

CRANFIELD UNIVERSITY

HAZEM ASHOR AMRAN ABOLHOLL

VORTEX CORE DETECTION IN TURBULENT
SIMULATIONS BASED ON MACHINE LEARNING
APPROACHES

SCHOOL OF AEROSPACE

TRANSPORT AND MANUFACTURING

CENTRE FOR COMPUTATIONAL ENGINEERING SCIENCES



PHD THESIS

ACADEMIC YEAR: 2022 – 2023

SUPERVISOR: DR TOM-ROBIN TESCHNER

ASSOCIATE SUPERVISOR: DR IRENE MOULITSAS

APRIL, 2023

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE

TRANSPORT AND MANUFACTURING

CENTRE FOR COMPUTATIONAL ENGINEERING SCIENCES

PHD THESIS

ACADEMIC YEAR: 2022 – 2023

HAZEM ASHOR AMRAN ABOLHOLL

VORTEX CORE DETECTION IN TURBULENT
SIMULATIONS BASED ON MACHINE LEARNING

APPROACHES

SUPERVISOR: DR TOM-ROBIN TESCHNER

ASSOCIATE SUPERVISOR: DR IRENE MOULITSAS

APRIL, 2023

This thesis is submitted in partial fulfilment of the requirements for the degree of
Doctor in Philosophy in Aerospace

©Cranfield University, 2023. All rights reserved. No part of this publication may
be reproduced without the written permission of the copyright holder

Abstract

The identification of vortex cores in fluid mechanics is a challenging task that requires sophisticated techniques. Commonly employed local detection methods, such as the Q, delta, or swirling-strength criterion, rely on the local velocity gradient tensor to locate the vortex cores. Despite their reasonable accuracy, these methods tend to produce false positives and negatives, necessitating user-defined tuning parameters to maintain an acceptable error level. Moreover, this method presupposes prior knowledge of the vortices, limiting its robustness and self-contained nature. To overcome this shortcoming, a hybrid computer vision and machine learning approach is proposed to enhance the detection features of vortex cores and reduce false positives and negatives. Initially, computer vision was employed to identify the areas of vortex structures, followed by machine learning to identify the vortex core within the vortex region. A convolutional neural network (CNN) was trained to analyse streamline plots based on line integral convolution (LIC) for the computer vision process. Through the use of computer vision, false positives and negatives in flow-specific problems are reduced without the need for calibrating user-defined parameters. Furthermore, the trained CNN was successfully applied to three test cases, indicating its universal applicability. As such, computer vision offers a reliable convolutional neural network approach to detect vortex areas, which requires training once and is suitable for a broad range of flow scenarios. To identify the exact location of vortex cores, machine learning was combined with the computer vision approach. Various sets of input features were tested for both hybrid and pure machine learning approaches, starting with primitive variables such as velocity and pressure and expanding to more derived quantities such as velocity gradients, pressure gradients, Q-criterion, vorticity, and magnitude of vector quantities. In addition, a method for automatically labelling the dataset using K-means clustering was proposed to preprocess input images for machine learning. Results demonstrated that the K-means clustering-based labelling approach had a mean square error of

only 0.45%, comparable to the manual labelling approach. The hybrid approach significantly reduced training time for all tested cases and led to fewer false positives and negatives when using primitive variables and their derivatives compared to pure machine learning using the Artificial Neural Network approach applied to the entire flow. At the same time, using the variable set with all possible inputs does not provide a more accurate prediction of vortex cores and thus the hybrid approach is demonstrated as an effective way to reduce false positives and negatives entirely using just the primitive variables and their derivatives.

Keywords

machine learning; computer vision; vortex core detection; fluid mechanics; line integral convolution; computational fluid dynamics; turbulent flows.

Acknowledgements

I want to convey my heartfelt appreciation to Dr Tom Teschner and Dr Irene, my supervisors, for their priceless guidance, feedback, and constant encouragement during my thesis. Their proficiency and motivation have played a crucial role in helping me overcome the obstacles encountered in my research journey. I consider myself extremely fortunate to have had the chance to collaborate with such committed and knowledgeable supervisors.

I am grateful to my incredible spouse, Fatma, for consistently demonstrating patience, understanding, and support. Her affection and motivation have been a continuous and powerful source of inspiration, and I appreciate her steadfast dedication to our mutual aspirations.

Finally, I extend my heartfelt thanks to my parents for setting me on the path to this PhD a long time ago. They have always been my greatest source of motivation and inspiration, and I am always grateful for their support and encouragement.

Contents

List of Figures	V
List of Tables	XI
1 Introduction	1
1.1 Motivation	1
1.2 Aim and Objectives	5
1.3 Contributions to the Scientific Knowledge	7
1.4 List of Publications	8
1.5 Thesis Structure	8
2 LITERATURE REVIEW	11
2.1 Vortex definition	11
2.2 Artificial Intelligence (AI)	16
2.2.1 Computer vision (CV)	16
2.2.1.1 Overview	16
2.2.1.2 Object Detection methods	18

2.2.1.3	Deep learning in computer vision to identify vortices	20
2.2.2	Machine learning (ML)	22
2.2.2.1	overview	22
2.2.2.2	Machine learning fundamentals	24
2.2.2.3	Deep learning algorithm	27
3	Methodology	31
3.1	Governing equations for incompressible CFD	31
3.1.1	Navier-Stokes Equations	31
3.1.1.1	Conservation of Mass	32
3.1.1.2	Conservation of Momentum	33
3.1.1.3	Conservation of energy	35
3.1.2	Large Eddies Simulation	36
3.2	Local vortex identification approaches	39
3.2.1	Q criterion	40
3.2.2	Delta criterion	41
3.2.3	Swirling Strength criterion	42
3.3	Streamline detection using the line integral convolution algorithm	43
3.4	Vortex regions labelling	44
3.5	CV-based vortex detection regions using YOLO algorithm	46
3.5.1	YOLOv3 Architecture	47
3.5.2	Loss function	49

<i>CONTENTS</i>	III
3.5.3 mAP Evaluation metric	50
3.5.3.1 Intersection over Union (IoU)	51
3.5.3.2 Precision and Recall	52
3.5.3.3 Average Precision	52
3.6 Unsupervised K-Means Clustering Algorithm	53
3.7 ANN approach to detect vortex core within the regions	56
3.8 Summary	59
4 Test Case	63
4.1 Taylor-Green Vortex (TGV)	63
4.2 Application to turbulent boundary layer flows	68
4.3 Application to rotating wings	71
5 Results and Discussion	75
5.1 Local vortex detection methods	75
5.1.1 Identifying vortices of Taylor-Green Vortex using Local detection methods	75
5.1.2 Identifying vortices of Application to turbulent boundary layer flows using Local detection methods	83
5.2 Identifying vortex regions through computer vision (CV)	85
5.2.1 Identifying vortex regions of Taylor-Green Vortex	85
5.2.2 Identifying vortex regions of Application to turbulent boundary layer flows	91
5.2.3 Identifying vortex regions of Application to rotating wings	92

5.3	Detecting vortex cores using hybrid and single approaches	94
5.3.1	Detecting vortex cores of Taylor-Green Vortex	94
5.3.1.1	Using automatic labelling for preparing the Taylor-Green vortex data-set	101
5.3.2	Detecting vortex cores of Application to turbulent boundary layer flows	103
5.3.3	Detecting vortex cores of Application to rotating wings	108
5.4	Summary	111
6	Conclusions and Future work	113
6.1	Conclusions	113
6.2	Recommendations for future work	116
	Bibliography	117

List of Figures

2.1	Computer vision and its sub-concepts [47].	17
3.1	Vector field visualization (Line integral Convolution).	44
3.2	Different labelling strategies: On the left-hand side, non-overlapping bounding boxes are used and on the right-hand side labelled bounding boxes are allowed to overlap, capturing more of the vortices' structures.	45
3.3	YOLO model detection.	46
3.4	YOLOv3 architecture with input image size 416×416 and 3 types of scales (13×13 , 26×26 , and 52×52) as output; Darknet-conv2D-BN-Leaky ("DBL" for short) is the main component of YOLOv3 which contains one convolution layer, one batch normalization layer, and one leaky relu layer. residual-like unit ("ResUnit" for short) is two "DBL" structures following one "add" layer; several "ResUnit" with one zero-padding layer and "DBL" structure forward generates a residual-like block, "ResBlock" for short, which is the module element of Darknet-53. Figure adapted from Wang et al. [138]	48
3.5	Ground truth box vs predicted box.	52
3.6	This plot showing how to interpolate Precision values.	53

3.7	Labelling data sets automatically using CV and the K-means clustering algorithm.	55
3.8	The figure shows a FFNN with one input layer, two hidden layers, and one output layer of the model.	56
3.9	Process flow diagram architecture to identify vortex cores.	60
4.1	64^2 element meshes.	64
4.2	Contour velocity plot. The left-hand side at the beginning initial state and the right-hand side at the end time step.	66
4.3	Kinetic energy dissipation rate. The reference Large Eddy Simulation (LES) and Direct Numerical Simulation (DNS) was taken from [159], the current simulation results shown in the dashed blue line were obtained using LES on a 64^3 grid at a Reynolds number of $Re=1600$	67
4.4	Temporal evolution of the turbulent kinetic energy. The reference Large Eddy Simulation (LES) and Direct Numerical Simulation (DNS) was taken from [160], the current simulation results shown in the dashed blue line were obtained using LES on a 64^3 grid at a Reynolds number of $Re=1600$	67
4.5	Velocity boundary layer development on a flat plate, [161]	69
4.6	Developing the boundary layer, where δ represents the boundary layer thickness.	70
4.7	Developing the boundary layer, where δ represents the boundary layer thickness.	71
4.8	Figure depicts the quarter domain of the Application to rotating wings (Caradonna and Tung rotor) case.	72

5.1	Test images showing vortical structures obtained with the line integral convolution-based streamline algorithm.	76
5.2	Detected vortices (shown on yellow and green) using the Q, delta, and Swirling Strength criterion with second-order, fourth-order and least square gradient reconstruction schemes for the first test image (16 vortices). The actual vortex core regions are shown by the red circles.	77
5.3	Detected vortices (shown on yellow and green) using the Q, delta, and swirling strength criterion with second-order, fourth-order and least square gradient reconstruction schemes for the second test image (24 vortices). The actual vortex core regions are shown by the red circles.	78
5.4	Test image representing vortical structures obtained with the line integral convolution-based streamline algorithm.	83
5.5	Detected vortex cores using the Q, delta and swirling strength criterion with different gradient reconstruction schemes for the turbulent boundary layer case taken from Szöke et al. [163]	84
5.6	The output result after training data set of (1 image), (40 images) and (100 images) for test images 1 and 2. The red box is the predicted bounding box based on bounding boxes that are allowed to overlap during labelling; see Section 3.4.	86
5.7	The output result after training data set of (1 image), (40 images) and (100 images). The red box is the predicted bounding box based on bounding boxes that are not allowed to overlap during labelling, see Section 3.4.	87
5.8	(The black solid and blue dish) false negatives curves for test image one and test image two respectively in the number of images.	89

5.9	Mean Average precision curve in the time of training epochs	90
5.10	loss values curve in the time of training epochs.	90
5.11	Detected vortex core regions using the computer vision approach for the turbulent boundary layer case taken from Szöke et al. [163]	92
5.12	Detected vortex core regions using the YOLO algorithm for differ- ent angles of application to rotating wings case taken from Ricci <i>et</i> <i>al.</i> [166].	93
5.13	The figures show the result of using the hybrid approach (YOLO, ANN) for different flow field files and different variable sets where the yellow dots are shown the right prediction of actual vortex cores(true positives) and blue dots are false positives, and the purple dots indi- cate the false negatives.	96
5.14	The figures show the result of using the single approach (pure ANN) for different flow field files and different variable sets where the yellow dots are shown the right prediction of actual vortex cores(true posi- tives) and blue dots are false positives, and the purple dots indicate the false negatives.	97
5.15	The values of True positive for different Variable sets with respect to the number of flow field files based on hybrid approaches with solid line and single approach (pure ANN) with dash line.	100
5.16	The values of False positive for different Variable sets with respect to number of flow field files based on hybrid approaches with solid line and single approach (pure ANN) with dash line.	100
5.17	The values of False Negatives for different Variable sets with respect to number of flow field files based on hybrid approaches with solid line and single approach (pure ANN) with dash line.	101

5.18	The figures show the result of using the computer vision and the K-means clustering algorithm to identify vortex cores for labelling data-set automatically.	102
5.19	The figures show the result of using the hybrid approach (YOLO, ANN) for different flow field files and different variable sets where the yellow dots are shown the right prediction of actual vortex cores(true positives), blue dots are false positives, and the purple dots indicate the false negatives.	106
5.20	The figures show the result of using the single approach (pure ANN) for different flow field files and different variable sets where the yellow dots are shown the right prediction of actual vortex cores(true positives), blue dots are false positives, and the purple dots indicate the false negatives.	108
5.21	The figures show the result of using the single approach (pure ANN) for different flow field files and five variable set where the yellow dots are shown the right prediction of actual vortex cores(true positives) and blue dots are false positives.	110
5.22	The figures show the result of using the hybrid approach (YOLO, ANN) for different flow field files and five variable set where the yellow dots are shown as the right prediction of actual vortex cores(true positives) and blue dots are false positives.	110

List of Tables

3.1	Variable sets exported from Ansys Fluent software	57
5.1	Number of predicted vortices for the Q, delta and swirling strength criterion using a second-order gradient reconstruction scheme, as well as their false positives and negatives and the threshold value ϵ required to match the number of predicted vortices to the number of expected vortices (EV)	80
5.2	Number of predicted vortices for the Q, delta and swirling strength criterion using a fourth-order gradient reconstruction scheme, as well as their false positives and negatives and the threshold value ϵ required to match the number of predicted vortices to the number of expected vortices (EV)	80
5.3	Number of predicted vortices for the Q, delta and swirling strength criterion using a least-square gradient reconstruction scheme, as well as their false positives and negatives and the threshold value ϵ required to match the number of predicted vortices to the number of expected vortices (EV)	80
5.4	Root mean square errors for the predicted vortex cores to the actual vortex core regions.	81

5.5	Maximum Difference between the predicted vortex cores and the actual vortex core regions.	82
5.6	Number of detected and expected vortices for the first test image, as well as false positives and negatives with an increasing number of images used for training. The computational time taken for training is provided along with the mean Average Precision (mAP)	88
5.7	Number of detected and expected vortices for the second test image, as well as false positives and negatives with an increasing number of images used for training. The computational time taken for training is provided along with the mean Average Precision (mAP)	88
5.8	Showing a number of expected and detected vortices for the sample test, as well as comparison of hybrid approaches and a single approach how false positives and negatives reduce as the number of flow field files in the training data set increases with different Variable sets from 1 to 5 as input features listed in table 3.1 and The computational time is taken for training in both methods.	98
5.9	Vortex core coordinates using manual and automatic labelling techniques for the same image test.	102
5.10	Showing a number of expected and detected vortices for the Application to turbulent boundary layer flows, as well as comparison of hybrid approaches and a single approach to how false positives and negatives reduce as the number of flow field files in the training data set increases with different Variable sets from 1 to 5 as input features listed in Table 3.1 and The computational time is taken for training in both methods.	104

5.11 Showing a number of expected and detected vortices for the Application to rotating wings test case, as well as comparison of hybrid approaches and a single approach how false positives and negatives reduce as the number of flow field files in the training data set increases with different Variable sets from 1 to 5 as input features and The computational time is taken for training in both methods. . . .	109
--	-----

List of Symbols

Latin Symbols

Symbol	Description	Units
$\mathbf{g}$	Gravitation	$[\text{m s}^{-2}]$
k	Kinetic Energy per Unit Mass	$[\text{m}^2 \text{s}^{-2}]$
p	Pressure	$[\text{N m}^{-2}]$
t	Time	$[\text{s}]$
Δt	Time-step	$[\text{s}]$
Δ_{LES}	Filter width	$[-]$
Δ_c	Delta criterion	$[-]$
$\mathbf{U}$	Velocity Vector	$[\text{m s}^{-1}]$
u_x, u_y, u_z	Velocity Vector Components	$[\text{m s}^{-1}]$
a	Acceleration	$[\text{m s}^{-2}]$
$\Delta x, \Delta y$	Mesh spacing	$[\text{m}]$
$\mathbf{x}$	Coordinate Vector	$[\text{m}]$
x, y	Coordinate Vector Components	$[\text{m}]$
b	Bias	$[-]$
f	Activation function	$[-]$
F	Force Vector	$[-]$
m	Mass	$[\text{kg}]$
f_{body}	Body force	$[\text{kg m s}^{-2}]$
$f_{surface}$	Internal force	$[\text{kg m s}^{-2}]$
C_S	Empirical constant	$[-]$
T	Temperature	$[\text{K}]$
D_{ij}	Velocity gradient tensor	$[-]$

Greek Symbols

Symbol	Description	Units
V	Volume	$[\text{m}^3]$
w	Weight	$[-]$
δ	Boundary layer thickness	$[-]$
ε	Threshold parameter	$[-]$
σ	Eigenvalue	$[-]$
λ_r	Real eigenvalue	$[-]$
λ_{cr}	Complex eigenvalue	$[-]$
v_r	Real eigenvector	$[-]$
$v_{cr} \pm iv_{ci}$	Corresponding eigenvectors	$[-]$
μ	Dynamic Viscosity	$[\text{Pa s}]$
ν	Kinematic Viscosity	$[\text{m}^2 \text{ s}^{-1}]$
ρ	Density	$[\text{kg m}^{-3}]$
σ	Shear Stresses	$[\text{Pa}]$
S	Rate-of-strain	$[-]$
Ω	Rate of rotation tensor	$[-]$
τ	Stress Tensor	$[-]$
∇	Nabla	$[-]$
δ_{ij}	Kronecker delta	$[-]$
$\mathcal{L}$	Loss function	$[-]$
η	Learning rate	$[-]$
ϵ	Threshold parameter	$[-]$

List of Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
AI	Artificial Intelligence
Adam	Adaptive Moment Estimation
API	Application Programming Interface
CFD	Computation Fluid Dynamics
CFL	Courant Friedrichs Levy (number)
CNN	Convolutional Neural Network
CV	Computer Vision
CW	Current Work
DNN	Deep Neural Network
DNS	Direct Numerical Simulation
FP	False-Positive
FN	False-Negative
FFNN	Feed-Forward Neural Network
MFFNN	Multilayered Feed-Forward Neural Network
GPU	Graphics Processing Unit
HPC	High-Performance Computing
IoU	Intersection over Union
IVD	Instantaneous Vorticity Deviation
LES	Large Eddy Simulation
LIC	Line Integral Convolution
mAP	mean Average Precision
ML	Machine Learning
MUSCL	Monotonic Upstream-centered Scheme for Conservation Laws

NN	Neural Network
PDE	Partial Differential Equation
RANSE	Reynolds-Averaged Navier-Stokes Equations
RCNN	Region-Based Convolutional Neural Network
ReLU	Rectified Linear Units
RHS	Right-hand Side
RMSE	Root Mean Square Error
RMSProp	Root Mean Square Propagation
RN	Reynolds Number
ROIs	Regions Of Interest
SSD	Single Shot MultiBox Detector
TanH	Hyperbolic tangent
TBL	Turbulent Boundary Layer
TP	True Positive
YOLO	You Only Look Once
WENO	Weighted Essentially Non-Oscillatory

Chapter 1

Introduction

1.1 Motivation

Detecting vortex cores reliably remains an open question in fluid mechanics research. It has been widely recognised that vortex dynamics play a significant role in explaining the behaviour of fluid flows. Identifying the exact location of vortex cores is useful during automated mesh refinement in computational fluid dynamics (CFD) simulations [1]. It also provides additional valuable insights, such as knowing the locations of vortices on aircraft objects, leading to reduced drag and lift by optimising aerodynamic devices to channel turbulent vortices [2]. Engineers can design more efficient aircraft that produce less drag and consume less fuel by studying vortices. Consequently, there has been a significant focus on researching vortices within flow fields in the past few decades. Simple velocity gradient-based approaches may be used to identify single vortex, but complex engineering applications may involve merging and mixing of vortices where these simplistic approaches may fail. Although knowing the precise location of vortices has many practical benefits, numerical tools are still unable to accurately predict these vortices, and they produce spurious detected vortex cores, which negates the utility of knowing exactly

where vortices are.

The governing equations of CFD are solved by dividing the computational domain into smaller elements (cells) to achieve higher accuracy and resolution [3]. Mesh quality dramatically impacts the final results; thus, mesh generation can be a significant factor in errors. The goal of mesh refinement is to identify the regions in the flow where the flow variables change rapidly so that these regions can be modelled with a finer mesh [4]. In order to identify areas of the flow that require mesh refinement, it is essential to identify the location of the vortices. Generally, vortices can be observed in many different types of flows, including boundary layers, turbulent flows, and free shear layers, which are characterised by high vorticity and high shear stress and therefore require a finer mesh to resolve their complex flow structure [5]. Once the regions of vortices have been determined, the mesh in these areas can be refined to produce a more precise depiction of the flow. This process is usually applied by subdividing the elements in these regions into smaller elements, resulting in a finer mesh in these areas.

Identification of the vortex region for mesh refinement has several advantages. It is a direct, efficient method for identifying regions of the flow that need to be refined. This can significantly reduce the computational time required to obtain accurate results and decrease the number of elements required to resolve the flow. Vortex detection for mesh refinement allows the user to control the degree of mesh refinement, which can be important when the flow is highly unsteady or when flow variables change rapidly. The user can control the degree of refinement to ensure the mesh can resolve the flow accurately while avoiding over-refinement, which can lead to unnecessary computational cost. Therefore, identifying the vortex provides a direct and efficient means of maintaining the degree of mesh refinement and can significantly reduce computational time and cost. In addition, knowing where the

vortex cores are located for mesh refinement is an essential tool for CFD simulation and can substantially improve the accuracy and resolution of simulation results. Therefore, identifying the vortices provides a direct and efficient means of maintaining the mesh refinement degree, significantly reducing computational time, and minimising memory consumption. In addition, knowing where the vortex is located for mesh refinement is an essential tool for CFD simulation and can substantially improve the accuracy and resolution of simulation results. For automatic mesh refinement to be executed effectively, it is imperative to possess knowledge regarding the areas necessitating refinement, typically associated with the location of vortex cores and regions exhibiting prominent gradients. This knowledge facilitates enhanced resolving partial differential equations (PDEs) resolution in such regions.

Vortex detection is an important aspect of fluid dynamics, and there are two commonly used methods for detecting these phenomena, local and global detection algorithms. The local method for identifying vortices relies on the decomposition of the local velocity gradient tensor [6], and there is a range of different criteria available that can be employed to detect vortices using this approach. Some of the most commonly used criteria include the Q criterion [7], Lambda 2 criterion [8], delta criterion [9], and swirling strength criterion [10]. These criteria are known as local detection methods, as they are designed to analyse the velocity gradient tensor in a localised manner. Despite their popularity, these methods have some shortcomings. These techniques are cheap to compute and could yield reasonable estimates, but their lack of robustness makes them prone to identifying spurious vortices known as false positives (no vortex exists but is identified) or false negatives (which exist but are not detected). A user must set appropriate parameters and prescribe a threshold below which vortices are filtered to reduce the number of false positives and negatives. It is necessary to re-calibrate this threshold value for each flow problem since it is flow-dependent. Global vortex detection methods typically use streamlines to

identify vortex regions. While global methods are generally more robust than local methods, they use neighbouring cells to identify vortical flows, which increases their computational time. On the other hand, local methods require users' input who have specific knowledge about the field, thus making them ineffective for detecting vortex cores automatically.

In order to address this problem, A novel approach involves employing convolution neural networks on CNN for YOLO (You Only Look Once) and utilizing a separate deep neural network exclusively for extracting features from the data files (raw ASCII files) and then combining their output to enhance the detecting vortex cores. The method begins by utilising CV techniques to pinpoint the regions where vortical structures are present [11]. Once these regions have been identified, the application of ANN is employed to detect the location of the vortex core with precision within the vortical area [12]. By incorporating both CV and ANN, the approach offers a comprehensive solution for detecting vortex cores, leveraging the strengths of both techniques to deliver accurate results.

The part of CV has been utilised the You Only Look Once (YOLO) algorithm [13] to identify specific areas where vortex cores are located. A recently introduced object detection approach called You Only Look Once (YOLO) utilises a deep neural network(NN) to the whole image instead of predicting multiple areas of the image. The objective here is to investigate bounding boxes that enclose vortex regions rather than the exact location of the vortex core. Through the utilisation of CV for the detection of vortex core regions, a decrease in the occurrence of false positives and negatives can be achieved. Additionally, this method proves to be beneficial for the purposes of local mesh refinement (which refines a volume and not just a single line) or tracking a vortex by itself (or rather its bounding box). The attention of this part of the study is directed towards providing a more reliable, robust, and

accurate method for predicting vortex regions, which can subsequently be employed with greater confidence in engineering design processes. In any 3D data set of vortical flows, the area of vortices can be identified by the CV framework as long as two-dimensional planes can be constructed through three-dimensional flow volumes in which vortices pass through and as long as surface streamline plots can be used to visualise these vortices, as discussed in Section 3.3, the framework will be able to identify the area of these vortices. It provides a more generalised approach to detecting vortices since vortices do not have to align with the generated 2D planes and can pass through these at an angle.

The ANN is then added on top, where this hybrid approach of CV using YOLO and ML using ANN will add new dimensions. ANN is a set of features that can be used not only to detect the vortices in general but also to identify the vortex core, which is an important application in fluid mechanics. The tracking of these vortices facilitates understanding the flow's evolution. The algorithm is implemented utilising Keras, a deep learning Application Programming Interface (API) acting as an interface for TensorFlow [14], to carry out the task. TensorFlow is an open-source written in Python for ML.

1.2 Aim and Objectives

This study aims to identify vortex cores in CFD simulations through a hybrid CV and ML framework. The thesis will address the challenge of detecting vortex cores in CFD simulations and improve the accuracy and efficiency of vortex detection. The following objectives will be worked towards in order to achieve the aim:

1. Identify suitable test cases to generate vortices to detect. In order to understand vortices under different fluid flow conditions, the study will examine three different fluid flow scenarios. This objective will provide a foundation

for the subsequent objectives and ensure the methodology is robust enough to be implemented in a wide range of CFD simulations.

2. Extract the vortices from the domain and detect the vortices using traditional methods. The study will evaluate the performance of local detection methods in detecting vortex cores and compare the results to the outcomes of the proposed hybrid framework. This objective will provide a baseline for the proposed framework's performance and highlight traditional detection methods' limitations.
3. Detect vortices using CV algorithm. The study will evaluate the performance of CV algorithms to identify the region of vortices using streamline plots based on line integral convolution (LIC) in order to reduce false positives and negatives and provide a basis for developing the hybrid framework.
4. Development of a surrogate framework consisting of ML and Machine vision to reduce false positives and false negatives. The study will integrate deep learning with CV algorithms to improve the accuracy and efficiency of vortex core detection. The hybrid framework will be trained on a large dataset of CFD simulations, and the results will be evaluated to demonstrate the effectiveness of the proposed approach in reducing false positives and false negatives.

1.3 Contributions to the Scientific Knowledge

The contribution to scientific knowledge may be summarised as follows:

1. The hybrid framework approach used in this research provides a novel and efficient method for detecting vortex cores in complex fluid flow scenarios and reducing false positives and negatives. This approach makes a substantial contribution to scientific understanding in the field of fluid dynamics and holds implications for various applications.
2. The use of the CV and ANN approach enables the analysis of large amounts of fluid flow data, providing valuable insights into the turbulence behaviour in fluids.
3. By utilising the K-means clustering algorithm to identify actual vortex cores for labelling data sets automatically, the process of preparing the dataset for training models to detect vortex cores becomes more efficient and streamlined. This automatic labelling of the dataset using the K-means algorithm allows for quicker and more accessible data preparation, resulting in a more efficient and rapid preparation dataset. This contribution offers a valuable tool for simplifying the process of detecting actual vortex cores and training models to detect them accurately in a timely and efficient manner.
4. The hybrid framework approach is scalable and flexible, making it applicable to a wide range of fluid flow scenarios, including complex industrial processes.
5. The following link presents a GitHub repository that demonstrates a project utilising Computer Vision based on YOLOv3 algorithm part to identify the region of vortices:

<https://github.com/Hazem-Abolholl/Surface-line-integral-convolution-based-vortex-detection-using-computer-vision>

The weights (checkpoint) after training the model can be downloaded through this link to evaluate the model:

https://cord.cranfield.ac.uk/articles/dataset/Surface_line_integral_convolution-based_vortex_detection_using_computer_vision/19214166/1

1.4 List of Publications

1. Abolholl HAA, Teschner T-R, Moulitsas I. (2022) A hybrid computer vision and machine learning approach for robust vortex core detection in fluid mechanics applications. In: UKACM 2022: 2022 Annual Conference of the UK Association for Computational Mechanics, 20-22 April 2022, Nottingham, UK
2. Abolholl, H. A. A., Teschner, T.-R., and Moulitsas, I. (January 11, 2023). "Surface Line Integral Convolution-Based vortex detection using Computer Vision." ASME. J. Comput. Inf. Sci. Eng. doi: <https://doi.org/10.1115/1.4056660>

1.5 Thesis Structure

The remaining structure of the Thesis is outlined in the following manner: Chapter 2 delves into a summary review of the current knowledge and understanding regarding the vortex definition, CV, and ML concept. This chapter presents an overview of the current state of research and development in these areas, identifying gaps in the existing knowledge that may be addressed.

Chapter 3 describes the Navier-Stokes Equations and Large Eddies Simulation. It explains the concepts of these equations and simulations and how they are used to analyse fluid flow. Additionally, the chapter explores various methods of identifying vortices using local detection methods, including the Q, Delta, and Swirling Strength criteria. The chapter also covers the line integral convolution algorithm

(Streamline plots) and how it can be generated. Another aspect of vortex detection discussed in the chapter is the use of CV-based vortex detection regions and the YOLO algorithm to detect the region of vortices. Furthermore, the chapter introduces the Unsupervised K-Means Clustering Algorithm and its use in automatic labelling. Lastly, the chapter examines using Artificial Neural Networks (ANNs) to detect vortex cores within regions. The chapter provides a comprehensive overview of the various techniques and algorithms used to analyse the fluid flow and identify the vortex core.

Chapter 4 explores three fluid flow scenarios: the Taylor-Green Vortex (TGV), Turbulent Boundary Layer flows, and Rotating Wings application. The TGV is a simple and well-known flow pattern used to study the turbulence behaviour in fluids. The Turbulent Boundary Layer flow is a more complex flow scenario commonly observed in engineering applications, such as aircraft wings, where the fluid flow is affected by a solid boundary. This chapter also covers the application of fluid flow principles to the design of rotating wings, such as those found in aircraft and wind turbines, is also discussed. The importance of understanding fluid flow patterns in these scenarios is highlighted to test and support the validation of the model. Overall, this chapter summarises three critical fluid flow scenarios in engineering and technology.

Chapter 5 presents the results for different flow test cases to test the hybrid framework. The performance of local vortex detection methods is first evaluated, and their results are compared to those acquired using the hybrid framework approach. The second part of the chapter involves identifying vortex regions through CV techniques. This method can accurately determine vortex regions in complex flow scenarios. Finally, the results of detecting vortex cores in fluid flow using the hybrid and single approaches are presented. The results show that the hybrid approach outperforms

the single approach, providing more accurate and reliable results. Overall, this chapter presents a comprehensive overview of the outcomes obtained from testing the hybrid framework in different fluid flow scenarios. It is demonstrated how the framework can be used to accurately detect vortex cores in fluid flow, which provides valuable insights into fluid turbulence behaviour.

In Chapter 6, The conclusion of the thesis will be condensed, providing a summary utilising the hybrid framework approach to detect vortex cores in fluid flow.

Chapter 2

LITERATURE REVIEW

Background information on fluid vortices and a brief overview of prior research conducted in the area of vortex core detection using both traditional methods and deep learning techniques will be presented throughout this chapter, along with an explanation of how the solution proposed in this project will contribute to the previous research.

2.1 Vortex definition

Advancements in transistor technology have paved the way for CFD to be used as a tool to study turbulence flows at even the smallest scales of motion [15]. As indicated in the introduction, vortices are a crucial aspect in a range of engineering and scientific problem fields where they play an essential role, such as aircraft navigation during nose-up or roll of a jet [16], medical [17, 18], as well as industrial applications [19]. Thus, understanding the behaviour of vortices is of prime importance when it comes to possible practical applications. In fluids, vorticity is the physical quantity which describes the rotation of fluid components as $\omega = \nabla \times v$, where ∇ is the del operator and v is fluid velocity [20]. If the value of ω is zero at every point in a flow, the flow is considered irrotational. On the other hand, if ω is non-zero,

the flow is referred to as rotational [21]. The vortex is composed of two distinct parts that are interrelated - the core and the swirling region around it. The core serves as the centre of the vortex, while the swirling region represents the fluid that is swirling around the core. The velocity of the fluid within the vortex core is zero compared to the vortex itself [22]. This unique feature makes the study of vortices so important, as it is important to understand the dynamics of fluids under such conditions.

