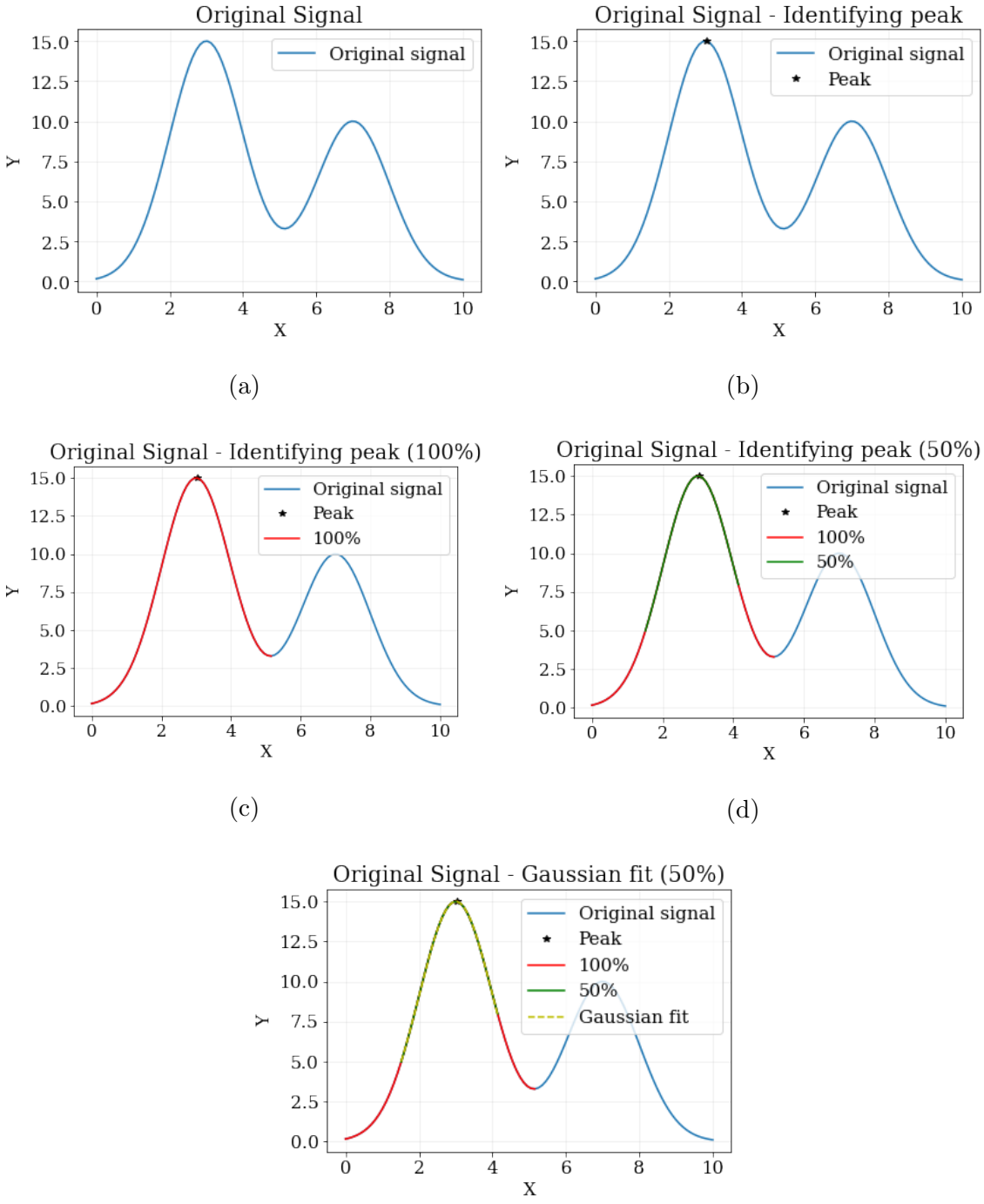


Figure 3.5: Gaussian fitting algorithm : step -1 (a), step -2 (b), step -3 (c), step -4 (d), step -6 (e).

3.1.5 Optimization using traditional minimization techniques

In all the results seen in the future sections, performance of the MCMC is compared against a traditional minimization process. This serves a few different objectives. It is well known that an MCMC based optimization is quite time consuming in comparison to traditional minimization procedures. For example, with the hyperparameters described in the previous section, an MCMC procedure on a 31 by 31 dimensional velocity field would take approximately 9 minutes to complete the extraction of vortex properties[5]. On the other hand, using Truncated Newtons Algorithm (TNC) for Maximum likeli-hood estimation (of the log-likelihood, $P(\theta|x)$) returns a solution (on an average) in less than 1 seconds. So, TNC can be used as a baseline to understand how well or badly the MCMC optimization performs comparatively. Also, since MCMC with a flat prior theoretically should converge at the same solution as a Maximum likelihood estimation, it would an interesting exercise to compare and validate the two solution. However, it is important to note that despite the relatively longer computational times, Bayesian optimization based methods including MCMC posses many clear advantages when compared against frequentist approaches like maximum likelihood estimation using TNC. MCMC is expected to be able to localize global minima which cannot be garunteed by TNC. Also, MCMC is expected to be more robust to noise and can be scaled to higher dimensional model which is important while shifting the focus to the identification of other turbulent coherent structures.

To conclude, MCMC based optimization uses the Log-posterior function (Eq. 3.5), while the TNC based optimization uses the Log-likelihood (Eq. 3.6) function. However, it should be noted that the fundamental approach to optimization is entirely different in both methods. MCMC is Bayesian and TNC is frequentist.

---

[5]these results were generated using an Intel (R) Core (TM) i5-5200U CPU @ 2.20 GHz, 2201 Mhz, 2 Core(s) with 8 GB of RAM.

3.2 *Performance on synthetic vortices*

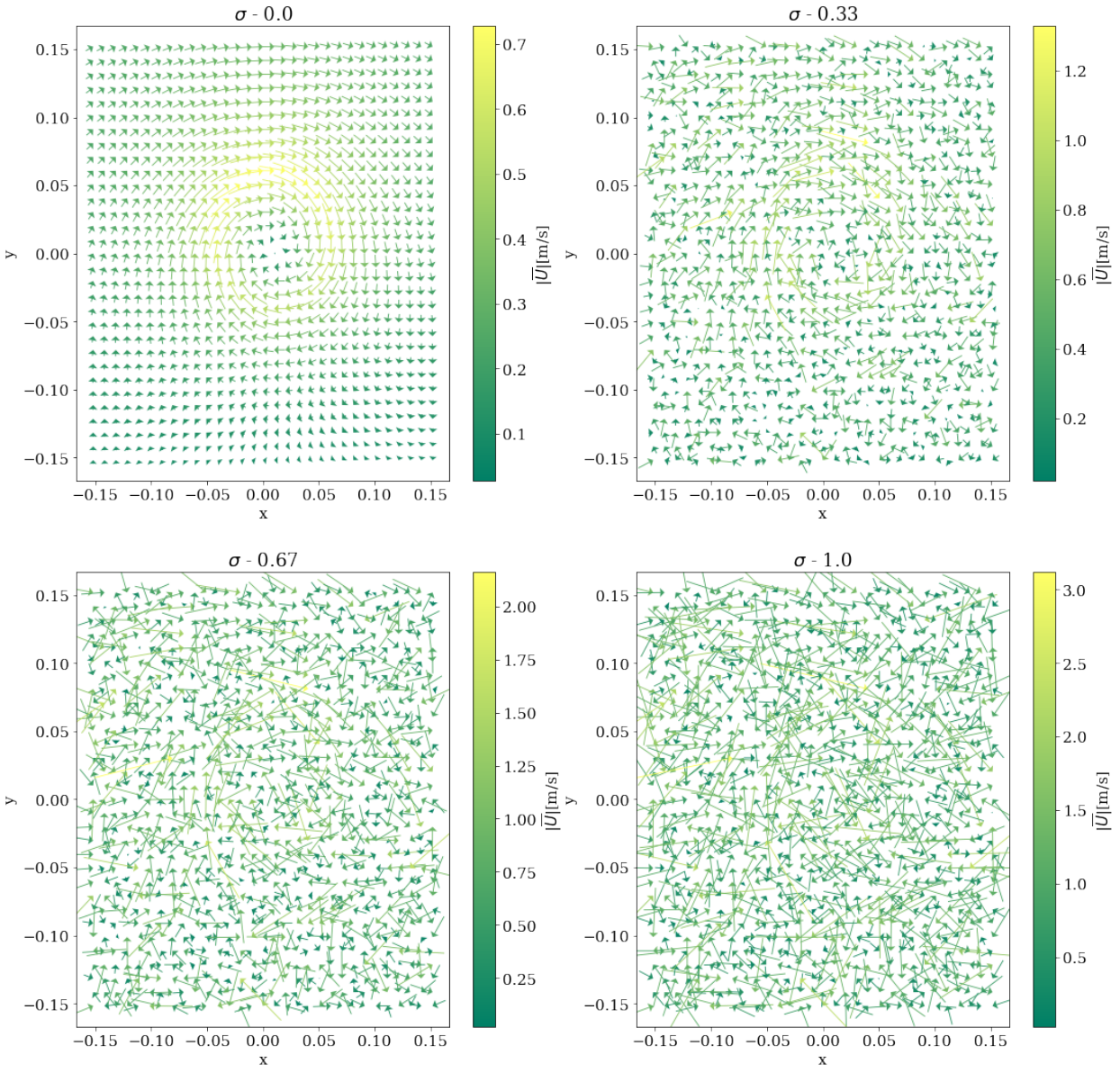
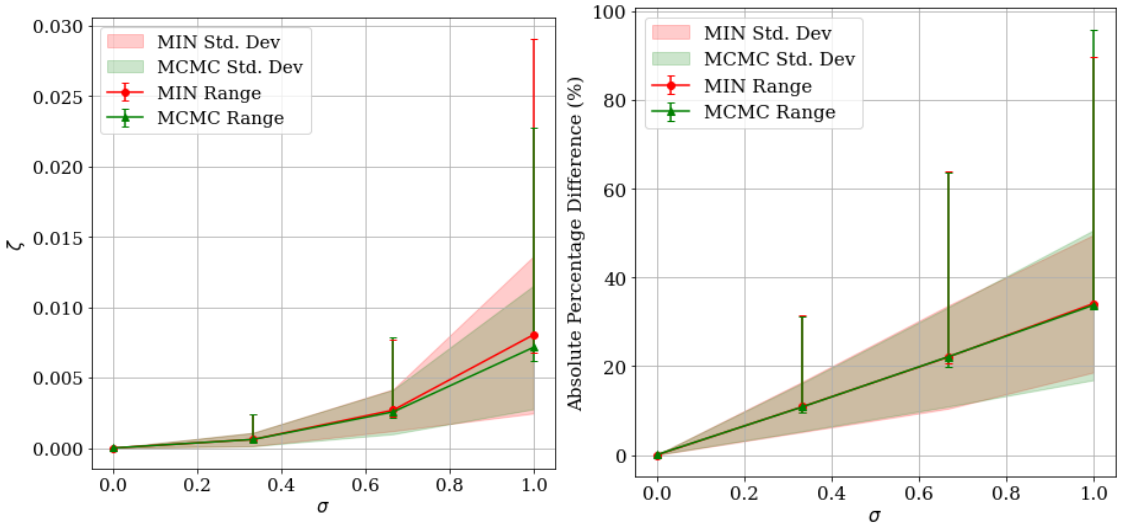


Figure 3.6: RVM vortices with increasing amounts of noise (σ) on a field with 31 × 31 pixels.

Now that the framework to extract vortex properties is established, it is crucial to estimate the limitation of MCMC in the presence of significant noise within the vector fields and

also lower than ideal number of vector samples within the vortex, which could arise due to insufficient resolution. In this section, the performance and robustness of the MCMC method in response to varied amounts of noise and resolution are evaluated and compared to TNC. Two different types of performance metrics are evaluated: These are the overall weighted mean square difference errors (ζ) and the percentage deviation of the derived parameters of vortex properties (θ) from true values. In order to accurately judge the quality of this analysis, synthetic vortices with known properties are generated and distorted with noise or resolution effects.



(a) (b)

Figure 3.7: (a) Overall weighted mean squared difference (ζ) error and (b) Absolute percentage difference with respect to true $V_C = 0.1$

3.2.1 Effects of noise on overall error and individual parameters

To evaluate the performance of current optimization techniques, Gaussian noise is chosen. More particularly, a 2-D grid of values are generated by sampling a normal distribution centered at 0 with certain variance. This variance is equal to the maximum value of all

vector magnitudes in the velocity field to which the noise is being added. (Mean = 0, variance = max($\overline{U}$)). Before adding this noise to the original velocity field, the variance is scaled by a factor (σ), so that now the resulting variance is σ*max($\overline{U}$). A few examples with increasing amounts of σ is shown in the Fig. 3.6. The top most subplot shows the true vortex ($\sigma = 0$), that was generated using a Rankine Vortex model with parameters as $\Gamma = 0.2, R = 0.05, X_c = 0.01, Y_c = 0.01, V_C = 0.1$. These are all quiver plots which show the velocity magnitude by taking the x and y velocity components. It should be noted that while generating the noise, different noise frames were added to both the $\overline{U}$ and $\overline{V}$ velocity components, which make the task of optimizing more challenging.

For these four plots shown in the Fig. 3.6, both MCMC and TNC optimizations are executed. The overall error in reconstruction is calculated by using the the ζ function, by comparing the new RVM vortex generated using the extracted solution with the original velocity vector fields ($\sigma = 0$).

While dealing with any random noisy parameter, the resulting values have to be independent of the noise seed value. Tn order to capture the true relationship that is not dependant on the nature of noise, the experiment was repeated with 10 different seeds, while generating the noisy velocity fields. In Fig. 3.7a-3.8d, each measurement is essentially the mean of all the repeated experiments taking 10 random seeds into account. For the sake of completeness, the standard deviation and the range are also added in the plots.

From Fig. 3.7a, which shows the overall error, the crucial observation is that both methods are extremely reliable when no noise is present. This can be confirmed by the ζ value which is very close to 0. Beyond that, both methods perform pretty good even when the σ value increases to 0.33 and also 0.67. However, beyond that MCMC shows a slight advantage compared minimization both in terms of all statistical measurements shown in the plot. This fact that MCMC reliably extracts solutions until $\sigma = 0.67$ is used in future evaluations to make sure that the noise is not too disruptive. From the remainder of the figures (Fig. 3.8a - Fig. 3.8d), it can be argued that the circulation strength parameter (Γ) is the most robust to noise and the location of the vortex core (X_C, Y_C) is the most sensitive

towards increasing amounts of noise.

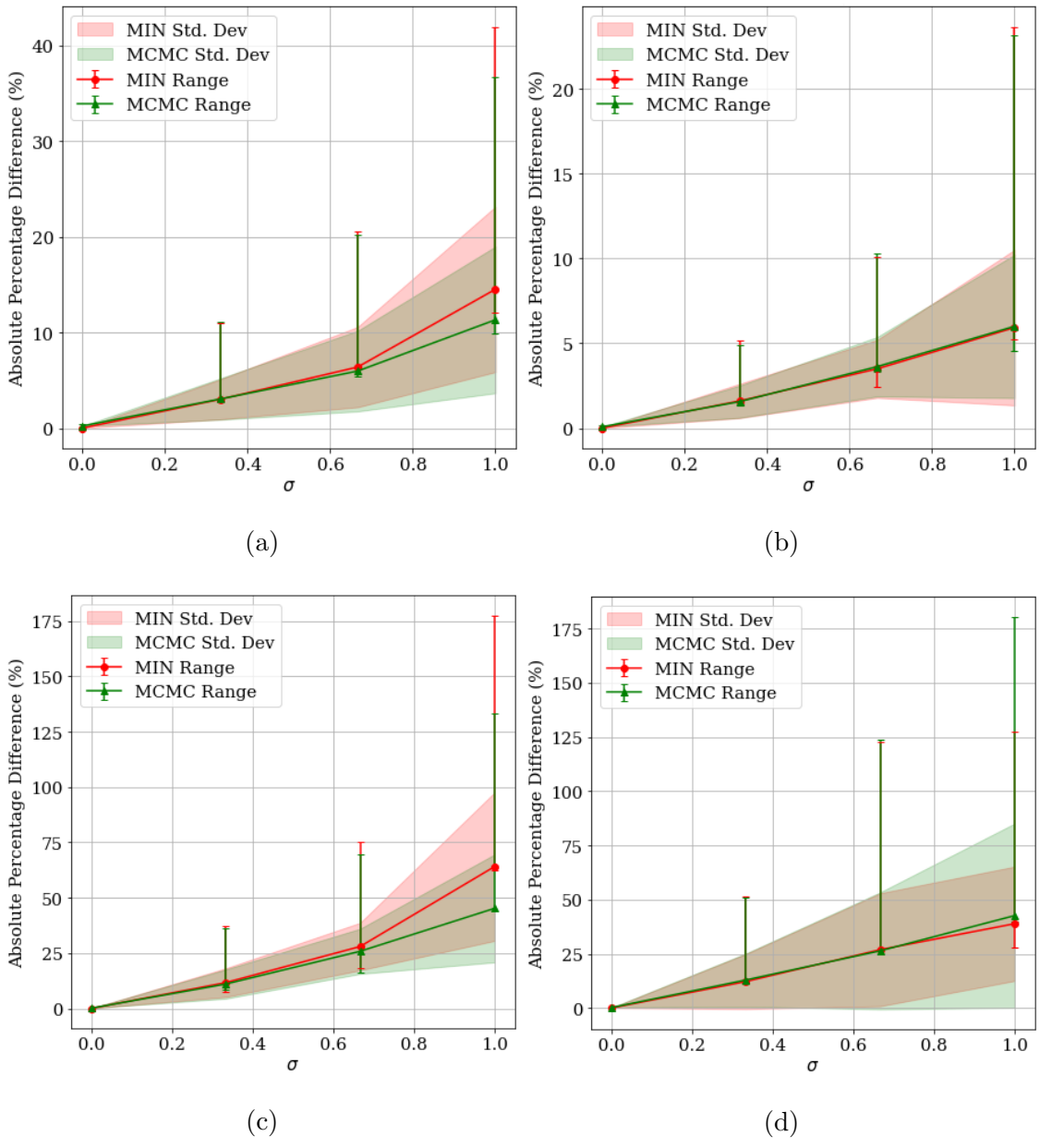


Figure 3.8: Absolute difference plots for individual parameters. with respect to (a) $R = 0.05$ (b) $\Gamma = 0.2$ (c) $X_C = 0.01$ (d) $Y_C = 0.01$

In the current analysis, a resolution of 31 by 31 pixels was chosen with varying amounts of noise. In the next subsection, the effects of resolution on the robustness of the optimization are evaluated.