For understanding turbulent flows, the vortex is essential, as vorticity distributions depend on the position of the vortices in highly sophisticated ways. During the 19th century, when aerodynamics was being explored, particularly in the context of human-powered flight, the concept of vorticity and vortex dynamics came to the forefront. In 1858, Helmholtz [23] presented his three vorticity theorems which emphasised the crucial role of vorticity in ideal incompressible fluid movements, which he referred to as “vortex motions” [24]. This sparked ongoing debates and extensive research into the mathematical definition of vortices in fluid dynamics, leading to the development of a vast field of study. Kelvin [25, 26] further built upon this work with his circulation theorem. According to this theory, the atomic structure of all elements is made up of knotted, linked vortex filaments in a hypothetical ideal background fluid referred to as “ether” that pervades the universe. However, due to the instability of these structures and the inviscid fluid flow, Kelvin was eventually forced to abandon his theory. Despite this, his pioneering research significantly contributed to revealing and understanding that turbulent flows, instead of laminar flows, are prevalent in all natural systems. As Hinshelwood observed during that period [27], “fluid dynamicists were divided into hydraulic engineers who observed what could not be explained, and mathematicians who explained things that could not be observed.” (Lighthill 1956). In the early 1900s, Richardson [28] composed the renowned verse that established and solidified the connection between turbulence

and predation:

“Big whirls have little whorls,
Which feed on their velocity.
And little whorls have lesser whorls,
And so on to viscosity.”

The author describes in his poem how large vortices feed smaller ones, which feed even smaller ones, eventually causing dissipation at extremely small scales, a phenomenon called molecular friction. Numerous studies have supported this idea of vortex dynamics in turbulence, particularly the discovery of turbulent coherent structures, which have provided further proof and precision of this concept. These findings have provided further evidence and refinement of Richardson’s theory, demonstrating the relationship between turbulence and vortices.

Despite the fact that vortices are easily recognisable through intuition, it is challenging to come up with a clear definition of what they actually are [29]. The concept of vortices is complex and has been the subject of much study and investigation. Vortices are found in many different physical systems, including fluid mechanics, atmospheric science, and even in plasma physics. They can take on various forms and have different physical properties, making it difficult to come up with a single definition that covers all of their different forms. There are several definitions of vortices in literature, which can be found in various sources [30]. In “Vortex Dynamics”, written by Saffman [31], a vortex is defined as “a finite volume of vorticity immersed in the irrotational fluid.”. This definition, however, does not satisfy since it requires that “the fluid is ideal to make strict sense, because viscosity diffuses vorticity.” [31]. According to Lugt [32], “a vortex is the rotating motion of a multitude of material particles around a common centre”. However, this definition has limitations and does not give flexibility in terms of its practical application in developing detection

algorithms [6]. Recent studies conducted by Wang et al. [33] have revealed that the vorticity decreases as the vorticity lines move into the vortex region, reducing the vorticity magnitude. This reduction is particularly pronounced near solid walls, and the vorticity in these areas is significantly lower compared to other regions. The findings, obtained through the use of Direct Numerical Simulations (DNS), highlight the need for a more comprehensive and flexible definition of vortices in order to improve their detection and analysis. Robinson [34] proposed a different definition of a vortex, stating that “a vortex exists when instantaneous streamlines mapped onto a plane normal to the vortex core exhibit a roughly circular or spiral pattern when viewed from a reference frame moving with the centre of the vortex.”. However, this definition has a self-referential nature as it requires prior knowledge of the location, orientation, and motion of the vortex core in order to identify it. In real fluids, identifying the exact vortex can be challenging since there is a lack of clear differentiation between the eddy flow regions and the non-eddy flow regions.

Despite the lack of an official definition, the study of identifying vortices through available methods has recently gained significant attention. As stated by Küchemann [35], “Vortices are the sinews and muscles of turbulence.”. While there has been considerable advancement in the development of qualitative definitions of vortices in classical vortex regions, the need for a comprehensive and quantitative mathematical description has become increasingly important in understanding the structure, evolution, and interactions of vortices in turbulence. This lack of a formal definition of vortices has not prevented researchers from making progress in the field. A number of methods have been developed and applied to identify vortices in various environments. One of the most common methods is the use of flow visualization techniques, such as Particle Image Velocimetry (PIV) [36] and Laser Doppler Velocimetry (LDV) [37], to observe and track the movement of vortices. Another approach involves the application of mathematical techniques, such

as the vortex identification algorithm, to quantify the characteristics of vortices in turbulence. Despite these efforts, much is still to be learned about the behaviour and dynamics of vortices in turbulence. In order to achieve a deeper knowledge of these complex structures, a rigorous mathematical description is required. This description will provide a foundation for developing accurate and predictive models of vortex behaviour in a turbulent flow.

Jeong and Hussain [38] have emphasised the importance of considering Galilean invariance when determining quantitative criteria for defining a vortex. Various vortex identification techniques have been developed in recent years, including the Eulerian and Lagrangian methods [39, 40]; these methods are based on local and global algorithms [41–43], which enable them to detect vortices. Local vortex detection methods [7–9] are a quick way to detect vortices, as they are based on the decomposition of the local velocity gradient tensor. This makes them relatively easy to implement and results are obtained rapidly. However, there are limitations to these methods, as they require careful selection of suitable parameters in order to produce robust results. This increases the time required to identify vortex cores and may lead to inaccurate results if the parameters are not correctly selected. On the other hand, global vortex identification methods typically rely on streamlines to identify vortex regions, providing a higher level of robustness in detecting vortices. These methods use neighbouring cells to identify vortical flows and therefore tend to be more accurate than local methods. Examples of global methods include the winding angle [41], instantaneous vorticity deviation [42], and elliptic objective Eulerian coherent structures method [43]. While global methods provide more accurate than local methods results, they also come with increased computational time [44], making them less efficient than local methods. Additionally, global methods require user input, making them less suitable for automatic vortex core detection tasks. Both local and global methods have drawbacks and cannot fully guarantee robust

and reliable results. In conclusion, both local and global methods have their own benefits and limitations when it comes to vortex identification. While local methods are quick and efficient, they may not provide the most robust results. On the other hand, global methods are more accurate, but they require increased computational time and user input. Thus, it is essential to consider a given application's specific requirements and limitations before selecting a method for vortex identification.

2.2 Artificial Intelligence (AI)

2.2.1 Computer vision (CV)

2.2.1.1 Overview

CV is a field of study that has made great strides since its inception in the late 1960s and early 1970s. Initially, researchers aimed to mimic the human visual system by exploring how cameras capture and project images and how photogrammetry measures and analyses those images [45]. Over time, the CV has come to play a major role in a wide range of fields, including pattern recognition, computer graphics, 3D reconstruction, robotic navigation, virtual reality, and augmented reality [46, 47]. CV also plays a crucial role in developing ML and Artificial Intelligence (AI). With the help of CV algorithms, machines can learn to recognise patterns and make predictions about the real world, just as humans do. This has tremendous potential for improving our ability to solve problems, from predicting the weather to diagnosing medical conditions. Even more exciting developments in the future can be expected as CV and related technologies continue to advance.

Researchers have concentrated their efforts on low-level vision tasks, such as edge detection and segmentation [48], to analyse images from the real world. Edge detection is the process of identifying the boundaries between different objects in an

image. Segmentation, on the other hand, is the process of dividing an image into smaller regions, or segments, based on similarities in colour, texture, and shape. These two tasks are often used as the first step in CV algorithms, as they provide a basic understanding of the structure of an image. The field of CV involves the use of algorithms to enable computers to generate intelligent digital images and videos. Figure 2.1 [47] illustrates that CV encompasses various scientific disciplines, such as mathematics, science, technology, physics, geometry, and probability. As a practical

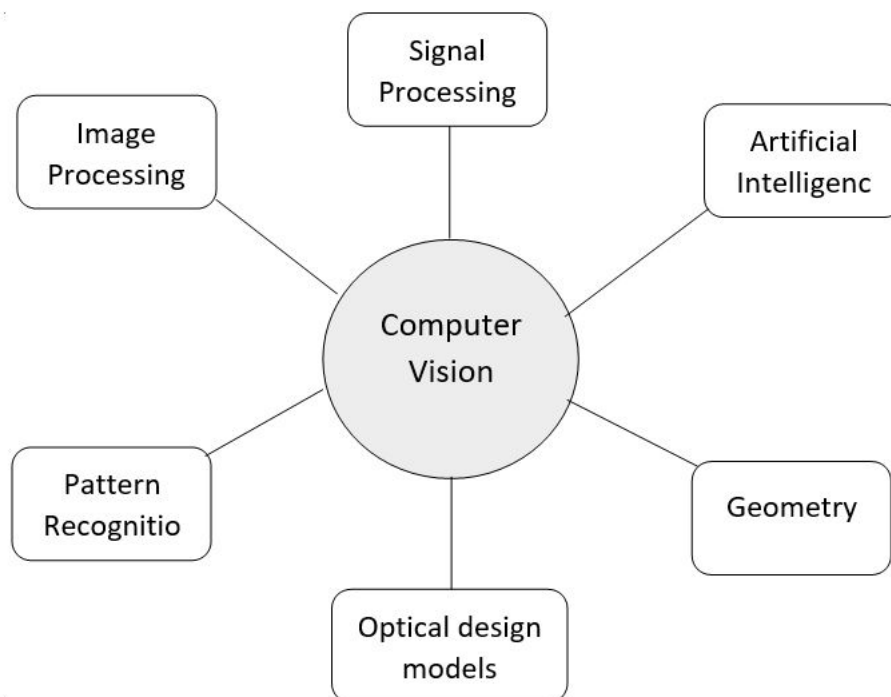


Figure 2.1: Computer vision and its sub-concepts [47].

matter, implementing AI techniques, such as deep learning and ML, necessitates the processing of vast amounts of data through CV in order to be efficient. CV is an emerging field that has acquired a lot of attention in recent years. It involves the development of algorithms and computational models that allow computers to recognise and interpret visual information in the same way that humans do. By leveraging these algorithms and models, computers can analyse images and videos and extract valuable information from them.

Several machine vision applications are found in manufacturing, Kinnell et al. [49] and Osorio-Comparan et al. [50] report on specific applications for manipulating parts in robotics systems. Krüger and Nguyen [51] study ergonomics in an industrial setting using machine vision. These systems often employ various alternative technologies, but they generally follow a similar sequence of steps, starting from image acquisition and ending with actuation. The process consists of several stages, including optical acquisition while considering the scene's limitations, preprocessing, segmentation, feature extraction, classification, and actuation [52, 53]. The optical acquisition involves capturing images using cameras or other image sensors. The constraints of the scene, such as lighting, contrast, and background, need to be considered in order to produce high-quality images for analysis.

Many techniques have been developed in CV to extract, track, and recognise features [54, 55]. Visualisation aims to understand, analyse, and identify physical or mathematical models. Abdurakipov et al. [56] Initially conducted experimental measurements of velocity and pressure fields to gather data. Subsequently, they applied computer vision techniques based on deep neural networks to interpolate and enhance the spatial resolution of the turbulent flow. The use of ANNs in CV solves complex problems involving pattern recognition and image analysis. These techniques were applied to solve problems in fluid dynamics, specifically in the reconstruction of turbulent flow velocity and pressure fields.

2.2.1.2 Object Detection methods

One of the primary tasks of CV is object detection, which involves identifying objects in an image or video stream based on their classification, such as animals, buildings, cars, and humans. Object detection is concerned with creating approaches and models that estimate the category and locations of objects within an image. Object detection is utilised in various practical applications in CV, including medical diag-

nosis [57], autonomous driving [58,59], face recognition [60,61], and robotics [62–66]. The objective is to enable the machine to identify and locate objects with the same level of accuracy as humans.

Object detection has undergone a significant acceleration in recent years due to the rapid progress of deep learning techniques. The application of deep learning networks and Graphics Processing Units (GPUs) has drastically improved the performance of object detection. In 2012, Krizhevsky et al. [67] introduced the AlexNet model for image classification in the ImageNet dataset [68] competition, which was based on convolutional neural networks (CNNs). Their algorithm was a game-changer, as it outperformed traditional algorithms by an accuracy advantage of 11%, making it the winner of the competition [69]. This breakthrough result paved the way for the widespread adoption of deep CNNs in object detection.

Deep learning techniques for detecting objects can be classified into two categories: **two-stage and one-stage methods**. **Two-stage methods** [70], such as (RCNN, Fast RCNN, Faster RCNN, Mask R-CNN, and G-RCNN), involves detecting regions of interest (ROIs) and classifying the object within them. In contrast, **one-stage methods**, such as (YOLO, RetinaNet, and SSD), directly predict bounding boxes without proposing regions. Compared to one-stage methods, which predict bounding boxes without proposing areas, this method consumes less time and can therefore be implemented in real-time. Regions with CNN features (RCNN) [71], the original two-stage approach, proposes ROIs using selective search, a method for identifying likely object locations based on image features. Fast R-CNN [72] builds on this by sharing computation across the entire image rather than processing each proposed region individually. Faster RCNN, which was introduced by Ren et al. in 2015 [73], goes a step further by using a separate network, called a Region Proposal Network (RPN), to generate region proposals, making the method even faster. Mask

R-CNN [74] extends Faster RCNN by adding a third branch to the network for predicting object masks. G-RCNN [75] takes a different approach by using a graph-based representation of the image, allowing it to better capture spatial relationships between objects. SSD (Single Shot MultiBox Detector) is a popular one-stage object detection algorithm proposed by Liu et al. in 2016 [76]. SSD uses a single neural network to generate object category scores and bounding box coordinates for multiple object locations. This makes it more efficient than the two-stage approaches, but it can have some limitations in detecting smaller objects. YOLO (You Only Look Once) was proposed by Redmon et al. in 2016 [13], and it is another one-stage object detection algorithm. YOLO divides the image into a grid and predicts class probabilities and bounding boxes for each cell. This makes it faster than the two-stage approaches, such as Faster R-CNN and well detection of smaller objects by increasing the number of cells in the grid.

2.2.1.3 Deep learning in computer vision to identify vortices

To address the limitations of traditional vortex identification methods, researchers have recently turned to deep learning based on CV techniques for detecting vortical structures in flow fields to achieve more robust and reliable results. Many of the techniques developed in deep learning can be leveraged to extract, track, and identify features [77, 78]. While the primary aim of visualisation is to comprehend, analyse, and identify physical phenomena or mathematical models, the criteria for tracking features in scientific simulations differ from those in most deep learning based on CV applications. Scientific simulations often involve dynamic features that can experience changes such as splitting, merging or disappearing [79]. One promising application of this technology in this context involves using deep neural networks (DNNs) to solve the reconstruction of velocity and pressure fields in a turbulent flow with high resolution, as well as the identification of characteristic flow features [80].

In recent years, deep neural network (DNN)-based ML has shown great success in natural language processing and CV [81–84] and has been applied to identify vortical flows. For instance, Lguensat et al. [85] used a DNN to identify ocean eddies from sea surface height maps. In order to detect ocean eddies, the whole visualisation image of the flow field was considered as input. Franz et al. [86] used a convolutional neural network (CNN) to detect ocean eddies and provided the labelled training data by local vortex identification that required specific parameters to achieve reliable results. In a similar vein, Ströfer et al. [87] created a CNN to detect horseshoe vortices in a 3D flow over a wing-body junction. In a separate study, Deng et al. [88] used a CNN-based method to identify vortices, combining the global instantaneous vorticity deviation approach of Haller et al. [42] with a local method to label each point in the flow field for training. Zhang et al. [89] applied adaptive boosting (AdaBoost) algorithm [90] to assign distinct weights to four local vortex detection methods, guided by expert insights. This led to decreased misclassification rates in computational fluid dynamics (CFD) datasets. The local vortex detection methods are used to identify vortex cores and train the network, which can be automated in this way. However, relying on local detection methods can result in spurious vortices, leading to false positives and negatives during network training. Building upon Deng et al.’s work, an alternative approach for using CNNs to track and identify vortices is proposed in this study. Rather than using local detection methods for labelling, vortices are manually tagged on streamline plots. While this approach requires significant manual effort upfront, it only needs to be done once. By tagging enough vortices for various flow situations, a trained network can be applied to detect vortices accurately and reliably. This approach eliminates false positives and negatives, enhancing the overall robustness and accuracy of CNNs for vortex identification. The following chapter provides a detailed explanation of this approach.

2.2.2 Machine learning (ML)

2.2.2.1 overview

ML is a widely used tool for research related to the fields of computer science and cybernetics. Because of high-performance computing architectures and improved experimental measurement abilities, algorithms have been developed that empowered the modelling and analysis of big data. Three significant aspects of ML have developed from the earliest starting point: the classical work in emblematic learning by Hunt et al. [91], the statistical techniques is described by Nilsson [92], and the neural network (NN) learning single-layer by Frank Rosenblatt [93]. Each of the three branches developed methods [94]: design recognition or measurable methods, such as the Bayesian, k-nearest neighbours, K-means cluster, and discriminant analysis classifiers [95], inductive learning of representative guidelines, for example, top-down induction of decision trees [96], creating decision rules, induction of logic programs, and utilising ANNs like multilayered feed-forward neural networks (MFFNNs) with back-propagation learning, Kohonen's self-organising networks [97], and Hopfield's associative memory [98]. ML algorithms are classified as categorical or continuous based on the type of data. A numerical data set, for instance, velocity or pressure is represented by continuous data, while the type of flow gives categorical data, such as (bubbly/slug) or (laminar/turbulent) [99].

ML is a field of study within AI that explores how computers can learn from data and make assumptions without explicit programming. This discipline overlaps with computational statistics and involves using test data or past experience to create algorithms that optimise the features and parameters of the data to generate a mathematical model. This model can be used to make predictions or extract useful information [100]. Rather than following a set of pre-determined instructions, ML algorithms are designed to learn and adapt through the processing of data. By

analysing patterns in the data, these algorithms can generate models that allow them to predict future outcomes. Developing the ML model involves selecting a dataset and an algorithm that can best process that data. The algorithm is then trained on the dataset, and the output is optimised to generate a model that accurately represents the data. This model is then tested on a separate data set to determine how well it can predict outcomes or recognise patterns. If the model performs well on the test data, it can be used to make predictions on new, unseen data.

The field of ML has seen tremendous progress in the last two decades. ML is a set of algorithms that are designed to determine the most effective model based on previous training experiences so that prediction performance can be satisfied [101,102]. It is often inescapable that the computational barrier exists for different problems, even though there are numerous computational methods. Increasing the accuracy of detecting the vortices is one of the primary goals of numerical modelling in recent years, a task that would be beneficial to many major industries. ML plays an important role in improving accuracy and performance by combining numerical methods with data-driven algorithms. The definition of ML provided by Mitchell et al. [103]. is that “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E ”. The concept of ML describes how computers can solve problems by learning from experience and test sets provided as data, as opposed to traditional programming methods, which use code and step-by-step instructions for solving problems. By partly replacing numerical methods with empirical methods, ML algorithms can reduce some of the computational load of simulations.

Obtaining meaningful scientific features from data is not always a straightforward process. In some cases, it may take as long as the computations or experiments are required to generate the data. Identifying significant features in the data often re-

quires advanced visualisation techniques, statistical methods, and signal-processing algorithms. One useful tool for this task is ML, which has been demonstrated to be effective in solving pattern recognition, regression, and classification problems. ML is particularly well-suited for signal and image processing, as evidenced by its use in fields such as natural language processing, optical character recognition, and self-driving vehicles. As such, it is becoming increasingly common for industries to invest time and money in ML techniques to improve the quality of their products and services [104]. It is because ML offers a powerful way to identify patterns and trends in data, enabling businesses to gain insights that would be difficult to obtain through traditional methods. However, the application of ML has its challenges. One of the primary difficulties is the need for large amounts of high-quality data to train the models effectively. In addition, the algorithms used in ML can be complex and difficult to interpret, making it challenging to understand why a particular decision was made. Despite these challenges, the benefits of ML make it an attractive option for many applications and industries.

2.2.2.2 Machine learning fundamentals

As a method of tackling difficult decision and regression problems, ML has proven to be highly effective. Instead of stating explicitly what a solution should do, an expert provides sample data for training to get desired behaviour on representative problem instances rather than modelling the solution explicitly. As a way to understand and analyse the big data generated by experiments and numerical simulations, ML algorithms can be used to identify patterns in the data and develop accurate prediction models based on prior experiences. The following three broad categories of ML tasks can be classified according to the learning scenario available to the learning model [105]:

Supervised learning

The majority of ML techniques utilise a method known as supervised learning, where the aim is to predict a target variable (referred to as dependent variable y) based on a given set of predictors (independent variables x) [106]. The algorithm is designed to understand the connection between the input and output, and the process of training will persist until the model achieves the intended level of precision. During the training process, the algorithm is presented with a set of labelled examples, where both the input and output are known, and the model adjusts its parameters to optimise the prediction performance. The training data calculates the difference between the predicted and actual values, and the algorithm adjusts its weights and biases accordingly to minimise this error. This process is repeated many times until the model can accurately predict the target variable for new, unseen data. In essence, supervised learning is a technique used in ML to train models to make accurate predictions by learning the relationship between input and output through labelled examples as the following equation.

$$Y = f(x) \tag{2.2.1}$$

The problems of supervised learning can also be grouped into the categories of regression and classification. Examples of Supervised Learning:

- classification
 1. Decision Tree
 2. Random Forest
 3. K-nearest neighbor
 4. Neural networks (NNs)
 5. Support vector machines

6. Logistic Regression

- Regression

1. Linear
2. Gaussian process
3. Generalized Linear

unsupervised learning

Unsupervised learning differs from supervised learning because it does not require a target or outcome variable to make predictions. Instead, it relies solely on input data (X) to learn more about its underlying structure and distribution. This approach enables the discovery of patterns and relationships within the data without the need for prior knowledge or labelling [107]. One popular application of unsupervised learning is clustering, which involves grouping similar items or observations into clusters. Clustering is used to segment things into distinct groups. The use of unsupervised learning in the analysis of complex data sets is generally seen as a powerful and flexible tool. Examples of Unsupervised Learning:

- Clustering

1. K-means
2. Spectral clustering
3. Apriori algorithm

- Dimensionality reduction

1. Diffusion maps
2. Self-organizing maps
3. principal component analysis PCA

4. proper orthogonal decomposition POD
5. Autoencoder and Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs)

reinforcement learning

Reinforcement learning is a type of ML where an algorithm is trained through trial and error in an environment. The algorithm learns from its past experiences and tries to make the most accurate decisions [108]. To achieve this, it needs to capture all available knowledge and use it to optimise its performance. A popular example of reinforcement learning is the Markov Decision Process. It is a mathematical model representing a decision-making situation where the outcome of each decision is uncertain. The Markov Decision Process considers the current state of the environment, the possible actions that can be taken, and the resulting rewards or penalties. In reinforcement learning, the algorithm's performance is evaluated based on how closely its actions match the desired outcome. The algorithm is then adjusted to maximise the expected reward while minimising the expected penalties. Through this process of trial and error, the algorithm becomes better at making decisions, and its performance improves over time.

2.2.2.3 Deep learning algorithm

The human brain is a sophisticated organ that contains massive NNs capable of performing complex cognitive, perceptual, and control tasks. These networks function by transmitting electric pulses through the axons, synapses, and dendrites of the brain [109], which allows the brain to perform computation-intensive perceptual functions such as image detection, speech recognition and efficient body control. The human brain is made up of several biological processing components that work in parallel to generate proper procedures, including thinking and understanding. These components function collectively to form NNs that enable the brain to perform complex tasks, such as language processing, memory consolidation, and

decision-making. The idea of ANNs was introduced in 1943 by McCulloch and Pitts. They developed mathematical models of biological nervous systems and modelled neurons as switches that receive input from other neurons. These neurons can be activated or deactivated based on the weighted sum of input from all neurons. This concept has been instrumental in the development of AI systems, as it provides a framework for designing algorithms that mimic the behaviour of biological NNs. One of the key features of NNs is their ability to learn from experience. This is achieved through a process called backpropagation, where the network adjusts its weights in response to a given input. As the network is exposed to more data, it can gradually improve its performance and accuracy.

ANNs are one of the subsets of ML. It consists of a set of algorithms and a set of rules that are based on the biological NNs that are found in our brains for the purposes of classification and regression [110]. In NNs, input layers are related to outputs using activation functions. Non-linear neuron theories can be expressed as follows [111]:

$$h_{w,n}(x) = f \left(\sum_{i=1}^{PL_n} w_i x_i + b \right) \quad (2.2.2)$$

Where b is the bias, w_i 's are weights of the network, PL_n is the number of neurons at layer n , and f is called the activation function, which can be chosen as a Sigmoid (logistic function), Tanh (hyperbolic tangent), ReLu (rectified linear unit), or Softmax. The NN functions by taking in various data sources or inputs and then transforming them through the use of nonlinear activation functions. This process is repeated for every hidden layer in the network until the output layer is eventually reached.

During the last few years, deep learning has been widely applied to fluid dynamics, as it is a component of ML [112–115]. Ling et al. [116] developed a DNN architecture

that incorporates Galilean invariance to model Reynolds stresses, demonstrating the superiority of combining deep learning and domain expertise. Tracey et al. [117] modelled the source terms in the turbulence model using a standard feed-forward neural network (FFNN). Zhang and Duraisamy [118] examined NNs as a means of predicting a turbulent production term correction factor. As a result, deep learning is one of the most powerful tools available today for extracting fluid dynamics features and predicting flow fields, deep learning is a promising field for the future.

Chapter 3

Methodology

In the following Chapter, the underlying methods used in this thesis are introduced and briefly described. The Navier-Stokes equations are presented in the form in which they are used to solve turbulent flow, and local vortex detection algorithms are described, as well as a proposed CV-based approach to detect the region of vortices using the line integral convolution method. Finally, ML approaches are used to detect vortex cores within the regions.

3.1 Governing equations for incompressible CFD

3.1.1 Navier-Stokes Equations

Navier-Stokes equations are nonlinear partial differential equations that describe fluid motion through laws that conserve mass, momentum, and energy. In spite of the fact that humans have explored fluid motion, mathematical models developed after the industrial revolution. The dynamic behaviour of fluids with constant viscosity was first explained in the book “Principia” by Sir Isaac Newton (1687). Earlier In the 18th century, Euler and Bernoulli formulated the equation of inviscid flow, now known as Euler’s inviscid equations. The theory of viscous fluids was

introduced in 1827 by Navier, making it much more realistic and challenging. The Navier-Stokes Equations were refined in 1845 by Sir George Stokes, who added Newtonian viscous terms to emanate the equation of motion of a viscous flow. Hence, the Navier-Stokes Equations were finally developed, which are used today to calculate fluid flow numerically [119, 120].

Fluid flow is driven by a variety of properties in the physical domain. In order to reveal fluid flow behaviour and construct a mathematical model, those properties must be defined appropriately so that a change between the physical and numerical space can be achieved. The main properties of fluid flow are velocity, pressure, density, temperature, and viscosity. Governing equations determine thermo-fluid incidents based on conservation laws. The Navier-Stokes equation (N-S) is a commonly used mathematical model to analyse the changes in these properties during dynamic and/or thermal interactions. Modify the equations in accordance with the content of the problem using the principles of conservation of mass, momentum and energy [120].

3.1.1.1 Conservation of Mass

The principle of conservation of mass states that the total mass of a substance within a closed system remains unchanged, as the substance can't be created or destroyed [121]. Therefore, in any given control volume, the mass entering the system through the inlet must be equal to the mass leaving through the outlet. This means that the mass flow difference between the two points must always be zero. In other words, the total amount of mass within the system must remain constant, and any changes in mass flow rate must be reflected equally in both the inlet and outlet. The continuity equation in the field of fluid dynamics represents

the principle of mass conservation and is formulated as follows [122]:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{U} = 0, \quad (3.1.1)$$

Where $\mathbf{U}$ is the velocity vector (u_x, u_y, u_z) and ρ the density of the fluid. The divergence operator *nabla* (Nabla) is expressed as:

$$\nabla = \vec{i} \frac{\partial}{\partial x} + \vec{j} \frac{\partial}{\partial y} + \vec{k} \frac{\partial}{\partial z} \quad (3.1.2)$$

While the flow in this study is considered single phase, and incompressible, the density is constant, so continuity is simplified as follows:

$$\frac{\partial \rho}{\partial t} = 0 \quad (\text{density is constant}) \quad (3.1.3)$$

$$\nabla \cdot \vec{\mathbf{U}} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \quad (\text{continuity equation}) \quad (3.1.4)$$

3.1.1.2 Conservation of Momentum

Conservation of momentum is an essential principle in fluid mechanics critical in CFD. The conservation of momentum principle states that in the lack of external forces, the total momentum of a closed system remains constant [123, 124]. This means that if a fluid is flowing through a closed system, the total momentum of the fluid at any given time must be equal to the total momentum of the fluid at any other time, provided no external forces are acting on the system. In other words, the momentum of the fluid cannot be created or destroyed, only transferred between different parts of the system. Generally, the law of momentum conservation in a control volume is constant. It is derived by applying a direct consequence of Newton's third law of motion by noting that mass must also be conserved:

$$\mathbf{F} = m \cdot \mathbf{a} \quad (3.1.5)$$

Where F represents the impulse of force applied to any particle, m is the mass, and a is the acceleration. The following equation provides a helpful way to express the equation above in terms of a molecule's volume:

$$\rho \frac{DU}{D_t} = f = f_{body} + f_{Internal} \quad (3.1.6)$$

Where:

- ρ the density, and t the time
- $\frac{D}{D_t}$ indicates the rate of change of momentum and the substantial derivative as follows:

$$\frac{D()}{D_t} = \frac{\partial}{\partial t} + u \frac{\partial}{\partial x} + v \frac{\partial}{\partial y} + w \frac{\partial}{\partial z} = \frac{\partial}{\partial t} + U \cdot \nabla() \quad (3.1.7)$$

- Body forces f_{body} : applies force on a mass of fluid inside of a volume, and the most common example is gravity force $\vec{g}$, which acts on the whole mass of fluid as follows:

$$f_{body} = m \cdot g \quad (3.1.8)$$

- Internal forces: These exist at the surface or the sink of a volume, such as surface forces, and can be expressed as pressure and viscous forces as follows:

$$f_{internal} = -\nabla p \delta_{ij} + \nabla \tau_{ij} \quad (3.1.9)$$

Where p represents pressure, δ_{ij} the Kronecker delta symbol (δ_{ij}) equals one when i and j are equal and equals zero when they are not equal, this symbol plays an important role in ensuring that the formula provides accurate results for normal stresses, and τ_{ij} tensor representing all viscous stresses acting parallel to a fluid body surface is known as the viscous shear stress tensor. Based

on Stokes general law for Newtonian viscous fluids, the stress tensor τ_{ij} can be expressed as follows [119]:

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + \delta_{ij} \lambda \nabla \cdot U \quad (3.1.10)$$

where μ is dynamic viscosity, λ is viscosity to relate to the volumetric deformation, and U is the velocity vector (u_x, u_y, u_z) .

Newton's equation of motion can be expressed as follows:

$$\rho \frac{\partial U}{\partial t} + \rho \nabla \cdot (UU) = \rho \cdot g - \nabla p \delta_{ij} + \nabla \tau_{ij} \quad (3.1.11)$$

Substitution of equation 3.1.10 into 3.1.11 results in the Navier-Stokes equations of Newtonian viscous fluid in one equation:

$$\rho \frac{\partial U}{\partial t} + \rho \nabla \cdot (UU) = \rho \cdot g - \nabla p \delta_{ij} + \nabla \left[\mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + \delta_{ij} \lambda \nabla \cdot U \right] \quad (3.1.12)$$

According to the assumption that the fluid's density is constant and $\nabla \cdot U = 0$, the conservation of momentum for incompressible fluid flow is given by:

$$\rho \frac{\partial U}{\partial t} + \rho \nabla \cdot (UU) = -\nabla p + 2\mu \nabla^2 U + \rho \cdot g, \quad (3.1.13)$$

3.1.1.3 Conservation of energy

Conservation of energy is an essential aspect of CFD simulations [120]. In a CFD system, energy conservation refers to conserving a fluid's total energy. In other words, the total amount of energy within a system must remain constant, including kinetic, potential, and internal energy. Conservation of energy is often expressed in CFD using the principle of the first law of thermodynamics, which states that the change in the total energy of a system is equal to the sum of the work and the heat

added to the system. In numerical simulations, this is expressed as:

$$\Delta E = W + Q \quad (3.1.14)$$

Where ΔE is the change in total energy of the system, W is the work done on the system, and Q is the heat added to the system. As a result of energy conservation, CFD simulations ensure that the numerical solution accurately represents the physical system that is being modelled. Energy conservation also helps identify errors and inaccuracies in the simulation, which can be caused by numerical errors, modelling approximations, or other factors. The use of numerical schemes that have been designed to conserve energy is a common method of ensuring energy conservation in CFD simulations. For instance, numerical schemes such as the finite volume method (FVM) [125] are inherently conservative, which means they automatically conserve mass, momentum, and energy.

3.1.2 Large Eddies Simulation

With computers' increased processing power and memory capabilities, engineers are using simulation of fluid flows to solve turbulent flows. As computer processors and memory have grown, CFD has become increasingly capable of simulating complex fluid flows and is therefore used by academia and industry. The three most commonly used CFD methodologies are [126]:

- Direct Numerical Simulation (DNS)
- Large Eddy Simulation (LES)
- Reynolds-Averaged Navier Stokes (RANS)

All of them solve differently with Navier-Stokes' turbulence closure problem. The most accurate numerical model of turbulent flow is Direct Numerical Simulation (DNS). DNS resolves even the smallest eddies, which have Kolmogorov scales, which

are theoretically the smallest length scale in a flow. However, DNS is too expensive to use in practice due to its computational cost [127]. In the RANS approach, the velocity field is split into two parts: a Reynolds-averaged (average in time) part and a fluctuating part, with the Reynolds-averaged part simulated and the unsimulated fluctuating part using a closure model. The method runs quickly since all eddy scales are modelled, but it provides less information on fluid flow. The LES method resolves the largest scales, which contain the large kinetic energy, while small eddies are modelled to reduce computing costs and provide adequate and accurate information about turbulent flows by introducing some filters, so this method has been used.

In order to describe fluid motion, Navier-Stokes equations are used. Essentially, it is a set of conservation laws, namely mass conservation, momentum conservation, and energy conservation. It describes the motion of fluids using five equations and six unknown quantities (density, pressure, energy (temperature), and three velocity components in the x-, y- and z-directions). The Navier-Stokes equations for a compressible fluid (a fluid with a variable density) can be closed using a suitable state equation, such as the ideal gas law. A constant density is possessed by the fluid being studied here, which means that no such relation exists as the fluid is incompressible. In this case, various pressure-velocity coupling schemes have been proposed that work in a predictor-corrector fashion and iteratively find the solution to a given problem, of which the SIMPLE scheme of Patankar is one of the most widely used one [128]. The conservation of momentum is given by

$$\rho \frac{\partial U}{\partial t} + \rho \nabla \cdot (UU) = -\nabla p + 2\mu \nabla^2 S - \nabla \tau, \quad (3.1.15)$$

where U is the velocity vector (u_x, u_y, u_z) , ρ the density, t the time, p is pressure, μ is dynamic viscosity and S and τ being the rate-of-strain and stress tensor, respectively.

The rate-of-strain tensor can be defined in tensor notation as

$$S_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \quad (3.1.16)$$

and the stress tensor as

$$\tau_{ij} - \frac{1}{3} \tau_{kk} \delta_{ij} = -2\nu_{SGS} S, \quad (3.1.17)$$

The conservation of mass, in turn, is given as

$$\nabla \cdot U = 0. \quad (3.1.18)$$

As we can see in the equations above, we have spatially filtered the Navier-Stokes equations where small-scale turbulence features that are smaller than a filter width Δ_{LES} have to be modelled through a subgrid-scale model. The filtering operation produces a stress tensor τ that is commonly modelled with the relation provided in Eq.(3.1.17), where the contribution of τ_{kk} can be ignored. the turbulent stresses are replaced through the strain-of-rate tensor S , which is scaled by the turbulent subgrid-scale viscosity and which, in turn, accounts for the filtered smallest turbulent eddies. In this approach, the smallest scale is modelled, and only the largest turbulent flow structure is resolved, making it known as large-eddy simulations or LES for short. To resolve turbulence and vortical flows using LES, a subgrid-scale model [129] must be introduced to compute the turbulent viscosity to close the set of momentum equations, i.e. Eq.(3.1.15). In this study, the Smagorinsky-Lilly [130] model is used, which is given by

$$\nu_{SGS} = C_S \Delta_{LES}^2 \sqrt{2SS}, \quad (3.1.19)$$

Where C_S is an empirical constant and tuned for a range of canonical turbulent flows, and Δ_{LES} is the filter width below which turbulent eddies are filtered (typically related to the local mesh size). LES offers the advantage that turbulent vortical

structures can be generated quickly while only removing the smallest length scales that are easily modelled. The alternative is to model the fully turbulent flow field, which still requires substantial computational resources. The Navier-Stokes equations are not filtered but instead ensure that the computational grid can capture all turbulent length scales (in which case the stress tensor τ does not need to be modelled and can be computed directly from the resolved velocity field). This approach is referred to as DNS. Since we are only interested in detecting macroscopic vortical structures, LES provides excellent accuracy and sufficient information about turbulent flows, which is suitable for capturing vortices for training the CNN model. Therefore, LES has been used in this study to generate data quickly.

3.2 Local vortex identification approaches

The detection methods rely on the velocity gradient tensor, ∇U , that can be written as

$$D_{ij} = \frac{\partial u_i}{\partial x_j}, \quad (3.2.1)$$

where $i = 1, 2, 3$ for x_i and u_i correspond to x, y, z and u, v, w , respectively. The velocity gradient tensor D_{ij} can be decomposed into a symmetric and anti-symmetric tensor as $D_{ij} = S_{ij} + \Omega_{ij}$ with

$$S_{ij} = \frac{1}{2} (D_{ij} + D_{ji}), \quad (3.2.2)$$

$$\Omega_{ij} = \frac{1}{2} (D_{ij} - D_{ji}), \quad (3.2.3)$$

S_{ij} is the component of the symmetrical velocity gradient tensor, often referred to as the rate-of-strain tensor. On the other hand, Ω is part of the tensor that is anti-symmetric and represents the rate of rotation tensor. The eigenvalues velocity gradient tensor are required to perform local detection methods. We assume that σ_1 ,

σ_2 , and σ_3 are three eigenvalues. The characteristic equation for D is given as [131]

$$\sigma^3 + P\sigma^2 + Q\sigma + R = 0, \quad (3.2.4)$$

where P , Q and R are the first, second, and third invariants of the velocity gradient tensor. Using the decomposition into symmetric and anti-symmetric tensors, these invariants can be expressed as

$$P = -(\sigma_1 + \sigma_2 + \sigma_3) = -\text{tr}(\nabla U) = 0 \quad (3.2.5)$$

$$Q = \sigma_1\sigma_2 + \sigma_2\sigma_3 + \sigma_3\sigma_1 = -\frac{1}{2} [\text{tr}(\nabla u^2) - \text{tr}(\nabla u)^2] \quad (3.2.6)$$

$$R = -\sigma_1\sigma_2\sigma_3 = -\det(\nabla U). \quad (3.2.7)$$

The following four criteria of eigenvalue-based criteria, namely the Q criterion, the Delta criterion, The Swirling Strength criterion, and λ_2 criterion, are the most common local detection methods based on the velocity gradient tensor D_{ij} [132].

3.2.1 Q criterion

The Q criterion is one of the most popular methods of identifying vortices proposed by Hunt et al. [7]). The Q criterion identifies vortices as flow regions with the positive second invariant of ∇U , i.e. $Q > 0$, which can be expressed as

$$Q = \frac{1}{2} (\|\Omega\|_F^2 - \|S\|_F^2), \quad (3.2.8)$$

where S is the symmetric part of the velocity gradient tensor, which is commonly known as the rate-of-strain tensor, and Ω is the anti-symmetric component representing the rate of rotation tensor:

$$\mathbf{S} = \frac{1}{2} (\nabla \mathbf{U} + \nabla \mathbf{U}^T) = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{1}{2} \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) & \frac{1}{2} \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \\ \frac{1}{2} \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right) & \frac{\partial v}{\partial y} & \frac{1}{2} \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \\ \frac{1}{2} \left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \right) & \frac{1}{2} \left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} \right) & \frac{\partial w}{\partial z} \end{bmatrix} \quad (3.2.9)$$

$$\mathbf{\Omega} = \frac{1}{2} (\nabla \mathbf{U} - \nabla \mathbf{U}^T) = \begin{bmatrix} 0 & \frac{1}{2} \left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} \right) & \frac{1}{2} \left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x} \right) \\ \frac{1}{2} \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right) & 0 & \frac{1}{2} \left(\frac{\partial v}{\partial z} - \frac{\partial w}{\partial y} \right) \\ \frac{1}{2} \left(\frac{\partial w}{\partial x} - \frac{\partial u}{\partial z} \right) & \frac{1}{2} \left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z} \right) & 0 \end{bmatrix} \quad (3.2.10)$$

and $\|\cdot\|_F^2$ represents the Frobenius norm. In practice, a $Q_{threshold}$ is used to define the region of vortices with $Q > Q_{threshold}$.

3.2.2 Delta criterion

The delta criterion uses the critical point theory of Chong et al. [9] and defines a vortex core for regions where D_{ij} has complex eigenvalues. These are identified using the discriminant of the characteristic equation, which is given by

$$\Delta_c = \left(\frac{1}{2} R \right)^2 + \left(\frac{1}{3} Q \right)^3 > \epsilon, \quad (3.2.11)$$

In a strict mathematical sense, ϵ is equal to zero, which can lead the result to falsely detected vortices. These false positives can be reduced by defining a value of $\epsilon > 0$, which improves the accuracy of the model but decreases robustness since each vortex is required a specific parameter. When set ϵ too low, results in vortices are detected in areas where there are no vortices exist (false positives), while when set ϵ too high, results in vortices are not detected despite existing in physical space (false negatives).

3.2.3 Swirling Strength criterion

The Swirling strength criterion of Zhou et al. [10] is similarly defined to the delta criterion as the region where D_{ij} has complex eigenvalues. Unlike the delta criterion, however, Zhou et al. [10] used the velocity gradient decomposition as follows

$$D_{ij} = \begin{bmatrix} v_r & v_{cr} & v_{ci} \end{bmatrix}^T \begin{bmatrix} \lambda_r & 0 & 0 \\ 0 & \lambda_{cr} & \lambda_{ci} \\ 0 & -\lambda_{ci} & \lambda_{cr} \end{bmatrix} \begin{bmatrix} v_r & v_{cr} & v_{ci} \end{bmatrix}^T. \quad (3.2.12)$$

Where λ_r is defined as the real eigenvalue which is related to the real eigenvector v_r , and we have the conjugate pair of complex eigenvalues $\lambda_{cr} \pm i\lambda_{ci}$ which is related to the corresponding eigenvectors $v_{cr} \pm iv_{ci}$. The local swirling strength of a vortex is determined via λ_{ci} , and for values greater than zero, a vortex is identified. This definition, however, produces similar falsely detected vortices and therefore, a more conservative method would be to define $\lambda_{ci} > \epsilon$. In this approach, the threshold parameter ϵ , which is again a vortex-specific parameter, has to be fine-tuned. As a result of the singular nature of the threshold parameter (ϵ), false positives and false negatives detection cannot be avoided since vortices may be excluded below the threshold, or spurious vortices are allowed to be identified above the threshold.

Lindner et al. [131] integrated the vortex detection techniques mentioned above into their Vortex Fitting toolbox, which is utilised to identify vortices by adjusting the threshold parameter ϵ specifically for each case. It will be demonstrated that the results are highly dependent on ϵ value and that even though the most suitable threshold value is searched for, false positives and negatives still occur. To overcome and address this shortcoming, a hybrid CV and ML approach is proposed to detect vortex cores and reduce false positives and negatives based on streamline-based vortex detection and numerical data.

3.3 Streamline detection using the line integral convolution algorithm

According to Cabral and Leedom [133], the LIC algorithm is used to extract the velocity vector from each pixel and calculate the local streamlines from that vector in order to prepare an image data set for a CV framework. The numerical algorithm integrates along streamlines, but the speed relies on the vector field at the point of integration, so the lines will be long in locations with fast flows and vice versa. In order to create a streamline plot, a white noise image value is applied to the surface at the resolution required for the plot. It then computes the following integral:

$$C(\mathbf{r}) = \int_{-L/2}^{L/2} k(s)N(\sigma(s))ds, \quad (3.3.1)$$

where $C(r)$ represents the colour value at location r , s being the arc length along the streamline, $k(s)$ is an appropriately chosen kernel function to limit colouring to local space and $N(\sigma(s))$ containing the gray-scale image. The along streamline ds is integrated for a distance L . It is applied for each pixel in the domain which produces LIC plots, as shown in Figure 3.1. This image was obtained by taking the snapshot of the X-plane for a simulation of the Taylor green vortex. Compared to traditional streamline plots, LIC streamline plots provide the advantage of covering the entire image, whereas traditional streamline plots require a starting location through which streamlines have to pass, which does not guarantee full coverage of the domain. Furthermore, in 3D, streamlines may leave a 2D plane on which visualisation is sought, making it a difficult task to generate surface streamline plots and extract vortex cores automatically. The LIC streamline plots are generated using ParaView and are then pre-processed using the Torchvision module — which is part of the PyTorch Python package. This involves applying various transformations to the images, such as rotation, cropping, and flipping, to make the training data more



Figure 3.1: Vector field visualization (Line integral Convolution).

robust. To construct the training and validation data set, the planes were randomly chosen in the x , y or z directions. The vortex regions on the LIC images were manually labelled for the supervised learning approach. Although this step is time-consuming, it only needs to be completed once to ensure that it can be effectively applied to various fluid problems confidently. It has been demonstrated that this technique can subsequently be used for detecting vortices in different fluid problems, even those that have not undergone any prior training, with a high level of accuracy.

3.4 Vortex regions labelling

The process of labelling data is a way of identifying an object class associated with raw data like images and videos in CV by adding tags to them. To identify a particular object class in unclassified data, the supervised YOLO model utilises labels, which help them associate meaning with data, thus helping them to train the model. The labelling of vortices is a challenging task and requires to be done carefully in order to avoid false positives and negatives. Due to the fact that vortices

are not well-defined structures (where does a vortex start and where does it end?), two different labelling strategies have been tested and analysed and compared their effects on the results. The two strategies are illustrated schematically in Figure 3.2. On the left-hand side, a method of labelling is presented where the bounding boxes

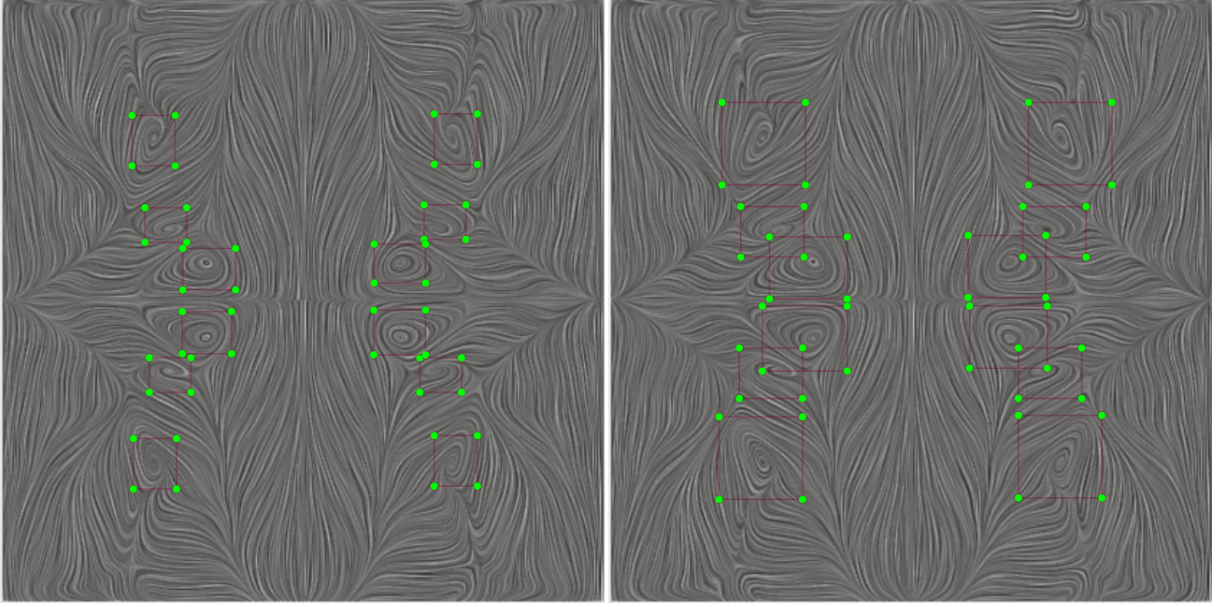


Figure 3.2: Different labelling strategies: On the left-hand side, non-overlapping bounding boxes are used and on the right-hand side labelled bounding boxes are allowed to overlap, capturing more of the vortices' structures.

do not overlap, ensuring that each bounding box takes into account only one vortex and that influences from a close adjacent vortex do not bleed into this bounding box. Unlike the left side, the bounding boxes on the right-hand side overlap and focus on capturing as much of the vortex as possible, allowing neighbouring vortices to influence the vortex detection. Based on this concept, neighbouring vortices will affect the shape and structure of the vortex itself, so utilising that method may be helpful in predicting vortices with non-perfectly spherical shapes more accurately.

3.5 CV-based vortex detection regions using YOLO algorithm

CV, which overlaps with the domain of AI, is a specialized area that involves the extraction of significant information from digital images and videos. Specifically, in this study, CV is utilised to extract vortices from streamline plots. To enhance the accuracy of vortex core region detection while minimizing the number of false positives and negatives, YOLO version 3 has been employed, a state-of-the-art algorithm proposed by Redmon et al. [134]. YOLO algorithm [13] has recently gained attention due to its impressive balance between speed and precision in object detection [135, 136]. This algorithm operates on a model that sends the entire image through a NN in a single detection stage (one-stage detection). It utilises CNNs to detect objects, requiring only one forward propagation through a network. In

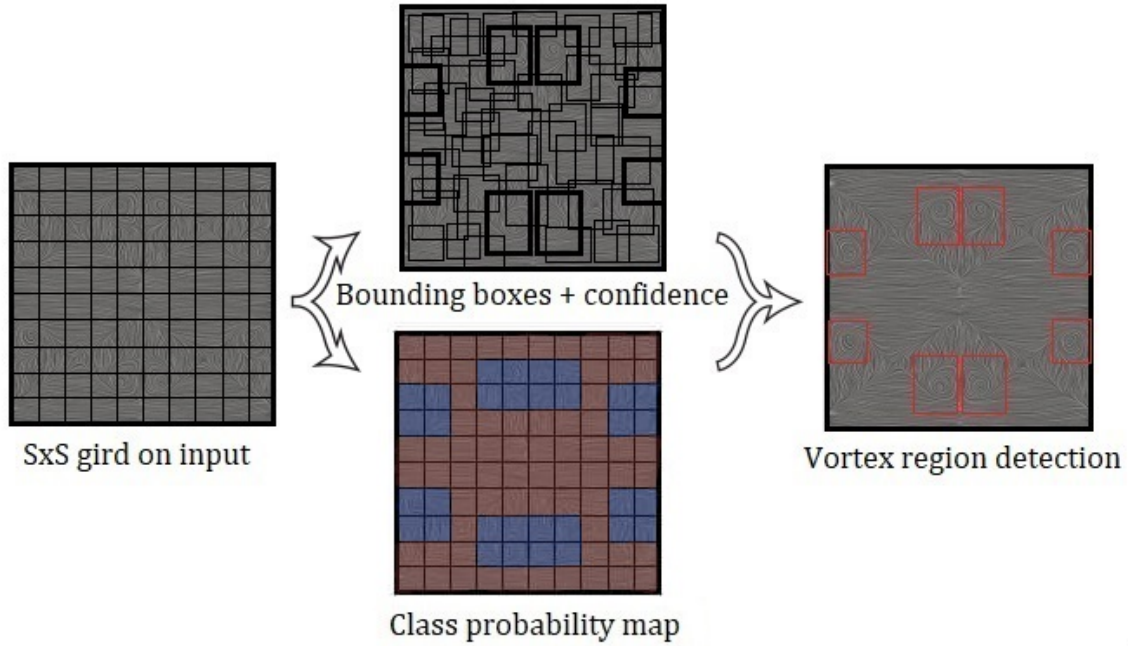


Figure 3.3: YOLO model detection.

Figure 3.3, the YOLO framework is illustrated. In this method, the input image is divided into a $S \times S$ grid cells and perform detections in each grid cell by providing probabilities for every cell. The value of S is a hyperparameter set before training

the YOLO model and is usually chosen based on the size of the input image and the desired output resolution. The higher the value of S , the more grid cells there will be, and the finer the spatial resolution of the predictions. A grid cell's task is to identify an object that is located within its centre. Each grid cell has the ability to predict B bounding boxes, where B is a number of a rectangular region that represents the predicted location and size of an object in an image which is equal to three bound boxes for each cell in this study. Each bounding box consists of $(5+C)$ values to predict the object as follows: x, y, w, h where (x, y) represent the centre coordinates of the box, and (w, h) represent the width and height of the box, confidence scores and C class probabilities respectively. These confidence scores can be used to ascertain whether an object is present in a particular grid cell. The confidence is formulated as follows:

$$Confidence = P_r(object) \times IoU_{pred}^{truth}, \quad (3.5.1)$$

Where $P_r(Object) \in [0, 1]$ and IoU is Intersection over Union between the predicted bounding box and the ground truth labelling [137]. In the case where there is no object in that cell, the confidence score equal zero, whereas if there is an object, the confidence score is equal the IOU .

3.5.1 YOLOv3 Architecture

The YOLO v3 model relies on a network called Darknet-53 as its backbone. This network has 53 CNN layers that extract feature images from the input. The basic components of YOLO's architecture are shown in Figure 3.4 and consist of residual blocks, skip connections and up-sampling layers. Each convolutional layer is followed by a batch normalization layer and Leaky ReLU activation function. YOLO v3 uses three different resolutions, responsible for detecting objects of different scales at three separate places in the network. These places are layers 82, 94, and 106.

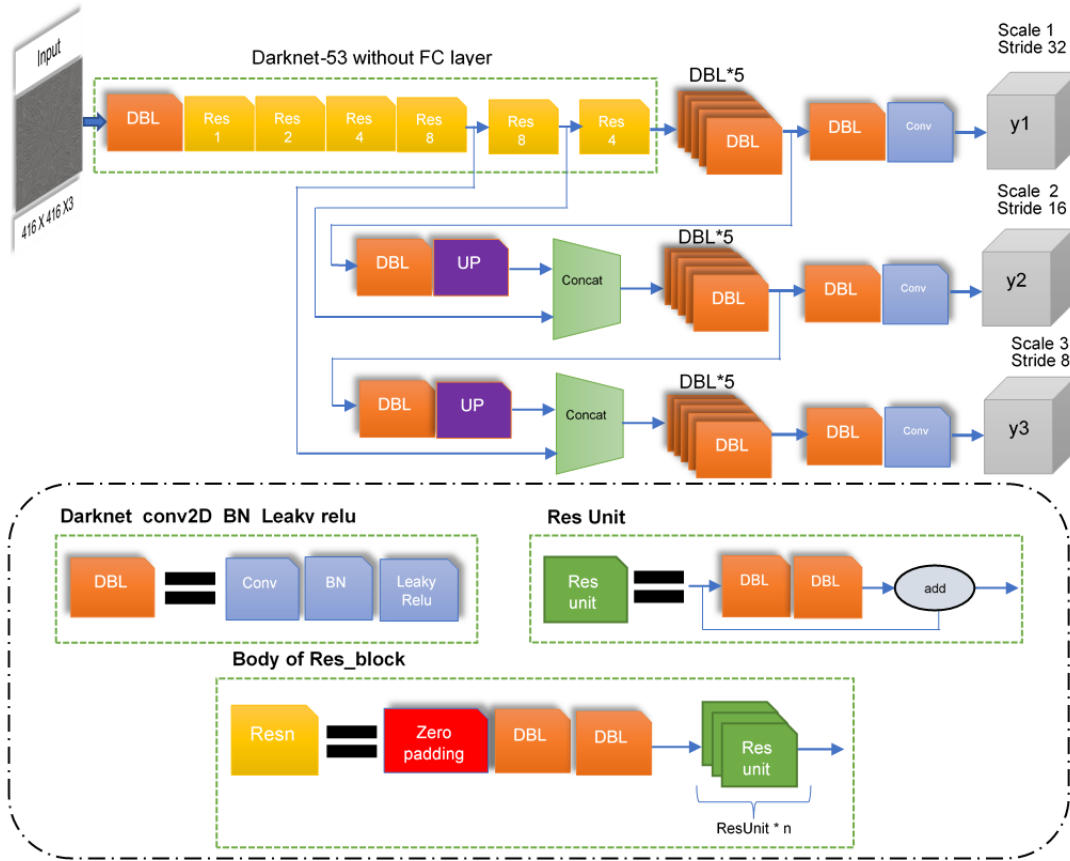


Figure 3.4: YOLOv3 architecture with input image size 416×416 and 3 types of scales (13×13 , 26×26 , and 52×52) as output; Darknet-conv2D-BN-Leaky ("DBL" for short) is the main component of YOLOv3 which contains one convolution layer, one batch normalization layer, and one leaky relu layer. residual-like unit ("ResUnit" for short) is two "DBL" structures following one "add" layer; several "ResUnit" with one zero-padding layer and "DBL" structure forward generates a residual-like block, "ResBlock" for short, which is the module element of Darknet-53. Figure adapted from Wang et al. [138]

Therefore, the original architecture of YOLO v3 adds 53 additional layers to the network, yielding a total of 106 layers for detection tasks. To detect objects of different scales, YOLO v3 uses three different strides, which correspond to different feature map resolutions. Strides 8, 16, and 32 are responsible for detecting small, medium, and large features in the images, respectively [138]. The input image to the network must be divisible by 32 without leaving a remainder, and in this case, the size of the input image is 416 pixels. With stride 32, the output size is 13 by 13, stride 16 generates an output size of 26 by 26, and stride eight results in an output

size of 52 by 52. YOLO v3 generates the same three different outputs of feature maps as YOLOv4 and subsequent versions to achieve multi-scale prediction. In previous versions, YOLOv1 and YOLOv2, detecting small objects was challenging because the images were divided into equal-sized grid cells. Each grid cell detected an object if the central point of the object fell into that cell. Although YOLOv4 and above may achieve slightly better accuracy than YOLOv3, especially in the MS COCO dataset, YOLOv3's detection speed is faster than that of YOLOv4 and the versions above [139]. For this reason, YOLOv3 is suitable for detecting small areas of vortices in this study, as it provides good speed and high accuracy.

3.5.2 Loss function

The loss function is an important factor that affects object detection precision. The loss function is a mathematical function that quantifies the difference between the predicted and ground truth values for a given task. In the case of object detection, the loss function measures the error in predicting the object's location, size, and class. A well-designed loss function can help the model to converge faster and produce more accurate predictions. Currently, there is no general loss function, and choosing one involves considering factors, including ML algorithms, model convergence times, and prediction confidence. In YOLOv3 There are three components of the loss function [140]:

- The bounding box position error (Coordinate loss)
- The bounding box confidence error (objectness loss- caused by an incorrect box-object IoU prediction)
- the classification prediction error between the actual vortices boxes (ground truth) and the predicted vortices boxes

$$\begin{aligned}
\mathcal{L} = & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\
& + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
& + \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\
& + \sum_{i=0}^{S^2} I_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned} \tag{3.5.2}$$

Where: S is the size of the image, B is the number of bounding boxes predicted by each grid cell, λ_{coord} is the arbitrary weight parameter and delivered in advance, the latter is much smaller than the former, and λ_{noobj} is the confidence loss weight when the grid cell contains no objects [141]. The parameters I_{ij}^{obj} and I_{ij}^{noobj} specify whether or not the j th bounding box of the i th grid cell is responsible for detecting the object. I_{ij}^{obj} denotes the the j th bounding box in the i th cell grid contains an object. while I_{ij}^{noobj} represents the bounding boxes that do not contain an object. $I_{ij}^{\text{obj}} = 1$ and $I_{ij}^{\text{noobj}} = 0$ If the object's centre point falls on the i th grid cell, the j th bounding box for that grid cell intersects with the ground-truth box with the greatest intersection ratio, otherwise, $I_{ij}^{\text{obj}} = 0$ and $I_{ij}^{\text{noobj}} = 1$. After normalisation, $(\hat{x}_i, \hat{y}_i, \hat{h}_i, \hat{w}_i)$ is the actual truth bounding box coordinates, and (x_i, y_i, h_i, w_i) is the predicted bounding box coordinates. $\hat{C}_i$ and C_i represent the confidence level of the ground-truth box and the confidence level of the predicted bounding box, respectively. $\hat{P}_i(c)$ denotes the class probability of the ground-truth box, and $P_i(c)$ depicts the class probability of the predicted bounding box.

3.5.3 mAP Evaluation metric

The mean Average Precision (mAP) is a commonly utilized measurement for assessing the precision of object detection techniques [142]. It takes into account both precision and recall values from 0 to 1, which are two crucial metrics for evaluating

the performance of an object detection model. The mAP computes a score based on the difference between the ground-truth bounding box and the predicted bounding box. The higher the score, the better the model's detection. The mAP is calculated by first computing the average precision (AP) for each class of object detected in an image. AP is a metric that measures the quality of the model's predictions by calculating the area under the precision-recall curve [143–145]. Then, the mAP is obtained by averaging the AP values across all classes. The following sub-metrics are used in calculating the mAP formula: True Positive (TP), False Positive (FP), and False Negative (FN). TP refers to the number of correct detections, while FP refers to the number of incorrect detections, and FN refers to the number of missed detections. These sub-metrics help to provide a more detailed evaluation of the model's performance and can be used to identify areas where the model needs improvement. Overall, mAP is an essential metric for evaluating the performance of object detection algorithms, and it is widely used in the CV community. By using mAP, researchers and practitioners can compare different models and identify areas for improvement, ultimately leading to more accurate and reliable object detection systems.

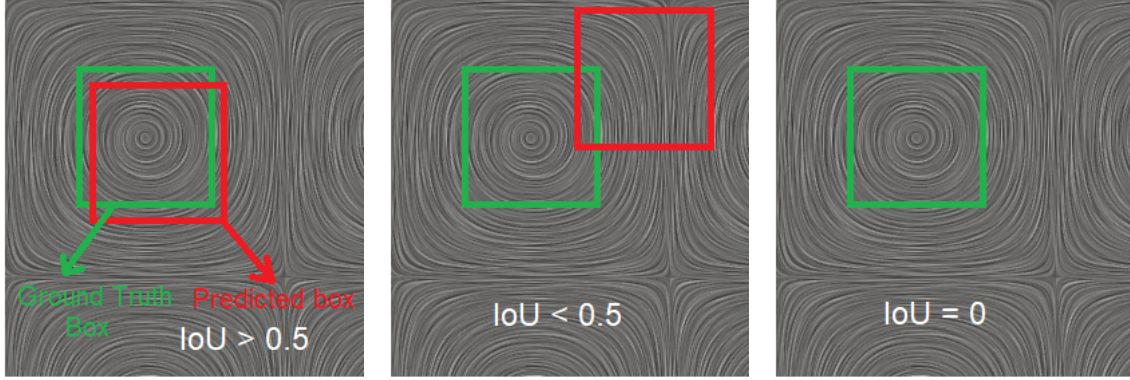
3.5.3.1 Intersection over Union (IoU)

Intersection over Union (IoU) indicates the overlap between the ground truth bounding box to the predicted bounding box coordinates. The IoU gets a high value when the predicted bounding box coordinates are closer to the ground truth, and the calculation formula is as follows [146]:

$$IoU = \frac{\text{Truth bounding box} \cap \text{predicted bounding box}}{\text{Truth bounding box} \cup \text{predicted bounding box}} \quad (3.5.3)$$

True positives are counted as those identified vortices of predicted bounding (red bounding boxes) as shown in Figure 3.5 that share an IoU with the ground truth

bounding box (the green bounding box that is manually created to label vortices), which is above an IoU threshold value of 0.5 in this case, whereas, the false positives are under the IoU threshold value.



True Positive $IoU > 0.5$.

False Positives $IoU < 0.5$.

False Negative $IoU = 0$.

Figure 3.5: Ground truth box vs predicted box.

3.5.3.2 Precision and Recall

The precision rate measures the accuracy of the predictions, and the calculation formula is as follows:

$$\text{Precision} = \frac{\text{True positives}}{(\text{True positives} + \text{False positives})}, \quad (3.5.4)$$

Meanwhile, recall measures how well all the true positives are found, which are defined as:

$$\text{Recall} = \frac{\text{True positives}}{(\text{True positives} + \text{False negatives})}. \quad (3.5.5)$$

3.5.3.3 Average Precision

The Average Precision (AP) is determined by finding the area under the precision-recall curve [143–145].

$$\text{AP} = \int_0^1 p(r) dr \quad (3.5.6)$$

In order to calculate AP for object detection, the actual pattern of precision is interpolated before calculating the area under the precision and recall curves, as

shown in Figure 3.6. Precision interpolated is equivalent to the maximum precision associated with recall greater than the current recall value.

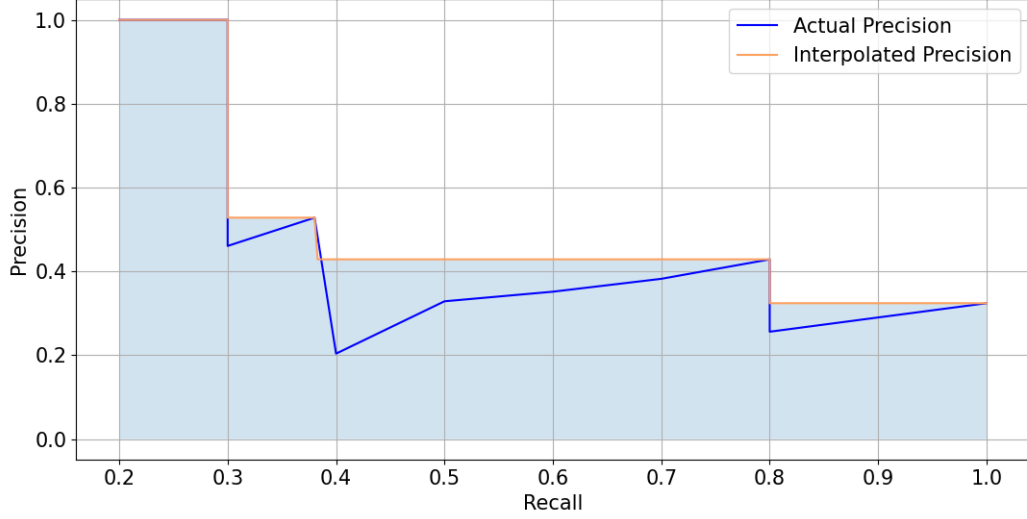


Figure 3.6: This plot showing how to interpolate Precision values.

The mAP is calculated by determining the Average Precision (AP) for each class and averaging them over all classes.