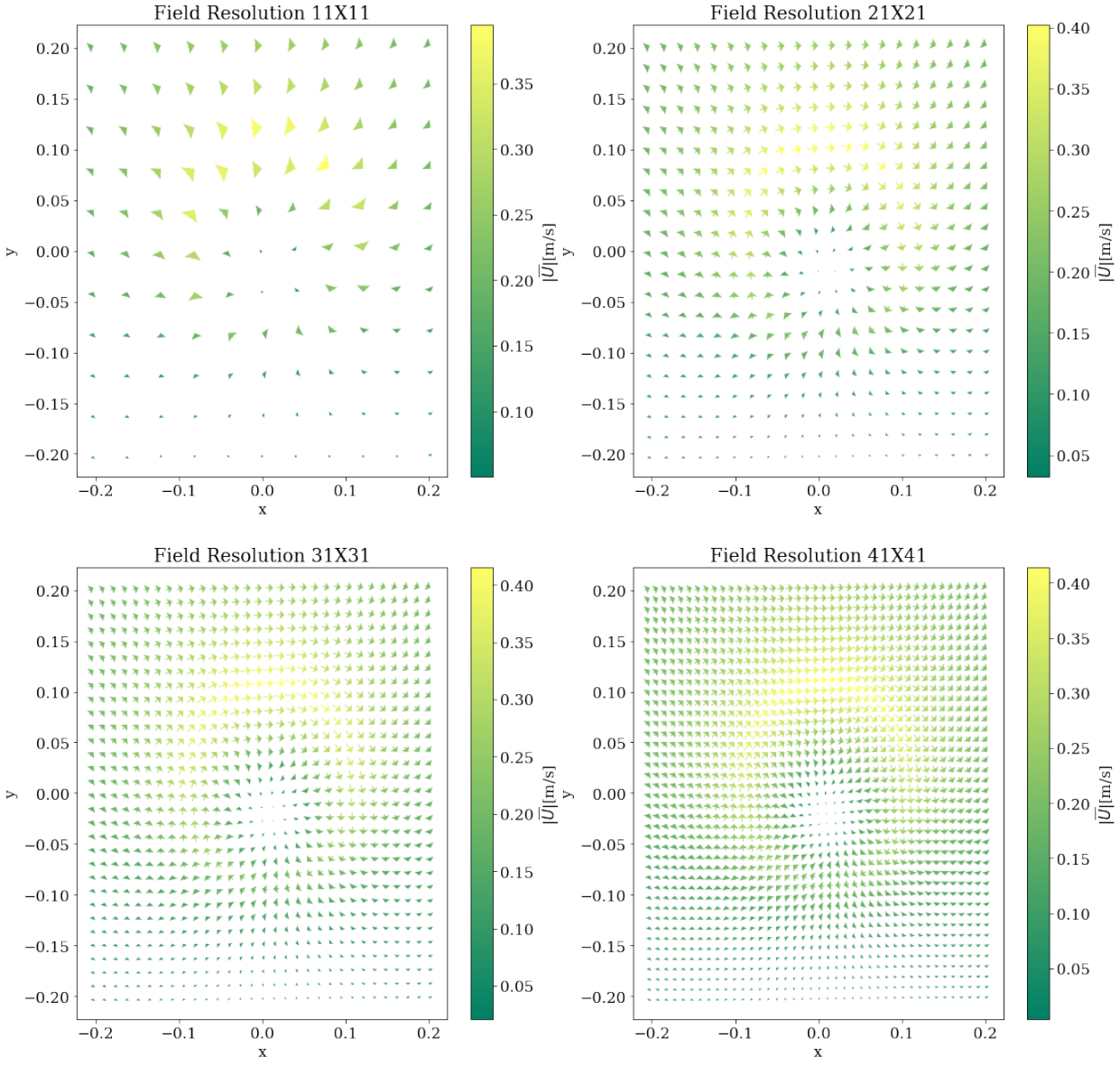


Figure 3.9: RVM quiver plots with different resolution

3.2.2 *Effect of resolution on total error and on individual parameters*

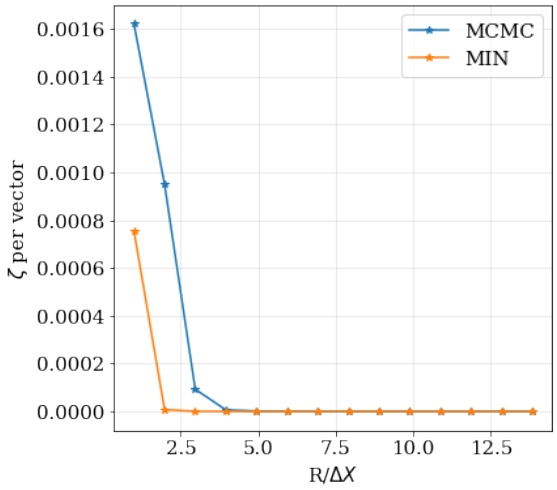


Figure 3.10: Average ζ error per vector

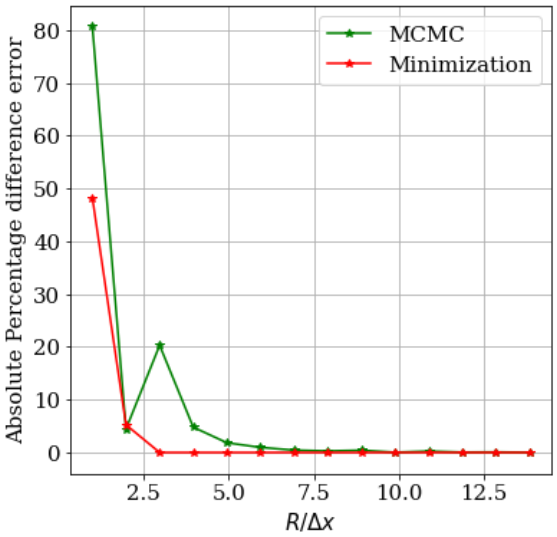


Figure 3.11: % difference : $\Gamma = 0.2$

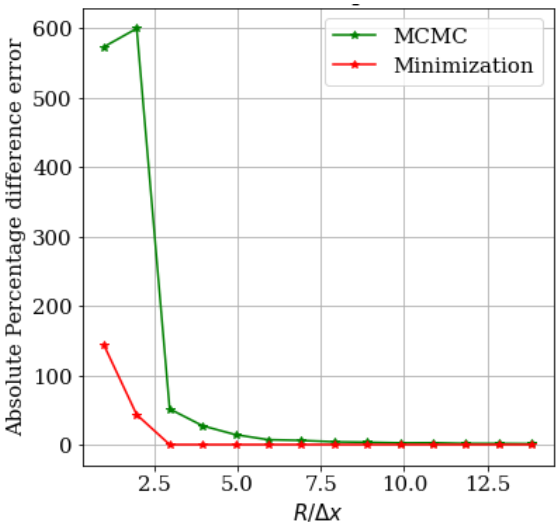


Figure 3.12: % difference : $R = 0.1$

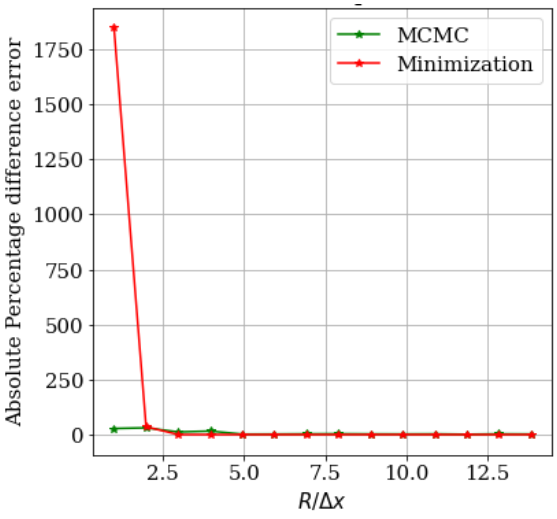


Figure 3.13: % difference : $X_C = 0.01$

A similar analysis is performed on to evaluate the threshold of the resolution below which the optimization methods fail. Fig. 3.9 shows example snapshots of quiver plots with different resolutions. All the vector fields used for analysis in this section contain a vortex with the following properties : $\Gamma = 0.2, R = 0.1, X_c = 0.01, Y_c = 0.01, V_C = 0.1$.

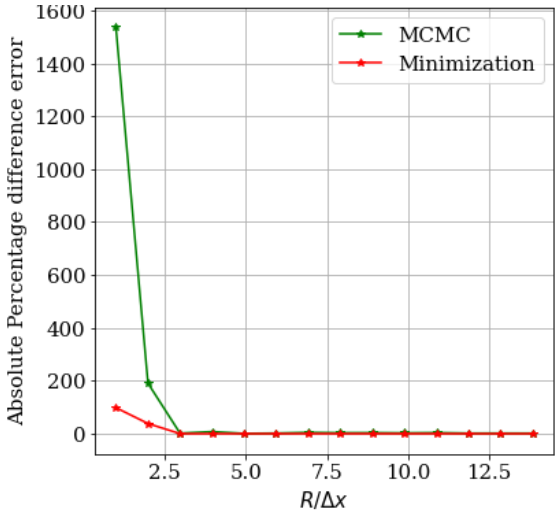


Figure 3.14: % difference : $Y_C = 0.01$

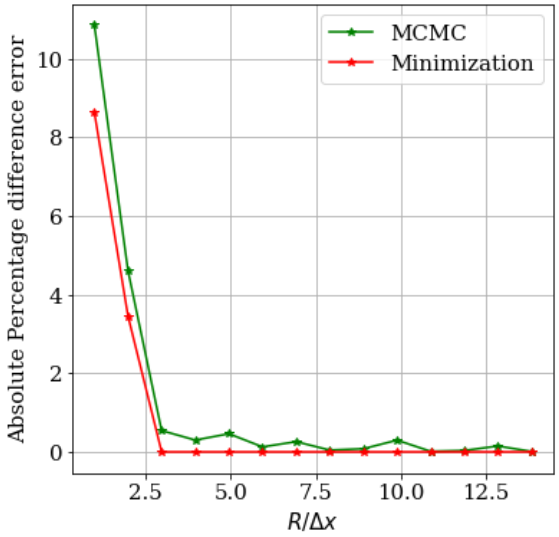


Figure 3.15: % difference : $V_c = 0.1$

Fig.3.10 shows the overall error per vector (pixel) as the resolution of the vector field is increased. Naturally, the error goes down as the resolution increases. The resolution is non-dimentionalized by taking $\frac{R}{\Delta X}$ instead of the simply taking the number of vectors across. Here, R refers to the radius of the original vortex and ΔX refers to the grid width between two vectors. Even though ζ is the weighted average difference error, larger regions of vector fields show larger errors. To make sure that this fact does not unfairly compare the errors, this analysis considers the ζ error per vector as the quantity of measure.

The Fig. 3.10 indicates how close the MCMC and TNC are, with respect to one another. But, the value of ζ could still arbitrarily large or small with out any intuitive meaning. Only by looking at the the later figures (Fig. 3.11 to Fig. 3.14) can sound conclusions be made. Vortex velocity fields with resolution $\frac{R}{\Delta X} \leq 5$ are not reliable. This analysis provides an

interesting conclusion: Velocity fields that are smaller than 3 by 3 to 7 by 7 pixel range are not ideal for current optimization methods. Since, the resolution of each PIV measurement is fixed, this would imply ignoring vortex candidates that are in this range, by considering them too small to be accurately characterized.

Another conclusion that could arise out of this analysis would be to always take PIV measurement with the highest resolution possible to make sure more vortex structures are considered as candidates ideal for further analysis.

Other than these values, there are many other hyper-parameters that could also influence the optimization process to varying degrees. Notably, the weighting function and the initial conditions given to the MCMC process. For all of the current analysis, the weighting function shown in Fig. 3.3 was used. It works well for the current problem set up. However, the flat region of this weighting function can be increased or decreased to take bigger or smaller regions of the difference field into consideration for the log likelihood function. This could prove to be crucial while working with other coherent structures or different turbulent flow data sets.

The initial conditions mentioned in Section. 3.1.3 have also been observed to have an effect on the overall optimization process. So, care should be given to make sure that the Initial conditions are not oblivious to the problem set up.

Chapter 4

RESULTS FROM BAYESIAN INFERENCE ON PIV DATA

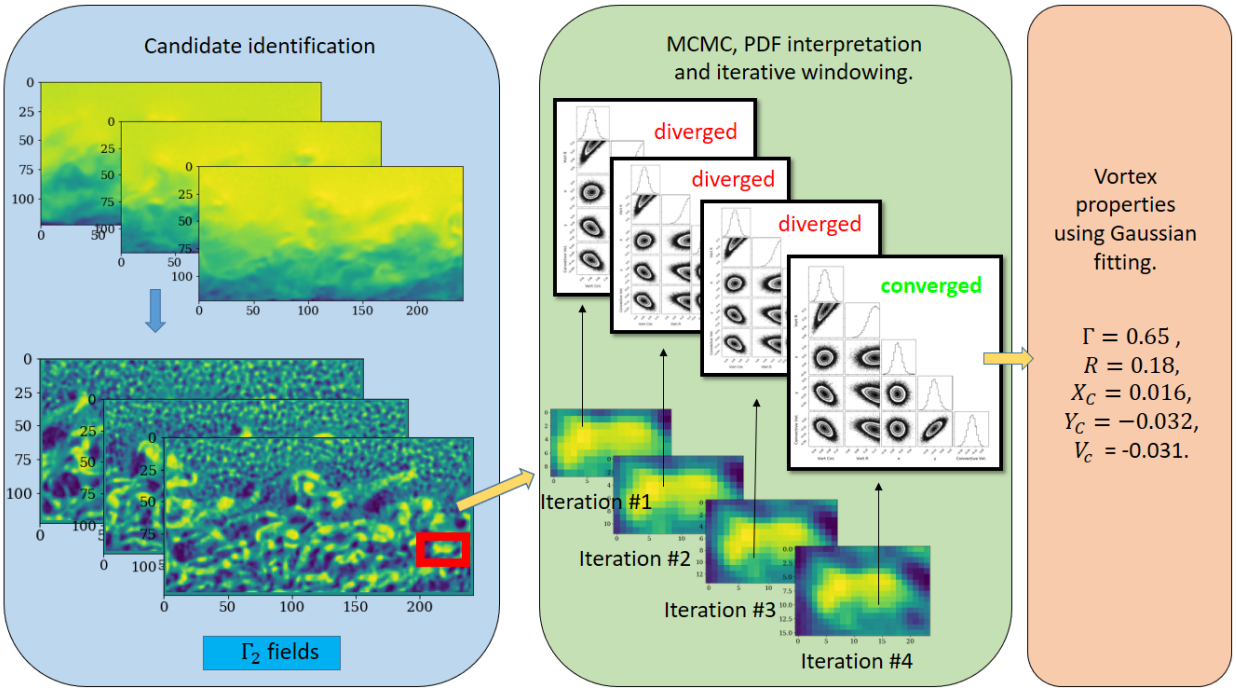


Figure 4.1: Overview of automated Identification of vortex properties in a PIV dataset

Having established the efficacy of this method for synthetic velocity fields, this approach will now be used to extract vortex information from an experimental turbulence data set. Fig. 4.1 shows an overview of this process on PIV data. Details about the particular dataset used are mentioned in the Section 4.1. This data is inherently more messy than the previously considered synthetic examples. A range of methods exist, that could be used to localize the vortex candidates in a PIV snapshot. Using these methods, properties like the radius and the

center of the vortex core can be approximated with high reliability. Once the local region of a velocity vector field is identified, MCMC optimization, as described in the previous chapters can be applying to extract further properties about the vortex like the circulation strength and, more importantly, convective velocity. It is also worth pointing out that MCMC also refines the initial estimates of the radius and the location of the vortex core.

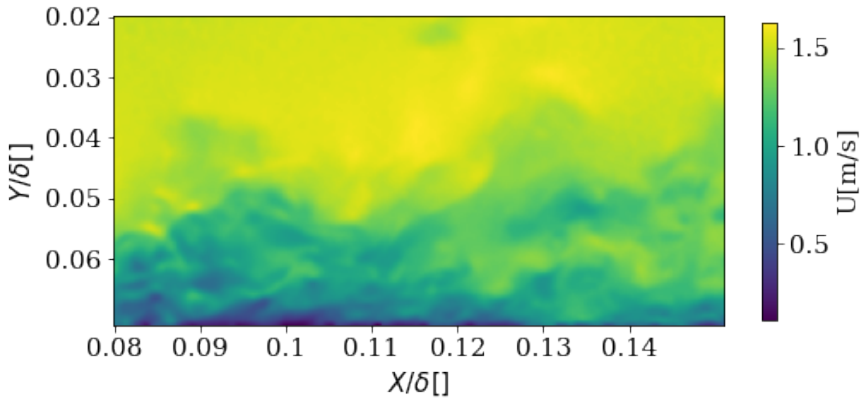
Adjacent vortices too close to one another are found to create challenges for the interpretation of MCMC's corner plots as some Markov chains (walkers) end up localizing the adjacent structures. This would result in a bi-modal distribution in the corner plots, which is not desirable. In this chapter, a reliable procedure to overcome this and some other issues are narrated to successfully isolate and extract vortex properties in an *automated* manner to go over all the PIV snapshots.

Before moving any further, it is necessary to introduce the PIV dataset of turbulent boundary layer flow which is taken upon, in the first session of this chapter. The later sections focus on detailing an established vortex localization method and some suggestions to set up the bounding box for a candidate to make it ideal for optimization. The bounding box refers to the localized region in the entire frame that is presumed to contain a single vortex. The next section focuses on measures to take, in case MCMC returns diverging PDFs in the corner plots. The final section showcases the results of using MCMC for vortex identification at a few different wall normal heights in the turbulent boundary layer flow.