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^N \text{AP}_i \quad (3.5.7)$$

The mAP involves the adjustment between both precision and recall, and with FPs and FNs taken into account, thus it is an effective and appropriate detection metric for most applications.

3.6 Unsupervised K-Means Clustering Algorithm

Clustering is a widely employed unsupervised learning technique in data science that involves grouping similar data points together in order to identify meaningful patterns and relationships [147]. Clustering aims to partition a given data set into K clusters, where K is a predetermined number of clusters to be identified. The

most common approach to clustering involves generating initial estimates for the K centroids, typically by random selection. Then, each data point is assigned to its nearest centroid based on a distance metric, such as the squared Euclidean distance. The centroids are updated iteratively until the clustering algorithm converges and the clusters are stable. One popular algorithm for clustering is the k-means algorithm, which seeks to minimise the sum of squared distances between each data point and its assigned centroid. K-means has been successfully applied in a wide range of fields, from customer segmentation [148] and image analysis to bioinformatics [149–151] and social network analysis [152]. Overall, clustering is a powerful tool in data science that can help uncover hidden patterns and relationships in complex data sets. Let us consider a data set $\mathbf{X} = x_1, x_2, \dots, x_n$, where each point is a d -dimensional real vector. As the first step in the algorithm, initial estimates for the K centroids are generated by random selection. Each data point x is assigned to its nearest centroid using the squared Euclidean distance. Let $M = m_1, m_2, \dots, m_c$ where m_c is cluster centres. The given data point x is categorised into a particular cluster by using the following:

$$J(w, M) = \sum_{k=1}^c \sum_{i=1}^n w_{ki} \|x_i - m_k\|^2 \quad (3.6.1)$$

Where $w_{ki} = 1$ for data point x_i when it belongs to cluster k ; otherwise, $w_{ki} = 0$. After that, the centroids are recalculated to update them by taking the mean of all data points assigned to that centroid's cluster as following:

$$m_k = \frac{\sum_{i=1}^n w_{ki} x_i}{\sum_{i=1}^n w_{ki}} \quad (3.6.2)$$

$$\Rightarrow w_{ki} = \begin{cases} 1, & \text{if } k = \operatorname{argmin}_j \|x^i - m_j\| \\ 0, & \text{otherwise} \end{cases} \quad (3.6.3)$$

As an iterative process, the K-means algorithm repeats each step until no data points change clusters, the distances are minimised, or there is some maximum number of iterations.

In the study, CV and a K-means clustering algorithm have been employed to label flow field files automatically by identifying and clustering regions around vortex cores. First, regions surrounding vortex cores were identified, and their corresponding coordinates were represented as the $\mathbf{X}$ vector for use in Equation 3.6.3. Then, the K-means clustering algorithm was applied to group these coordinate values into distinct clusters. Figure 3.7 on the left-hand side illustrates the use of K-Means to group pixels in vortex core regions from each LIC image into similar groups, resulting in the identification of the centre of each cluster, referred to as the "centroid". This centre is calculated as the average of the pixels within the cluster and repre-

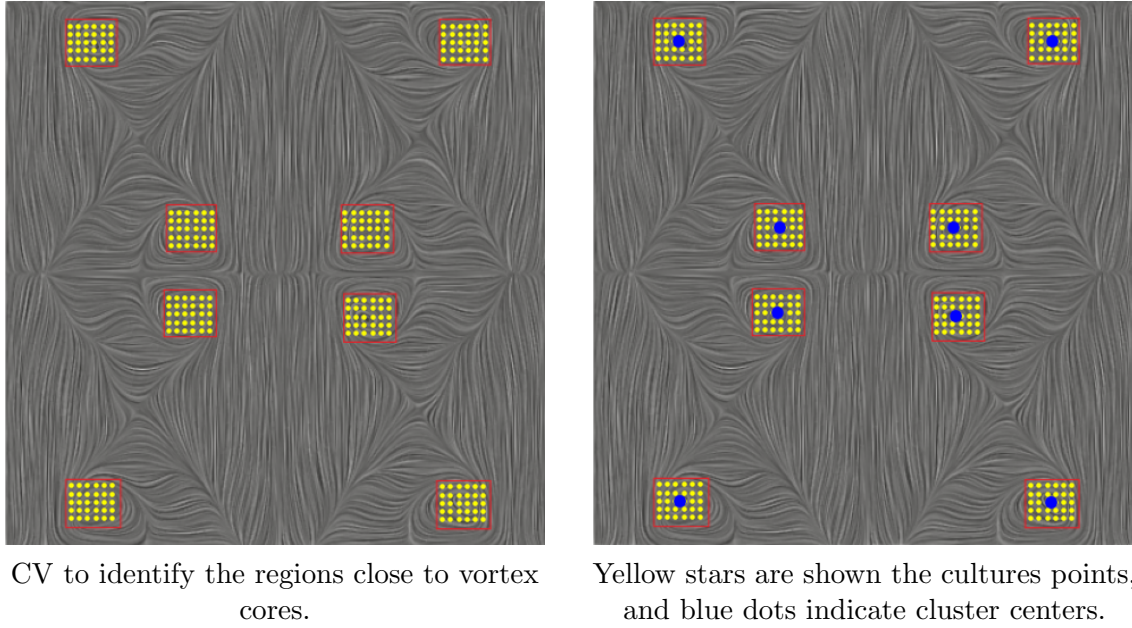


Figure 3.7: Labelling data sets automatically using CV and the K-means clustering algorithm.

sents the vortex core, as shown in The right-hand side of Figure 3.7. In other words, the K-means algorithm was used to group together pixels in the vortex core regions of the images to identify the location of the vortex cores.

3.7 ANN approach to detect vortex core within the regions

The ANN is constructed based on a mathematical model called a neuron, which is a representation of the fundamental structure and operation of NNs. The neuron model comprises synapses that regulate input signals through connection weights, and the neurons have non-linear properties via activation functions, also referred to as transfer functions. The artificial neuron operates by taking in weighted input values, which implies that each input value is multiplied by a particular weight. Then the neuron calculates the sum of all the weighted input signals and bias. At the neuron output, the summed-up weighted inputs and bias are fed into the activation function, as depicted in Figure 3.8. Flow fields (ASCII files) have been exported

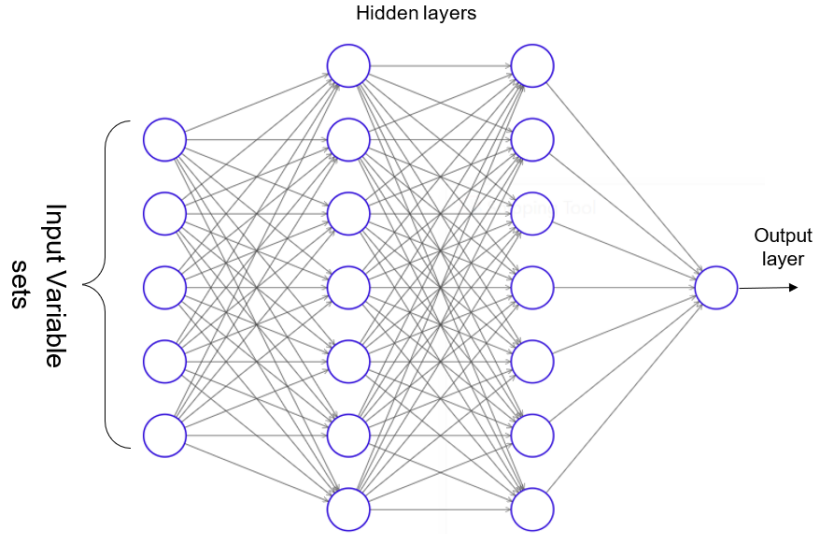


Figure 3.8: The figure shows a FFNN with one input layer, two hidden layers, and one output layer of the model.

from the test cases for the input. These flow fields consist of x , y , and z coordinates, and each time a variable is selected from the variables listed in Table 3.1. Once the input features have been normalised including coordinates (x,y,z) using the Keras library and partitioned into separate training and testing sets, they are subsequently used as input for training the neural network. The model's output comprises coor-

ordinates (x, y, z) that have been transformed back into the image space or physical as represented in flow field files which are exported from test cases and accompanied by a binary value of 1 if it represents a vortex core, and 0 if it does not.

Table 3.1: Variable sets exported from Ansys Fluent software

S	Variable sets
Set 1	x-velocity, y-velocity, w-velocity, velocity magnitude and pressure
Set 2	Set 1 + velocity gradient tensor and pressure gradient
Set 3	Set 1 + u-vorticity, v-vorticity, w-vorticity and vorticity magnitude
Set 4	Set 2 + u-vorticity, v-vorticity, w-vorticity and vorticity magnitude
Set 5	Set 4 + dynamic pressure, normalised Q-criterion and absolute Q-criterion

The purpose of this study is to accurately identify and detect the cores of vortices in a flow field using a combination of YOLO algorithm and ANN. By using YOLOv3 to identify the region of vortices and obtaining the numerical data around the vortex cores, the amount of numerical data needed to be analysed is significantly reduced, allowing for more efficient and accurate analysis. Additionally, the study aims to compare the results of the hybrid approach with a pure ANN approach, which uses whole numerical data of flow field files for the LIC image. To ensure accurate analysis, the ANN model's training data set was labelled manually at first and automatically using CV and k-means clustering. The study's findings can be used to improve the understanding of the behaviour of vortices in flow fields and their impact on various engineering applications, including aerodynamics and fluid dynamics. The study may also have broader implications for other fields that rely on analysing complex data sets.

The data set is split into 70% training data set, 20% validation data set to overcome overfitting, and 10% test data set to test the model. The features of the data-set for the single approach (pure ANN) and the hybrid approach (YOLO and ANN) are

divided into five variable sets as shown in table 3.1 [12]:

In this study, TensorFlow and the Keras API, open-source Python libraries, are used to train the model. In input and hidden layers, Rectified Linear Units (ReLU) activation function is utilised. Activation functions play a crucial role in NNs, as a NN without them is simply a linear regression model. This means that the activation function makes the NN nonlinear. The activation function ReLU is extensively utilised in deep learning models. This function outputs a value of 0 when given any negative input and outputs the input value when given a positive input. This can be expressed in the following manner. [153]:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.7.1)$$

While in the output layer, the softmax activation function is employed. With the Softmax activation function, the actual values are converted into probabilities mathematically, where the probabilities of each value are proportional to their relative scale. It's only used as an output layer of the NN, and the higher probability represents the actual output. SoftMax's activation function is shown in the following equation [154].

$$\text{softmax}(a_i) = \frac{e^{a_i}}{\sum_k e^{a_k}} \quad (3.7.2)$$

Where the a represents the vector values from the neurons of the output layer, the exponential indicates the non-linear function, and these values are then normalised and converted into probabilities by dividing by the sum of the exponential in the output layer. For optimisation, Adaptive Moment Estimation (Adam) is used to update the model based on the output of the loss function. The Adam algorithm is capable of handling problems that involve a large amount of data and parameters, while also being computationally efficient and not requiring a significant amount of memory [155]. In the Adam algorithm, Momentum and Root Mean Square Prop-

agation (RMSProp) heuristics are combined. In RMSProp, past squared gradients are accounted for by exponentially decaying averages of past squared gradients v_t , whereas Momentum represents the exponential decaying average of past gradients m_t . To calculate the decaying averages of past and past squared gradients (m_t and v_t , respectively), the following process is used [156]:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla_w J(w) \quad (3.7.3)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \nabla_w^2 J(w) \quad (3.7.4)$$

$$\alpha = \eta \frac{\sqrt{(1 - \beta_2^t)}}{(1 - \beta_1^t)} \quad (3.7.5)$$

Where m_t and v_t are moving averages of the gradients' first moment (the mean) and second moment (the variance), ∇ is the gradient for the current mini-batch, β_1 and β_2 represent hyper-parameters of the optimisation algorithm with default values of 0.9 and 0.999 respectively, and η represents the learning rate with value 0.001. Then these are used to update the weights parameter until a global minimum value is reached, which triggers the Adam update rule as follows:

$$w_t = w_{t-1} - \alpha \frac{m_t}{\sqrt{v_t} + \epsilon} \quad (3.7.6)$$

Where ϵ is equal $1e - 8$ for stability.

3.8 Summary

The architecture diagram in Figure 3.9 illustrates the end-to-end process through which different software and tools are used to identify vortices for various purposes. Geometry and mesh refinement of the test cases are created using ICEM CFD as part of the Pre-processing. With the mesh file generated, the LES model is used to solve the turbulent flow using Ansys Fluent 2019 software to produce Tecplot files

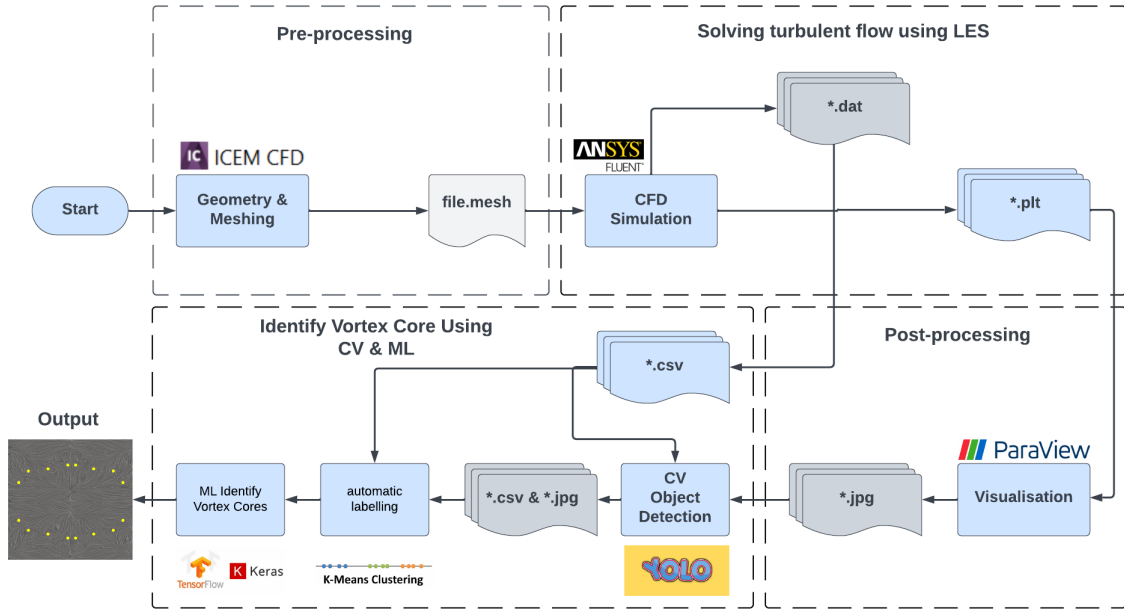


Figure 3.9: Process flow diagram architecture to identify vortex cores.

for visualisation and flow field files with numerical data, such as velocity, pressure, etc., to be utilised in the ML using ANN algorithm. Then streamline plots are generated for x, y, and z planes in the computational domain using ParaView which the velocity vector is extracted and then compute local streamlines for each pixel with the LIC algorithm as input to the CV framework. Once the images have been labelled, they are passed to the YOLOv3 algorithm in the CV framework to determine the region of the vortices. After identifying the regions that contain vortical structures using CV, ANN machine learning technique is employed to pinpoint the exact location of the vortex core within each identified vortex area. Within these regions, which were identified by YOLOv3, the values of different variable sets of components are obtained to train the model using ANNs (TensorFlow 2, Keras). Only velocity and pressure variables are initially considered, and subsequently, add more derived quantities (velocity gradients, pressure gradients, Q-criterion, vorticity, and magnitudes of vector quantities), which this study is composed of five different sets of variables is shown in Table 3.1. Due to this, a hybrid approach decreases the training time for all tested cases, and also it reduces the number of false positives

and negatives where we can see the comparison and difference between the hybrid approach and the single approach (pure ML using ANN) in the result section.

Chapter 4

Test Case

This study will investigate three different fluid flow scenarios: the Taylor-Green Vortex (TGV) problem, turbulent boundary layer flows, and rotating wings. The objective is to analyze and predict the vortices present in each of these scenarios. The Taylor-Green Vortex problem was selected due to its status as a well-known benchmark in fluid mechanics and computational fluid dynamics (CFD). Moreover, this problem exhibits a wide range of vortices at various scales, making it convenient for prediction and analysis. The two additional test cases involve distinct simulations and feature specific types of vortices, namely horseshoe vortices and turbulent boundary layers, which are not present in the Taylor-Green Vortex problem to test the hybrid model for identifying vortex cores.

4.1 Taylor-Green Vortex (TGV)

The Taylor-Green Vortex (TGV) flows are commonly used as models for validating numerical solvers. The solution provided by Taylor and Green [157] is a turbulence series in time that explains the small eddies created by vortex stretching, energy dissipation, diffusion, and generating decaying vortex in three-dimensional (3D) flow fields. It has a closed form of the incompressible Navier-Stokes equation solution

using Cartesian coordinates. Periodic 3D TGV flows have an initial analytical condition and are simple dynamical systems. The TGV provides an ideal benchmark for evaluating turbulent flow simulation approaches since it possesses various fundamental physical processes in turbulence. In this test case, periodic boundary conditions are applied to all boundary surfaces of a cubic fluid volume with an initial allocation of vorticity. Because of no external forces, small-scale turbulent motion dissipates all of the energy in the fluid over time. Eventually, the vortices stretch and interact with each other dissipating into turbulence.

The TGV problem is utilised to evaluate the current approach. In this scenario, the flow of TGV is simulated through LES with a Reynolds number of $Re = 1600$ [158]. To simulate the large-scale turbulence, the Navier-Stokes equations are solved directly, while the small-scale turbulence is modelled using the Smagorinsky-Lilly subgrid-scale model. The geometry has been discretized by a uniform mesh of resolution 64^3 as shown in Figure 4.1. All six sides of the cube of length $2\pi L$ are

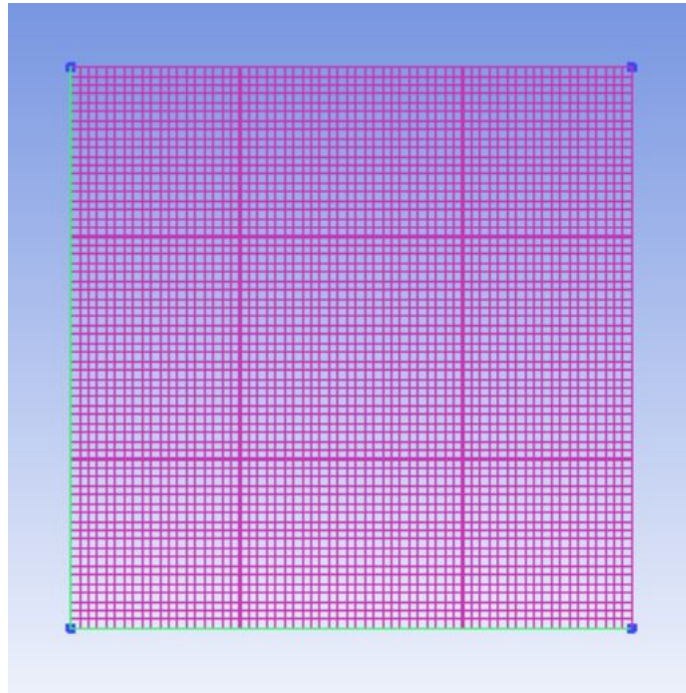


Figure 4.1: 64^2 element meshes.

set to periodic boundary conditions where L is the inverse of the wavenumber of the minimum frequency ($L = 1$). The smallest length scale can be captured on the mesh $\Delta x = (2\pi L/N) \approx 0.01$ where N is the number of mesh points ($N = 64$). Taking a time-step of $\Delta t^* = (CFL \Delta x/U) \approx 0.06$ with ($U = 1$), corresponds to a CFL value of 0.6. Here, the non-dimensional time t^* is used, which is expressed as $t^* = tU_0/L$. The initial conditions are set in such a way that an initial distribution of large, energy-containing vortices exists, which then decays through vortex stretching in three-dimensional space. The initial conditions are given by

$$u = U_0 \sin\left(\frac{x}{L}\right) \cos\left(\frac{y}{L}\right) \cos\left(\frac{z}{L}\right), \quad (4.1.1)$$

$$v = -U_0 \cos\left(\frac{x}{L}\right) \sin\left(\frac{y}{L}\right) \cos\left(\frac{z}{L}\right), \quad (4.1.2)$$

$$w = 0, \quad (4.1.3)$$

$$p = p_0 + \frac{\rho_0 U_0^2}{10} \left(\cos\left(\frac{2x}{L}\right) + \cos\left(\frac{2y}{L}\right) \right) \left(\cos\left(\frac{2z}{L}\right) + 2 \right). \quad (4.1.4)$$

Here, putting $U_0 = 1 \text{ m s}^{-1}$, $p_0 = 0 \text{ Pa}$ and $\rho_0 = 1 \text{ kg m}^{-3}$. Setting the kinematic viscosity to $\nu = 1/1600 \text{ m}^2 \text{ s}^{-1}$ then results in the desired Reynolds number of $Re=1600$. The left-hand side of Figure 4.2 displays the distribution of vortices at $t^* = 0$ using the magnitude of the velocity vector. The right-hand side shows the state of the velocity vector at $t^* = 20$, after vortex stretching has redistributed the containing energy in space, causing the energy at the smallest turbulent scales to dissipate into heat. From the initial conditions, it can be seen in Figure 4.2 that a range of vortices will be developed, a reliable mechanism to generate vortical flow patterns will be provided, from which streamlines on the symmetry planes of the x-, y- and z-direction can be extracted which are then used training data for the CV based framework.

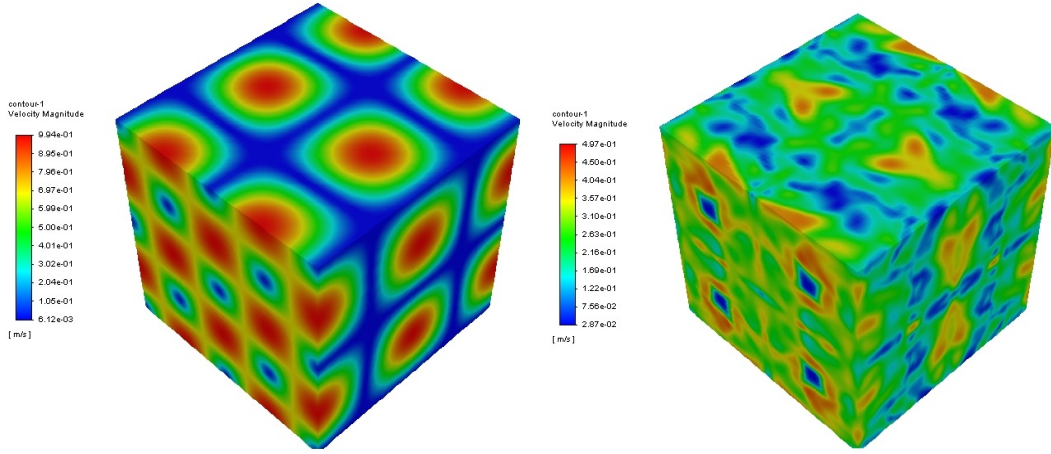


Figure 4.2: Contour velocity plot. The left-hand side at the beginning initial state and the right-hand side at the end time step.

To validate the current simulation setup, the Taylor-Green vortex problem has been run until $t^* = 20$ and compare the computed overall kinetic energy with that of Beck and Gassner [159] and DeBonis [160]. The temporal evolution of kinetic energy, based on the instantaneous velocity vector, is computed as

$$E_k = \frac{1}{\rho_0 V} \int_V \rho \frac{U \cdot U}{2} dV. \quad (4.1.5)$$

The temporal evolution of the kinetic energy dissipation rate is computed as

$$\varepsilon = -\frac{dE_k}{dt}, \quad (4.1.6)$$

These two quantities are plotted and compared against reference data in Figure 4.3 and Figure 4.4, respectively. The data here is compared against direct numerical simulation (DNS) results, as well as LES results are done on the same mesh size as the current simulation. From Figure 4.3, we can see that the results prematurely capture an increase in dissipation rate compared to the DNS results. However, this trend is explained by the coarse resolution of the 64^3 grid cells in the current

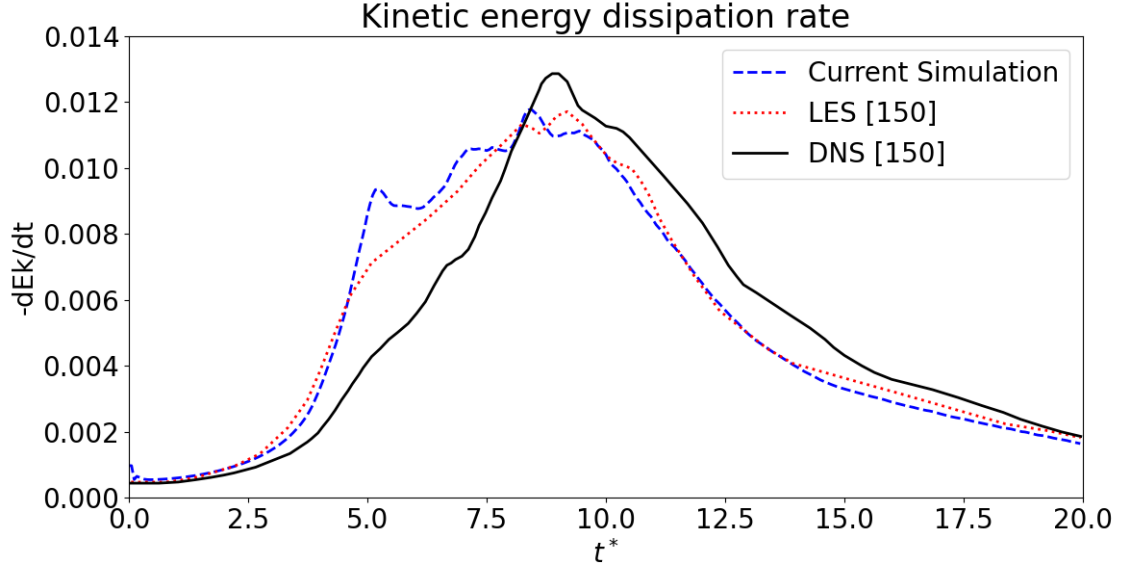


Figure 4.3: Kinetic energy dissipation rate. The reference Large Eddy Simulation (LES) and Direct Numerical Simulation (DNS) was taken from [159], the current simulation results shown in the dashed blue line were obtained using LES on a 64^3 grid at a Reynolds number of $Re=1600$.

simulation. Comparing current results against LES data obtained on a similar grid shows that data is matched much better. Looking at Figure 4.4, the kinetic energy

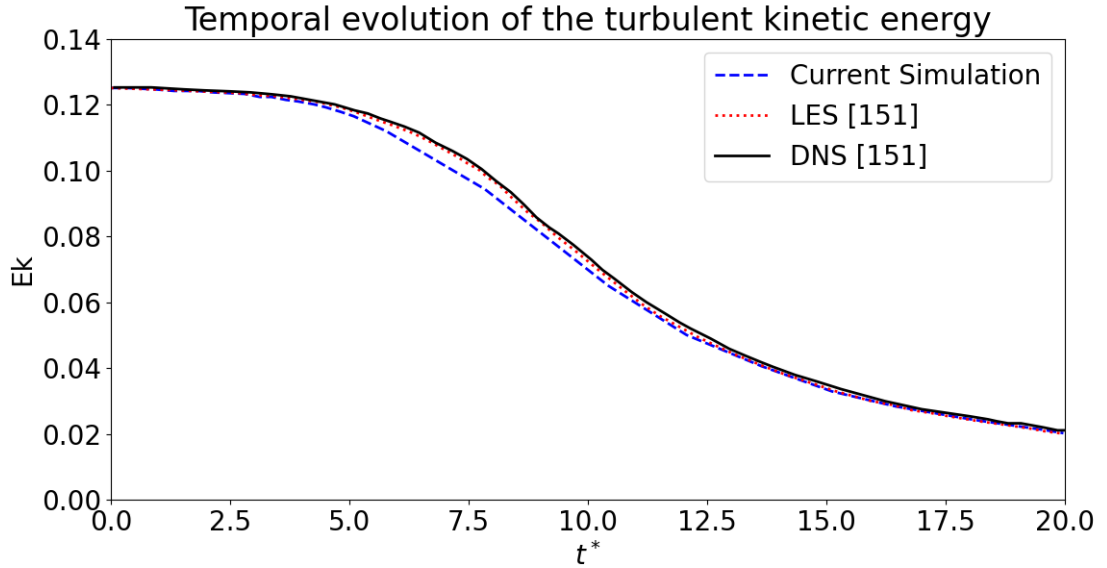


Figure 4.4: Temporal evolution of the turbulent kinetic energy. The reference Large Eddy Simulation (LES) and Direct Numerical Simulation (DNS) was taken from [160], the current simulation results shown in the dashed blue line were obtained using LES on a 64^3 grid at a Reynolds number of $Re=1600$.

temporal evolution is captured well, and there are only minute differences between the LES and DNS reference data. It should be stressed here that the primary concern is to generate vortical structures quickly while capturing the overall physical process of turbulence and its decay. It is shown by the validation that this is the case here, and it can be concluded that the streamline plots for the vortex core detection task can be generated by these simulations.

In order to advance the solution from $t^* \geq 0$ to $t^* \leq 20$, a total of 335-time steps are required. In each time step, the velocity vector on the symmetry planes in the x-, y-, and z-direction is stored. In this study, a total of 100 images are obtained from the symmetry plane norm to the x, y, and z-direction as training data. The 100 images are labelled with vortices using the line integral convolution method in Paraview, which was developed to visualise the vortices.

4.2 Application to turbulent boundary layer flows

The turbulent boundary layer (TBL) plays a critical role in many modern engineering applications, which motivates researchers to study it rigorously. Incompressible TBLs over a flat plate has been the subject of many theoretical and experimental investigations. In 1904, Ludwig Prandtl proposed the concept of boundary layers, whereby when fluid moves over solid surfaces, the fluid particles do not slip because of the effect of viscosity, also known as no-slip conditions. This no-slip or no-velocity-offset boundary condition for viscous fluids indicates that at a solid boundary (surface), the fluid has zero velocity relative to the boundary. Meaning the fluid velocity layer at the surface must be identical to the velocity of the surface. An important part of his study was the introduction of a boundary layer and its importance in stalling and drag. As the flow flows over the surface, a boundary layer forms between the solid surface and the free-stream flow. Upon reaching the transition point, the boundary layer transitions from a laminar to a turbulent state

depending on the value of the Reynolds number, as shown in Figure 4.5. This is because the flow tends to separate from the surface.

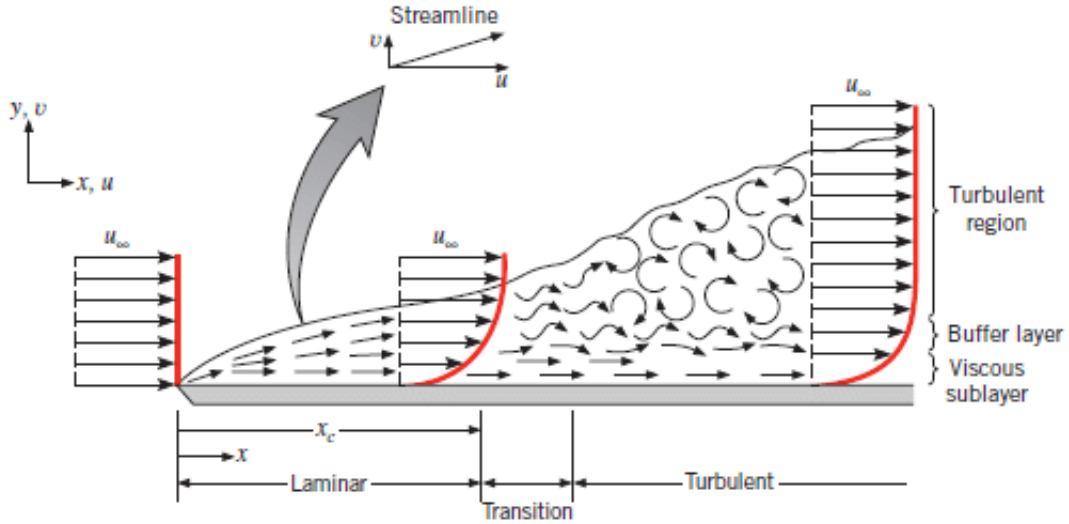


Figure 4.5: Velocity boundary layer development on a flat plate, [161]

The Reynolds number is a numerical value that expresses the relationship between the forces caused by inertia and viscosity in a fluid system. It can be used to predict whether a flow will be laminar or turbulent and is derived as follows:

$$Re_L = \frac{\rho UL}{\mu} = \frac{UL}{\nu} \quad (4.2.1)$$

Where ρ density of the fluid, U represents the mean flow velocity, L represents a characteristic linear dimension, μ represents dynamic viscosity, and ν is kinematic viscosity. Figure 4.5 illustrates that when the Reynolds number has a small value, the boundary layer remains laminar, and there is a uniform change in the streamwise velocity as one moves away from the wall. However, the flow becomes unstable as the Reynolds number grows. As the fluid flow moves away from the surface, the horizontal velocity component of the fluid (represented as u) progressively increases until it reaches the velocity of the free stream u_∞ . Figure 4.6 is shown the boundary layer's thickness δ developing, which is the distance between the wall and the point

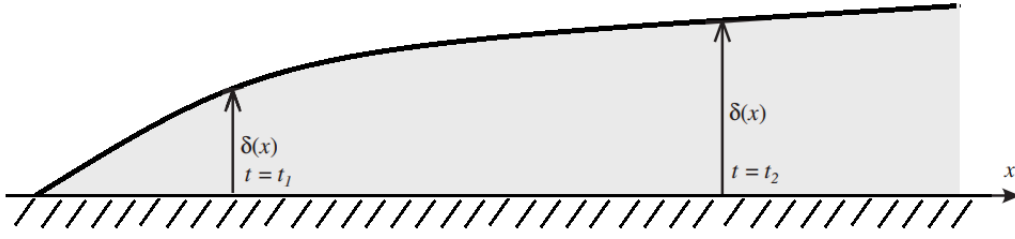


Figure 4.6: Developing the boundary layer, where δ represents the boundary layer thickness.

at which 99% of the velocity of free streams is presented. According to Blasius' solution of the flow governing equations over a flat plate, laminar boundary layers over the plate are [162]:

$$\delta \approx \frac{5}{\sqrt{Re_x}}x \quad (4.2.2)$$

Re_x refers to the Reynolds number, which is calculated based on the length of the plate. Whereas in turbulent flow, the thickness of the boundary layer can be determined using the following expression:

$$\delta \approx \frac{0.37}{Re_x^{1/5}}x \quad (4.2.3)$$

A case of TBL that develops over a flat plate, which has features such as streaks and hairpin vortices, is studied. In this study, the LES data of Szöke et al. [163] is used, where the turbulence within a channel with streamwise rods was simulated to investigate their role in attenuating pressure fluctuations near the wall. As shown in Figure 4.7, the streamlines are taken from the numerical data in [163] using ParaView, This is looking straight onto the rods, where the bottom part is assigned a wall boundary condition (and hence this is where fundamentally different vortical structures are expected). A periodic boundary condition is assigned to the left and right sides, while a slip wall is assigned to the bottom.

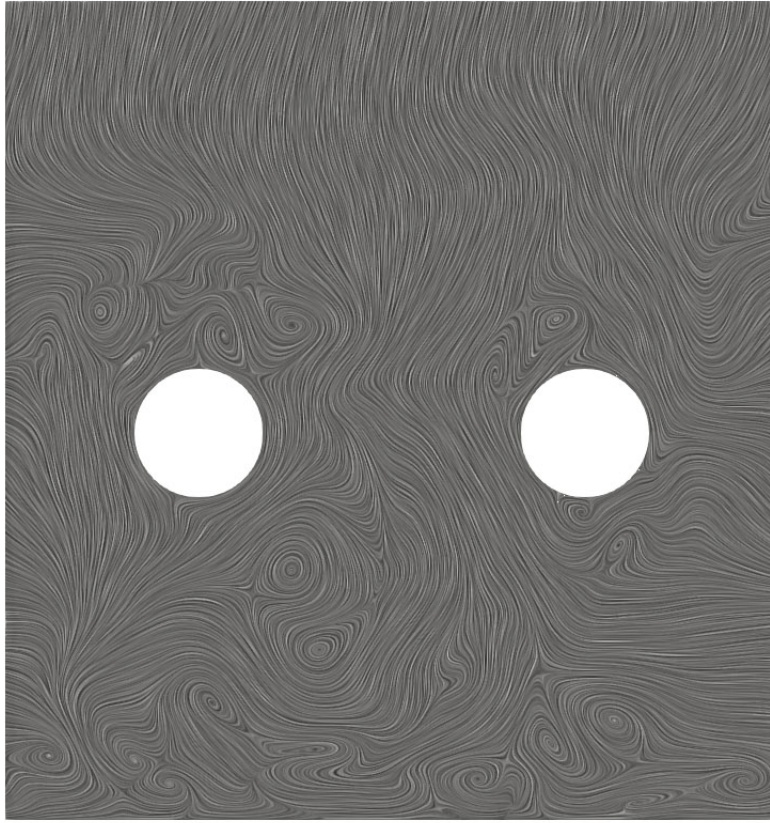


Figure 4.7: Developing the boundary layer, where δ represents the boundary layer thickness.

4.3 Application to rotating wings

A rotating wing's performance and stability can be significantly impacted by vortices, which are of particular interest when it comes to vortices. An essential application of vortices in rotating wings is the generation of lift, which occurs when rotating wing motion creates a pressure differential that generates lift [164]. The lift is generated by the movement of air over the wing's surface, resulting in the development of a vortex trail behind the wing. The strength of these vortices is affected by the velocity of the air over the wing's surface and the shape and size of the wing. A rotating wing's lift is maximised when the distribution and strength of vortices along its edge are controlled. Therefore, engineers use various techniques to control the strength and distribution of vortices along rotating wings. The most effective technique is to predict the vortices along these structures, which allows them to

optimise the design of the wing to achieve desired performance characteristics, such as lift, stability, and efficiency.

In order to test and support the validation of the model, real-world data from Hovering Rotor Solutions (Caradonna and Tung) [165] is utilised. NASA Ames Research Center leads the Caradonna and Tung experiment in its steady wind tunnel, which is equipped with the two-bladed rotor. As a basis for this study, the published data of Ricci *et al.* [166] has been used, who provided numerical simulations for this test case. Figure 4.8 shows the helicopter blade is located in the middle of a cylindrical domain with a radius of 40m and a length of 80m. The blade has a six-aspect ratio



Figure 4.8: Figure depicts the quarter domain of the Application to rotating wings (Caradonna and Tung rotor) case.

and a radius of 1.143 meters. It is derived from the NACA 0012 aerofoil that has an untwisted and untapered surface. The simulations were conducted at a blade tip Mach number of $M_{tip} = 0.89$, a pitch angle of $\theta_c = 8^\circ$, and an angular velocity of 265.9 rad/s. In section 5.2.3 and section 5.3.3, the results of the hybrid approach

are presented for identifying vortex regions and vortex cores using the CV and ANN machine learning algorithms, respectively, within rotating wings, which are critical to the performance and stability of rotating wings.

Chapter 5

Results and Discussion

In this chapter, the results of applying the current model are presented to identify vortex cores in three different test cases (Taylor-Green Vortex, Application to turbulent boundary layer flows, and Application to rotating wings). First, local traditional vortex detection approaches (Q criterion, Delta criterion, and Swirling Strength criterion) are used to detect vortices and to show the limitation of these methods. Then, a hybrid CV and ANN approach is used to identify vortex cores, where CV is used to determine the regions of vortices for each test case. While ANI has been implemented to identify vortex cores within the regions.

5.1 Local vortex detection methods

5.1.1 Identifying vortices of Taylor-Green Vortex using Local detection methods

Two images are selected from the symmetry plane of TGV, which are not part of the training data set. First, these two test images will be used to detect vortices using the Vortex fitting toolbox of Lindner et al. [131] and show results for that in this section. Then, the same test images are applied to the hybrid CV and ANN frame-

work and show how the number of images used for training influences the accuracy. The two images are shown in Figure 5.1, where test image 1, on the left-hand side, contains 16 vortices and test image 2, on the right-hand side, contains 24 vortices. The test images are observed to contain mostly symmetrical vortex structures along the main axes.

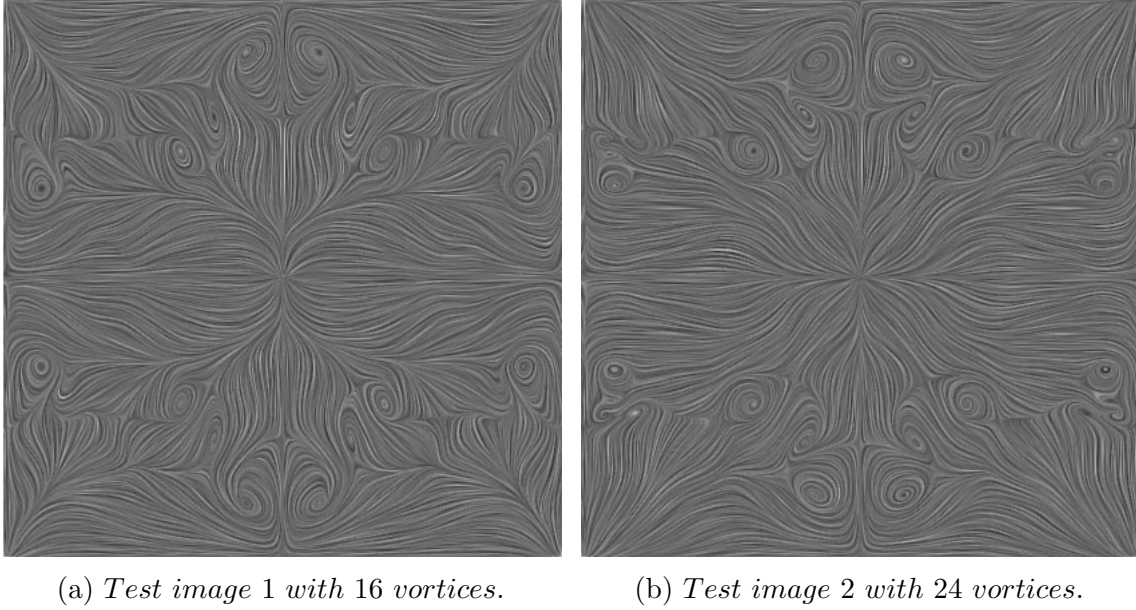


Figure 5.1: Test images showing vortical structures obtained with the line integral convolution-based streamline algorithm.

As shown in Figure 5.2 and Figure 5.3, vortices have been detected in test image one and test image two, respectively, by using the Vortex fitting toolbox [131]. The numerical discretisation schemes used to approximate the derivatives within the velocity gradient tensor have been varied, employing the Q, delta, and swirling strength criterion.

Second-order, fourth-order, and least-square gradient reconstruction schemes are used. Moreover, the threshold value is modified to match the desired number of vortices for both cases, 16 vortices for test image 1 (Figure 5.2) and 24 vortices for test image 2 (Figure 5.3). Delta and swirling strength criteria are the only ones that allow fine-tuning of thresholds, whereas Q does not.

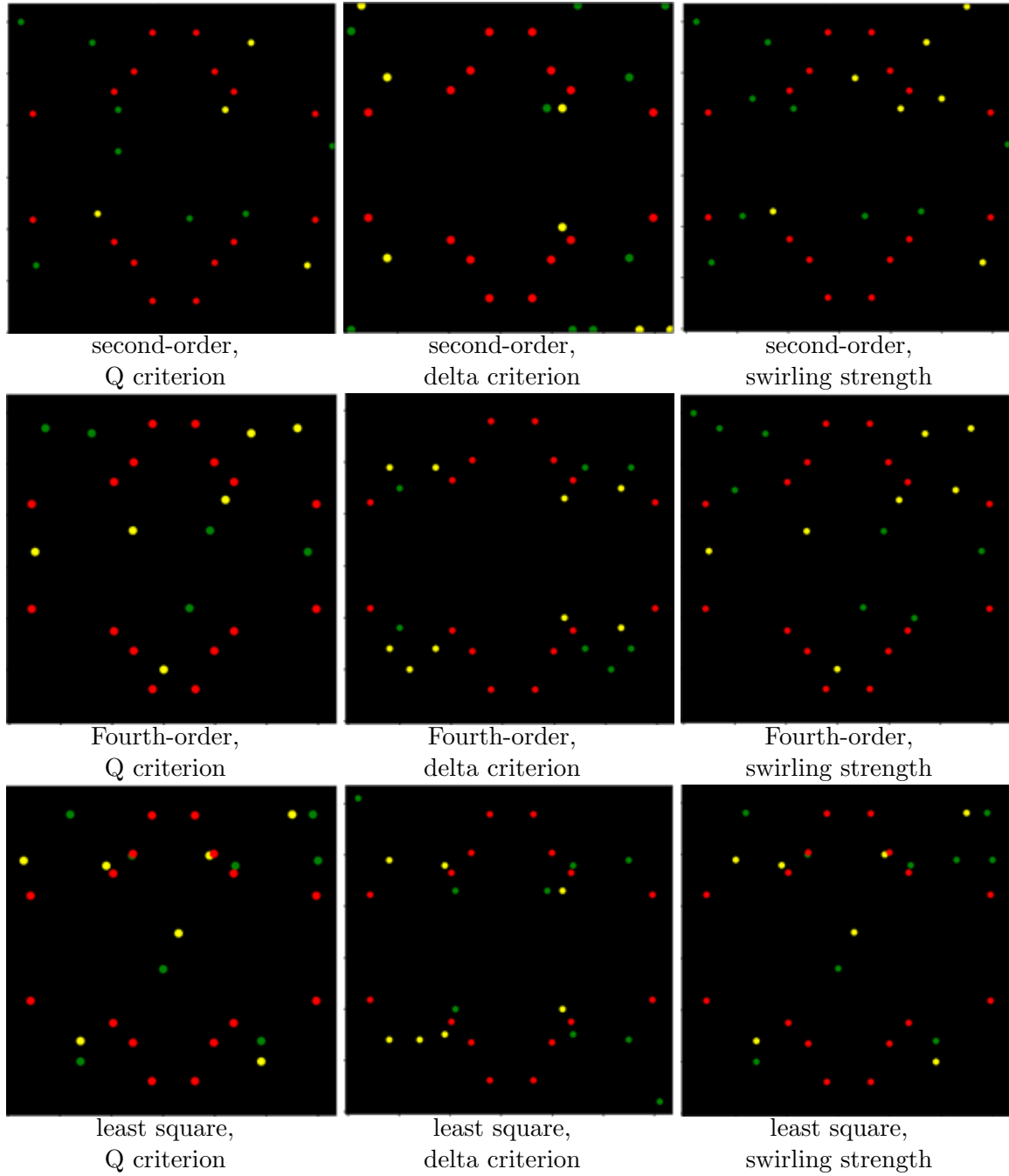


Figure 5.2: Detected vortices (shown on yellow and green) using the Q, delta, and Swirling Strength criterion with second-order, fourth-order and least square gradient reconstruction schemes for the first test image (16 vortices). The actual vortex core regions are shown by the red circles.

The yellow dots in the images indicate clockwise rotating vortices, the green dots indicate counter-clockwise rotating vortices, and the red dots show the actual location of vortex cores, extracted manually from Figure 5.1.

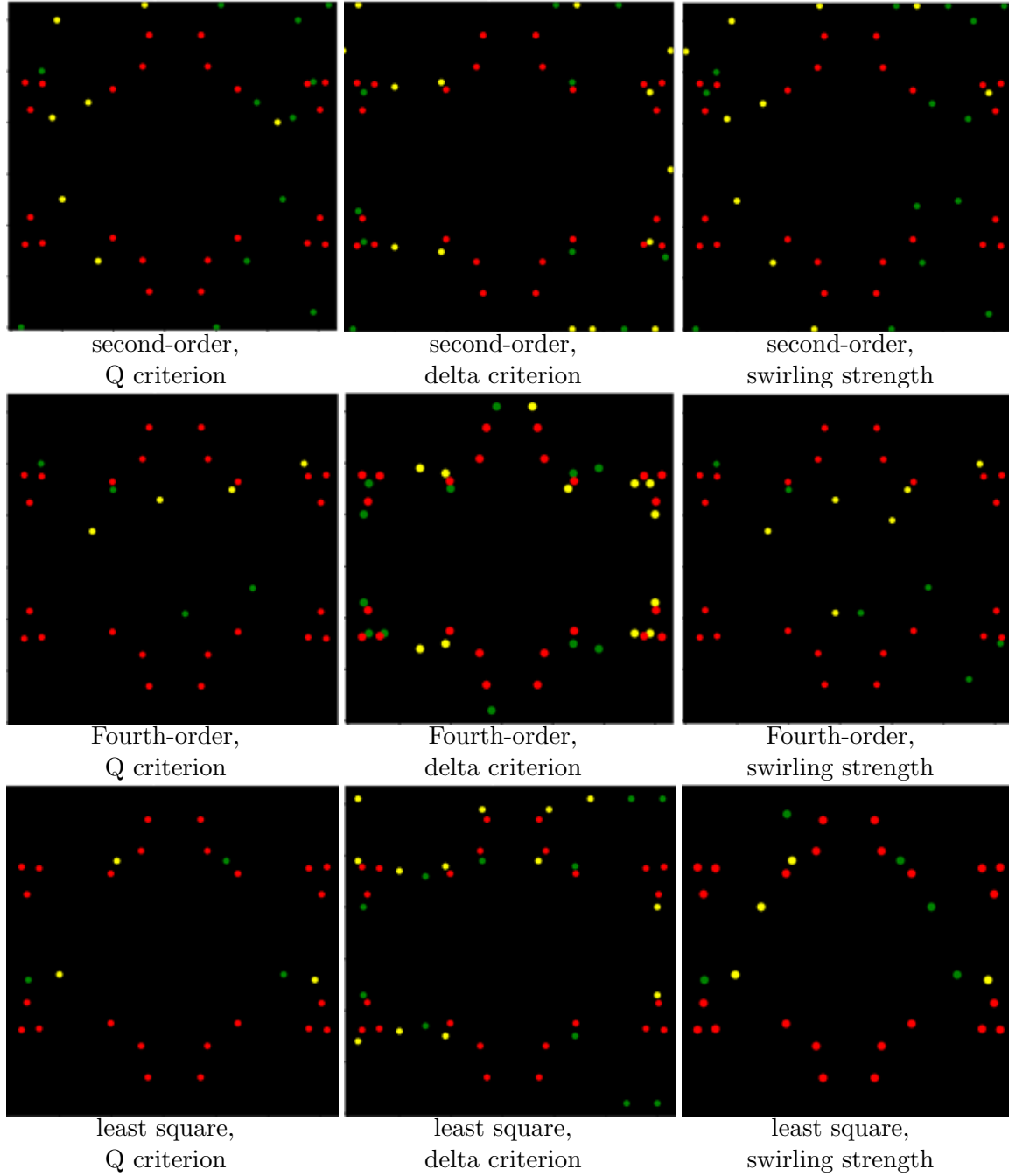


Figure 5.3: Detected vortices (shown on yellow and green) using the Q, delta, and swirling strength criterion with second-order, fourth-order and least square gradient reconstruction schemes for the second test image (24 vortices). The actual vortex core regions are shown by the red circles.

Both Figures show performance results are similar behaviour between the three different detection algorithms. There is little coherence between the different methods. In addition, to a certain extent, there is also a lack of coherence between the differ-

ent gradient reconstruction schemes for the same approach. It is interesting to note that even though the vortex local detection algorithms are given a nearly symmetrical flow field, this symmetry does not reflect in the predicted vortices. There are some detection algorithms that produce more symmetrical vortex core predictions, but in general, a non-symmetrical prediction pattern emerges. This shows that the different methods have sensitivity to the threshold parameter ϵ (for the delta and swirling strength criterion). When the threshold parameter value ϵ is modified, some of the mirrored vortices are removed due to a slight difference in vortex strength. Additionally, all of these local detection methods provide, at best, an approximation of detecting the actual vortex core regions. However, qualitative point of view, it appears that the delta criterion can accurately predict regions with high vortex flows, despite not being able to identify the exact locations of vortex cores. In the second image of the test, this can be seen especially clearly. The Q and swirling strength criterion methods accommodate rotational flow at the centre; there may be vortical flow at the centre, but within the plane in which the flow is observed, the centre part of the streamlines cannot be classified as a vortex core. Despite getting a reasonable initial prediction of areas where vortices will be located, it is difficult to determine their exact location without manually detecting where vortices will be located. Furthermore, it is challenging to judge if the reported results are physical or contain false positives and negatives.

The tables 5.1–5.3 summarise how the different local detection approaches compare in terms of predicting false positives and negatives for the three different gradient reconstruction schemes. It is considered that vortices have been correctly predicted if they are near actual vortex cores. These three tables provide in numbers what has been discussed above but further indicate that for all detection algorithms and reconstruction schemes, a large number of false positives and negatives exist. Aside from the Q criterion, we can see that arriving at these values needs fine-tuning of

Table 5.1: Number of predicted vortices for the Q, delta and swirling strength criterion using a second-order gradient reconstruction scheme, as well as their false positives and negatives and the threshold value ϵ required to match the number of predicted vortices to the number of expected vortices (EV)

Test image	Threshold value			Second-order						False negatives			
				Detected vortices			False positives						EV
	Q	delta	swirl	Q	delta	swirl	Q	delta	swirl	Q	delta	swirl	
1	n/a	1.2e-09	0.9	12	16	16	4	8	4	8	7	4	16
2	n/a	4.3e-08	1.37	19	24	24	4	3	3	9	4	4	24

Table 5.2: Number of predicted vortices for the Q, delta and swirling strength criterion using a fourth-order gradient reconstruction scheme, as well as their false positives and negatives and the threshold value ϵ required to match the number of predicted vortices to the number of expected vortices (EV)

Test image	Threshold value			Fourth-order						False negatives			
				Detected vortices			False positives						EV
	Q	delta	swirl	Q	delta	swirl	Q	delta	swirl	Q	delta	swirl	
1	n/a	2.0e-10	0.0	11	16	15	2	2	3	7	2	4	16
2	n/a	2.4e-08	0.0	8	24	12	0	1	1	16	1	13	24

Table 5.3: Number of predicted vortices for the Q, delta and swirling strength criterion using a least-square gradient reconstruction scheme, as well as their false positives and negatives and the threshold value ϵ required to match the number of predicted vortices to the number of expected vortices (EV)

Test image	Threshold value			Least square						False negatives			
				Detected vortices			False positives						EV
	Q	delta	swirl	Q	delta	swirl	Q	delta	swirl	Q	delta	swirl	
1	n/a	1.4e-08	3.2	15	16	16	3	3	6	4	1	4	16
2	n/a	1.9e-07	0.0	6	24	9	0	4	0	18	2	15	24

the threshold parameter value, which does not follow any identifiable pattern.

The root mean square error of all predicted vortices is calculated to the closest actual vortex present in the domain in order to quantify the spread of vortex cores

detected by these local detection methods, Using the following equations:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} (\mathbf{x}_i - \hat{\mathbf{x}}_i)^2}, \quad (5.1.1)$$

Where n is the total number of predicted vortices, $\mathbf{x}_i$ is the predicted vortex core, and $\hat{\mathbf{x}}_i$ is the vortex that is present in the domain and closest to $\mathbf{x}_i$. Here, $\mathbf{x}_i$ denotes the yellow and green dots Figure 5.2 and Figure 5.3, while $\hat{\mathbf{x}}_i$ is represented by the red dots. The results are shown in Table 5.4. Interestingly, for test images 1 and 2, as shown in this table, the delta criterion using a second- or fourth-order gradient reconstruction scheme yields the smallest and highest errors, respectively. This can be demonstrated by calculating the variance of all the results for each method. Delta, Q, and swirling strength criterion have variances of $5.9e - 3$, $7.9e - 4$, and $6.4e - 4$, respectively. It is true that the delta criterion provides some reasonably

Table 5.4: Root mean square errors for the predicted vortex cores to the actual vortex core regions.

Test image	Root Mean Square Error values								
	Delta Criterion			Q Criterion			swirling strength Criterion		
	Second Order	Fourth Order	Least Square	Second Order	Fourth Order	Least Square	Second Order	Fourth Order	Least Square
1	0.275	0.1112	0.1347	0.1457	0.1603	0.158	0.1503	0.1603	0.1589
2	0.1419	0.0535	0.1115	0.1384	0.1046	0.0930	0.1389	0.1092	0.1022

accurate prediction of the vortex core regions for test image 2, but the predictions for the first test image 1 seem to be rather scattered. Although the swirling strength criterion does not produce overall the lowest errors, it provides the most consistent method with the lowest variance compared with other approaches. When it comes to overall errors, the Q criterion is similar to the swirling strength and requires no fine-tuning of the threshold parameter value ϵ to achieve this. Thus, the Q criterion may be viewed as a suitable compromise, which may explain its popularity as a visualisation tool in computational fluid dynamics (CFD). In spite of this, it continues to yield false positives and negatives.

For each criterion, Table 5.5 shows the maximum distance between the actual vortex cores and the predicted location based on second-order, fourth-order, and least-square gradient reconstruction schemes. In order to calculate the maximum difference between the predicted vortices and the actual vortices in the domain, the following formula is used:

$$MaxDiff = Max \left(\sqrt{\Delta x_i^2 + \Delta y_i^2} \right) \quad i = 1, 2, \dots, n, \quad (5.1.2)$$

Where Δx_i represents the difference between the predicted and actual of a vortex core in the x direction, while Δy_i represents the difference between the predicted and actual of a vortex core in the y direction. As shown in Table 5.5, for test image one, the delta criterion with the second order gradient reconstruction scheme has the highest maximum distance value, as well as the highest RMSE value in the table 5.4. In test image 2, it appears that the second-order gradient reconstruction scheme of Q, delta, and swirling strength criterion gives the highest maximum distance value. Table 5.4 also indicates that they have the highest RMSE value. There is little spread between the data, indicating that all methods produce a similar difference in terms of maximum difference to the actual vortex cores. Section 5.2.1, shows that implementing CV can remove the need to specify this free parameter while reducing the false positives and negatives entirely.

Table 5.5: Maximum Difference between the predicted vortex cores and the actual vortex core regions.

Test image	Maximum Difference values								
	Delta Criterion			Q Criterion			swirling strength Criterion		
	Second Order	Fourth Order	Least Square	Second Order	Fourth Order	Least Square	Second Order	Fourth Order	Least Square
1	1.129	0.919	1.089	1.080	0.989	1.025	1.080	1.080	1.025
2	1.190	1.016	1.161	1.190	1.0126	0.958	1.190	1.045	0.958

5.1.2 Identifying vortices of Application to turbulent boundary layer flows using Local detection methods

The results for this section are shown for a plane image of turbulent boundary layer flows used as the test image to detect vortices using the Vortex fitting toolbox of Lindner et al. [131]. Using the same test image for training demonstrates how this image affects accuracy in the hybrid CV and ANN framework. The test image is shown in Figure 5.4, which contains 16 vortices. The Vortex fitting toolbox has

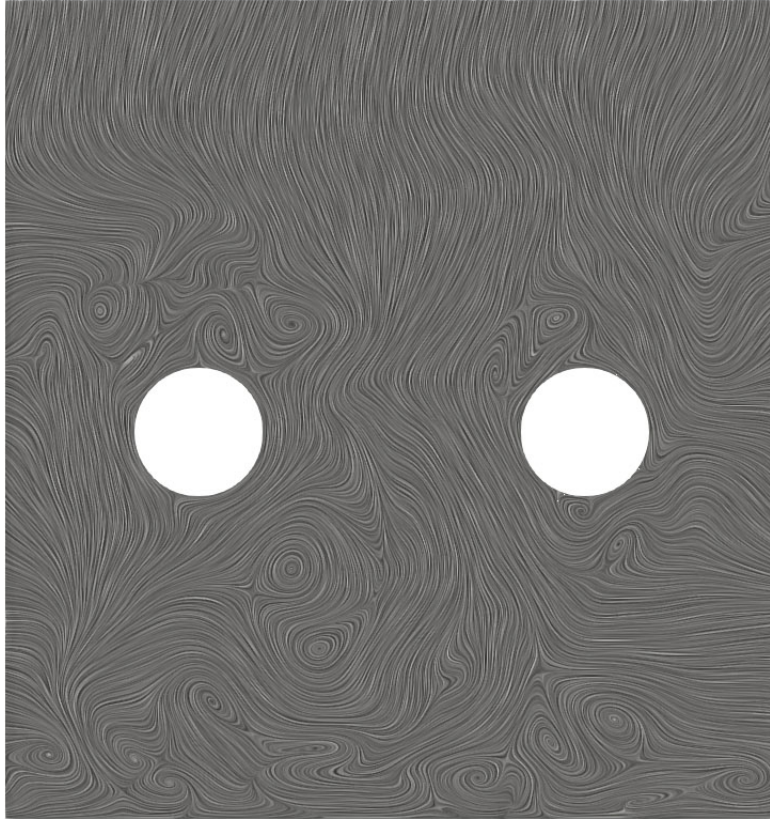


Figure 5.4: Test image representing vortical structures obtained with the line integral convolution-based streamline algorithm.

been used to detect vortices in the test image as shown in Figure 5.5, and numerical discretisation schemes have been used to approximate the derivatives of the velocity gradient tensor employing Q, delta, and swirling strength criteria.

Similar to the TGV test images in the previous section, second-order, fourth-order,

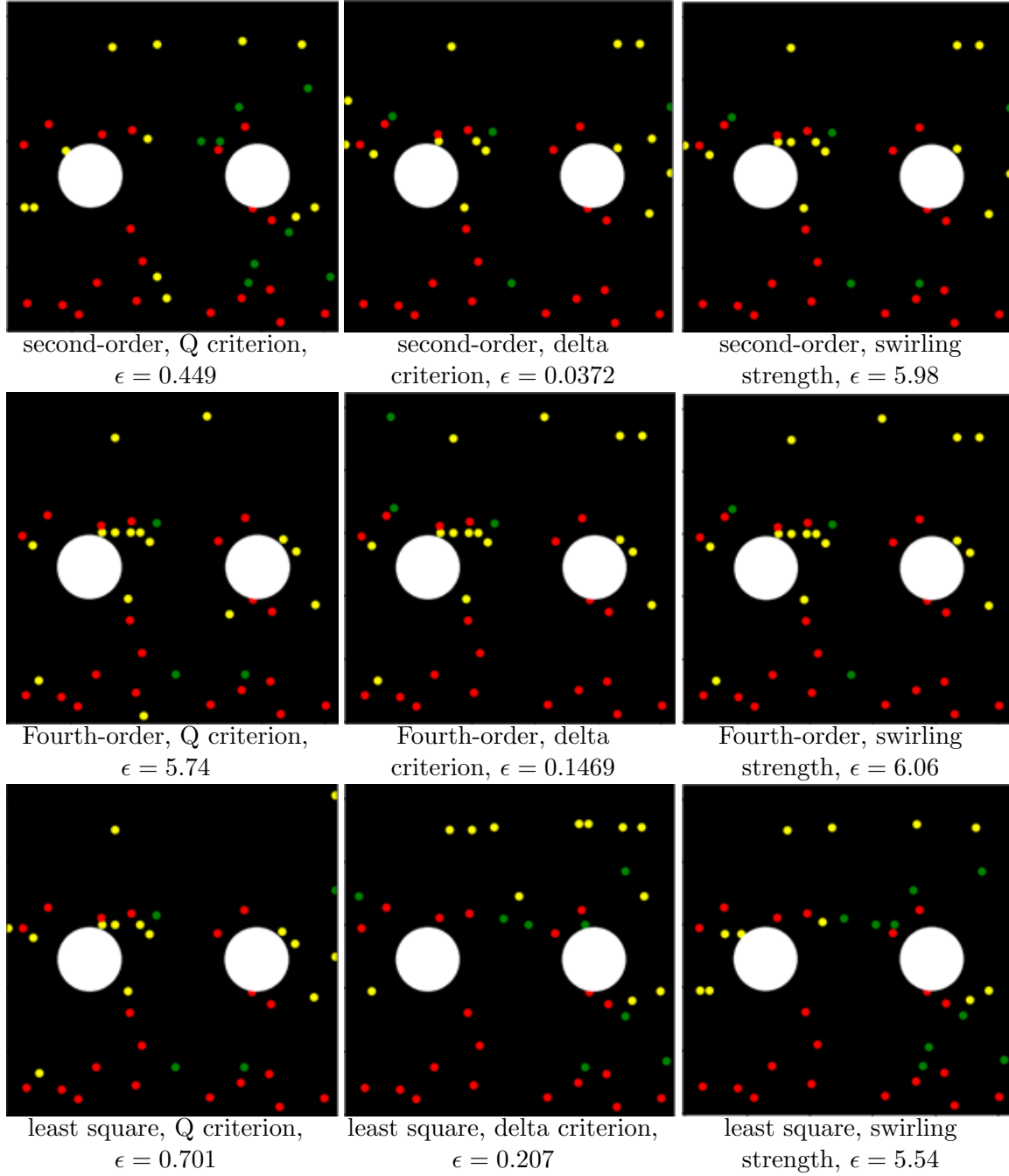


Figure 5.5: Detected vortex cores using the Q, delta and swirling strength criterion with different gradient reconstruction schemes for the turbulent boundary layer case taken from Szöke et al. [163]

and least-square gradient reconstruction schemes are utilised as shown in Figure 5.5, and the threshold value is tuned to match the desired number of vortices for this test case, 16 vortices. According to the previous observations, we can see that there are scattered false positives and negatives in the results, so the threshold parameter has

to be fine-tuned to reduce the number of false positives and negatives. While there is little coherence among the different methods, the CV based approach, as shown in the section 5.2.2, provides better predictions despite a lack of training. With these results of TGV and turbulent boundary layer flows, without needing to implement the third scenario, reveal that relying on traditional tools to detect vortices can lead to inaccurate outcomes. Despite adjusting threshold values, false positives and negatives still persist. Therefore, a CV framework is developed to identify regions of vortex cores to eliminate false positives and negatives. The framework efficiently identifies regions of vortex cores, eliminating false positives and negatives. This approach enhances the accuracy of vortex detection, providing more reliable results. By using CV techniques, the detection of vortices becomes more efficient and accurate, leading to a better understanding of fluid dynamics and its applications in various fields.