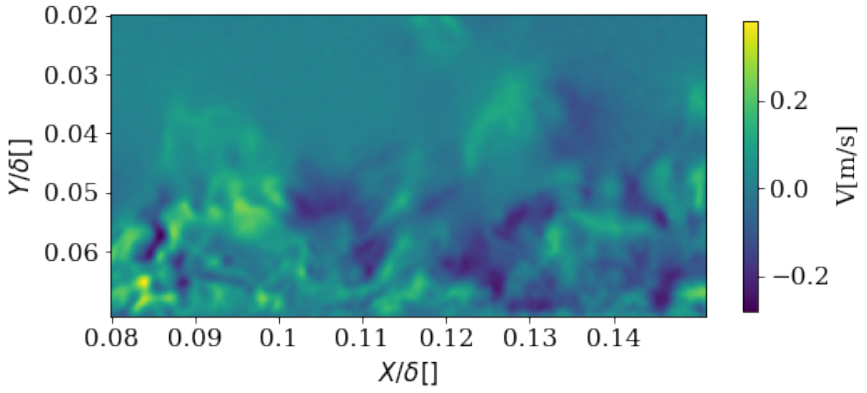
4.1 Turbulent boundary layer Data set

The vortex Identification framework developed throughout this work is initially tested on synthetic vortices (Chapter 3) and then on real vortex measurements captured using PIV methods. This PIV dataset contains a turbulent smooth wall bounded boundary layer flow. It was originally collected by [53] to study the effect of stable thermal stratification. This dataset contains 750 frames of non-time resolved PIV measurements along the streamwise-wallnormal plane. The vortex structures in this dataset were identifies manually by hand in this work by Williams[54]. Current work forms a natural extension to this work by automat-

ing the identification procedure. The Reynolds number based on momentum thickness, Re_θ is 1514, with a vector spacing in inner-units $\delta x^+ = \delta x u_\tau / \nu$ is 0.01011. As a result, the flow is considered well-resolved. It is recommended that the reader refers to [53] to learn more about the experimental setup of the dataset. Fig. 4.2a, Fig. 4.2b show the stream-wise ($\bar{U}$) and wall-normal velocity ($\bar{V}$) components of an example snapshot.



(a)



(b)

Figure 4.2: Example snapshots from the boundary layer flow dataset. (a) shows a stream-wise velocity field and (b) shows a wall-normal velocity field. Here $\delta = 0.12752905$

4.2 The Γ_2 vortex localization function

The localization of a single vortex in the entire frame is the first step towards automating vortex identification. The entire process of automated identification in all PIV snapshots can be seen in the Fig. 4.1. The selection of vortex candidates can be done a number of ways, notably including the swirling strength, the helicity and the vorticity. However, for PIV data, alternatively using the Γ_2 operator function, proposed in[4] proved to be the most successful. It operates on velocity fields from which the mean stream-wise velocity has been subtracted.

The Γ_2 operator is defined as,

$$\Gamma_2(P) = \frac{1}{S}\int_{M\in S} \frac{[PM \wedge (U_M - \tilde{U_P})].z}{[\|PM\|.\|U_M - \tilde{U_P}\|]} ds \tag{4.1}$$

The Γ_2 operator provides a scalar value for every point P in the frame, which is seen to be the input in Eq. 4.1. $\wedge$ refers to an inner product. S refers to a small local region around P. The integration is over all the points M in this region S, surrounding P. $\overline{PM}$ is the velocity vector between M and P. z is the unit normal to the measurement plane and $\tilde{U_P}$ is the local averaged convective fields calculated using Eq. 4.2. This subtraction ensure that Γ_2 is an objective operator [7]. U_M is the velocity velocity vector a M.

$$\tilde{U_P} = \frac{1}{S}\int U ds \tag{4.2}$$

Fundamentally, the Eq. 4.1 is a way to mathematically write the summation of all the angles which a point P makes with the velocity vectors of its neighbouring points. So, if P were to be at the center of the vortex core, it would make right angles will all of the neighboring points and would be a global maximum of the Γ_2 field.

For a perfectly symmetric vortex, the Γ_2 would take a maximum value (equal to 1) at the center of the vortex core. This value decreases as we move away from the center of the core. Also, Γ_2 is theoretically proven to be greater than $\frac{2}{\pi}$ if a point P is inside the vortex core [4].

Γ_2 is calculated for all points, P within the PIV snapshot. However, the current formulation of calculation is extremely slows since every point P in the PIV snapshot would not only require the element wise calculation of the angles with respect to the neighbouring vectors, but also needs the calculation of the average convective velocity (with a big enough S region) around it. So, a slight modification to the $\tilde{U_P}$ is proposed. Prior to evaluation, the average mean velocity is evaluated as a function of wall-normal position by averaging over all velocity fields and in the stream-wise direction. This is then subtracted from the instantaneous velocity fields to obtain a $\tilde{U_P}$ vector field, which is the size of a single PIV snapshot. This is maintained constant for all points. Now, the individual evaluation of $\tilde{U_P}$ can be skipped and the entire Γ_2 function for a single frame can be evaluated in a single go. A comparative analysis between the two approaches showed that the resulting output fields are extremely similar and the current modification is significantly faster.

It should be noted that, in order to pragmatically evaluate the Γ_2 function, a discretized version of the 4.1 is evaluated as shown in Eq. 4.3. In current analysis, the regions S is made up of a region with a radius of 5 pixels[1].

$$\Gamma_2(P) = \frac{1}{N}\sum_{S}\frac{[PM \wedge (U_M - U_P)].z}{[\|PM\|.\|U_M - U_P\|]} \tag{4.3}$$

Locations of vortex core of a possible candidates is identified as the local maximum of the Γ_2 field. To obtain converged solutions for a single vortex, it is desirable to select an interrogation region around each candidate that is marginally larger than a measure of its radius based on Γ_2. In this way, we have the greatest chance of avoiding a solution that incorporates aspects of two or more adjacent vortices. At the same time, we know from the synthetic vortex analysis above, that vortices with sufficiently poor resolution are not suitable for analysis using this technique.

[1]The bigger the region of S, the more accurate are the results. However, increasing beyond 5 pixels showed no significant improvement in the overall output.

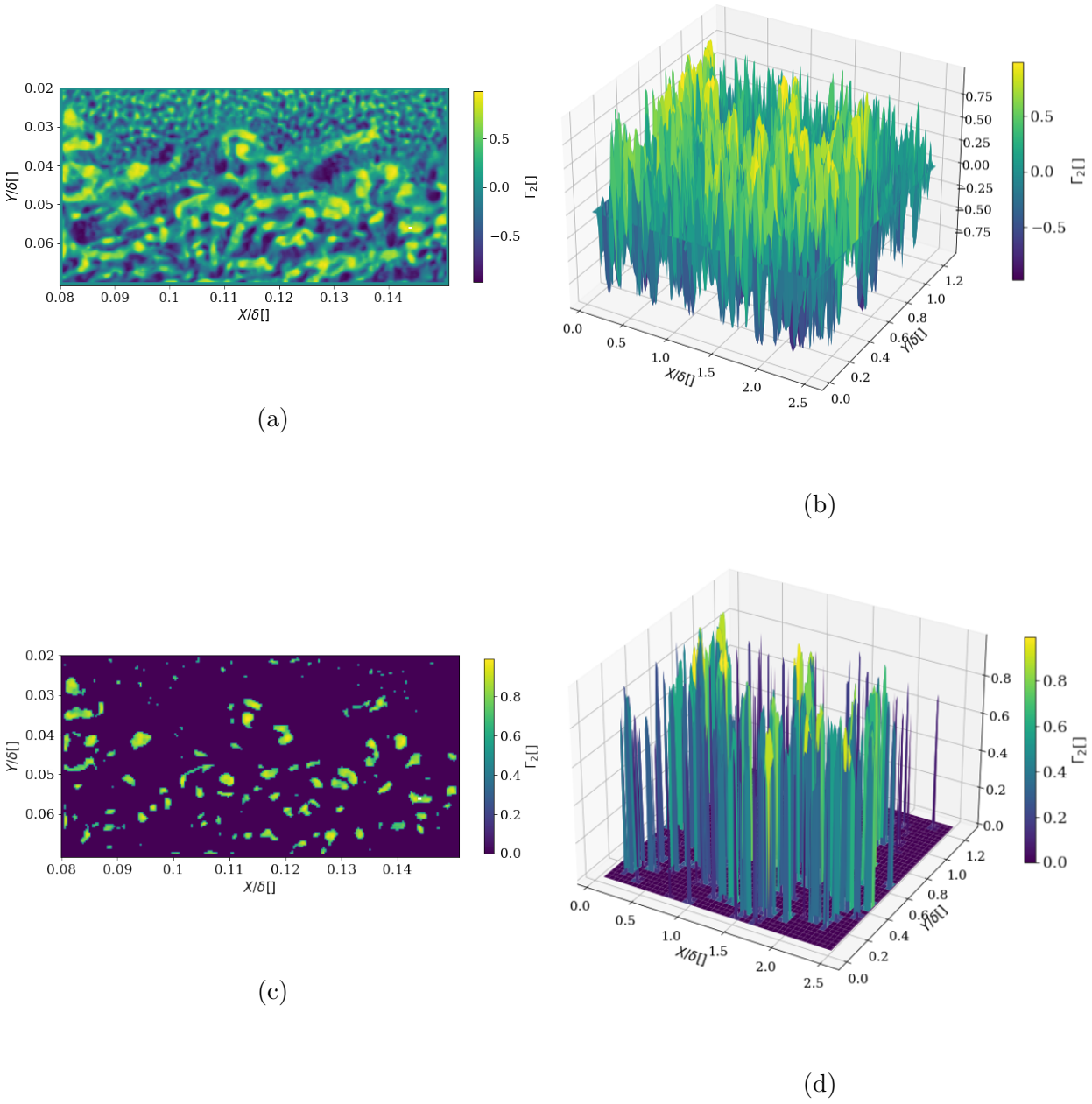


Figure 4.3: Scalar fields generated by Γ_2 operator. (a) and (b) show the 2-D and surface plots views of the entire field for an example snapshot. (c) and (d) shows the same plots after applying the threshold $\Gamma_2[\Gamma_2 < \frac{2}{\pi}] = 0$.

While the radius of the vortex could be estimated by size of the region for which $\Gamma_2 > 2/\pi$, the two dimensional width of the half-prominence of the Γ_2 function has been found to be a reliable estimate for the size of the bounding box. The two-dimensional half-prominence evaluation is not found in popular Python modules and it has been taken from an outside source[2] after careful scrutiny and customization to fit the current needs. Using this size estimation from Γ_2, all vortex candidates for which their estimated $R/\Delta X$ was less than 5, were discarded, deeming them to be insignificant for ideal optimization.

It has also been determined that vortices that are entirely (or exactly) occupying the field of consideration (i.e. the bounding box for optimization) are bad candidates. This suggests that once the initial vortex candidates are identified in a flow field, it is ideal to take a bounding box around this candidate that is slightly ($\sim$ 10-15%) bigger than the crude estimate of the radius.

Once the vortex candidates are identified, the MCMC procedure outlined for synthetic vortices in the previous chapter is used to extract a more complete set of vortex properties for the candidate. Once again to avoid interactions between the solution of adjacent vortices as much as possible a narrow, rectangular bounding box region of $\sim 1.5\ R_{est}$ is initially selected around each vortex candidate. Once, the candidate has been identified, the rectangular Bounding box is isolated and MCMC based optimization is executed, with set up conditions as mentioned in Section 3.1.3. This framework is repeated to cover all the vortex candidates identified in a frame and then repeated over all available frames.

4.3 Improving the radius estimate of the bounding box.

The corner plots shown in Fig. 4.1 are idealized version. In reality, they might not always encompass PDFs that are entirely converged or clearly show a sigle peak. The most common types of failures are either diverging PDFs or PDFs with multiple peaks in them.

Examples of corner plots exhibiting these characteristics are shown with their candidate

[2]https://github.com/Xunius/python_peak_promience2d

vortex in Fig. 4.4 and Fig. 4.5. In Fig. 4.4, the bounding box of the vortex candidate is either too small or includes part of another vortex that is dominating[3] the overall optimization. It resulted in producing a diverging PDF (extremum seeking peak) corresponding to the radius parameter in the corner plot. Fig. 4.5, on the other hand, shows multiple peaks (and divergence) in the PDF corresponding to the radius parameter. The conclusion is that this region contains of multiple vortices.

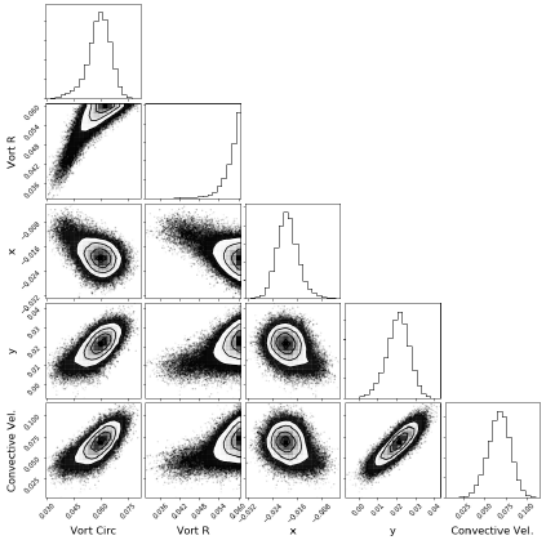


Figure 4.4: Divergence in Corner plot

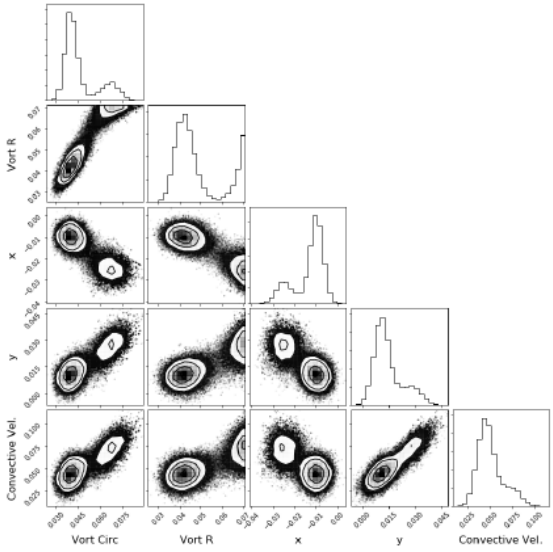


Figure 4.5: Multiple peaks in Corner plot

To conclude, If more than two peaks, are detected, this was the result of a partial adjacent vortex within the interrogation region. The choice of a small interrogation region minimized the probability of this occurring and ensured that any secondary peaks were often small within the distribution. So, corner plots as shown in Fig. 4.5 are quite rare. In case, they do occur the peak corresponding to the tallest signal is exclusively considered to be the

[3]An extensive study was conducted to see which particular parameter (say Γ, R, or V_c) or combination of parameters would make one vortex more dominating than the other, in the optimization process. But, this study was inconclusive and it is recommended to separate vortex candidates to the maximum possible extent to avoid interference from one another.

best estimate. This is inherently the governing principle behind the custom Gaussian fitting algorithm described in Section 3.1.4. This is a simple fix. But handling the divergence errors is slightly more convoluted.

At this point in narration, it is important to note that it is not by chance that both failing PDFs in Fig. 4.4 and Fig. 4.5 correspond to the radius parameter. This is a common occurrence which reinforces the importance of accurate dimensions of the bounding box, which highly influences the extracted radius of the candidate.

The fix to the divergence plots in based on the assumption that PDFs in corner plots are always tending towards the maximum of the bounds which suggests that the true radius of the vortex candidate is actually higher than what is estimated and ultimately accounted for, while designing the bounding box. So, to rectify the issue with divergence in the PDFs, the dimensions of the bounding box are iteratively increased until one of the following criterion is met: the divergence is rectified or the bounding box reaches one of the edges of the PIV frame or when the current bounding box is already (say) twice the size of the original estimate. These conditions are the stopping criterion for the corrections to the automation algorithm.

The first and second stopping criteria are obvious. The third stopping criterion needs context and works in two parts. Firstly, there has to be some limit on the number of expansions since MCMC processes are expensive to compute. So, this criterion acts as an alternative (non-dimensional) to hard thresholds. Secondly, it makes sure that extending the region of consideration does not work on identifying another more dominating vortex which is in the current candidate's neighbourhood and ultimately prevents the bounding box from becoming too big.

4.4 Cumulative results from turbulent PIV dataset

This section presents the cumulative results of using the automated framework for the identification of vortex properties on the experimental PIV turbulent boundary layer dataset. The algorithm was deployed at a non-dimensionalized height of $\frac{y}{\delta} = 0.15, 0.3$ and 0.45 over all of the 750 frames present in the dataset.

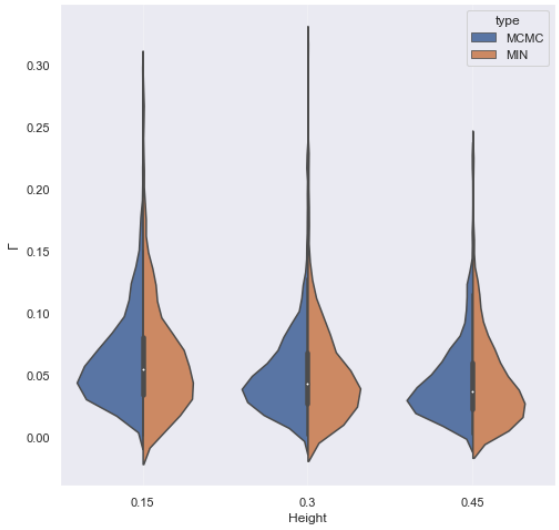


Figure 4.6: Γ distributions in PIV data

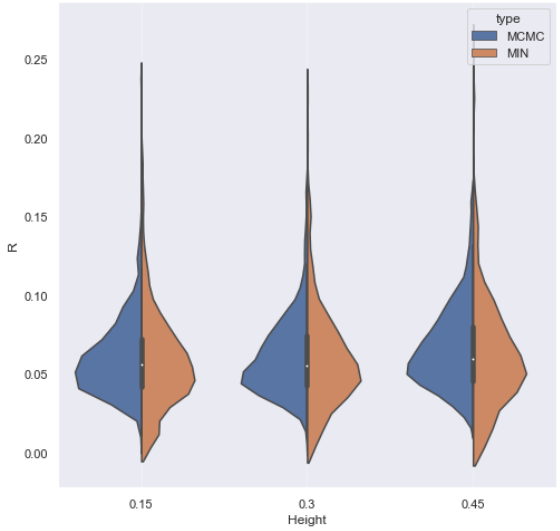


Figure 4.7: R distributions in PIV data

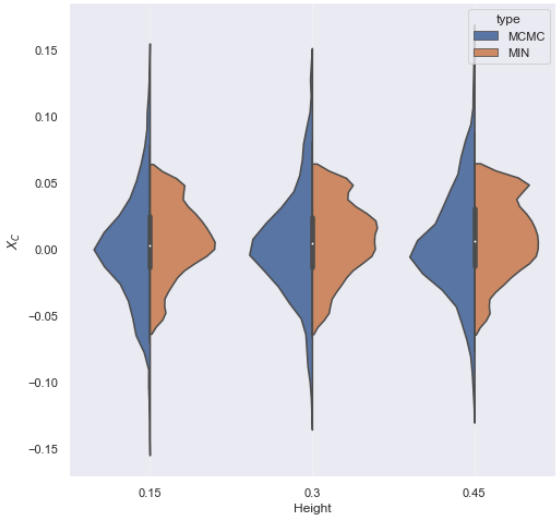


Figure 4.8: X_C distributions in PIV data

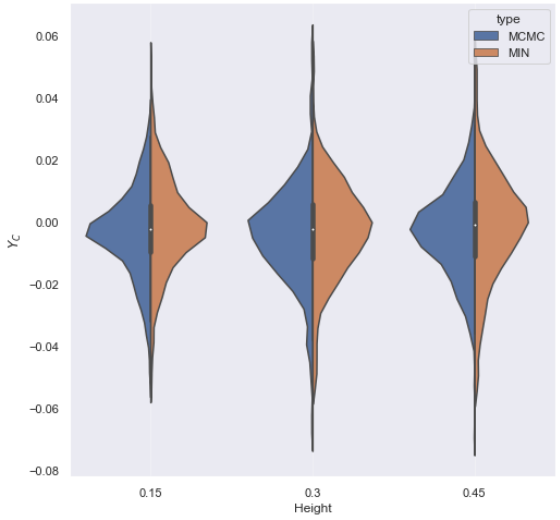


Figure 4.9: Y_C distributions in PIV data

A total of 368, 300 and 299 vortices were successfully identified at the aforementioned three heights respectively. The overall distribution of each parameter of the vortex properties can be visualized in the violin plots shown in Fig. 4.6-Fig. 4.10.

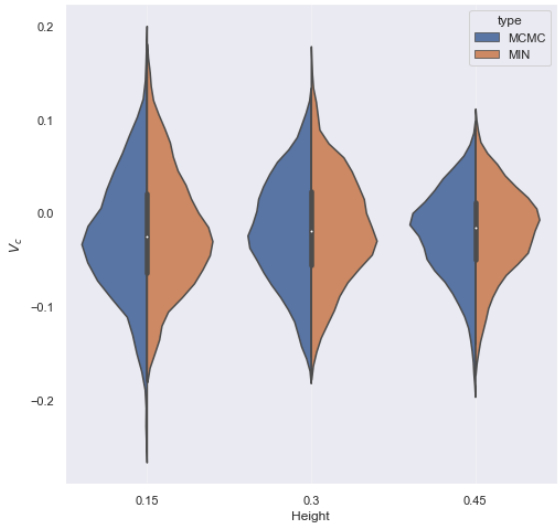
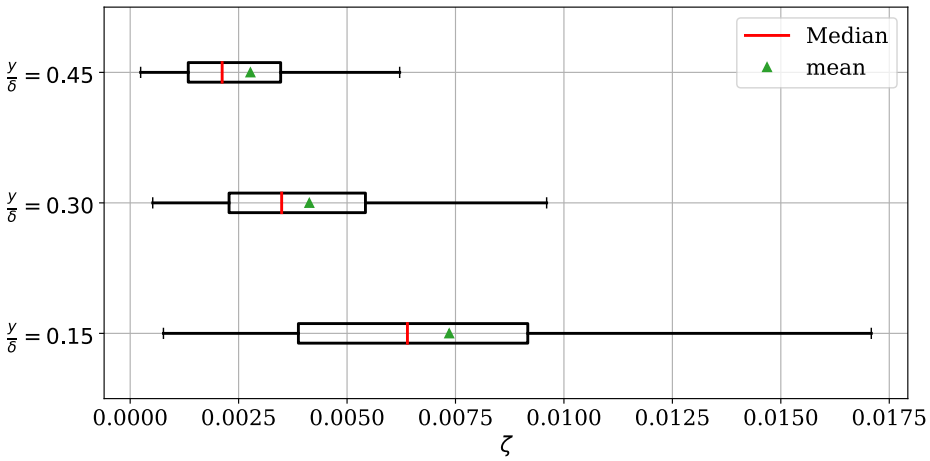


Figure 4.10: V_c distributions in PIV data

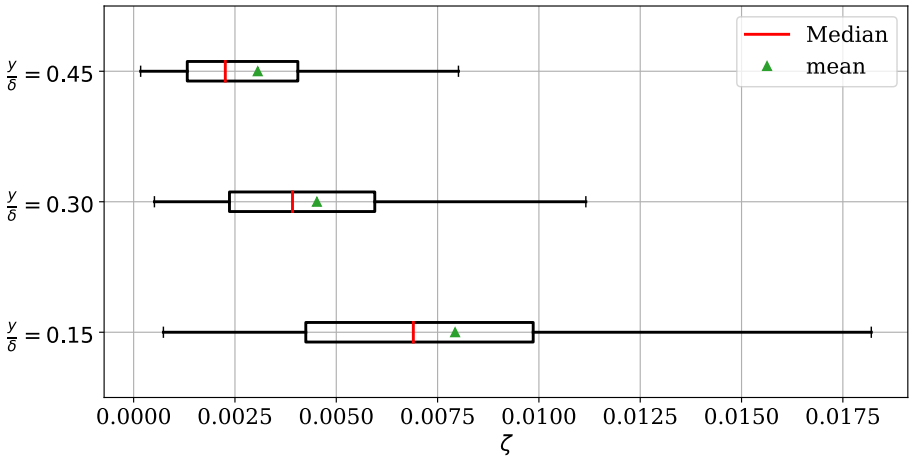
These figures are called as violin plots because of their violin like structure. Each figure has 6 distribution, with 2 distributions corresponding to each height (i.e. $y/\delta = 0.15, 0.3, 0.45$). The distributions corresponding to solutions of parameters based on Minimization (MIN) and MCMC are plotted next to each other for effective comparison, in different colors. The distributions are minimally smoothed by specifying a scaling factor while calculating the kernel bandwidth using the bw argument ($bw = 0.25$) in the Seaborn module's violinplot function.

There are quite a few interesting observations that can be made from these plots. But, it should be first noted that there were quite a few ($\sim 40\%$) cases that did not give converged solution with MCMC while using the initial bounding box dimensions calculated from the Γ_2 operator. The size of the bounding box was incrementally increased by checking for divergence in any of the parameters in the corner plots as described in the previous section. This technique gave rise to better bounding boxes and hence helped in identifying better regions of vortex candidates which gave converged solutions in corner plots. This correction would not have been possible by using any minimization (also holds for TNC) procedure

as they the user would be not as informed regarding the convergence properties of the optimization. However, for the sake of fair comparison, the minimization solution noted in the above violin plots is executed on the improved vortex candidate regions.



(a)



(b)

Figure 4.11: Box and whiskers plot showing the average ζ error for vortices identified at different heights using (a) MCMC and (b) Truncated Newton's method (Minimization).

Most plots shows smoother distributions of parameters for MCMC compared to Minimization which is often seen to be multi-modal. Both X_C and Y_C distributions are centered around 0. These are the values of displacement from the center of the vortex region. This indicates that the regions of the vortex candidates are localized with high accuracy and did not need additional corrections from future optimization. The distributions of convective velocity V_c are closely centered around 0, which means that the vortices are convecting about the stream wise mean velocity of the PIV measurements, which is consistent with boundary layer theory. It is also worth noting that the distribution of convective velocities becomes narrower and are approaching the stream-wise mean velocity as the height of consideration increases.

In order to judge the quality of the obtained solutions, it is not possible to perform a study similar to the one with synthetic vortices as the true solutions of the vortex properties are not know a priori. However, one possible measure of accuracy would be to look at how well the resulting solutions reconstruct the original field. This would mean looking at the residual after each optimization. So, the error plots are plotting using the ζ function using a box and whiskers plot to look at the accuracy of reconstruction at each height.

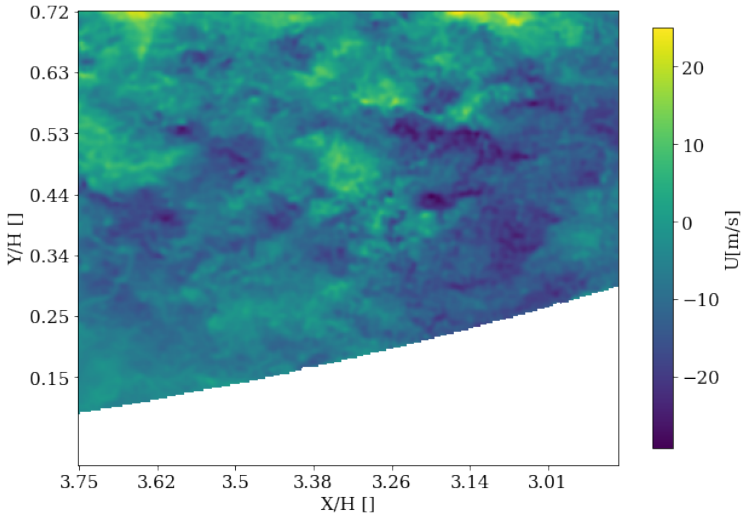
Fig. 4.11 shows that the mean and median ζ errors of reconstruction are better with using MCMC at all heights and also the range of error is universally narrower. The value of ζ error is decreasing as the height of the region of consideration is increasing. This suggests better optimization performance for vortices which are convecting at velocity closer to the stream-wise mean of the flow. This could also suggest that the Rankine vortex model is better at approximating vortices that are farther away from the wall, but further analysis is required to validate this observation.

Chapter 5

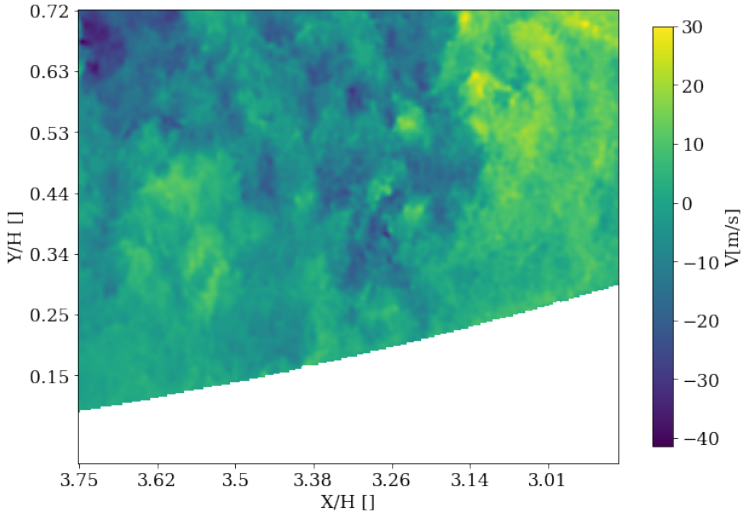
SUPER RESOLUTION (SR) OF TURBULENT SEPARATED FLOW FIELDS

This chapter examines the effectiveness of a range of SR techniques involving ANNs and interpolation based methods particularly on PIV datasets. First, the turbulent separated flow dataset is introduced. Then, the preliminary pre-processing on this data to handle the null pixel entries is detailed. The next sections will focus on the particular choice of down sampling with its advantages, disadvantages and possible alternatives.

The later sections will focus on looking at a few examples of various traditional interpolation based SR techniques and quantifies the error in reconstruction. These errors are then compared against a simple Artificial Neural Network (ANN) architecture. The size of the data available for training is around $40Gb$ and is made up of $50,000 snapshots$ of turbulent flowfields. So, before moving on to implement the training phase, it is important to address an critical question, given the vast magnitude of the dataset at hand. How much of the available 50,000 frames are sufficient for training the ANN? This is crucial for the training phase of ML. For, example if it is found out that 100 frames are just as good as all 50,000 (unlikely), it would significantly decrease the training time. The later sections would focus on arriving at a simple yet optimal architecture for the ANN. Finally, this chapter concludes by showcasing example super resolution outcomes using the ANN. This includes discussions on how well the turbulent statistics are reconstructed using various methods by observing the difference fields.



(a)



(b)

Figure 5.1: Example snapshots from the separated flow dataset. (a) shows a stream-wise velocity field and (b) shows a wall-normal velocity field. The X and Y positions are normalized with respect to the bump height, $H = 0.0766m$

5.1 Turbulent Separated flow Dataset

This section introduces the turbulent separated flow data set captured over a speed bump with a Gaussian cross sectional profile along the stream-wise wall-normal plane. This will be the primary data set used for training the ML models and deploying the super resolution pipeline subsequently. This dataset is experimentally collected using PIV at the the University of Washington's (UW-AA) $3' \times 3'$ wind tunnel.

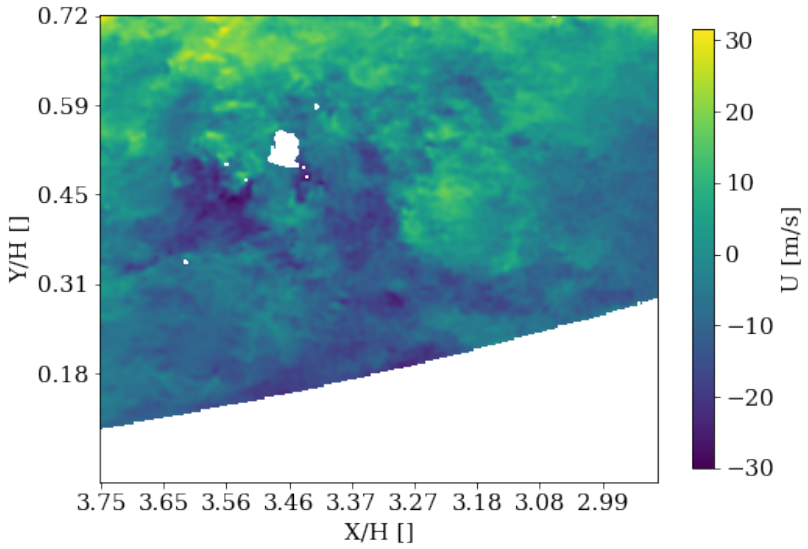
The Gaussian speed-bump geometry is mounted to a splinter plate upon which a fully turbulent boundary layer is generated. The flow is separated and approximately 2 dimensional primarily due to the fact that the geometry is significantly wider in the span-wise direction compared to the stream-wise direction. This separated flow is captured using PIV in the stream-wise wall-normal plane. The PIV set up is run on an open-return type wind tunnel with an inlet velocity of $40\frac{m}{s}$. The Reynolds number based on the tunnel width, L is $Re_L = 2.095 \times 10^6$ and based on the Gaussian bump height, h, is $Re_h = 1.795 \times 10^5$. Further details about the PIV setup and flow conditions surrounding the bump can be found in the the works of Williams et al. [55, 56].

In the current work, only the stream-wise velocity components are considered for SR. The stream-wise velocity fields ($\tilde{u}$) are decomposed as shown in Eq. 5.1.

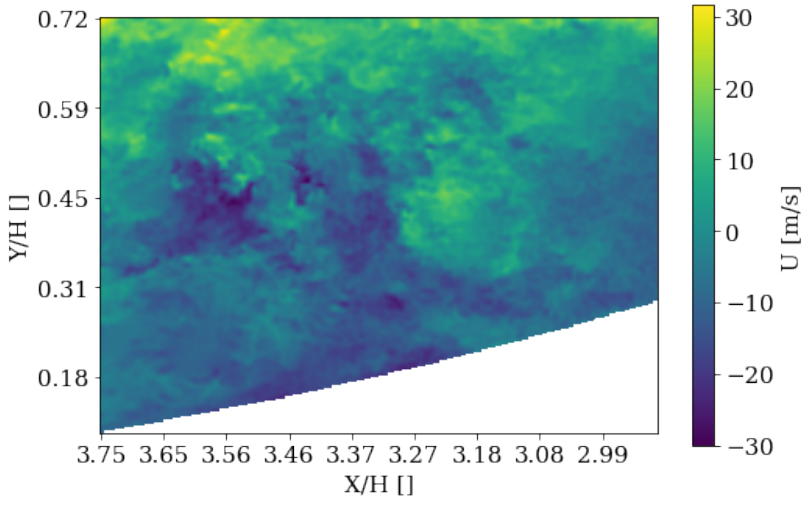
$$\tilde{u} = \overline{U} + u' \tag{5.1}$$

This is known as Reynolds decomposition and separates the mean flow profile ($\overline{U}$) from the velocity fluctuations (u'). It is very important to normalize the inputs and the outputs given to any ML model during the training phase. To implement this, the mean flow profile was initially subtracted and the fluctuations were normalized with respect to the local maximum. These normalized field of fluctuations are referred to as u'_n.

5.2 *Pre-Processing PIV data*



(a)



(b)

Figure 5.2: An example stream wise velocity snapshot before imputing (a) missing values and after imputing (b) missing values.

Particle Image velocimetry (PIV) is a robust technique to capture the instantaneous velocity measurements in a flow field in an approximately non-invasive manner[39]. In this method, the flow is embedded with sufficiently small tracer particles and their displacement is measured over time to calculate the velocity vectors. This is an extremely robust and widely accepted technique to capture the intricate turbulent flow characteristics. However, a very small percentage of flow field pixels are usually corrupted due to various forms of noise. These corrupted pixels can be detected as outliers as strongly differ from the magnitudes of the neighbouring pixels[39]. For simplicity there pixels are replaced by null values during the data processing stage of PIV, thus giving the individual user of this dataset more freedom regarding how they want to handle these outliers, at a later stage.

Before importing the PIV dataset into the training stage for an ANN, it is important to modify the null values of these outliers based on what seems appropriate to the application at hand. This will be the primary goal of pre-processing the current PIV dataset of turbulent separated flow.