5.2 Identifying vortex regions through computer vision (CV)

5.2.1 Identifying vortex regions of Taylor-Green Vortex

The CV model predicts vortex regions (i.e. bounding boxes of vortices) rather than vortex cores themselves, which is important since the test case is unsteady. In unsteady flows, the vortex core may not be inferred from streamlines such as LIC surface streamline plots, but rather path lines must be studied. Even though the centre of rotation may differ, the vortex region remains approximately the same, so avoid limiting to studying only steady-state flows by focusing on the vortex bounding box of the vortex. As a result of this approach, vortices can be created easily and in a wide range of sizes and shapes when using simple geometries such as the Taylor-Green vortex problem. To demonstrate how CV can be used to decrease false

positives and negatives, the YOLO-based CNN model is trained using a sequence of 12 steps, each step containing an increasing number of images used for labelling vortices, which are then used for training. The first sequence possesses a single image for training which is then increased to 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100 images. The number of vortices that are manually labelled in each sequence is 18, 106, 202, 394, 562, 722, 926, 1086, 1246, 1326, 1486 and 1646, respectively. The figures 5.6 and figures 5.7 illustrate, using both test images, how training evolves with and without overlapping bounding box labelling techniques, respectively, for sequence 1 (1 image), sequence 6 (40 images), and sequence 12 (100 images). the results were found that were not affected by the labelling strategy method used for identifying vortices, which is shown through a red bounding box. Therefore, in the

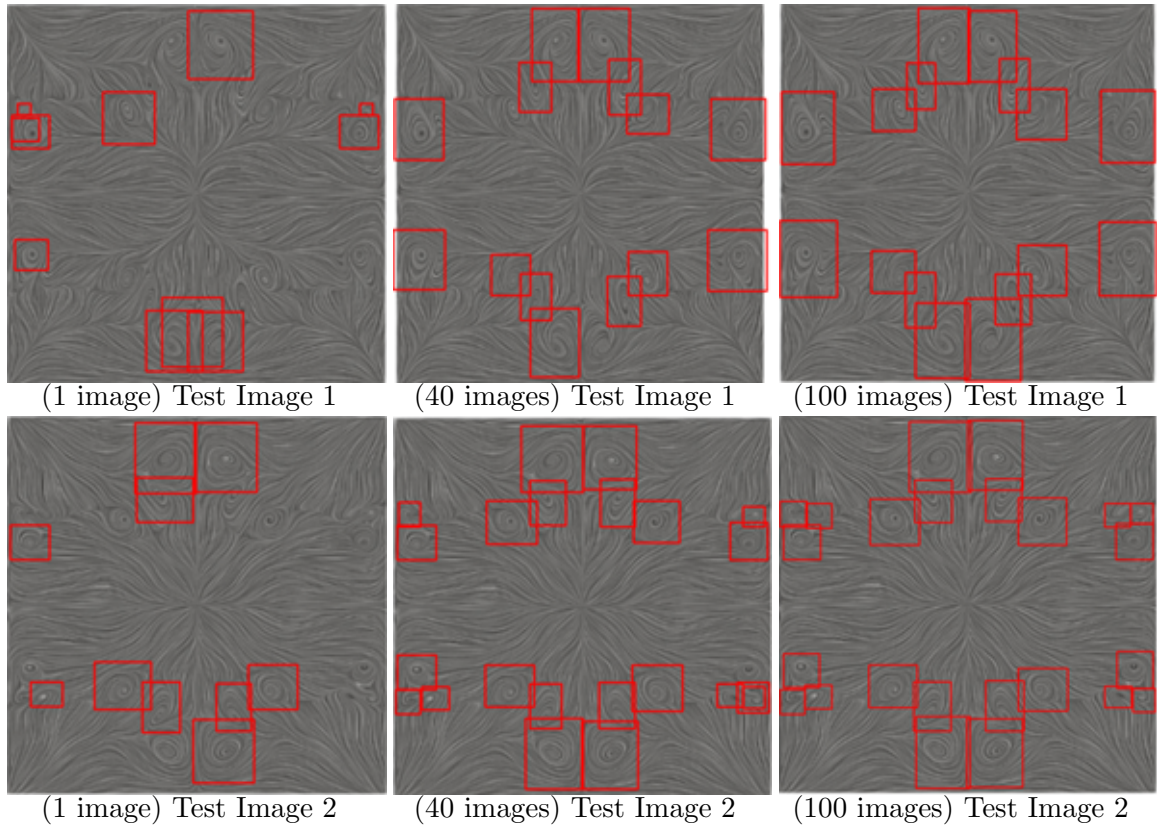


Figure 5.6: The output result after training data set of (1 image), (40 images) and (100 images) for test images 1 and 2. The red box is the predicted bounding box based on bounding boxes that are allowed to overlap during labelling; see Section 3.4.

remainder of this section, the overlapping bounding boxes presented in Figure 5.6

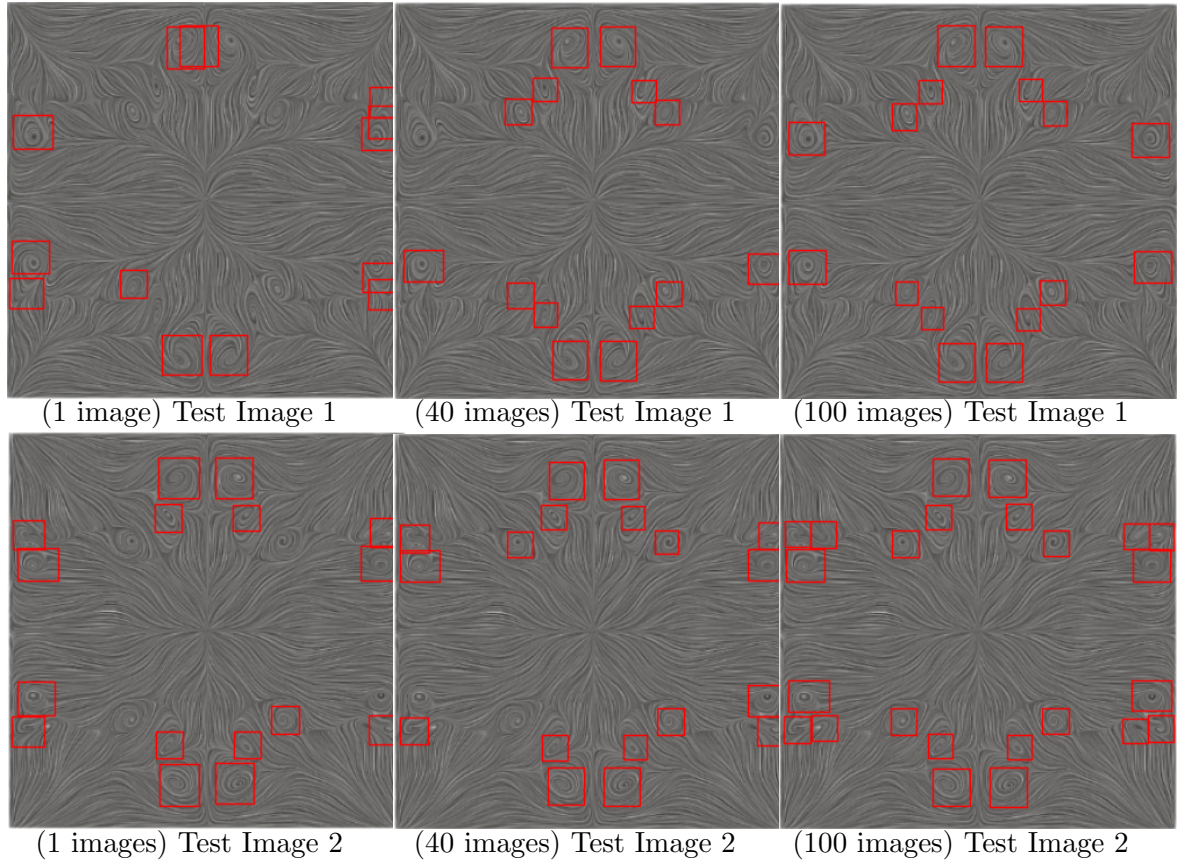


Figure 5.7: The output result after training data set of (1 image), (40 images) and (100 images). The red box is the predicted bounding box based on bounding boxes that are not allowed to overlap during labelling, see Section 3.4.

will be discussed since the labelling method does not influence the results, and the bounding box labelling captures the entire vortex region. Taking a look at sequence 1 for the first test image (top left) in Figure 5.6, 11 vortices are predicted, 4 of which are false positives and should be removed, and nine false negatives, i.e. vortices that are present but not detected. Shifting to the right, i.e. increasing the number of images used in the sequence, we can see that more vortices are detected increasingly for sequences 6 and 12, and the number of false positives and negatives are decreased completely with a sufficiently large training set. For the second test image, similar observations can be made. Table 5.6 and Table 5.7 provide tabulated data regarding the decrease in false negatives and positives. As shown in Table 5.6, for test image 1, it takes 60 images for training to reliably detect all physical vortices without detecting any spurious vortices. Table 5.7 shows that it takes the full training set

Table 5.6: Number of detected and expected vortices for the first test image, as well as false positives and negatives with an increasing number of images used for training. The computational time taken for training is provided along with the mean Average Precision (mAP)

Number of images	Number of vortices labelled	Expected vortices (EV)	Vortices detected	False positives	False negatives	Time of training	mAP
1	18	16	11	4	9	01:18:45	0.96
5	106	16	11	3	8	02:34:42	0.91
10	202	16	12	2	6	04:07:39	0.93
20	394	16	13	1	4	08:18:26	0.91
30	562	16	14	1	3	08:50:05	0.90
40	722	16	14	0	2	11:16:21	0.91
50	926	16	15	1	2	11:41:32	0.91
60	1086	16	16	0	0	12:05:55	0.90
70	1246	16	16	0	0	13:13:54	0.91
80	1326	16	16	0	0	13:50:58	0.92
90	1486	16	16	0	0	14:20:29	0.91
100	1646	16	16	0	0	22:00:38	0.96

Table 5.7: Number of detected and expected vortices for the second test image, as well as false positives and negatives with an increasing number of images used for training. The computational time taken for training is provided along with the mean Average Precision (mAP)

Number of images	Number of vortices labelled	Expected vortices (EV)	Vortices detected	False positives	False negatives	Time of training	mAP
1	18	24	12	4	16	01:18:45	0.96
5	106	24	14	3	13	02:34:42	0.91
10	202	24	17	3	10	04:07:39	0.93
20	394	24	17	2	9	08:18:26	0.91
30	562	24	18	1	7	08:50:05	0.90
40	722	24	22	1	3	11:16:21	0.91
50	926	24	22	1	3	11:41:32	0.91
60	1086	24	21	0	3	12:05:55	0.90
70	1246	24	22	0	2	13:13:54	0.91
80	1326	24	23	1	2	13:50:58	0.92
90	1486	24	24	2	2	14:20:29	0.91
100	1646	24	24	0	0	22:00:38	0.96

of 100 images to reduce all false positives and detect all physically present vortices accurately for test image 2. We also see this trend in Figure 5.8, where the false negatives are plotted against the number of training images used. It appears that the decrease is monotonous. For both tables, training times for CNNs trained on Nvidia

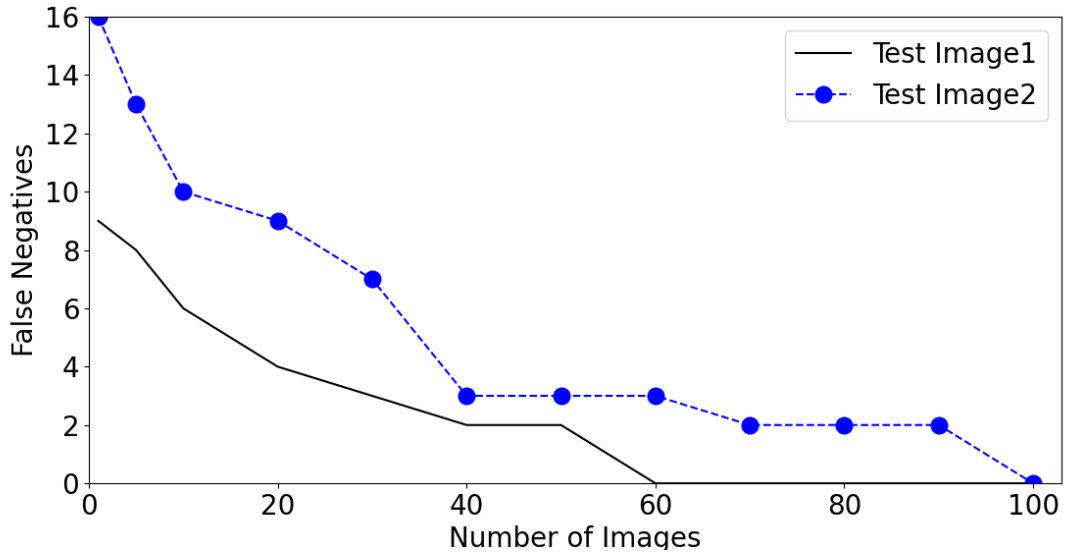


Figure 5.8: (The black solid and blue dish) false negatives curves for test image one and test image two respectively in the number of images.

V100 GPUs are provided. This step is performed only once since both test images rely on the same 100 images dataset are used for training. Therefore, all training times are the same in both tables and are provided here only for completeness. Additionally, Table 5.6 and 5.7 show the mean Average Precision (mAP) for each sequence, and the model is trained until 0.9 is reached except for the last sequence that is increased to 0.96 to capture all vortices.

Figure 5.9 illustrates that the mAP value initially increases at a steady rate after only a few epochs are required to pick up any vortices. The mAP is calculated by calculating the Precision (measures the accuracy of the predictions) and Recall (measures the number of true positives).

True positives are counted as those identified vortices (red bounding boxes) that share an intersection over union (IoU) with the ground truth bounding box (the bounding box that is manually created to label vortices), which is above an IoU threshold value of 0.5 in this case. The definitions of false positives and negatives are the same as in Table 5.6 and Table 5.7. Precision and Recall are obtained using IoU values for a given threshold value, and then the average area under the precision

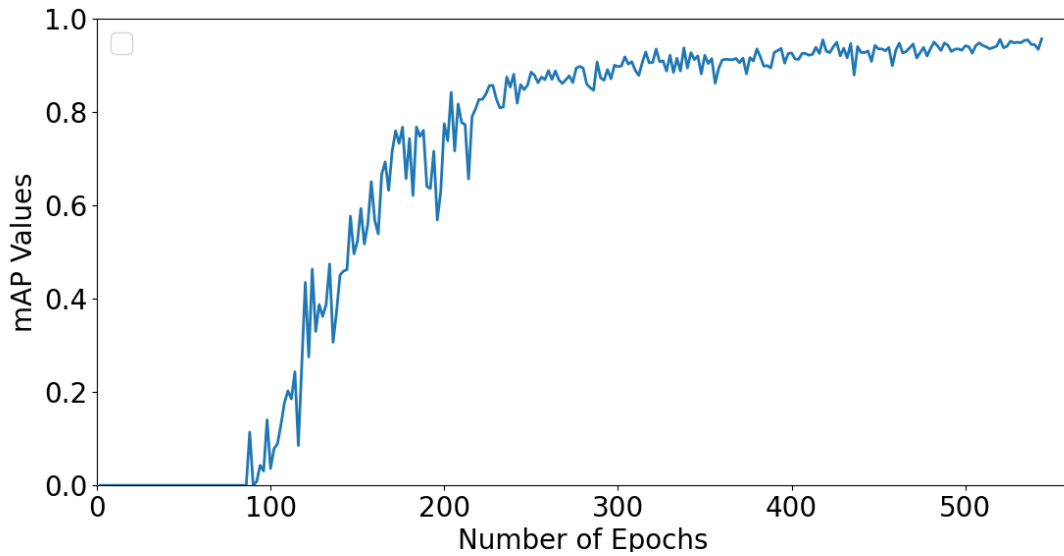


Figure 5.9: Mean Average precision curve in the time of training epochs

and recall curve is calculated. As part of the training process, the error between the target value and the predicted value is calculated using the loss function. This is calculated using the concept of error back-propagation in a neural network. In addition, the weight of each layer in the network is altered to achieve training of the model. As described in [13, 167], the loss function is calculated by adding up the

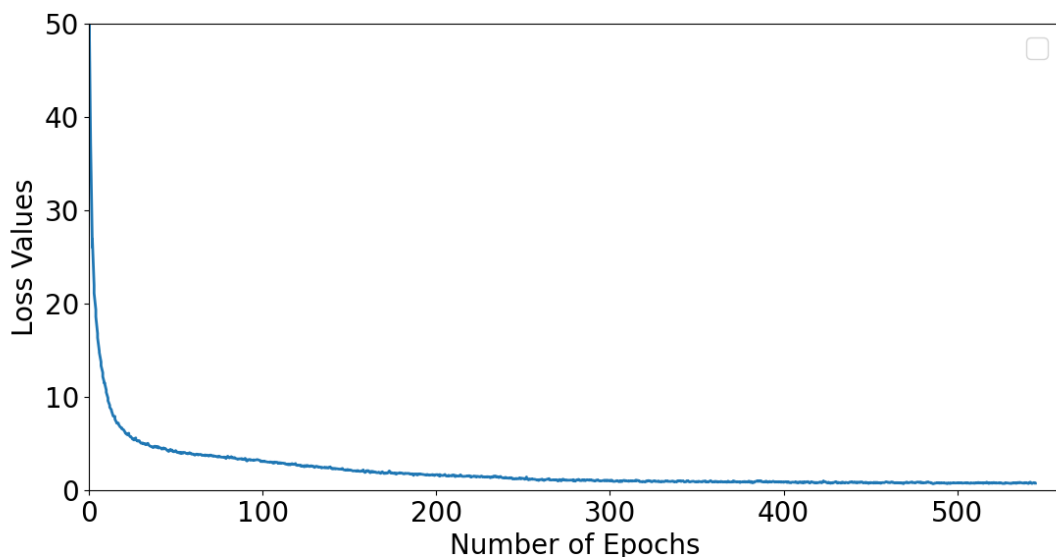


Figure 5.10: loss values curve in the time of training epochs.

localisation loss (error between the predicted bounding box and the truth bounding box), the classification loss, and the confidence. We can see in Figure 5.10, the CNN algorithm is well-fit after about 300 epochs with a monotonic decreasing loss function.

5.2.2 Identifying vortex regions of Application to turbulent boundary layer flows

In Section 5.2.1, homogeneously decaying turbulence in a box has been focused on. Nevertheless, real fluid-dynamics applications can occur in a wide range of complex flow structures, including hair-pin vortices, Gortler-Taylor vortices, Dean vortices, jets, and streaks, (to name but a few). Thus, the purpose of this section is to examine how the developed framework performs when dealing with such situations. This study case investigates the development of a turbulent boundary layer over a flat plate with features, for instance, hairpin vortices and streaks. The LES data of Szöke et al. [163] is used, which simulated the turbulence within a channel with streamwise rods near the wall to investigate their role in attenuating pressure fluctuations. With the turbulent boundary layer forming along the wall, a flow structure is expected to develop, which wasn't present in the Taylor-Green Vortex setup. The trained framework using CV has not seen such structures during the training dataset. However, their vortical appearance may be similar enough to allow vortices to be detected regardless of the underlying shape. This investigation of this framework is being conducted to provide evidence of its universal nature.

As shown in Figure 5.11, the streamlines were calculated from the numerical data in [163]. As we can see from Figure 5.11, the bottom portion of the rods is assigned a wall boundary condition (and is therefore expected to have fundamentally different vortical structures). There are periodic boundary conditions on both the left and right sides, as well as a slip wall on top. It is evident that the vortices are still

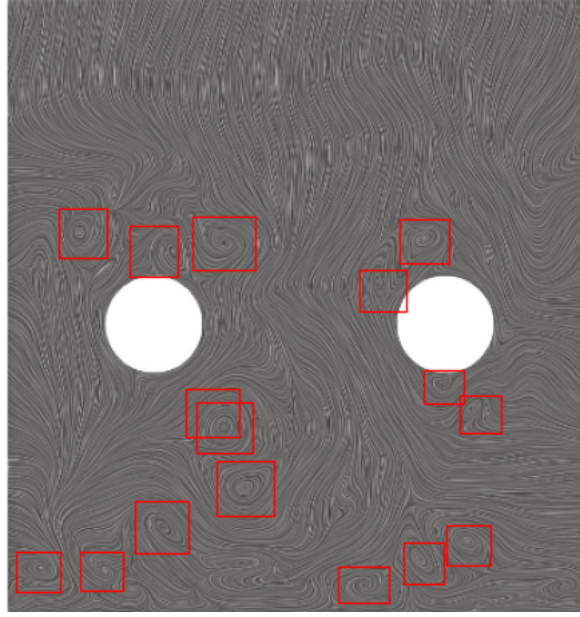


Figure 5.11: Detected vortex core regions using the computer vision approach for the turbulent boundary layer case taken from Szöke et al. [163]

detected with reasonable accuracy when using the trained CNN. At least one false positive is located below the right rod, and at least one false negative is located in the bottom left corner, but overall the correct vortex region can be predicted, and it is likely that false positives and negatives will be completely eliminated by additional images from turbulent boundary layer flow. Furthermore, using different types of flow that produce different types of vortices will reduce the risk of overfitting. The Taylor-Green Vortex problem in this case produces a wide range of vortices, but additional training would improve the robustness of the model for a different type of flow.

5.2.3 Identifying vortex regions of Application to rotating wings

Caradonna and Tung obtained experimental data applied to rotor-craft problems in 1981 [165] at NASA Ames Research Centre, which is widely used by the helicopter community for validating CFD codes. In this test case, the location of helicopter blade is in the middle of a cylindrical domain whose radius is 40m and length is

80m. A NACA 0012 aerofoil is used for the blade, which has a radius of 1.143m and an aspect ratio of 6 [166]. The MUSCL3 [168] and WENO4 [169] numerical schemes are used for the numerical solution and using unstructured meshing (triangle mesh). In this simulation, the tip Mach number M_{tip} was set to 0.89, and the pitch angle θ_c was set to 8° . Throughout the domain, an angular velocity of 265.9 rad/s is applied. The numerical CFD data obtained from Ricci *et al.* [166] contains velocity com-

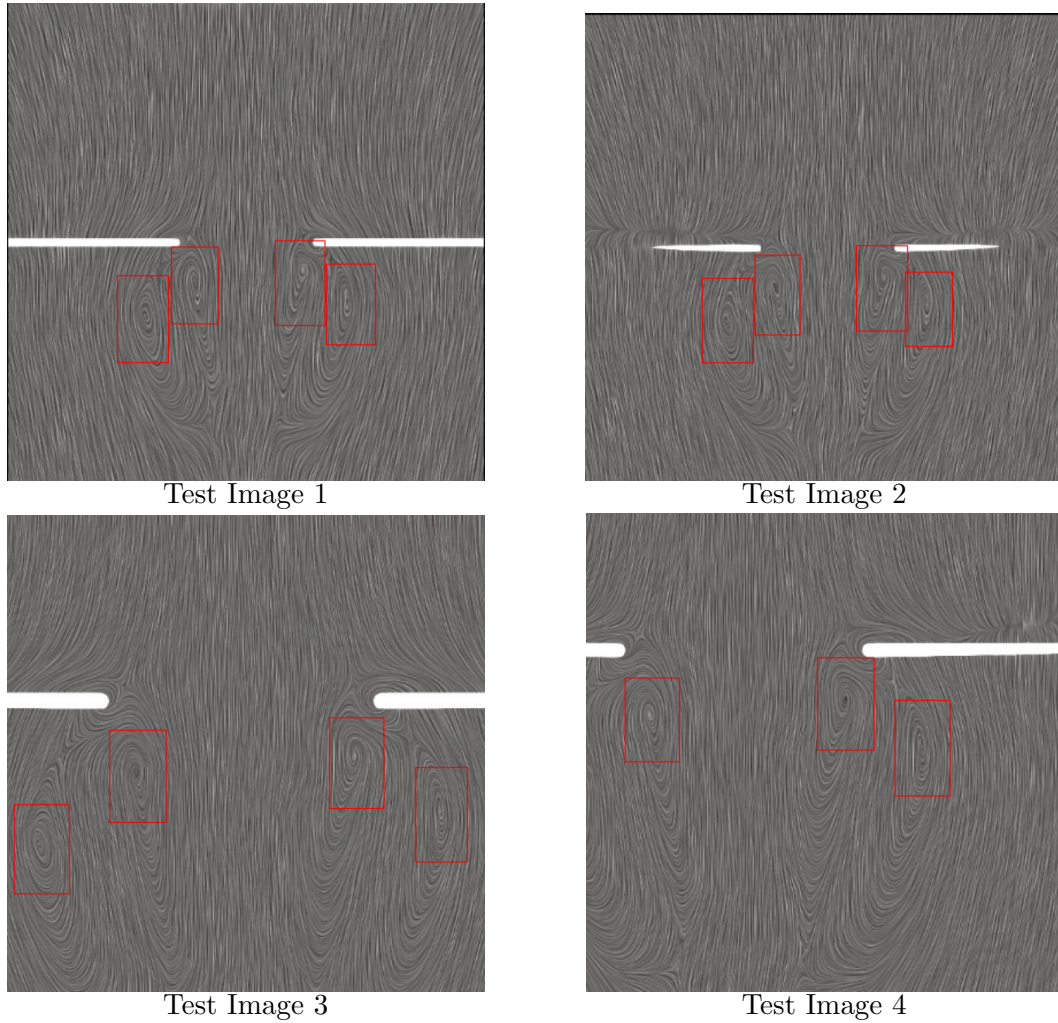


Figure 5.12: Detected vortex core regions using the YOLO algorithm for different angles of application to rotating wings case taken from Ricci *et al.* [166].

ponents for each direction (x, y, and z). These velocity components are used for visualisation and to produce streamline plots using the line integral convolution method in ParaView. The next step is to extract ten slices at different angles and then clip the streamline plots near the rotors. These streamline plots are then used

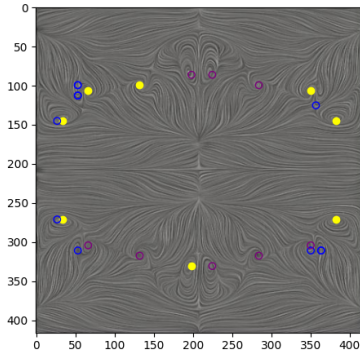
as a test data-set to identify the region of their vortices in order to validate the CV model. Figure 5.12 demonstrates that the YOLO algorithm successfully detects and captures the regions of vortex cores around the rotor with different high degrees of confidence. Therefore, with these results of TGV, turbulent boundary layer flows and application to rotating wings, it can be concluded that the developed CV framework has a universal nature and can be applied to flow cases for which it has not been trained, in order to eliminate false positives and negatives.

5.3 Detecting vortex cores using hybrid and single approaches

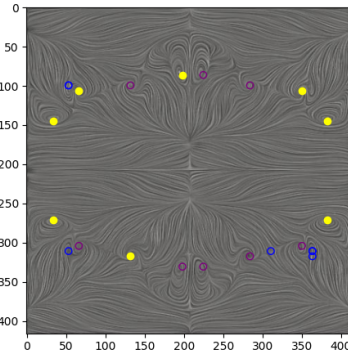
5.3.1 Detecting vortex cores of Taylor-Green Vortex

For testing the model, a flow field exported from the Taylor Green vortex case among the 335-time steps in the symmetry plane in the x, y, and z-direction, which was not included in the training data, was selected. In this section, vortices are predicted using the hybrid approaches (YOLOv3 and ANN algorithm) as well as a single approach (pure ANN machine learning without using CV) with different input variable combinations (see Chapter 3 Table 3.1) for the ANN machine learning part in order to detect vortex cores, as shown in the results.

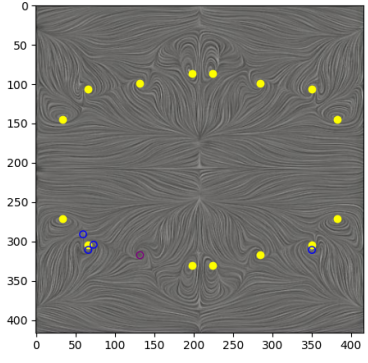
Figure 5.13 and Figure 5.14 illustrate the results obtained based on using the hybrid approach and the single approach (pure ANN) for different variable sets which are listed in Table 3.1 and number of datasets (flow field files) which consist of information recorded for every grid point on the plane, with data of each variable set stored for each corresponding coordinate X, Y, and Z, respectively.



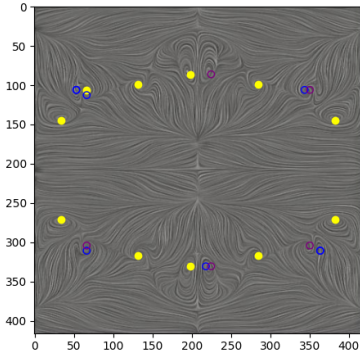
Variable set : 1
Number of datasets : 5



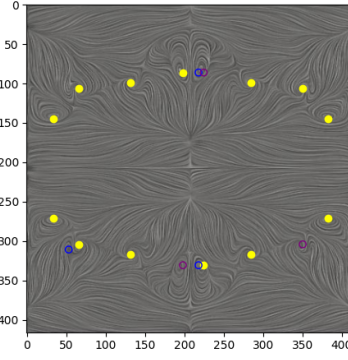
Variable set : 1
Number of datasets : 10



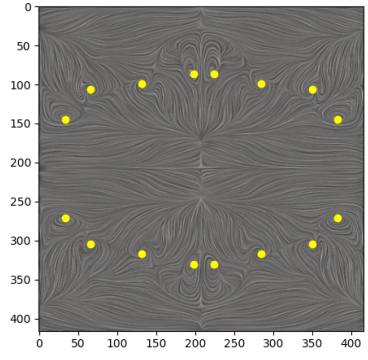
Variable set : 1
Number of datasets : 25



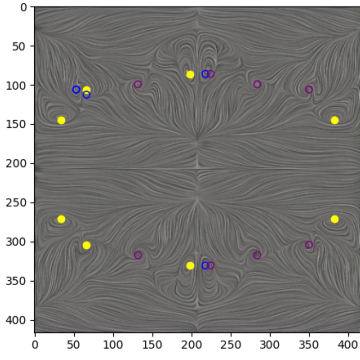
Variable set : 2
Number of datasets : 5



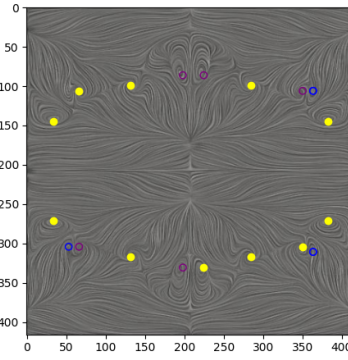
Variable set : 2
Number of datasets : 10



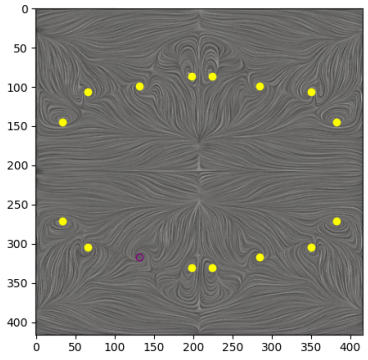
Variable set : 2
Number of datasets : 25



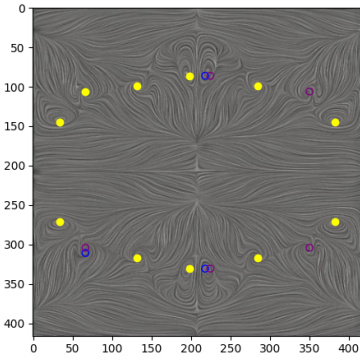
Variable set : 3
Number of datasets : 5



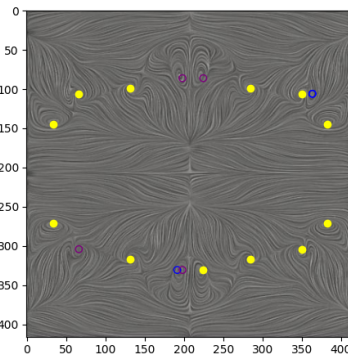
Variable set : 3
Number of datasets : 10



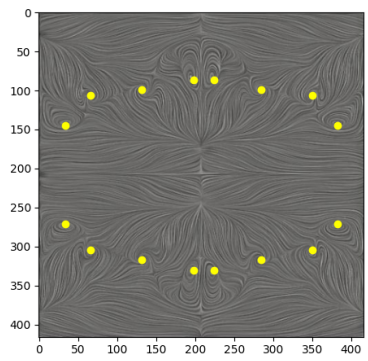
Variable set : 3
Number of datasets : 25



Variable set : 4
Number of datasets : 5



Variable set : 4
Number of datasets : 10



Variable set : 4
Number of datasets : 25

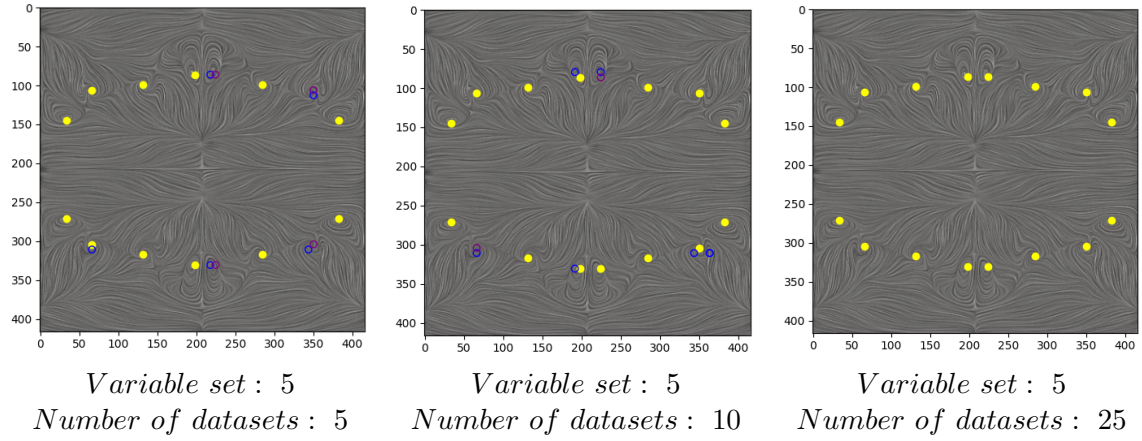
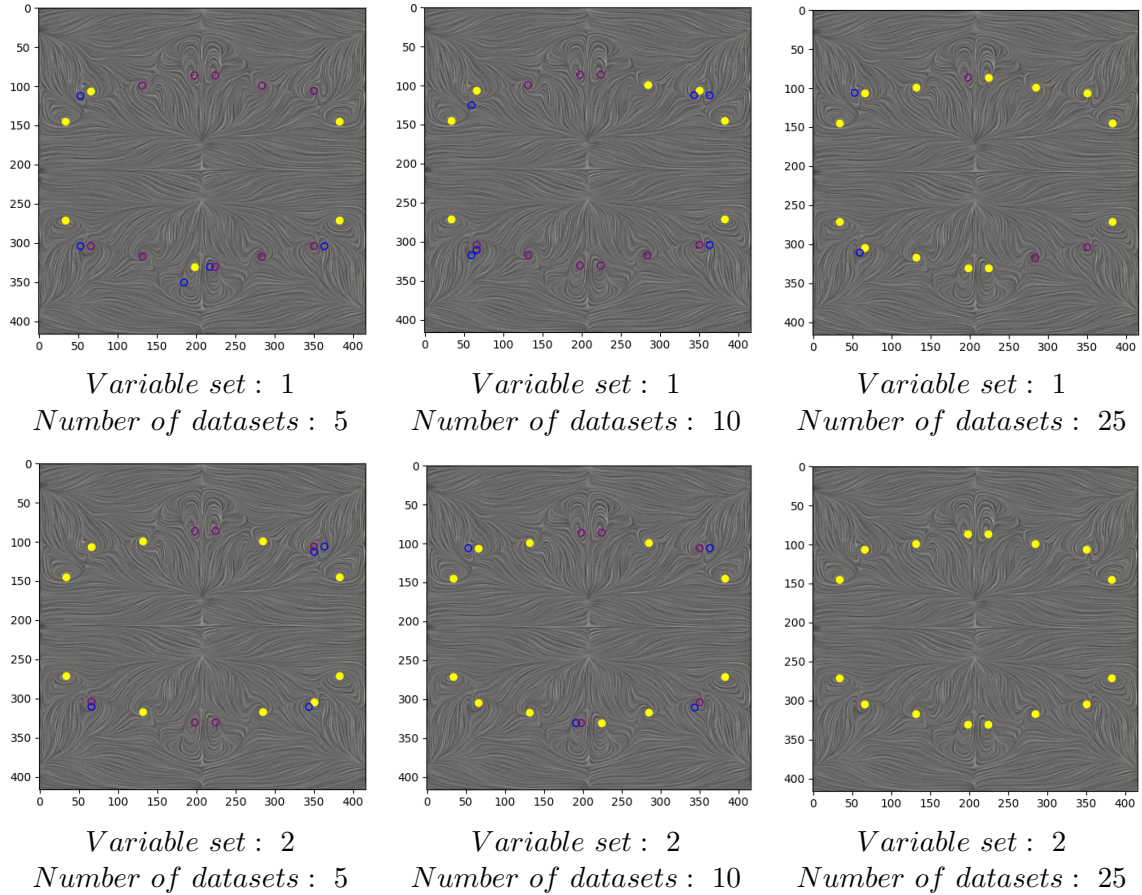


Figure 5.13: The figures show the result of using the hybrid approach (YOLO, ANN) for different flow field files and different variable sets where the yellow dots are shown the right prediction of actual vortex cores(true positives) and blue dots are false positives, and the purple dots indicate the false negatives.