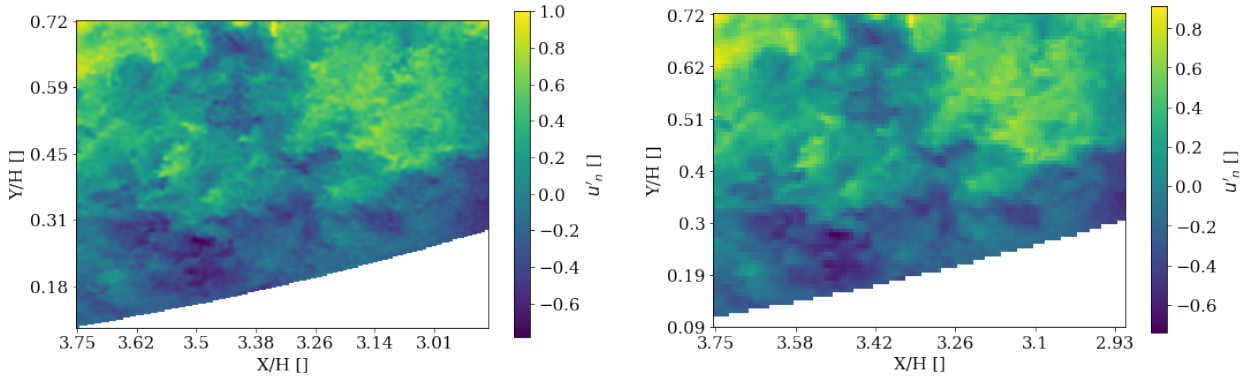
To motivate the reason behind this particular objective, it is valid to point out that manipulating null values is always an issue, that could raise errors in numerous stages. Even at the elementary level of handling basic operation like addition, multiplication and gradient based operations, might fail based on the specific type of python module at hand. Even popular modules like SciPy fail to effectively handle NaN values, unless consciously taken care during pre-processing. Eg. Scipy Interpolate module's Interp2d function. This could drastically influence the training procedure while tuning the weights and biases of ML models, due to the excessive reliability of the backpropagation algorithm on calculating precise gradients.

Interestingly, there are quite a lot of simple fixes to overcome this problem. For current application, the KNNImputer function from Sci-kit learn's impute module is chosen. The KNN stands for K nearest neighbours and they impute (assign something by inference) the missing values by replacing them with the average of the k nearest neighbors. For current operations k was chosen as 2.

Fig. 5.2 shows an example snapshot and how the impute function helped filled the null values. It should be noted that this example snapshot is not arbitrary and it was deliberately chosen to magnify the impact of the missing values for visualization purposes. Most snapshots only have a few, if any, missing pixels. It should be noted that in Fig 5.2 that the pixels corresponding to the bump region also exist as NaN values. This bump region was masked out before imputing the missing values inside the flow field.

To conclude, only a minute proportion ($< 0.023\%$) of the total number of pixels are benefited from this step, however, it is important to ensure that any future operations do not raise any errors and also contributes to the general robustness of this dataset. An example of a more advanced technique to rectify corrupted PIV data can be found in the works of Scherl et al.[57]

5.3 Downsampling high resolution flow fields



(a) (b)

Figure 5.3: An example of a high resolution snapshot (a) and its low resolution counterpart (b) down-sampled by 3 times.

Downsampling refers to the process of converting a high resolution image/ simulation into a low resolution one. Since, super resolution techniques aim to improve the resolution of

a flowfield, we initially prepare a low resolution sample that will later be upscaled and compared with the original high resolution sample. There are numerous methods to acquire downsampled/low-resolution data.

This current work utilizes average pooling to reduce the dimensions of PIV snapshots. In average pooling the value of a pre defined block/part of an image is replaced by the average of the all the pixels making up this block.

Fig. 5.3, shows an example snapshot that has been downsampled by using average pooling. It was downsampled by a factor of 3. The original dimensions of the high resolution snapshot are 311 pixels by 261 pixels. This was converted to a low resolution image of size 103 pixels by 87 pixels. This was achieved by using a bi-linear (weighted average pooling) interpolation from the image module in Python's Pillow library. The factor of downscaling, three, was maintained throughout the remainder of this work.

5.4 Traditional up-sampling methods for super resolution

Interpolation based techniques are simple and classical techniques that interpolate the value in between two pixels based on the neighbouring pixel information. Linear interpolation solves for the linear regression (1^{st} order) and takes two points (each in horizontal and vertical directions) as the inputs. Bicubic and quintic interpolation solve for the cubic (3^{rd} order) and quintic (5^{th} order) polynomial by employing 4 and 6 points respectively on either directions. There can be employed to implement super resolution by using the Scipy's interpolate function.

In this section, the results of using such interpolation techniques are presented and the losses of reconstruction are evaluated so that they can be compared against ML based results in the future section. The crucial takeaway is to establish a benchmark that could measure the effectiveness of current proposed advancements.

Figs. 5.3 shows the high and low resolution snapshots of an example frame. The following plots in Fig.5.4 show the high resolution snapshots generated from the low resolution image (Fig. 5.3) using Linear, bi-cubic and quintic interpolations respectively. The output of these

algorithms, i.e. the high resolution images have a resolution of 311 by 232 pixels.

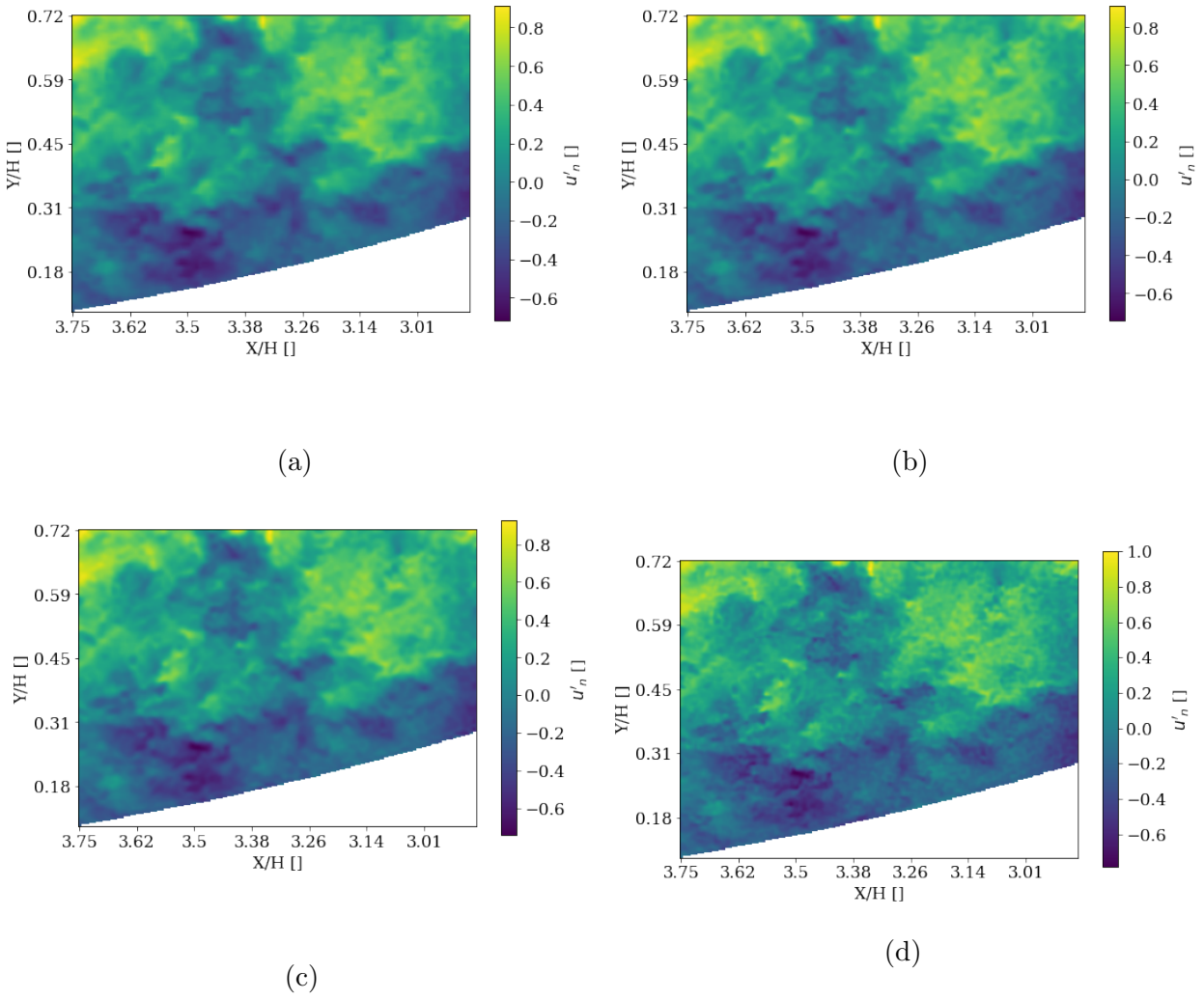


Figure 5.4: Super resolution using Linear (a), Bi-cubic (b) and Quintic (c) interpolation techniques. Their corresponding $\mathcal{L}_{SR}$ loss values, based on comparison with true high resolution snapshots are 0.2012, 0.2031 and 0.2039 respectively. (d) the true high resolution snapshot.

A Forbenius norm based loss function is introduced to examine the differences between high resolution and interpolated images. For simplicity this is referred to as the $\mathcal{L}_{SR}$ in

future.

$$\mathcal{L}_{SR} = \frac{\|X_{OG} - X_{RC}\|_F}{\|X_{OG}\|_F} \tag{5.2}$$

where X_{OG}, X_{RC} refer to the original (say Fig. 5.3a) and the reconstructed (say Fig. 5.3b) flow fields and $\|.\|_F$ refers the the Frobeinus norm. Alternatives to measure the loss include the Mean Squared Error, Mean Absolute Error, Mean Absolute Percentage Error or the Mean squared Logarithmic error. These can also be used to quantify the error, however the magnitudes of these errors do not indicate how close the reconstruction is to the true solution. The $\mathcal{L}_{SR}$ almost varies in between 0 and 1 and this number can be viewed as an approximate percentage estimate of the amount of data that has been not resolved by the super resolution technique.

The $\mathcal{L}_{SR}$ values for different interpolation techniques can be observed in the Fig.5.4. It should be noted that the difference in the loss values for different interpolation techniques are exaggerated for clarity and the values on the y-axis make this clear. The true values are within 1% of each another.

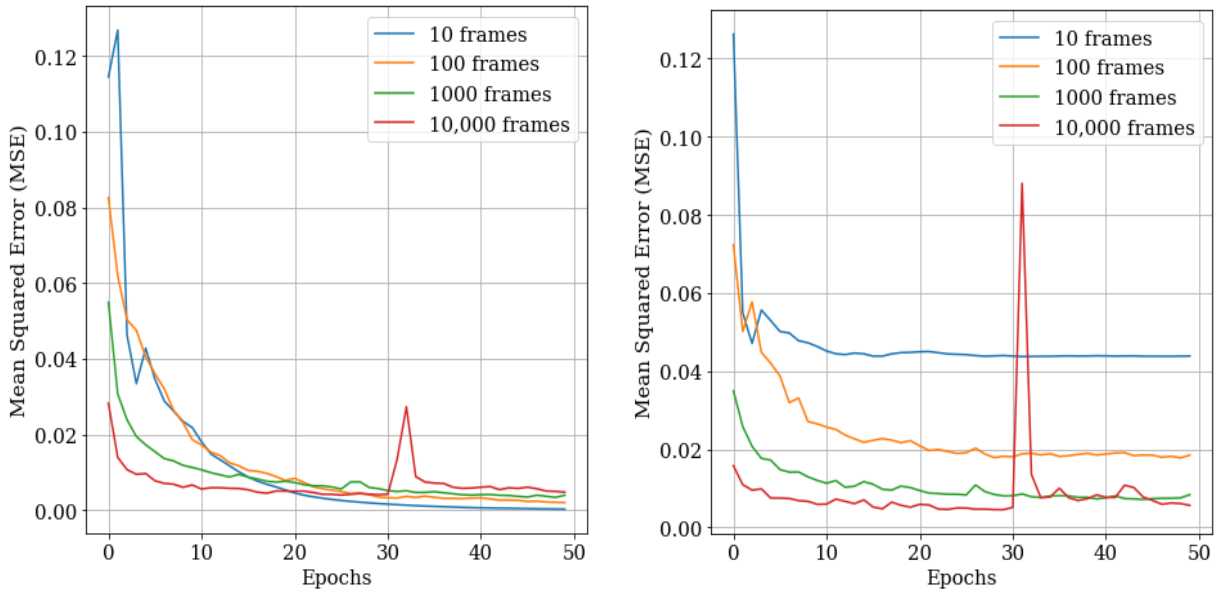
The loss value indicates that the loss due to reconstruction is approximately around 13%. This quality can be confirmed through a juxtaposition with their high resolution output plots. Apart from the pixel wise measurement of the reconstruction, the quality of reconstruction for the variables measuring the turbulence in the snapshot is also an important measure. This will be calculated and compared against the reconstruction performance of an artificial neural network in the next section.

5.5 Super Resolution using ANNs

In order to begin any ML pipeline, it is important to form an understanding of the nature of the data at hand, the kind of ML architecture that would be ideal for this data and the computational resources for executing the training and later pre-processing stages.

The current working dataset has $50,000$ snapshots of high resolution streamwise and

spanwise velocity vector components. They amount to almost $40GB$ in memory[1]. It is not a trivial task to load the entire dataset into RAM and begin training. Also, it wouldn't be wise to train using the entire dataset, if using half of it would give equally valid results. It is ideal to have a large dataset, so that it can be truncated (or not) based on the computational/ memory limitations of the working setup, because of the broadband nature of turbulence which is anticipated to be harder to train. *So, how much data is actually required?*



(a) Training Losses

(b) Validation losses

Figure 5.5: Training (a) and Validation (b) loss (MSE) trends while using datasets with different magnitudes.

To simplify this quest, a simple Artificial Neural Network (ANN) architecture was chosen to run on an Intel (R) Xeon (R) CPU E3 -1240 V2 @3.40Gz with 8GB of RAM.

After preliminary tuning with a small dataset, an initial ANN architecture was selected with one hidden layer that contains 2500 neurons and Exponential linear unit (elu) activation

[1]The prefetch() functions in tensorflow provide effective alternatives to tackle this issue.

function was used with an adaptive moment estimation based optimizer (ADAM) and Mean Squared Error based loss. The output layer has a linear activation function. ADAM is one of the most popular optimizer for updating ANN weights that has shown progressive results and the elu activation function is a fast and robust alternative to widely used relu activation for handling input data centered around 0. Explicit details about the ultimately utilized ANN architecture, tuning and training phases are detailed in the next section.

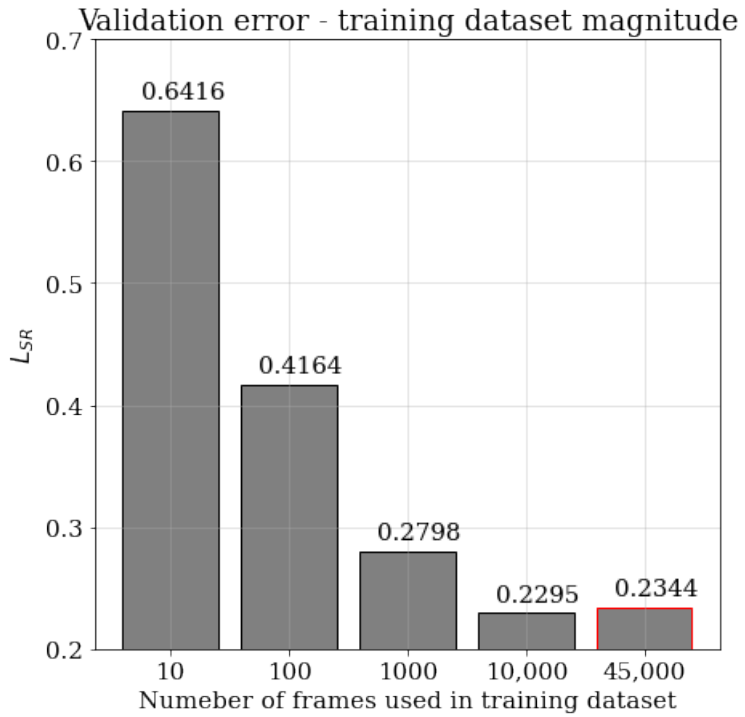


Figure 5.6: Average L_{SR} for different magnitudes of training datasets

The inputs to the ANN are flattened arrays of the low resolution snapshots (as seen in Fig.5.3) with a size of 6709 pixels. That is 103 times 76 pixels and then removing the pixels corresponding to the Gaussian Bump. This translates to a 3 time up-scaling using ANNs to uncover the original resolution of the PIV data.

From the Fig. 5.5a, Fig.5.5b, the loss trends (MSE) for training procedure with different orders of magnitudes can be observed. The training MSE loss begins to converge at around

30 epochs. For safe conclusions, they were executed until 50 epochs. Clearly, the training loss (Fig. 5.5a) converges to similar values for data with any order of magnitude. The testing loss (Fig. 5.5b) is a more certain measure of how well the model learnt the underlying mapping without over-fitting. This shows an interesting pattern, where datasets with 10 and 100 frames clearly converge at comparatively higher errors, indicating overfitting in those runs. It suggests that a dataset with an order of magnitude of 1000 or $10,000$ frames is the way to move forward. The similarity of these validation loss values proves that it is sufficient to train on a subset of entire $50,000$ frames to generate optimal benefits with the current ANN architecture and this specific amount of downsampling (3).

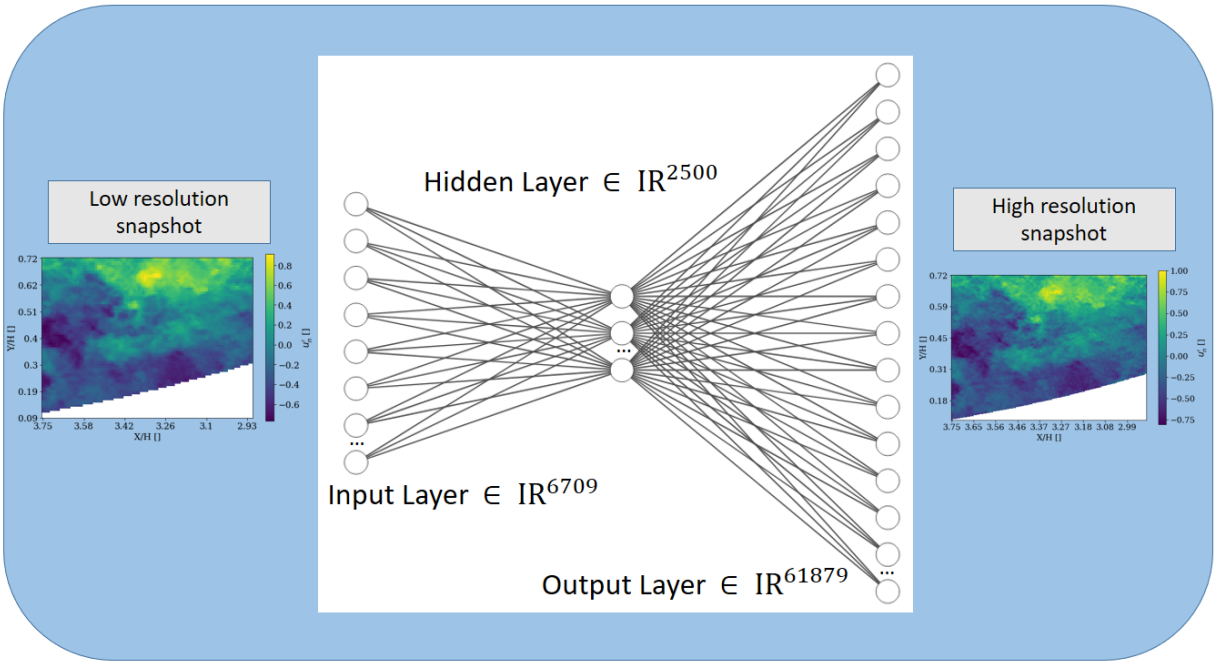


Figure 5.7: ANN architecture for super resolution of a 3 times down-sampled low resolution field from 103×76 pixels to 311×232 pixels.

Fig. 5.6 shows that beyond $10,000$ frames the loss value does not improve further when utilizing the entire dataset. The value for $50,000$ is highlighted in red. Here $45,000$ frames

were used in the training and a subset of the remaining $5,000$ frames were used for validation and testing. This conclusively recommends the use of approximately $10,000$ frames of high resolution data without any noticeable differences in the resulting outcomes for the current ANN based architecture.

The spike in the training and testing losses corresponding to the dataset with $10,000$ frames can be attributed to using mini-batches which is a compromise between using Stochastic Gradient Descent (SGD) and complete batch with a ADAM optimizer. In order to more closely clip the range of acceptable order of the magnitude of the dataset, the mean $\mathcal{L}_{SR}$ loss value provides an alternative overview.

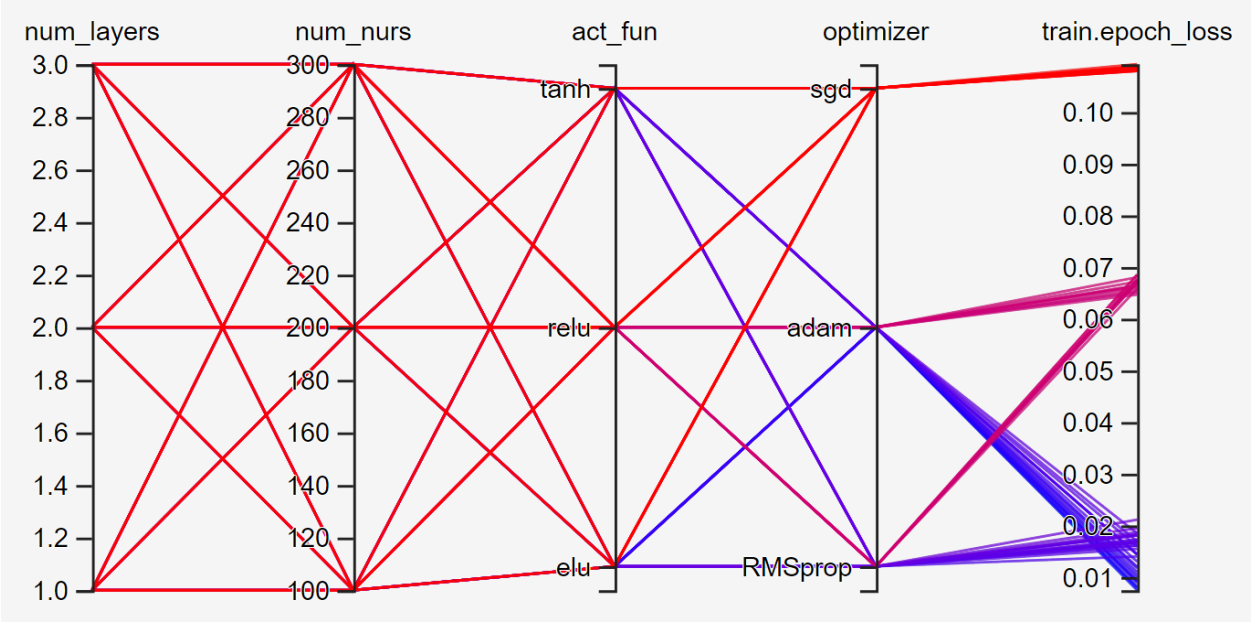


Figure 5.8: Tensorboard's HP parallel coordinates view dashboard.

5.5.1 Hyper parameter tuning and training the ANN

This subsection focuses on elaborating the specific structure of the ANN that will be employed to generate upcoming results. Initially, the procedure employed for tuning the model hyper-parameters using Tensorflow's Tensorboard and details about the ultimately deployed

model is described. Finally, this subsection is concluded by presenting the progression of training on the aforementioned ANN model.

Any ML model, including ANN, is governed by a significant number of hyper parameters (HPs) that critically affect the performance of the model. However, these are not like the parameters (weights and biases) that are modified during the training process and are defined apriori. Hyperparameters in ANNs include the number of hidden layers in the model, the number of neurons in each of those hidden layers, the type of activation function, the learning rate, the loss function, the type of optimizer employed and so on. Some of these are numerical while the others are categorical variables. There present a large number of possible combinations (search space) and it is a challenging task to find the optimal set of HPs which is within reason given the computational capabilities.

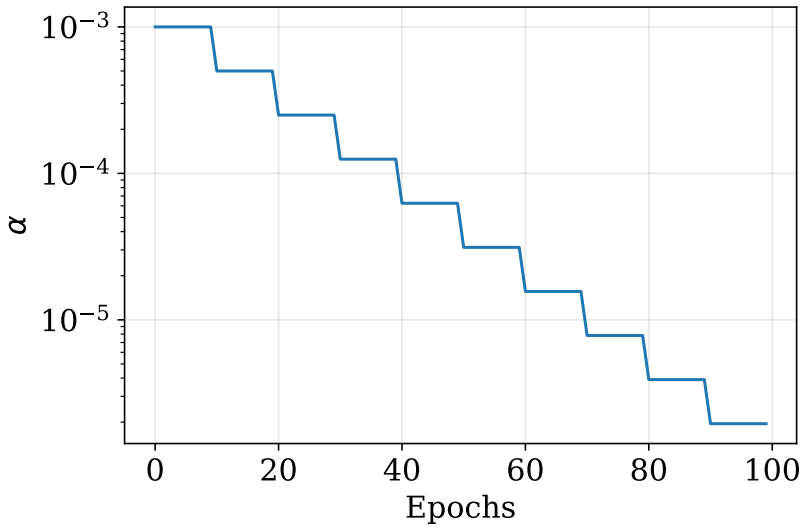


Figure 5.9: A step-wise decaying learning rate (α) over 100 epochs.

For preliminary testing, the number of layers was varied amongst 1, 2, and 3. The activation functions[2] were varied amongst tanh(), relu(), and elu(). The number of neurons

[2]This activation function was given in all the hidden layers and the output layer.

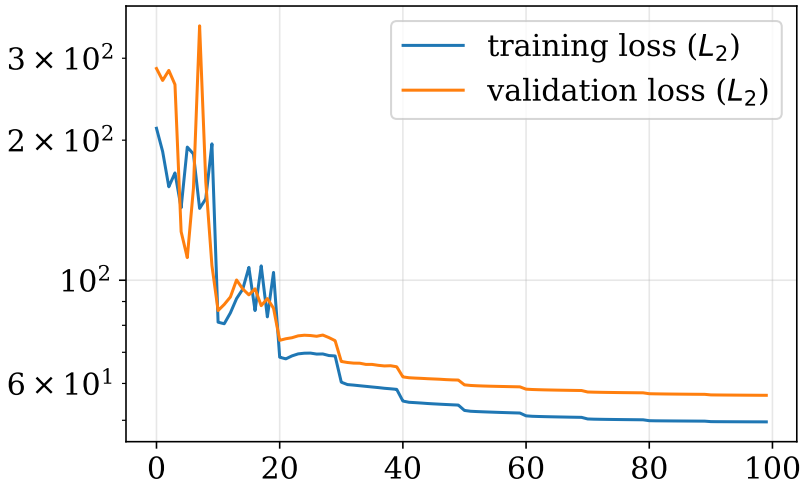
was varied amongst 100, 200 and 300. The choice of optimizer was narrowed down to ADAM, SGD and RMSprop. The model was trained by giving the $\mathcal{L}_2$ norm of the difference between the true and predicted fields as the loss function, $\mathcal{L}_2$ (Eq. 5.3) that being optimized.

$$\mathcal{L}_2 = \|Y_{predicted} - Y_{true}\|_2 \tag{5.3}$$

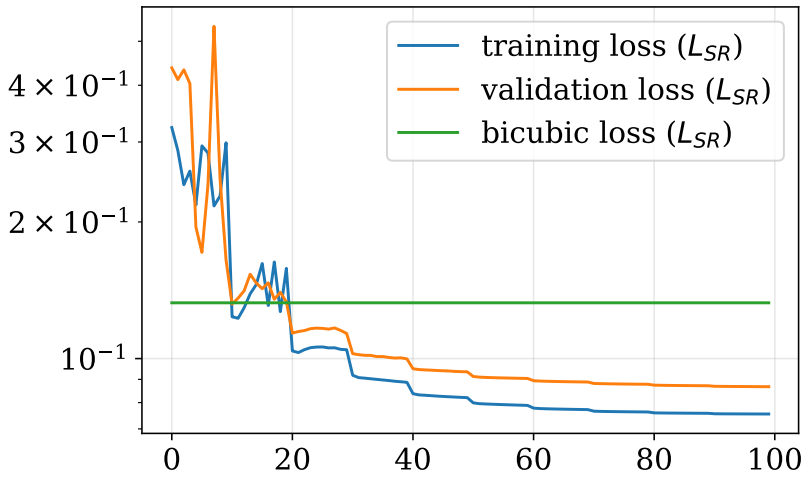
Python's Tensorflow module has a comprehensive resource for tuning Hyper tuning called as Tensorboard. By enabling a callback feature during training all these different combination of HPs (81 models), Tensorboard's parallel coordinates view (Fig. 5.8) provides a clear visualization to not only understand the best combination of HPs, but also learn about the most crucial hyper parameter. This map shows how each combination of HPs, map to the training loss (*train.epoch_loss*) after 3 epochs on $5,000$ snapshots, which is shown in the last column of the Fig. 5.8. Each combination is colored based on the final loss value, varying between red for high and blue for low loss values. From Fig. 5.8, it is clear that the best combination, with the lowest training loss value, has one hidden layer with 300 neurons and uses an Exponential linear unit (elu()) as the activation function and the ADAM optimizer. It can also be noticed that tanh() also an equally good enough choice. But relu() clearly performed the worst. This can attributed to the fact that the inputs and outputs are normalized between -1 and 1 and the outputs from relu are always positive. However, the results concerning the number of neurons and the number of layers was not as clearly obvious.

This set of hyper parameters were used as an initial guide to settle on the model that was ultimately used, after further hyper parameter tuning: one hidden layer with 2,500 neuron, with an elu() activation function and a linear activation function at the output layer. A learning rate scheduler was integrated into the model that begins to reduce the learning rate (α)[3] in a step wise manner after the first 10 epochs, based on the Eq. 5.4.

[3]learning rate determines the step size at each iteration while moving towards minimizing loss. A low learning rate would indicate small changes to ANN weights, and a higher one would indicate bigger changes. It is advisable to begin with a higher learning rate and gradually decrease its value.



(a)



(b)

Figure 5.10: The $\mathcal{L}_2$ loss during the training phase (a). The $\mathcal{L}_{SR}$ loss (b) is chosen as an observable training metric. This loss is compared against the average of $\mathcal{L}_{SR}$ loss for the validation dataset using bicubic interpolation to capture if and when the ANN shows superior efficiency.

Here $\alpha_{n=[1,2,..10]} = 0.001$, where n is the the current epoch number and % indicates the modulus operator.

$$\alpha_{n+1} = \begin{cases} \alpha_n/2 & \text{if } n\%10 = 0 \\ \alpha_n & \text{if } n\%10 \neq 0 \end{cases} \tag{5.4}$$

The α profile can be seen in Fig. 5.9. Here, Y-axis is in log scale. An early stopping callback is also included to preserve the best weights of the model, if it is found out that the model is starting to overfit[4], with a patience of 10 epochs. Weight decay and other regularizers did not show any significant improvements. The model was allowed to train for 100 epochs using $20,000$ snapshots of stream-wise velocity flowfields of turbulent separated flow dataset. A batch size of 64 was found to be ideal for implementation after preliminary trials.

This ANN model contains approximately 170 million parameters. It took 16 hours to train the model for 100 epochs on an Intel (R) Xeon (R) CPU E3-1240 V2 @ 3.40 GHz and 8 Gb of RAM. Fig. 5.7 gives an overview of the architecture employed in the current research[5].

A validation dataset of $5,000$ (25% the size of training datset) frames was used during the training. The training losses ($\mathcal{L}_2$) and the training metrics ($\mathcal{L}_{SR}$) are shown in Fig. 5.10. In this figure, the average value of $\mathcal{L}_{SR}$ while implementing SR with bicubic interpolation is plotted as a flat line. It can be seen that the performance of an ANN surpasses the performance of a classical bicubic interpolation after 20 epochs of training. A significant drop in the training loss and metrics can be observed directly as a consequence of a corresponding reduction in the learning rate.

---

[4]when validation loss begins increasing

[5]The pixels corresponding to the bump region, that are NaNs were discarded before inputting into the ANN. This is reflected in the size of the input layer and output layers, which are slightly smaller than 103×76 and 311×232 respectively

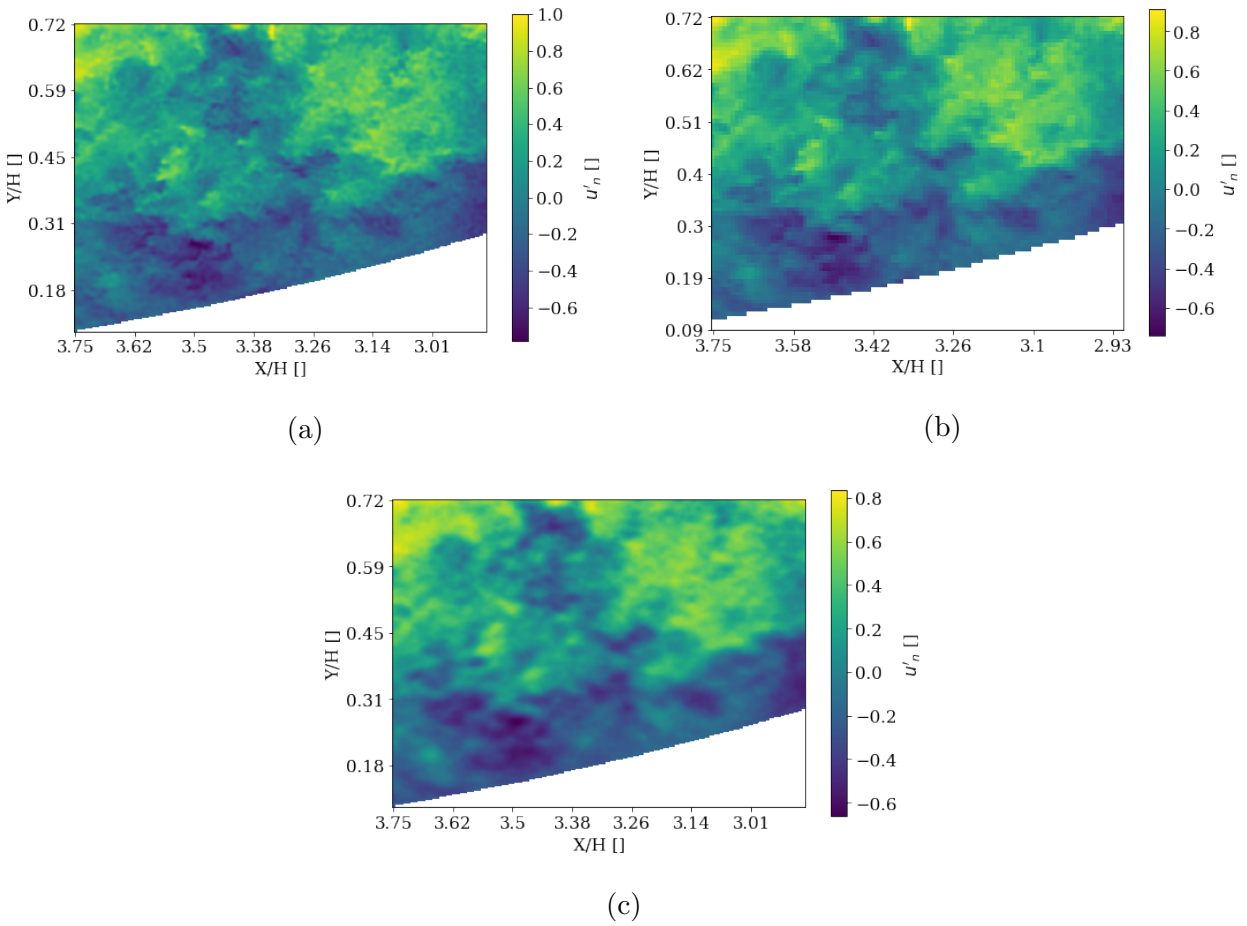


Figure 5.11: The example high resolution (a) and low resolution (b) are repeated here for convenient comparison. (c) shows the SR produced using an ANN. The corresponding $\mathcal{L}_{SR}$ loss while using the ANN is 0.1766.