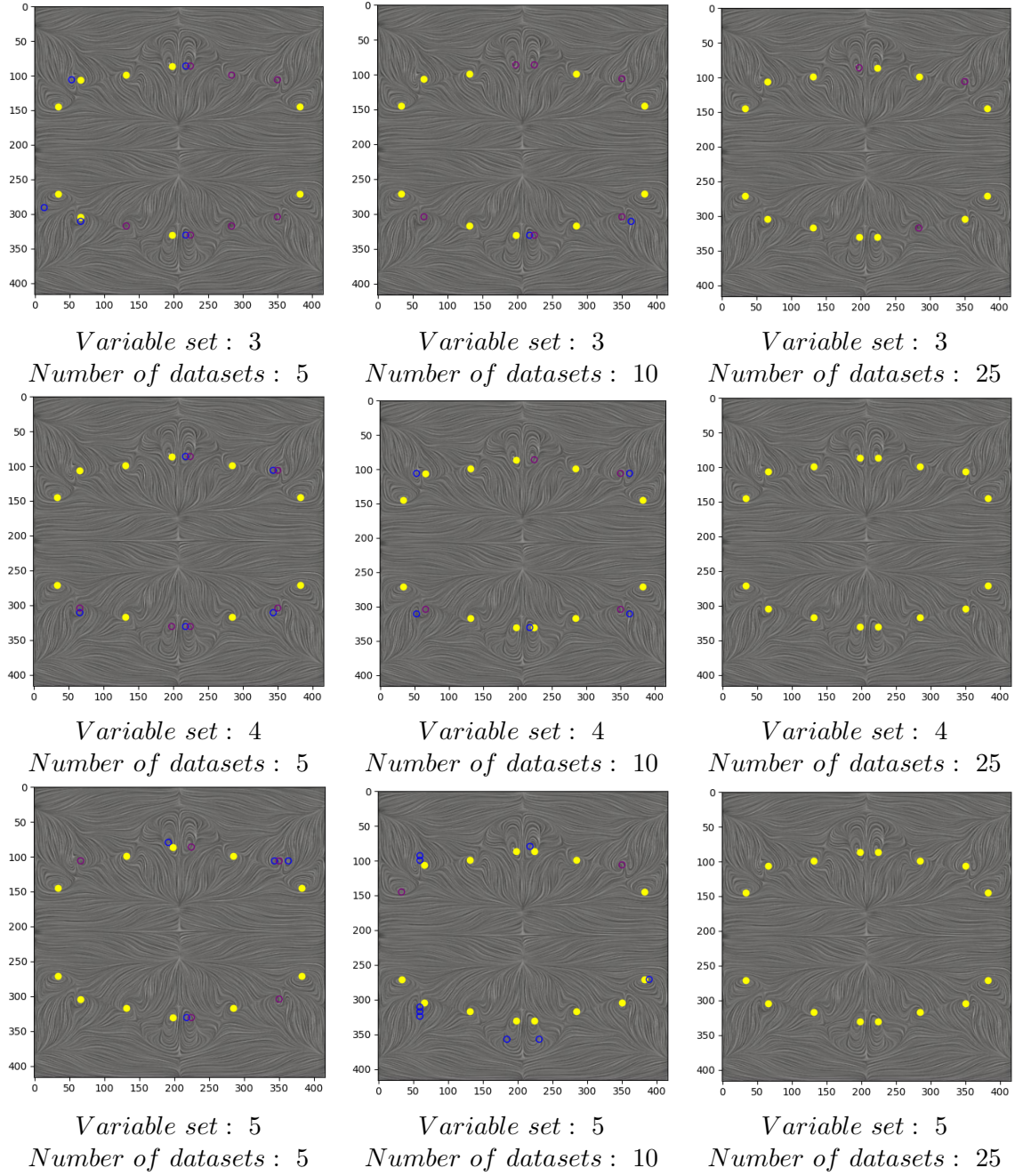


Figure 5.14: The figures show the result of using the single approach (pure ANN) for different flow field files and different variable sets where the yellow dots are shown the right prediction of actual vortex cores(true positives) and blue dots are false positives, and the purple dots indicate the false negatives.

Yellow dots indicate that a vortex core has been correctly detected (true positives), while blue dots indicate that a vortex core has been wrongly identified (false positives). Vortex cores that have not been identified (false negatives) are represented by purple dots. These figures show where the test case's true positives, false pos-

itives, and negatives are located. It is also notable that for both hybrid and pure ML approaches, false positives and negatives are in the same region of the vortex cores.

Based on data from Table 5.8, we can see the number of flow field files, variable

Table 5.8: Showing a number of expected and detected vortices for the sample test, as well as comparison of hybrid approaches and a single approach how false positives and negatives reduce as the number of flow field files in the training data set increases with different Variable sets from 1 to 5 as input features listed in table 3.1 and The computational time is taken for training in both methods.

Test image										
Number of files	Variable set	Total vortices	Both approaches			Training time	Single approach			Training time
			TP	FP	FN		TP	FP	FN	
5	1	16	8	11	8	00:42:27	6	5	10	01:39:33
	2	16	11	8	5	00:37:48	10	4	6	01:43:24
	3	16	8	5	8	00:38:28	9	5	7	01:46:18
	4	16	11	3	5	00:42:10	10	5	6	01:53:45
	5	16	12	5	4	00:43:18	11	4	5	01:59:37
10	1	16	8	7	8	01:41:59	7	6	9	02:57:52
	2	16	13	3	3	01:44:38	11	4	5	03:12:14
	3	16	11	5	5	01:46:02	10	2	6	03:20:09
	4	16	12	3	4	01:51:11	12	5	4	03:24:20
	5	16	14	7	2	01:55:28	14	9	2	03:36:43
20	1	16	13	9	3	02:24:50	10	8	6	05:38:34
	2	16	15	2	1	02:40:48	13	4	3	05:42:27
	3	16	14	1	2	02:33:17	12	6	4	05:40:01
	4	16	15	3	1	02:47:10	13	2	3	05:49:52
	5	16	16	0	4	02:56:19	15	3	1	05:54:11
25	1	16	15	4	1	03:11:16.	13	2	3	07:53:40
	2	16	16	0	0	03:23:38	16	0	0	08:00:03
	3	16	15	0	1	03:15:19	13	0	3	07:58:06
	4	16	16	0	0	03:48:58	16	0	0	08:13:13
	5	16	16	0	0	04:04:18	16	0	0	08:34:01

sets, true positives, false positives, and negatives in hybrid and single approaches, as well as the training time for each. Table 5.8 shows the result of using 5, 10, 20 and 25 different flow field files for the ANN machine learning step for both the hybrid and single approaches for each variable set. Using just five flow field files as a data set, we notice that both single and hybrid approaches provide high false positive

and negative rates, and just using the single approach (pure ANN) gives fewer false positives. In 20 flow field files with variable set 5, the hybrid approach produces zero false positives, and at this point, with half the training time, the hybrid approach is beginning to produce fewer false positives than the single approach. As long as the training data-set contains 25 flow field files, neither approach produces false positives or false negatives when 2, 4, or 5 variable sets are used. As a result of the results, using the hybrid approach of YOLO for identifying the vortex core regions, combined with ANN for detecting the vortex core within these regions, was more efficient and faster than training the models with pure ANN. Furthermore, it can be concluded that applying gradients or variable set 2 always produces better results than using variable set 1. The method uses absolute velocity and pressure components along with velocity gradient tensors, whereas the traditional methods use only velocity gradient tensors.

As shown in Figure 5.15, the True Positives for different Variable sets relative to flow field files using the hybrid approach and pure ANN are represented by a solid line and a dashed line, respectively. With each variable set described by a different colour. Variable set one in Figure 5.15 has the strongest gradient; therefore, it detects true positives the fastest after ten images. As opposed to variable set two, variable set two has a weaker gradient but is initially more likely to produce true positives. Even though variable set two does not increase as strongly as variable set one, it has the most optimal balance between input images and accuracy and can detect more true positives right away as a result. The rest of the variable sets lie in between, yet in the end, they obtain the 16 true positives. Overall, it is obvious that the true positives curve increases proportionally to the number of flow field files for all variable sets, regardless of which approach is used. However, the hybrid approach captures all vortex cores slightly better than pure ANN machine learning. Since it requires just 20 flow field files per variable set four, while the pure ANN

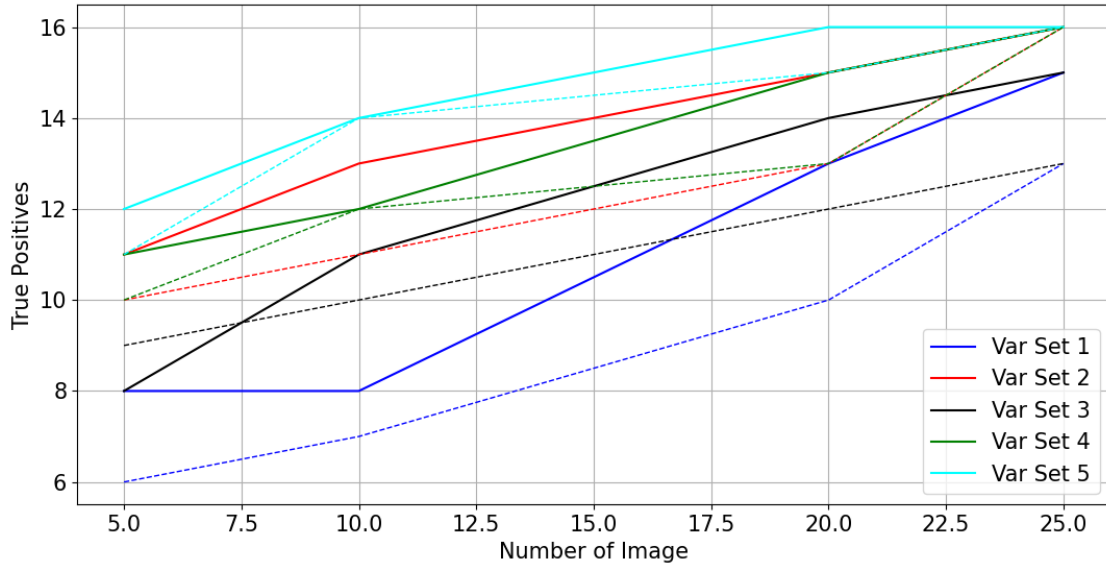


Figure 5.15: The values of True positive for different Variable sets with respect to the number of flow field files based on hybrid approaches with solid line and single approach (pure ANN) with dash line.

approach requires 25 flow field files per variable set two, four, and five.

Figures 5.16 and 5.17 illustrate false positives and false negatives with flow field

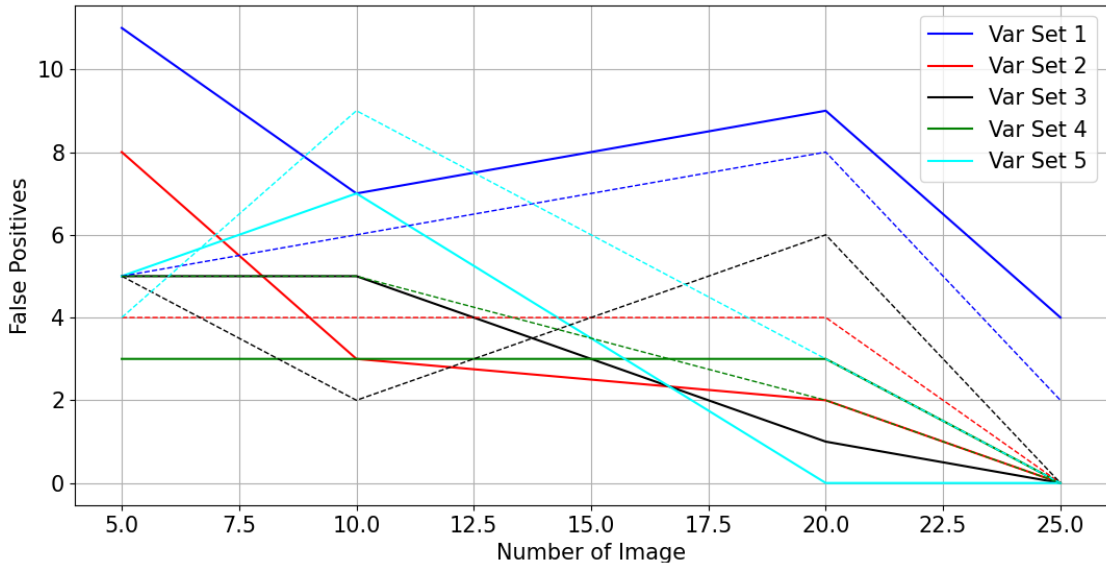


Figure 5.16: The values of False positive for different Variable sets with respect to number of flow field files based on hybrid approaches with solid line and single approach (pure ANN) with dash line.

files for different variable sets as selected by the hybrid approach with the solid line and the single approach with the dashed line. Variable set one has the highest false

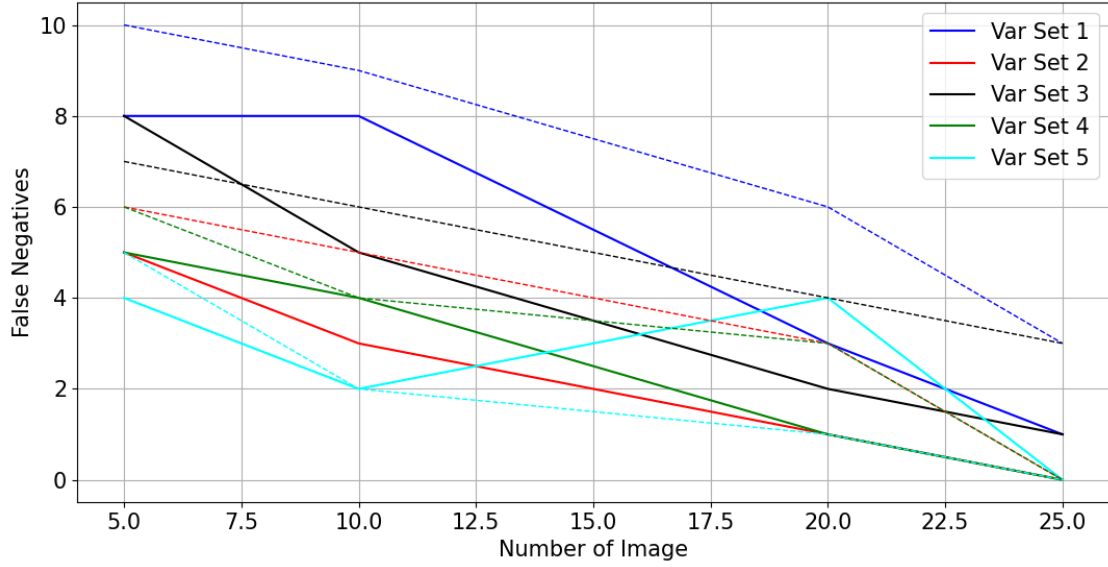


Figure 5.17: The values of False Negatives for different Variable sets with respect to number of flow field files based on hybrid approaches with solid line and single approach (pure ANN) with dash line.

positives and negatives as shown in Figure 5.16 and Figure 5.17, which explains why Figure 5.15 shows a dramatic increase in true positives with variable set one, whereas variable set two is always one of the lowest. Consequently, variable set 1 is indifferent to hybrid or pure ML approaches, whereas all other variable sets benefit from the CV approach.

5.3.1.1 Using automatic labelling for preparing the Taylor-Green vortex data-set

A limitation of supervised deep learning is the requirement for large, manually labelled data sets. Geometries with fine meshes are difficult to label manually because not all samples are equal, and where vortex cores are located is difficult to determine. In order to automatically label the data set, CV together with the K-means clustering algorithm have been used. The left side of Figure 5.18 illustrates automatically identifying regions close to vortex cores is done using CV. The CV results are represented by different-sized boxes around vortex cores (red squares). These regions are then grouped into k groups according to the K-means algorithm. In

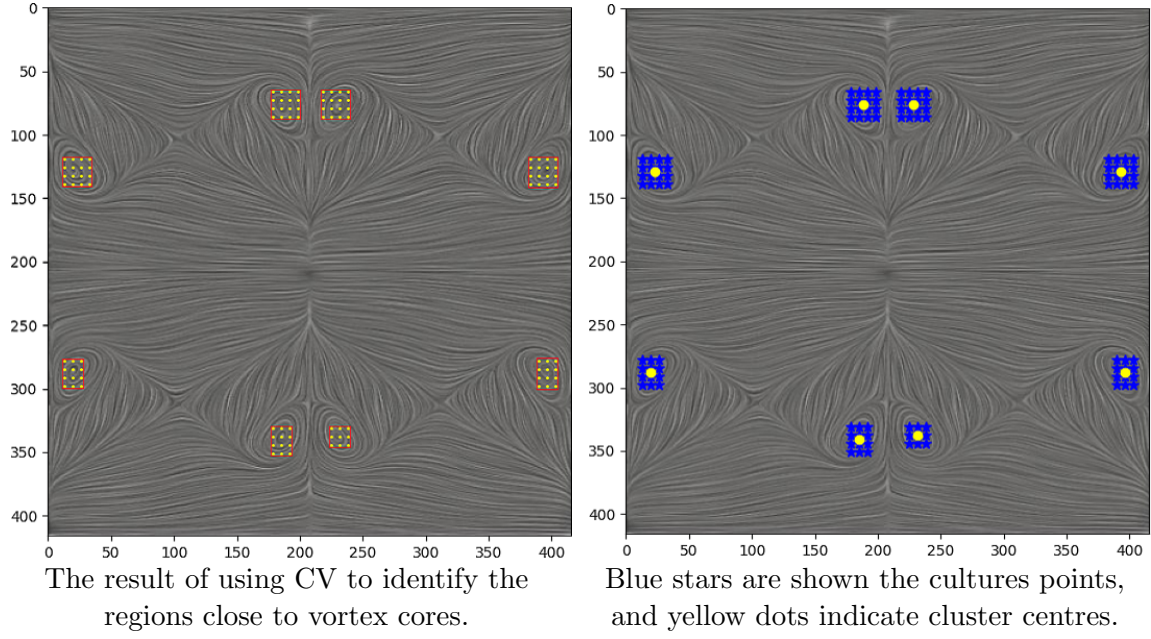


Figure 5.18: The figures show the result of using the computer vision and the K-means clustering algorithm to identify vortex cores for labelling data-set automatically.

order to determine the number of clusters K , the silhouette score library is used and assumes that each cluster has a centroid in the centre. As shown in Figure 5.18, the right-hand side of the figure shows the results of the automatic labelling process after using CV and the K-mean clustering algorithm. Blue stars describe clustering coordinates, and yellow dots illustrate cluster centres. The cluster centre's coordinates are defined as vortex cores (one) in the label field on the data set of flow field files; otherwise, they are set to zero. Automatically labelling allows the data set to be prepared efficiently and quickly for training the model to detect vortex cores. Ac-

Table 5.9: Vortex core coordinates using manual and automatic labelling techniques for the same image test.

Vortex core coordinates									
manual	x	0.305	2.815	3.469	5.978	0.305	2.815	3.469	5.978
	y	4.339	5.101	5.101	4.339	1.944	1.182	1.182	1.944
Auto	x	0.299	2.790	3.540	5.980	0.299	2.790	3.490	5.978
	y	4.340	5.190	5.190	4.290	1.990	1.100	1.200	1.990

cording to table 5.9, the Mean Square Error (MSE) of vortex core coordinates that were calculated using manual and automatic labelling techniques for the same test image with eight vortex cores (not included in the data-set) was 0.45%. It appears that the automatic labelling approach is suitable for labelling large data sets due to its relatively low MSE value. Continuing to test this hypothesis, the new automatic labelling is applied to a steady state, turbulent boundary layer flows, and rotating wing test cases in the next sections 5.3.2, and 5.3.3.

5.3.2 Detecting vortex cores of Application to turbulent boundary layer flows

numerical data from Szöke et al. [163] has been exported, where turbulence within a channel with streamwise rods was simulated to determine whether they attenuated pressure fluctuations near the wall. During the training of the CV part, the trained framework has not seen any such structures, as the turbulent boundary layer, in this case, forms along the wall, none of which was present in the Taylor-Green Vortex setup for detecting the region of vortices. Despite of their vortical appearance may be sufficiently similar to detect vortices, regardless of their underlying shape, an investigation is underway to provide proof of the universal nature of the CV framework. After identifying the region of vortices using the YOLO algorithm, the numerical data of different variable sets are obtained (see chapter 3 Table 3.1) for each image within these regions. Therefore, the amount of data is reduced and the training time for the ML model is saved as a result. In the next step, the data set for each variable set is automatically labelled by identifying the areas closest to the vortex core, and then the K-Means algorithm is used to find clusters and the centres of clusters labelled as vortex cores(see section 5.3.1.1).

Table 5.10 compares the results of hybrid (CV and ML) and single (pure ML) approaches and contains flow field files in the data set, feature variable sets, true

positives, false positives, and negatives, and their training times. Table 5.10 presents

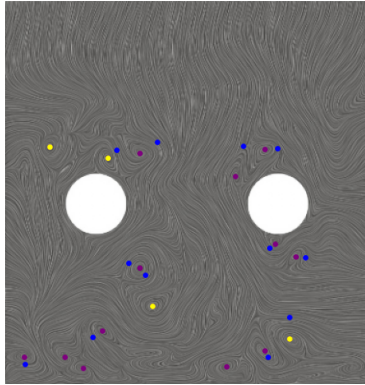
Table 5.10: Showing a number of expected and detected vortices for the Application to turbulent boundary layer flows, as well as comparison of hybrid approaches and a single approach to how false positives and negatives reduce as the number of flow field files in the training data set increases with different Variable sets from 1 to 5 as input features listed in Table 3.1 and The computational time is taken for training in both methods.

Test image										
Number of files	Variable set	Total vortices	Both approaches			Training time	Single approach			Training time
			TP	FP	FN		TP	FP	FN	
1	1	16	4	12	12	00:21:15	3	10	13	00:42:26
	2	16	7	8	9	00:24:55	5	11	11	00:44:23
	3	16	6	11	10	00:25:34	4	9	12	00:41:12
	4	16	8	6	8	00:22:54	7	8	9	00:45:34
	5	16	9	5	7	00:25:45	8	6	8	00:46:32
5	1	16	9	8	7	00:45:34	7	8	9	01:15:26
	2	16	11	6	5	00:43:36	9	5	7	01:16:14
	3	16	9	8	7	00:42:15	8	9	8	01:18:28
	4	16	11	4	5	00:46:47	10	7	6	01:17:39
	5	16	12	3	4	00:56:43	11	4	5	01:14:23
10	1	16	13	4	3	01:27:40	12	5	4	03:13:22
	2	16	14	1	2	01:29:34	13	3	3	03:24:16
	3	16	13	2	3	01:23:56	12	4	4	03:18:21
	4	16	15	0	1	01:36:54	15	1	1	03:32:55
	5	16	15	0	1	01:49:33	15	1	1	03:46:21

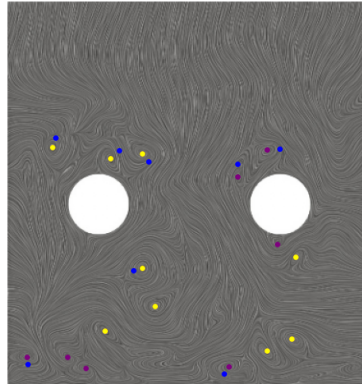
the results for the hybrid and single methods for each variable set based on one, five, and ten different flow field file data sets for the ANN machine-learning step. From the table, it is observed that in the single approach (pure ANN), fewer false positives are detected in the 1 and 3 variable sets as compared to the hybrid method when just one flow field file is used as a data set. In contrast, variable sets 2,4,5 produce fewer false positives in the hybrid approach. The hybrid approach begins producing fewer false positives and negatives in 5 flow field files with all variable sets than the single approach with less training time. When the training data set contains ten flow field files, both approaches capture the majority of vortex cores (True positives) with low false positives, and in variable sets 4 and 5, the hybrid method has zero

false positives. However, it is noted that not all vortex cores are identified by either approach, so it is necessary to have more flow field files to increase convergent training. Results indicate that hybrid approaches of YOLO for identifying vortex core areas, combined with ANN for detecting vortex cores within these regions, are more efficient and faster than pure ANN for training the models.

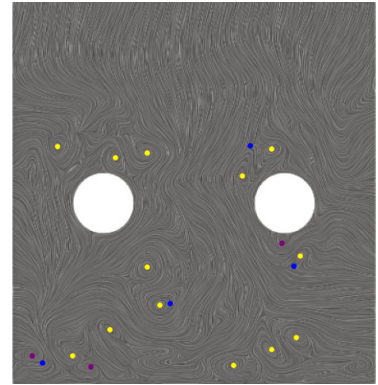
As shown in Figure 5.19 and Figure 5.20, results obtained from using the hybrid and single approaches (pure ANN), respectively, are compared for different variable sets and flow field files. It indicates that a vortex core has been correctly detected by yellow dots (true positives) and that a vortex core has been incorrectly identified by blue dots (false positives), and Vortex cores that have not been identified (false negatives) are represented by purple dots. These figures demonstrate where true and false positives are located.



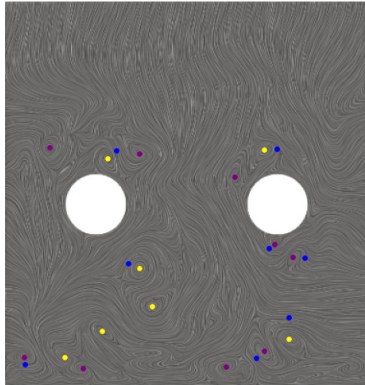
Variable set : 1
Number of datasets : 1



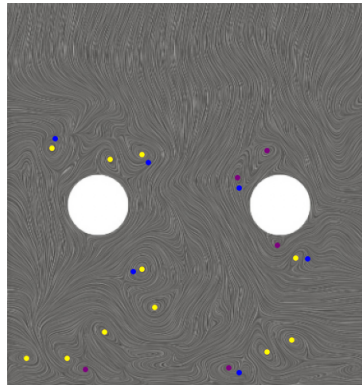
Variable set : 1
Number of datasets : 5



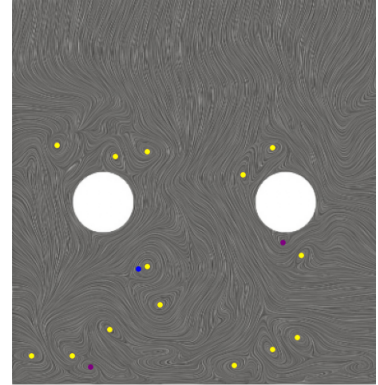
Variable set : 1
Number of datasets : 10



Variable set : 2
Number of datasets : 1



Variable set : 2
Number of datasets : 5



Variable set : 2
Number of datasets : 10

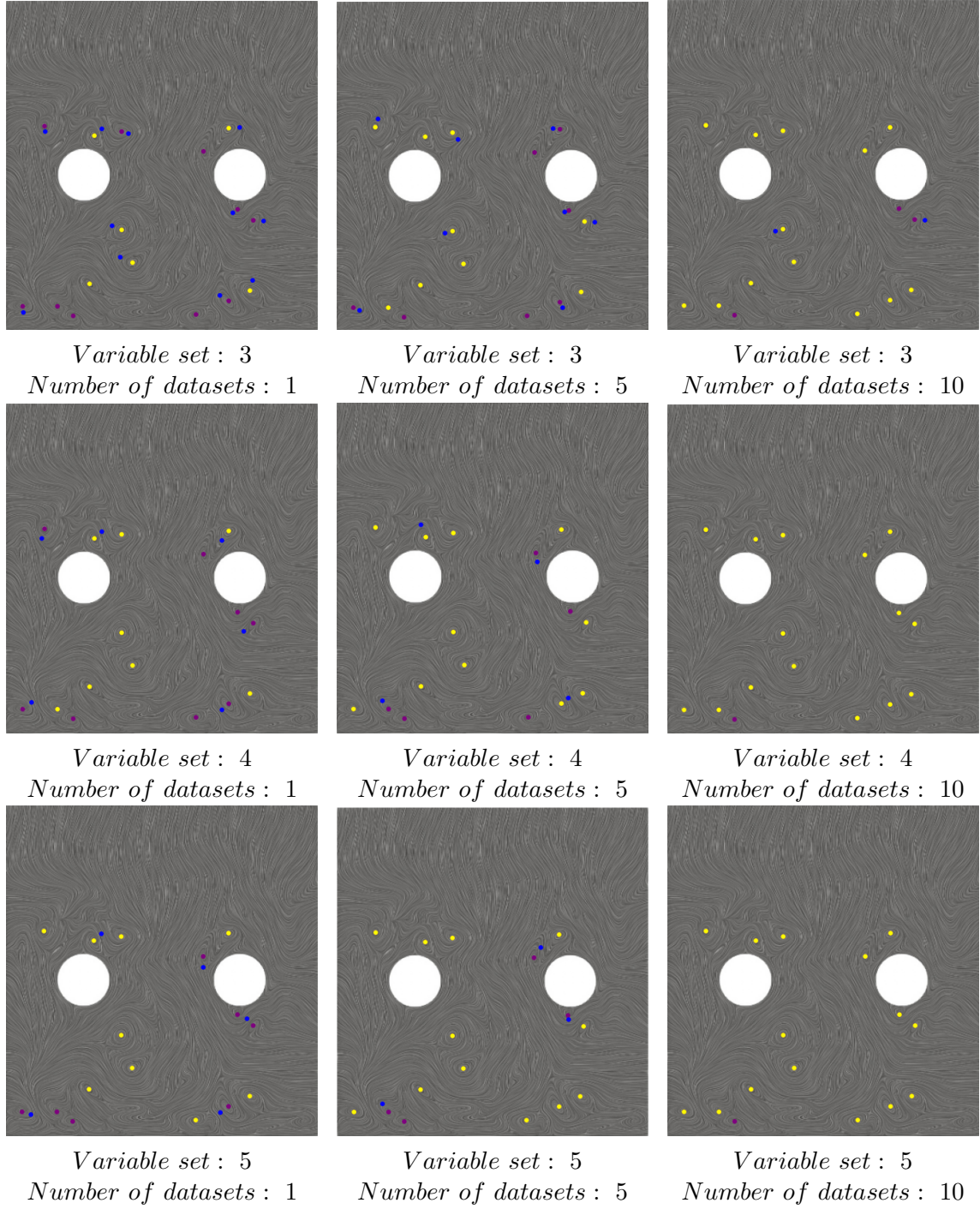
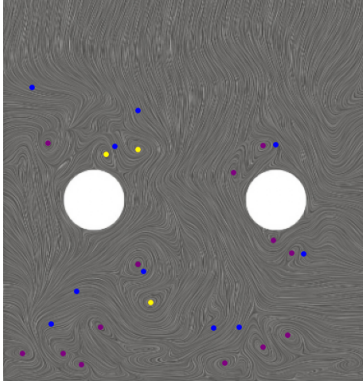


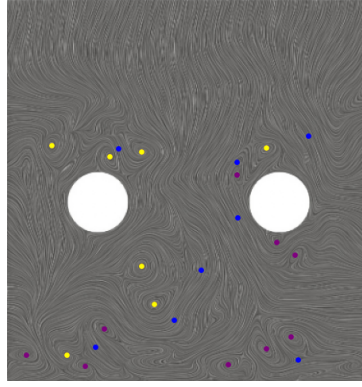
Figure 5.19: The figures show the result of using the hybrid approach (YOLO, ANN) for different flow field files and different variable sets where the yellow dots are shown the right prediction of actual vortex cores(true positives), blue dots are false positives, and the purple dots indicate the false negatives.