5.5.2 Analysing ANN results and comparing them with interpolation methods.

Fig. 5.11 shows an example snapshot taken from the validation dataset to show how well the ANN performs its task. This shows lower L_{SR} loss values compared to reconstructions seen with interpolation methods (Fig. 5.4).

The average $\mathcal{L}_{SR}$ loss value for SR for all the $1,000$ samples present in the validation dataset is compared against the quintic, bicubic and linear interpolation methods in Fig.

5.12. This definitively proves the superiority of the ANN model for SR in comparison to other interpolation methods.

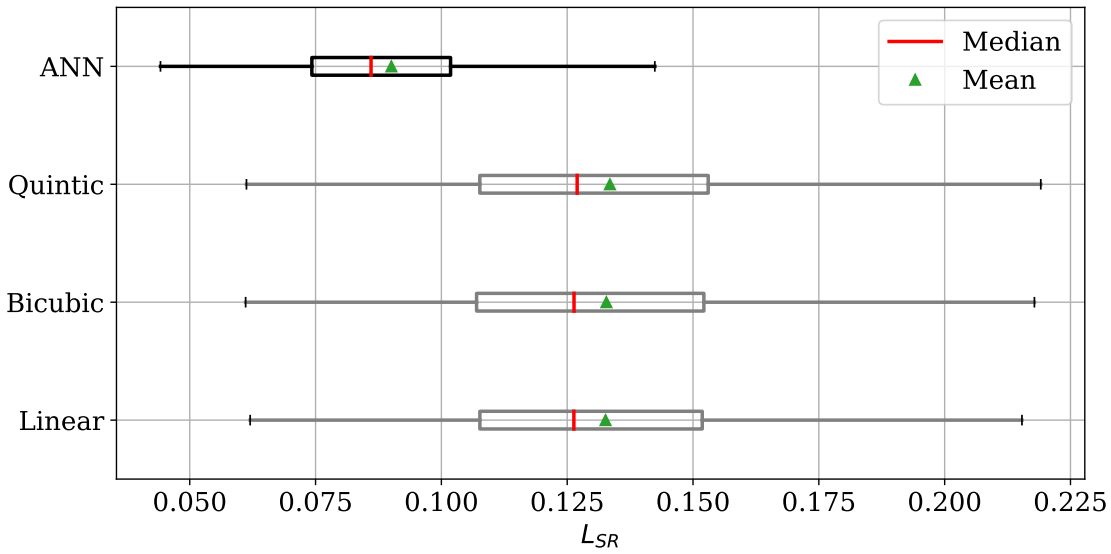


Figure 5.12: Comparison of the average of $\mathcal{L}_{SR}$ loss values for $1,000$ frames present in the validation dataset.

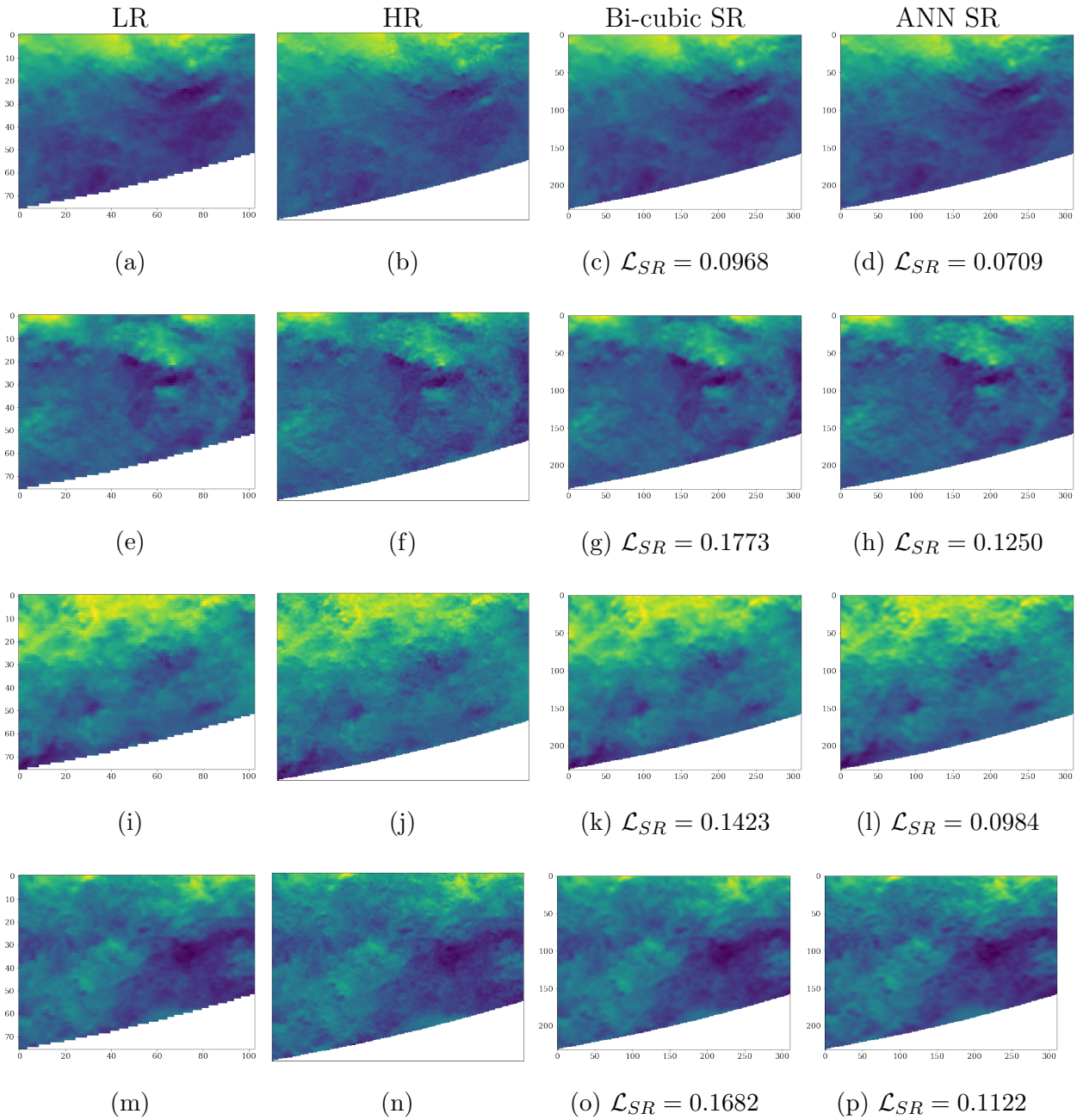


Figure 5.13: Super resolution examples. The figures in the first and second columns represent the low (LR) and high (HR) resolution snapshots. The LR samples are down sampled by 3 times. The third column show reconstructions using bi-cubic interpolation and the final column shows ANN based interpolation. The $\mathcal{L}_{SR}$ losses are also mentioned for the last two columns.

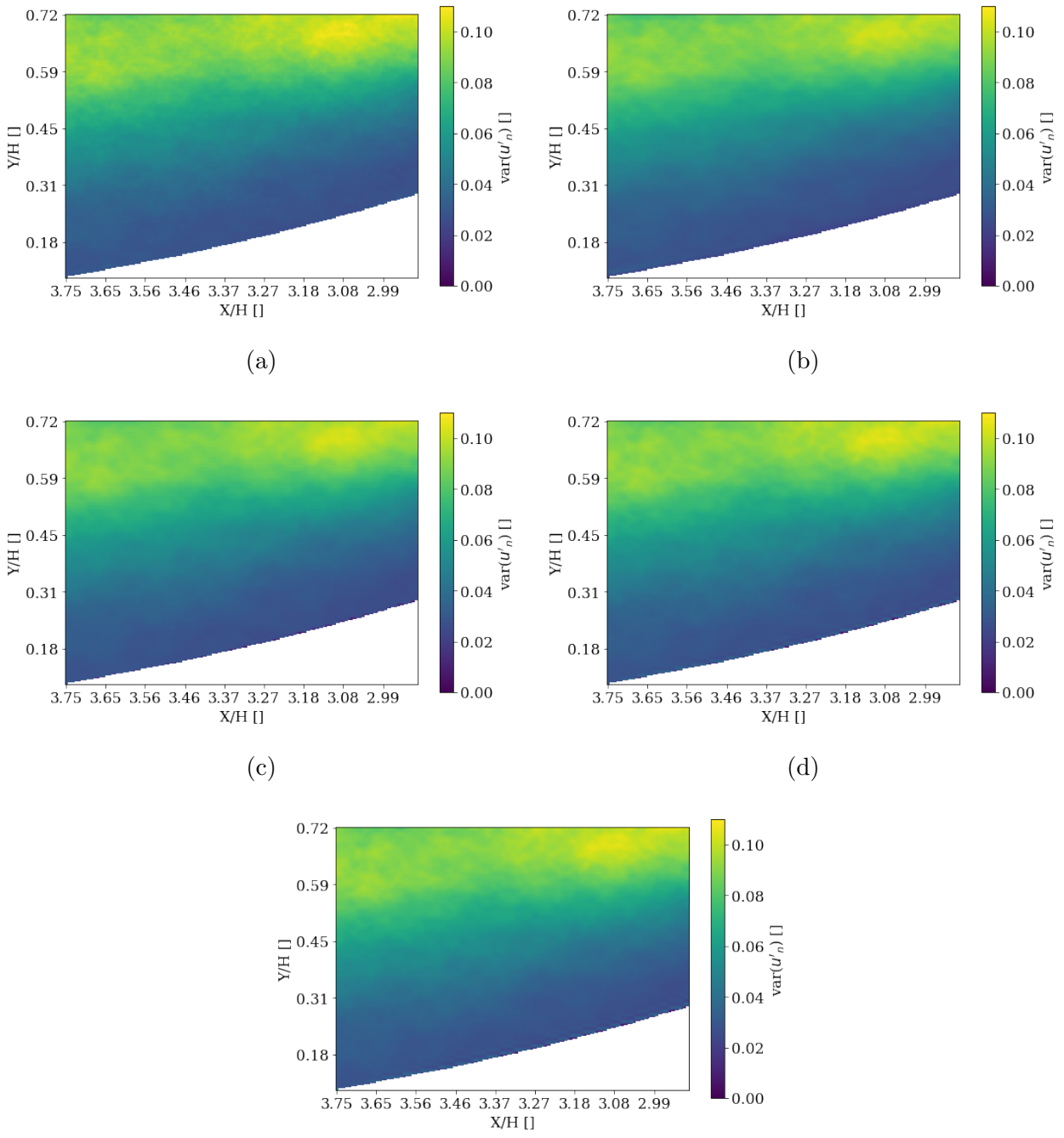
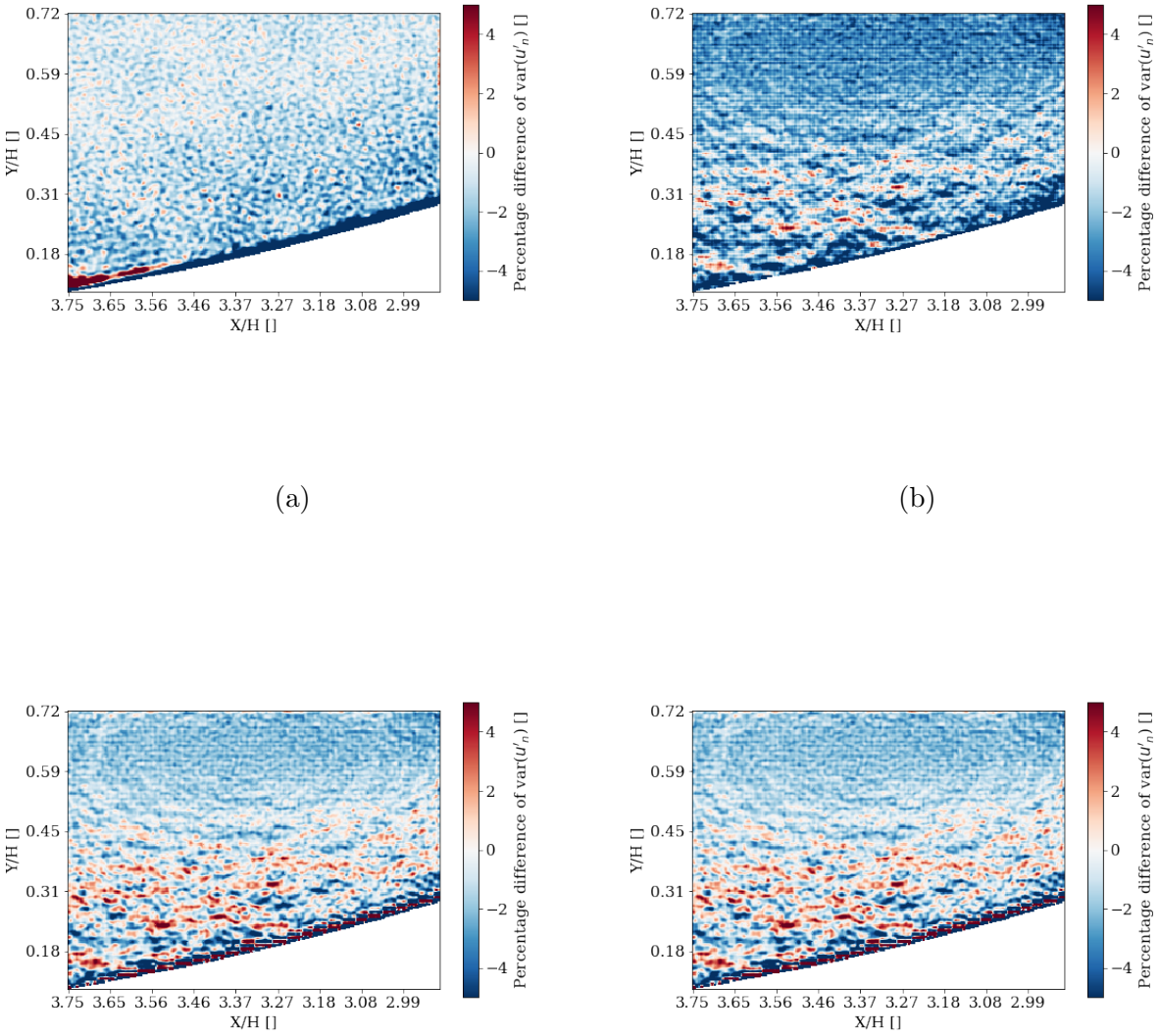


Figure 5.14: True variance of the velocity fluctuations (u'_n), over $1,000$ snapshots selected from validation dataset is shown in (a). Its counterparts generated using the ANN (b), linear interpolation (c), bi-cubic interpolation (d) and quintic interpolation (e) are also shown here. Their corresponding $\mathcal{L}_{SR}$ values are 0.0164, 0.027, 0.0271 and 0.0352 respectively.



(a) (b)

(c) (d)

Figure 5.15: Absolute percentage difference with respect to the true variance of fluctuations for the ANN (a), linear (b), bicubic (c) and quintic interpolations (d).

Fig. 5.14 shows the variance of streamwise velocity fluctuations. Turbulent flow fields are innately different from natural images like pictures of animals, tress or humans in the sense that they are fundamentally derived from first principles and hence have to obey certain restrictions. In this spirit, it is an interesting exercise to check how well these flow field reconstructions translate to accurate reconstructions of turbulent derived quantities like the the in-plane turbulence intensity along the stream wise velocity components i.e. the variance of the velocity fluctuations. These figures are generated utilizing $1,000$ frames selected from the validation dataset.

The true variance of the chosen subset of data show smooth turbulent characteristics, which is consistent with all the other SR based reconstructions. It is worth noting that the $\mathcal{L}_{SR}$ loss values produced by the ANN is better than the remaining interpolation based cases.

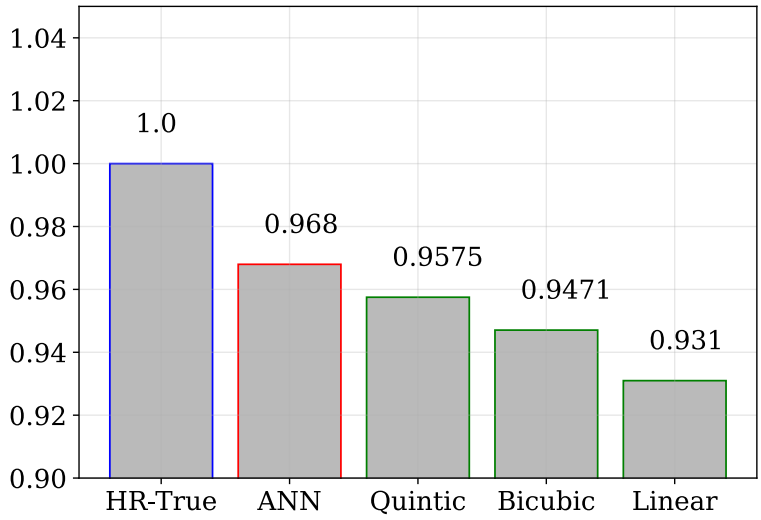


Figure 5.16: Average of the frame-wise maximum absolute value of the velocity fluctuations (u'_n) in all frames of validation data.

In order to further analyze the quality of the variance plots, the difference plots of each

SR method with respect to the original variance field, are shown in Fig. 5.15. The colorbar is clipped to vary between -5 and 5 %, to clearly visualize the subtleties. The crucial observation is that the performance of ANNs show great promise which is evident from the largely while regions, corresponding to a % error. The major contribution of error from interpolation based techniques is failure to accurately replicate properties close to the bump wall. The ANN, on the other hand, not only shows a lower error closer to the wall, but also exhibits much smoother behavior.

Interpolation based super resolution methods are not expected to capture the true richness of turbulent coherent structures as they simply fill-in missing values in between pixels using neighbouring cell values. As a result, some turbulent energy that was filtered by down sampling does not get restored by interpolation i.e. the magnitude of interpolated velocity field variances are always smaller than the original high resolution dataset. However, the average flow characteristics are retained through interpolation methods. However ANNs are unbounded by such restraints and can reconstruct variations to much closer extents.

Fig. 5.16 shows the magnitude of the average maximum pixel value of the instantaneous velocity fluctuations calculated over the 1000 frames in the validation dataset. Since the data is normalized, the truth value (from high resolution snapshots) of this measure is 1. It can be observed that interpolations with higher order schemes are more closely preserving this average measure, but yet again the ANN clearly outperforms them.

Chapter 6

CONCLUSIONS

For the sake of clarity, this chapter on conclusions and future works is divided into two parts. The first part focuses on the identification of turbulent coherent structures and the next part focuses on the super resolution of turbulent PIV measurements.

6.1 Coherent structure identification.

The current work presents a completely automated framework to extract vortex properties of all the vortices present in a PIV snapshot based a single measurement of the stream-wise and wall-normal velocity components. Previous approaches towards identification of vortex properties heavily relied on the a priori knowledge or iterative searching for the convective velocity of the vortex (V_c). However, this barrier was successfully overcome by the current MCMC framework, which predicts the V_c along with other vortex properties. This framework is robust to reasonably bad initial estimations of vortex candidates properties and can iteratively update the regions of consideration in an automated fashion, which is shown to be of critical importance. Finally, current work generates the probability distribution functions of properties of all vortices present in the dataset along with a height wise segregation, which can be very helpful in analyzing interactions between coherent structures in turbulent flow.

A traditional minimization based optimization using Truncated Newton's (TNC) algorithm for maximum likelihood estimation was used a baseline to compare the performance of MCMC based optimization. It has been found out that the Bayesian Optimization and TNC perform equally well for low noise, well resolved synthetic data but MCMC was better with higher noise.

It was found out that the quality of MCMC quality broke down when radius of vortex was

less than 5 vectors, which was used as a threshold to ignore smaller vortex candidates from the optimization process. In that sense, the synthetic vortex results informed optimization parameters for matching within PIV data.

Using this MCMC framework, it was possible to extract distributions of vortex properties for chosen distances from the surface within a turbulent boundary layer. From these distributions, it was found that the minimization solutions were found to be broader than MCMC, indicating poorer matching to RVM for at least some candidates. MCMC procedure extracted convective velocities whose values were more likely to be the mean boundary layer velocity at the wall, as expected for turbulent boundary layers. This is consistent in both optimization methods (MCMC and TNC). However, the parameter distributions from TNC were often multi-modal, particularly in the X_C and Y_C distributions. Unlike MCMC, where they were much more smoother, and more in agreement with the theoretical expectations.

In the absence of good vortex identification methods to get an initial estimate of the vortex candidate, a small window of consideration can be moved across the flowfield to identify vortices. This would be very useful to identify other complicated coherent structure.

The overarching objective of the current work is to present an automated framework for identifying the properties of coherent structures. By modifying the log posterior function in MCMC, this framework can be extended to work on other coherent structures. An immediate extension to the current work could be the integration of a saddle point model in addition to the Rankine vortex model to capture Hairpin vortex signatures and hairpin packets.

Bayesian Inference is an excellent tool to handle complex and high dimensional objective functions which hints that the current work is scalable to handle more complicated and higher dimensional coherent structure models, where traditional minimization approached would completely fail or take impractically long times to converge. Future direction could include modifying the type of sampling algorithm used for MCMC and utilizing other vortex models (other than Rankine) for developing the log posterior function.

The current work utilizes a flat prior, which is proven to work well. Developing better priors could lead to faster and more robust MCMC based optimization. It is very likely

that the key to successfully isolate two vortex candidates present in a single region of consideration could be the development of better prior functions. As one possible alternative, the distribution of vortex properties over different heights that is available at hand could be made to influence the prior function. This will continually be improved as more vortices are identified. However, it should also be noted that strong priors can significantly influence the optimization process. This task is a double edged sword and is left for the future.

Recalling the effect of resolution on the overall optimization process, it have been identified that a vortex with an estimated radius of 5 pixels is considered as a bad candidate for optimization. However, it would be interesting to explore the effects on machine learning based super resolution techniques to improve the resolution of the candidate as a pre-processing step, before implementing the matching process. This could significantly remove the influence of resolution in measurement and contribute towards the robustness of the bayesian optimization process.

It was initially planned to extract the information about the length scales of vortices at different heights by looking at the buffer and overlap regions of the boundary layer. However, the limitations posed by the resolution restricted that study as vortices that were too close to the wall (and hence too small) were excluded as bad candidates. By removing the effects of resolution, it could be possible to clearly visualize the various length scales of vortices present at different vertical heights from the wall.

Current framework is entirely written in Python and is openly available [1] and fairly simple to use. This framework is composed of many auxiliary functions that could be useful for a lot of other applications involving signal and image processing, that could be utilized independently.

[1] https://github.com/kommalapatisahil/CoherentStructures

6.2 Super resolution in turbulent PIV measurements

The current work explores the possibility of utilizing Machine Learning (ML) for the super resolution (SR) of mutli-scale turbulent PIV data. This work successfully proves that a simple shallow ANN could be utilized to outperform traditional interpolation based methods. Recalling the initial objective, the questions on the required magnitude of the training dataset and the accuracy in reconstructing turbulent characteristics are also explored. It was shown that a training dataset of the order of $10,000$ frames is sufficient for best results, which is training a model to produce low average L_{SR} loss values. For safety, $20,000$ frames of PIV were utilized in the training phase. Tuning the hyper parameters of the model was the most challenging task.

Hyper parameter tuning showed that the best model contains a single hidden layer with 2500 units and uses *elu()* activation function. The activation function in the output layer was linear. An ADAM optimizer was used to update the weights with a step wise decaying learning rate. Kernel, bias and activity based regularizers did not show any improvements in validation loss. The ANN model with these hyper parameters showed an average of 9% error, while the errors corresponding to all the interpolation based super resolution was higher than 13%.

The post-training results show that the ANN outperforms the interpolation methods in terms of reconstructing both the instantaneous velocity measurements and also the derived turbulent fields. The loss in reconstructing the stream wise variance of velocity fluctuations with the ANN was 1.6%. For linear, bicubic and quintic interpolations, it was 1.64%, 2.71%, and 3.52% respectively. Additionally, a measure of predicting turbulent intensities is also measured using the average of the maximum of absolute velocity vector value. The ANN still outperforms the interpolation methods, and it is estimated that it falls short by 3.2% in meeting results from the high resolution control. The plots in Fig. 5.15 show the performance of ANN and other methods in analyzing the absolute difference with respect to the true variance of fluctuations. From the the difference plot for the ANN, it could be inferred that

the majority of the error arises close to the bump region. Based on this observation, it can be concluded that the performance of an ANN in sub regions of turbulent flows that are away from the bump could report significantly lower average loss values.

It should be noted that the above results are only valid for super resolution with 3 times upscaling and no conclusions can be made about upscaling with higher factors.

The next steps towards development could include testing how well these learnt weights and biases can be transferred for the super resolution of other turbulent flow fields, with different configurations. Future direction could focus on making the model more robust and less erroneous.It would be interesting to explore the limits of super resolution using ML and particularly at higher factors of down sampling. It is worth pointing out that shallow networks (like the current implementation, with one hidden layer) are easier to tune compared to their deeper counterparts. The error in reconstruction could potentially be reduced by using deeper networks with more hidden layers, but they would require finely tuned hyper parameters to do so, which is a strenuous and computationally expensive task left for the future. An easy boost could aslo be expected by the use the convolutional layers before adding fully connected layers, as proven in the past literature.

The super resolution field from the ANNs still look blurry compared to traditional interpolation methods, despite showing lower loss L_{SR} loss values. This hints at room for improvement in the development of loss functions that are more conscious of the flow physics. This would suggest integrating the wall-normal velocity vector fields along with the stream wise velocity vector fields into the training data set. It would also be interesting to explore various kinds of down sampling algorithms and how they effect the overall SR performance.

If the ML architecture is too over constrained by the disproportionately larger size of the high resolution snapshots with respect to the low resolution snapshots, an alternative would be initially learn the latent representation of a high resolution snapshot using non linear auto-encoders or simply SVD and then train the ML model to learn the mapping between the low resolution snapshot and the latent representation of the high resolution snapshot. There are a lot of creative directions to pursue which could improve the performance of the

model.

Training deep learning models on such large datasets is not a trivial problem and requires the data to be handled in precisely tailored methods and the use of high performance computational resources. One of the exciting outcomes of the research from this thesis is that all the trained models are available openly. The model architecture with the trained weights and biases can be used to implement transfer learning with fine tuning to suit other applications.

The codes used for this work are build using open source machine learning libraries and an example workflow of the current pipeline in Python is available online [2].

[2]https://github.com/kommalapatisahil/superResolution

REFERENCES

[1] Adrian, R. J., "Hairpin vortex organization in wall turbulence," *Physics of Fluids*, Vol. 19, No. 4, 2007, pp. 041301.

[2] Adrian, R. J., Meinhart, C. D., and Tomkins, C. D., "Vortex organization in the outer region of the turbulent boundary layer," *Journal of fluid Mechanics*, Vol. 422, 2000, pp. 1–54.

[3] Mulleners, K. and Raffel, M., "The onset of dynamic stall revisited," *Experiments in fluids*, Vol. 52, No. 3, 2012, pp. 779–793.

[4] Graftieaux, L., Michard, M., and Grosjean, N., "Combining PIV, POD and vortex identification algorithms for the study of unsteady turbulent swirling flows," *Measurement Science and technology*, Vol. 12, No. 9, 2001, pp. 1422.

[5] Kommalapati, S., Agrawal, A., and Duryodhan, V. S., "Enhancing miscible fluid mixing by introducing pseudo turbulent flow in Golden ratio spiral microchannel," *Industrial & Engineering Chemistry Research*, Vol. 59, No. 9, 2019, pp. 3784–3793.

[6] Sirovich, L., "Turbulence and the dynamics of coherent structures. I. Coherent structures," *Quarterly of applied mathematics*, Vol. 45, No. 3, 1987, pp. 561–571.

[7] Epps, B., "Review of vortex identification methods," *55th AIAA aerospace sciences meeting*, 2017, p. 0989.

[8] Vétel, J., Garon, A., and Pelletier, D., "Vortex identification methods based on temporal signal-processing of time-resolved PIV data," *Experiments in fluids*, Vol. 48, No. 3, 2010, pp. 441–459.

[9] Lennie, M., Steenbuck, J., Noack, B. R., and Paschereit, C. O., "Cartographing dynamic stall with machine learning," *Wind Energy Science*, Vol. 5, No. 2, 2020, pp. 819–838.

[10] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A., "Hyperband: A novel bandit-based approach to hyperparameter optimization," *The Journal of Machine Learning Research*, Vol. 18, No. 1, 2017, pp. 6765–6816.

[11] Friston, K. J., Penny, W., Phillips, C., Kiebel, S., Hinton, G., and Ashburner, J., "Classical and Bayesian inference in neuroimaging: theory," *NeuroImage*, Vol. 16, No. 2, 2002, pp. 465–483.

[12] Erichson, N. B., Mathelin, L., Yao, Z., Brunton, S. L., Mahoney, M. W., and Kutz, J. N., "Shallow neural networks for fluid flow reconstruction with limited sensors," *Proceedings of the Royal Society A*, Vol. 476, No. 2238, 2020, pp. 20200097.

[13] Fukami, K., Fukagata, K., and Taira, K., "Super-resolution reconstruction of turbulent flows with machine learning," *Journal of Fluid Mechanics*, Vol. 870, 2019, pp. 106–120.

[14] Bradshaw, P., *An introduction to turbulence and its measurement: thermodynamics and fluid mechanics series*, Elsevier, 2013.

[15] Theodorsen, T., "The structure of turbulence," *50 Jahre Grenzschichtforschung*, Springer, 1955, pp. 55–62.

[16] Richardson, L. F., "The supply of energy from and to atmospheric eddies," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, Vol. 97, No. 686, 1920, pp. 354–373.

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

[18] Levy, Y., Degani, D., and Seginer, A., "Graphical visualization of vortical flows by means of helicity," *AIAA journal*, Vol. 28, No. 8, 1990, pp. 1347–1352.

[19] Zhang, S. and Choudhury, D., "Eigen helicity density: a new vortex identification scheme and its application in accelerated inhomogeneous flows," *Physics of fluids*, Vol. 18, No. 5, 2006, pp. 058104.

[20] Saffman, P. G., *Vortex dynamics*, Cambridge university press, 1992.

[21] Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B., *Bayesian data analysis*, CRC press, 2013.

[22] VanderPlas, J., "Frequentism and bayesianism: a python-driven primer," *arXiv preprint arXiv:1411.5018*, 2014.

[23] Brooks, S., Gelman, A., Jones, G., and Meng, X.-L., *Handbook of markov chain monte carlo*, CRC press, 2011.

[24] Lewis, H. W., *Why Flip a Coin?: the art and science of good decisions*, Wiley New York, 1997.

[25] Foreman-Mackey, D., Hogg, D. W., Lang, D., and Goodman, J., "emcee: the MCMC hammer," *Publications of the Astronomical Society of the Pacific*, Vol. 125, No. 925, 2013, pp. 306.

[26] Goodman, J. and Weare, J., "Ensemble samplers with affine invariance," *Communications in applied mathematics and computational science*, Vol. 5, No. 1, 2010, pp. 65–80.

[27] Chib, S. and Greenberg, E., "Understanding the metropolis-hastings algorithm," *The american statistician*, Vol. 49, No. 4, 1995, pp. 327–335.

[28] Andrieu, C., De Freitas, N., Doucet, A., and Jordan, M. I., "An introduction to MCMC for machine learning," *Machine learning*, Vol. 50, No. 1, 2003, pp. 5–43.

[29] Hoffman, M. D. and Gelman, A., "The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo." *J. Mach. Learn. Res.*, Vol. 15, No. 1, 2014, pp. 1593–1623.

[30] Patil, A., Huard, D., and Fonnesbeck, C. J., "PyMC: Bayesian stochastic modelling in Python," *Journal of statistical software*, Vol. 35, No. 4, 2010, pp. 1.

[31] Team, S. D. et al., "PyStan: the Python interface to Stan," *Version 2.16. 0.0*, 2017.

[32] Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M. A., Guo, J., Li, P., and Riddell, A., "Stan: a probabilistic programming language." *Grantee Submission*, Vol. 76, No. 1, 2017, pp. 1–32.

[33] Brunton, S. L., Noack, B. R., and Koumoutsakos, P., "Machine learning for fluid mechanics," *Annual Review of Fluid Mechanics*, Vol. 52, 2020, pp. 477–508.

[34] Kutz, J. N., "Deep learning in fluid dynamics," *Journal of Fluid Mechanics*, Vol. 814, 2017, pp. 1–4.

[35] Sahil, K. and Bhattacharya, A. K., "Accurate Replication of Simulations of Governing Equations of Processes in Industry 4.0 Environments with ANNs for Enhanced Monitoring and Control," *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, 2019, pp. 1873–1880.

[36] Duraisamy, K., Iaccarino, G., and Xiao, H., "Turbulence modeling in the age of data," *Annual Review of Fluid Mechanics*, Vol. 51, 2019, pp. 357–377.

[37] Dong, C., Loy, C. C., He, K., and Tang, X., "Learning a deep convolutional network for image super-resolution," *European conference on computer vision*, Springer, 2014, pp. 184–199.

[38] Raissi, M., Perdikaris, P., and Karniadakis, G. E., "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, Vol. 378, 2019, pp. 686–707.

[39] Adrian, L., Adrian, R. J., and Westerweel, J., *Particle image velocimetry*, No. 30, Cambridge university press, 2011.

[40] Hart, D. P., "Super-resolution PIV by recursive local-correlation," *Journal of Visualization*, Vol. 3, No. 2, 2000, pp. 187–194.

[41] Takehara, K., Adrian, R., Etoh, G., and Christensen, K., "A Kalman tracker for super-resolution PIV," *Experiments in Fluids*, Vol. 29, No. 1, 2000, pp. S034–S041.

[42] Milano, M. and Koumoutsakos, P., "Neural network modeling for near wall turbulent flow," *Journal of Computational Physics*, Vol. 182, No. 1, 2002, pp. 1–26.

[43] Hubel, D. H. and Wiesel, T. N., "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, Vol. 160, No. 1, 1962, pp. 106–154.

[44] Hornik, K., Stinchcombe, M., and White, H., "Multilayer feedforward networks are universal approximators," *Neural networks*, Vol. 2, No. 5, 1989, pp. 359–366.

[45] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al., "Mastering the game of go without human knowledge," *nature*, Vol. 550, No. 7676, 2017, pp. 354–359.

[46] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning internal representations by error propagation," Tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[47] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y., *Deep learning*, Vol. 1, MIT press Cambridge, 2016.

[48] LeCun, Y., Bengio, Y., and Hinton, G., "Deep learning," *nature*, Vol. 521, No. 7553, 2015, pp. 436–444.

[49] Giaiotti, D. B. and Stel, F., "The Rankine vortex model," *University of Trieste-International Centre for Theoretical Physics*, 2006.

[50] Loiseleux, T., Chomaz, J., and Huerre, P., "The effect of swirl on jets and wakes: Linear instability of the Rankine vortex with axial flow," *Physics of Fluids*, Vol. 10, No. 5, 1998, pp. 1120–1134.

[51] Foreman-Mackey, D., "corner. py: Scatterplot matrices in Python," *The Journal of Open Source Software*, Vol. 1, 2016, pp. 24.

[52] Foreman-Mackey, D., "corner.py: Scatterplot matrices in Python," *The Journal of Open Source Software*, Vol. 1, No. 2, jun 2016, pp. 24.

[53] Williams, O., Hohman, T., Van Buren, T., Bou-Zeid, E., and Smits, A. J., "The effect of stable thermal stratification on turbulent boundary layer statistics," *Journal of Fluid Mechanics*, Vol. 812, 2017, pp. 1039–1075.

[54] Williams, O. J., "Density effects on turbulent boundary layer structure: from the atmosphere to hypersonic flow," *Princeton University*, 2014, pp. 1–330.

[55] Williams, O., Samuell, M., Sarwas, E. S., Robbins, M., and Ferrante, A., "Experimental study of a CFD validation test case for turbulent separated flows," *AIAA Scitech 2020 Forum*, 2020, p. 0092.

[56] Williams, O. J., Samuell, M., Robbins, M. L., Annamalai, H., and Ferrante, A., "Characterization of separated flowfield over Gaussian speed-bump CFD validation geometry," *AIAA Scitech 2021 Forum*, 2021, p. 1671.

[57] Scherl, I., Strom, B., Shang, J. K., Williams, O., Polagye, B. L., and Brunton, S. L., "Robust principal component analysis for modal decomposition of corrupt fluid flows," *Physical Review Fluids*, Vol. 5, No. 5, 2020, pp. 054401.