In addition, it can also be observed that the false positives are far from vortex cores when pure ANN is utilised, whereas the false positives obtained with the hybrid

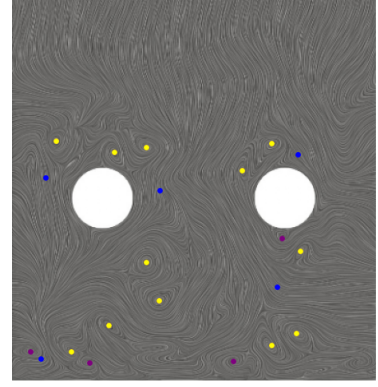
approach are located near vortex cores. It can be concluded from these results that the hybrid approaches have another advantage besides reducing training time.



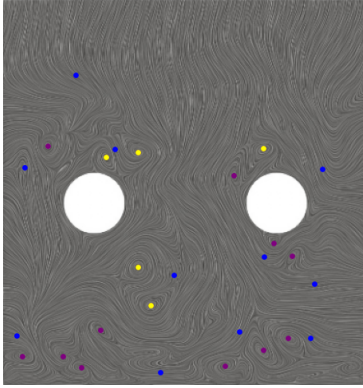
Variable set : 1
Number of datasets : 1



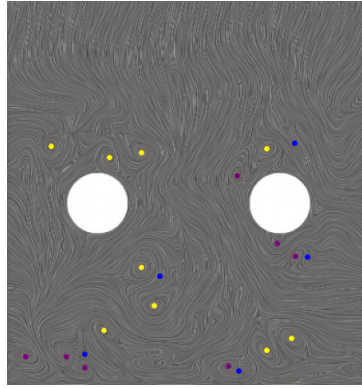
Variable set : 1
Number of datasets : 5



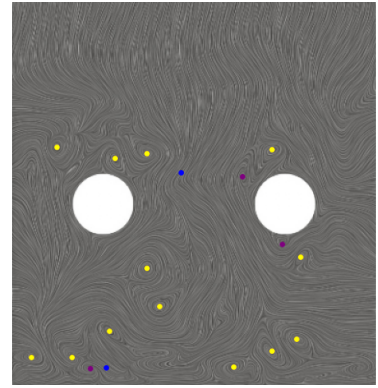
Variable set : 1
Number of datasets : 10



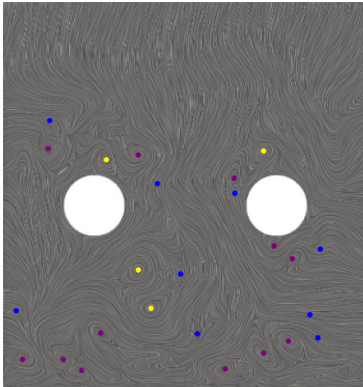
Variable set : 2
Number of datasets : 1



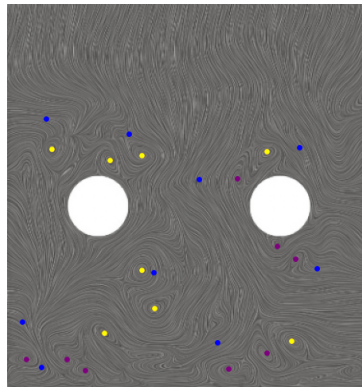
Variable set : 2
Number of datasets : 5



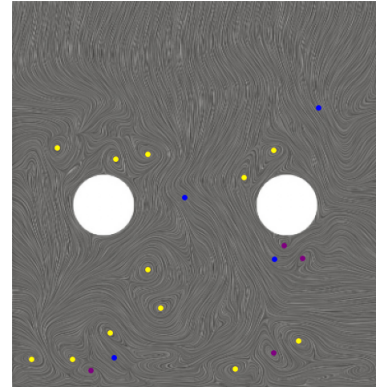
Variable set : 2
Number of datasets : 10



Variable set : 3
Number of datasets : 1



Variable set : 3
Number of datasets : 5



Variable set : 3
Number of datasets : 10

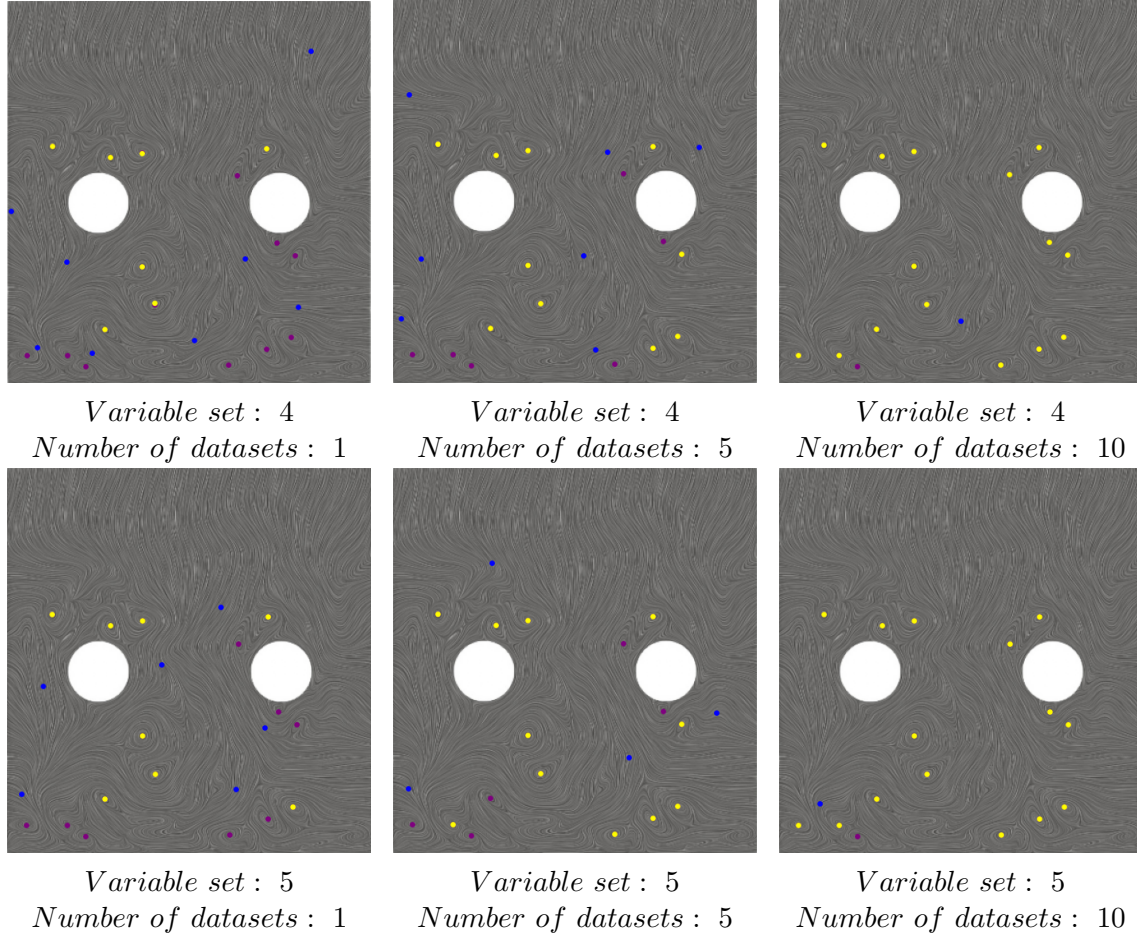


Figure 5.20: The figures show the result of using the single approach (pure ANN)

for different flow field files and different variable sets where the yellow dots are shown the right prediction of actual vortex cores(true positives), blue dots are false positives, and the purple dots indicate the false negatives.

5.3.3 Detecting vortex cores of Application to rotating wings

After obtaining CFD data from [166], using Paraview to produce 10 slices with different angles, and clipping all images near the rotor, then these images have been used as a training data set to train the CV model to identify the region of vortices. For the ML part, the data set is labelled for each variable set automatically by using CV which identifies the regions closest to the vortex core and uses the K-Means clustering algorithm to find clusters, from which then the centre of clusters are obtained, which are labelled as vortex cores. A comparison is made in Table 5.11 between hybrid (CV and ML) and single (pure ANN) approaches, which includes

flow field files in the data set, feature variable sets, true positives, false positives, and negatives, and their training times. The outcomes of hybrid and pure ANN approaches using one, five, and ten different flow field files were performed for each variable set in the ANN machine learning step, as shown in Table 5.11. It appears

Table 5.11: Showing a number of expected and detected vortices for the Application to rotating wings test case, as well as comparison of hybrid approaches and a single approach how false positives and negatives reduce as the number of flow field files in the training data set increases with different Variable sets from 1 to 5 as input features and The computational time is taken for training in both methods.

Test image										
Number of files	Variable set	Total vortices	Both approaches			Training time	Single approach			Training time
			TP	FP	FN		TP	FP	FN	
1	1	4	0	4	4	00:28:32	0	5	4	01:19:43
	2	4	1	5	3	00:31:16	0	4	4	01:13:53
	3	4	0	3	4	00:34:13	0	5	4	01:23:12
	4	4	1	2	3	00:37:21	0	3	4	01:26:42
	5	4	2	2	2	00:35:29	1	4	3	01:39:12
5	1	4	1	2	3	00:51:02	1	3	3	01:53:16
	2	4	2	3	2	00:55:17	2	4	2	02:24:38
	3	4	1	3	3	00:57:29	1	4	3	02:19:58
	4	4	3	3	1	00:53:27	2	2	2	02:16:52
	5	4	4	2	0	00:56:43	3	3	1	02:53:36
10	1	4	2	3	2	01:48:50	1	2	3	04:53:11
	2	4	3	2	1	01:55:14	2	3	2	05:04:35
	3	4	2	2	2	01:50:37	2	3	2	05:08:11
	4	4	4	0	0	01:53:20	4	1	0	05:12:12
	5	4	4	0	0	01:59:54	4	0	0	05:14:11

from Table 5.11 that variable set two is an acceptable alternative, achieving almost as good accuracy as the results of the first test case in Table 5.8. For variable set two, however, slightly more flow field files would be needed to increase convergent training and decrease false positives and false negatives. Consequently, variable set two, which comprises velocity, pressure, and their derivatives, always produce more accurate results than variable set 1. In addition, Table 5.11 shows that the hybrid approach achieves zero false positives and negatives for 10 flow field files with

variable sets 4 and 5, while pure ANN machine learning algorithms achieve zero false positives and negatives for 10 flow field files with variable sets 5. Nevertheless, hybrid approach training requires less time than the pure ANN machine learning method. Based on variable set five and different flow field files, Figure 5.21 and Figure 5.22

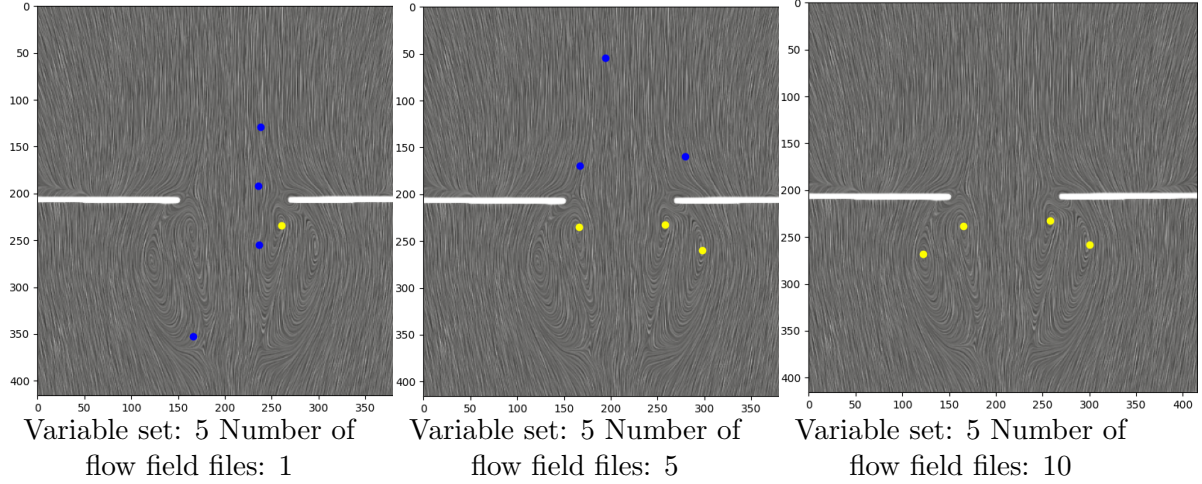


Figure 5.21: The figures show the result of using the single approach (pure ANN) for different flow field files and five variable set where the yellow dots are shown the right prediction of actual vortex cores(true positives) and blue dots are false positives.

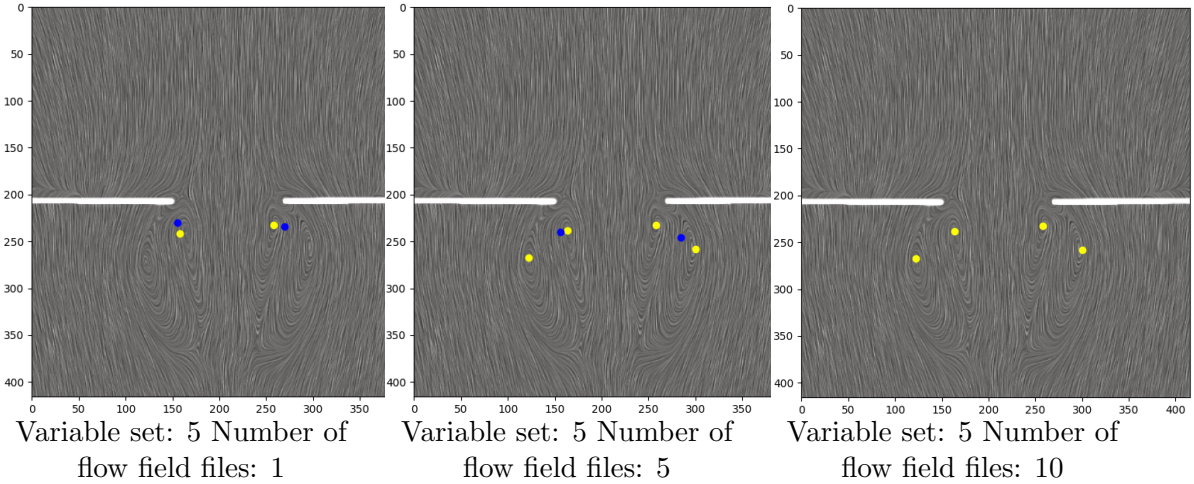


Figure 5.22: The figures show the result of using the hybrid approach (YOLO, ANN) for different flow field files and five variable set where the yellow dots are shown as the right prediction of actual vortex cores(true positives) and blue dots are false positives.

demonstrate results obtained using pure ANN and hybrid approaches, respectively.

The figures show that the false positives obtained with pure ANN are far from vortex cores, while the false positives obtained with the hybrid approach are around vortex cores. As a result, hybrid approaches have another advantage, apart from reducing training time.

5.4 Summary

Initially, local vortex detection methods were utilised to identify vortex cores in the Taylor-Green vortex and turbulent boundary layer flows. According to the result, these techniques exhibited limitations, particularly in their reliance on adjusting and fine-tuning threshold parameters to achieve accurate results. In response to these challenges, the study explores alternative approaches involving YOLO algorithm and DNN techniques to enhance vortex core detection performance and improve robustness in these flow scenarios.

The YOLO approach was employed to detect the regions of vortex cores by utilising line integral convolution-based streamline plots. Velocity components (x , y , and z) from the symmetry plane were exported for the Taylor-Green Vortex (TGV) problem using Large Eddy Simulation (LES). These components were then used to generate streamline plots using ParaView. To create the dataset, the streamline plots were manually labelled by placing bounding boxes around the vortex core regions, facilitating the training of the YOLO algorithm. The outcome indicated that the YOLO algorithm effectively eliminated all false positives and negatives for the Taylor-Green Vortex (TGV) scenario. This demonstrates the algorithm's capability to accurately and reliably detect vortex core regions, enhancing the overall performance and accuracy of the detection process. The manual labelling process ensured the dataset's quality and enabled the algorithm to learn and generalise effectively for future vortex core identification tasks. Furthermore, the trained network was used for turbulent boundary layer flows and rotating wing test cases, which were

not included in the original training dataset. The results showed that the network accurately detected most vortices, with only a limited number of false positives and negatives. To improve performance, retraining the network with additional input images may help minimise these errors, enhancing the algorithm's capability to generalise and identify vortices in various scenarios beyond the original training data. Once the YOLO algorithm identifies the areas of vortices, numerical data is extracted within these identified regions, comprising the coordinates and variable sets (see Chapter 3 Table 3.1). Subsequently, the datasets undergo automatic labelling through the utilisation of the K-means clustering algorithm. Additionally, the Keras library is used to normalise the datasets, ensuring they are ready for training the model. Next, the dataset is fed into the input layer of the Deep Neural Network (DNN), with the input shape being determined by the number of features in each variable set. The neural network contains two hidden layers, with dropout layers inserted between them to mitigate the risk of overfitting. The Rectified Linear Unit (ReLU) activation function is employed for the input and hidden layers, while the Softmax function is utilised for the output layer, comprising a single unit to indicate the presence of a vortex (1) or nonvortex (0). Afterwards, the coordinates of vortex cores are extracted from the dataset and converted into the image domain to visualise the obtained results.

Chapter 6

Conclusions and Future work

6.1 Conclusions

The detection of vortex cores in computational fluid dynamics is still challenging. In computational fluid dynamics simulations, knowing the precise location of vortex cores can be helpful during mesh refinement. It also provides valuable insights, for instance, to engineers trying to optimise aerodynamic devices to reduce drag by channelling turbulent vortices around objects. In simple engineering applications, velocity gradient-based approaches can be used to identify single vortices. However, complex engineering applications may merge and mix vortices where these simplistic approaches may fail. Despite the obvious practical benefits of getting knowledge of the exact location of vortices, numerical tools cannot reliably predict these vortices, and they generate spurious detected vortex cores, negating their value.

This study introduces a hybrid computer vision (YOLO algorithm) and ANN machine learning (TensorFlow, Keras) approach to detect vortex cores for the Taylor-Green vortex, turbulent boundary layer flows, and rotating wing cases. The first step was to present a novel computer vision-based method for detecting vortex

core regions in fluid dynamics applications through line integral convolution-based streamline plots using ParaView software. With these streamline plots, the YOLO algorithm has been used to train the model. Large-eddy simulations of the primary test case (Taylor-Green vortex problem) have been performed at a Reynolds number of 1600 using ANSYS software and extracted surface plots at the domain's symmetry plane (x , y , and z direction) in order to test the new algorithm. By computing the bounding boxes of the vortex core regions, it is demonstrated that all false positives and negatives are able to be eliminated for the Taylor-Green problem, regardless of the labelling procedure and how the labelling is performed. In addition, the trained network has been applied to turbulent boundary layer flows and rotating wing test cases where it hasn't been part of the training dataset and found that most vortices are detected correctly, with only a few false positives and negatives; retraining with additional input images may further remove these errors.

Traditional local vortex detection methods have also been employed based on the Q , Δ , and swirling strength criteria. Although these techniques are able to approximate the region of vortex cores, they generate a number of false positives and negatives, which is necessary to adjust and fine-tune a threshold parameter in order to keep the number of detected vortices bounded. In order to set the appropriate threshold parameter, local vortex detection algorithms need prior knowledge of the flow field. However, they still produce spuriously detected vortex cores. It is demonstrated that these can be removed entirely using the computer vision-based algorithm.

However, even though the CV (YOLO) approach is sufficient to eliminate false positives and negatives, it only provides bounding boxes of vortex regions. In order to detect the vortex cores themselves, the model has been extended using a hybrid CV and deep neural network. Once computer vision is employed to locate

the area where vortices exist and isolate the vortical flow patterns, the subsequent step involves the implementation of machine learning techniques where the regions of vortices are detected to extract the location of the vortex core. Consequently, the numerical data is reduced by only taking values around the vortex cores once the region of vortices are identified using the CV algorithm. Furthermore, the complete numerical data for the flow field files of the LIC image is obtained so that pure ANNs can be trained and their results can be compared with the hybrid approach. The Taylor Green Vortex test case data set is prepared using manual and newly proposed automatic labelling techniques; for automated labelling, computer vision and the k-means clustering algorithm have been used to locate the cluster's centre as the vortex core. According to the result, the value of Mean Square Error (MSE) between manual and automatic labelling techniques is low, making automatic labelling a viable alternative to labour-intensive manual labelling. This new approach to automatic labelling was applied to turbulent boundary layer flows and the rotating wing case that exhibited similar trends to the Taylor-Green vortex problem.

Through this study, it has been demonstrated that velocity and pressure are crucial information for predicting vortex cores correctly. The rate at which false positives and negatives are identified can also be reduced by incorporating gradient information about velocity and pressure. In addition, the hybrid approach increased detection speed to three times when compared with pure ANN machine learning, especially for the rotating wing case, and reduced false positives and negatives more efficiently than utilising the single method. Therefore, the newly developed hybrid computer vision + ANN approach not only significantly improves detection time but also reduces pre-processing time through the new K-means clustering-based automatic labelling approach.

6.2 Recommendations for future work

Identifying vortex cores using a hybrid CV/ML approach can significantly improve the understanding of fluid dynamics; however, there is still work to be done. Here are a few recommendations for future research:

1. **Improve accuracy:** While ML algorithms have shown promising results, the accuracy of the algorithms can still be improved, especially for the Turbulent boundary layer scenario, by expanding the dataset. Furthermore, to enhance its robustness and versatility, the algorithm should be tested with additional datasets for various other applications. Future studies should explore new ML techniques and incorporate physical models to improve the accuracy of vortex core identification.
2. **Develop real-time applications:** Many applications require real-time detection of vortex cores, such as flow control and turbulence control. Future studies should aim to develop real-time vortex identification techniques that can be applied in practical situations.
3. **Subsequent research is needed to advance the methodologies capable of detecting unsteady flows,** as current approaches for tracking vortices and identifying their cores are limited to steady-state flows.

By addressing these recommendations, future research can improve the understanding of fluid dynamics and develop practical applications that benefit many industries.

Bibliography

- [1] S. Kamkar, A. Jameson, and A. Wissink, “Automated grid refinement using feature detection,” in *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, p. 1496, 2009.
- [2] M. Hróz, P. Pecho, M. Bugaj, and J. Rostáň, “Investigation of vortex structure behavior induced by different drag reduction devices in the near field,” *Transportation Research Procedia*, vol. 65, pp. 318–328, 2022. 11th International Conference on Air Transport à INAIR 2022, Returning to the Skies.
- [3] J. D. Anderson and J. Wendt, *Computational fluid dynamics*, vol. 206. Springer, 1995.
- [4] K. Almohammadi, D. Ingham, L. Ma, and M. Pourkashan, “Computational fluid dynamics (cfd) mesh independency techniques for a straight blade vertical axis wind turbine,” *Energy*, vol. 58, pp. 483–493, 2013.
- [5] S. B. Pope and S. B. Pope, *Turbulent flows*. Cambridge university press, 2000.
- [6] M. Jiang, R. Machiraju, and D. Thompson, “Detection and visualization of vortices,” *Visualization Handbook*, pp. 295–309, 2005.
- [7] J. C. R. Hunt, a. a. Wray, and P. Moin, “Eddies, streams, and convergence zones in turbulent flows,” *Center for Turbulence Research, Proceedings of the Summer Program*, no. 1970, pp. 193–208, 1988.

- [8] C. Q. Liu, Y. Q. Wang, Y. Yang, and Z. W. Duan, “New omega vortex identification method,” *Science China: Physics, Mechanics and Astronomy*, vol. 59, no. 8, 2016.
- [9] M. S. Chong, A. E. Perry, and B. J. Cantwell, “A general classification of three-dimensional flow fields,” *Physics of Fluids A*, vol. 2, no. 5, pp. 765–777, 1990.
- [10] J. Zhou, R. J. Adrian, S. Balachandar, and T. M. Kendall, “Mechanisms for generating coherent packets of hairpin vortices in channel flow,” *Journal of Fluid Mechanics*, vol. 387, pp. 353–396, 1999.
- [11] H. A. A. Abolholl, T.-R. Teschner, and I. Moulitsas, “Surface line integral convolution-based vortex detection using computer vision,” *Journal of Computing and Information Science in Engineering*, pp. 1–18, 2023.
- [12] H. A. A. Abolholl, T.-R. Teschner, and I. Moulitsas, “A hybrid computer vision and machine learning approach for robust vortex core detection in fluid mechanics applications,”
- [13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.
- [14] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: A system for large-scale machine learning,” *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, pp. 265–283, 2016.
- [15] X. Wu and P. Moin, “Direct numerical simulation of turbulence in a nominally zero-pressure-gradient flat-plate boundary layer,” *Journal of Fluid Mechanics*,

- vol. 630, pp. 5–41, 2009.
- [16] D. Kenwright and R. Haimes, “Vortex identification - applications in aerodynamics: a case study,” *Proceedings of the IEEE Visualization Conference*, pp. 413–415, 1997.
 - [17] B. Kohler, R. Gasteiger, U. Preim, H. Theisel, M. Gutberlet, and B. Preim, “Semi-automatic vortex extraction in 4D PC-MRI cardiac blood flow data using line predicates,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2773–2782, 2013.
 - [18] S. Oeltze-Jafra, J. R. Cebal, G. Janiga, and B. Preim, “Cluster Analysis of Vortical Flow in Simulations of Cerebral Aneurysm Hemodynamics,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 757–766, 2016.
 - [19] A. D. Gutak, “Experimental investigation and industrial application of Ranque-Hilsch vortex tube,” *International Journal of Refrigeration*, vol. 49, no. 0, pp. 93–98, 2015.
 - [20] J. Z. Wu, H. Y. Ma, and M. D. Zhou, *Vorticity and vortex dynamics*. No. January 2006, 2006.
 - [21] S. Tian, Y. Gao, X. Dong, C. Liu, and W. Jian, “A Definition of Vortex Vector and Vortex,” *University of Science Technology, Nanjing, Jiangsu, China*, vol. 1, no. 1, pp. 1–26, 2016.
 - [22] R. P. Shaw, S. E. Gorrell, R. S. Woodley, and M. R. Gosnell, “Vortex core line extraction and tracking from unsteady computational fluid dynamics simulations using subjective logic,” *50th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 2012.
 - [23] H. Helmholtz, “Über integrale der hydrodynamischen gleichungen, welche den wirbelbewegungen entsprechen,” *Journal für die reine und angewandte*

- Mathematik*, vol. 55, pp. 25–55, 1858.
- [24] H. K. Moffatt, “A Brief Introduction to Vortex Dynamics and Turbulence,” pp. 1–27, 2011.
- [25] Thomson W. (Lord Kelvin), “On vortex motion,” *Proc. Roy. Soc. Edin.*, vol. 6, pp. 94–105, 1867.
- [26] Thomson W. (Lord Kelvin), “On vortex motion,” *Trans. Roy. Soc. Edin.*, vol. 25, pp. 217–260, 1869.
- [27] L. Debnath, “Sir james lighthill and modern fluid mechanics - a memorial tribute,” 2000. Copyright - © 2000. Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the associated terms available at <https://www.witpress.com/elibrary> ; Last updated - 2019-07-23.
- [28] L. F. Richardson, “The Supply of Energy from and,” pp. 354–373, 1920.
- [29] J.-Z. Wu, H.-Y. Ma, and M.-D. Zhou, “Fundamentals of vorticity dynamics,” in *Vorticity and Vortex Dynamics*, pp. 131–200, Springer, 2006.
- [30] B. P. Epps, “Review of vortex identification methods,” *AIAA SciTech Forum - 55th AIAA Aerospace Sciences Meeting*, no. January, 2017.
- [31] P. G. Saffman, *Vortex Dynamics*. Cambridge Monographs on Mechanics, Cambridge University Press, 1993.
- [32] J. Förste, “Lugt, H. J., Vortex Flow in Nature and Technology. New York et al., John Wiley & Sons 1983. XV, 297 S., zahlr. Abb., £ 47.45. ISBN 0-471-86925-2,” *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 64, no. 11, p. 502, 1984.

- [33] Y. Wang, Y. Yang, G. Yang, and C. Liu, “Dns study on vortex and vorticity in late boundary layer transition,” *Communications in Computational Physics*, vol. 22, no. 2, p. 441â459, 2017.
- [34] S. K. Robinson, “Coherent Motions in the Turbulent Boundary Layer,” *Annual Review of Fluid Mechanics*, vol. 23, no. 1, pp. 601–639, 1991.
- [35] D. KÃ¼chemann, “Report on the i.u.t.a.m. symposium on concentrated vortex motions in fluids,” *Journal of Fluid Mechanics*, vol. 21, no. 1, p. 1â20, 1965.
- [36] C. E. Willert and M. Gharib, “Digital particle image velocimetry,” *Experiments in fluids*, vol. 10, no. 4, pp. 181–193, 1991.
- [37] J. Salmon, T. Bogar, and M. Sajben, “Laser doppler velocimeter measurements in unsteady, separated, transonic diffuser flows,” *AIAA journal*, vol. 21, no. 12, pp. 1690–1697, 1983.
- [38] J. Jeong and F. Hussain, “On the identification of a vortex,” *Journal of Fluid Mechanics*, vol. 285, p. 69â94, 1995.
- [39] Y. Huang and M. A. Green, “Detection and tracking of vortex phenomena using Lagrangian coherent structures,” *Experiments in Fluids*, vol. 56, no. 7, pp. 1–12, 2015.
- [40] Y. Zhang, K. Liu, H. Xian, and X. Du, “A review of methods for vortex identification in hydroturbines,” *Renewable and Sustainable Energy Reviews*, vol. 81, pp. 1269–1285, 2018.
- [41] I. A. Sadarjoen, F. H. Post, B. Ma, D. C. Banks, and H. G. Pagendarm, “Selective visualization of vortices in hydrodynamic flows,” *Proceedings of the IEEE Visualization Conference*, vol. 98, pp. 419–422, 1998.
- [42] G. Haller, A. Hadjighasem, M. Farazmand, and F. Huhn, “Defining coherent vortices objectively from the vorticity,” *Journal of Fluid Mechanics*, vol. 795,

pp. 136–173, 2016.

- [43] M. Serra and G. Haller, “Objective eulerian coherent structures,” *Chaos*, vol. 26, no. 5, 2016.
- [44] Y. Zhang, K. Liu, H. Xian, and X. Du, “A review of methods for vortex identification in hydroturbines,” *Renewable and Sustainable Energy Reviews*, vol. 81, no. March 2017, pp. 1269–1285, 2018.
- [45] M. D. Levine, *Vision in man and machine*. McGraw-Hill College, 1985.
- [46] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.
- [47] V. Kakani, V. H. Nguyen, B. P. Kumar, H. Kim, and V. R. Pasupuleti, “A critical review on computer vision and artificial intelligence in food industry,” *Journal of Agriculture and Food Research*, vol. 2, p. 100033, 2020.
- [48] D. Marr, *Vision : a computational investigation into the human representation and processing of visual information*. San Francisco: W.H. Freeman, 1982.
- [49] P. Kinnell, T. Rymer, J. Hodgson, L. Justham, and M. Jackson, “Autonomous metrology for robot mounted 3d vision systems,” *CIRP Annals*, vol. 66, no. 1, pp. 483–486, 2017.
- [50] R. Osorio-Comparan, E. d. J. Vázquez, I. López-Juárez, M. Peña-Cabrera, M. Bustamante, and G. Lefranc, “Object detection algorithms and implementation in a robot of service,” in *2018 IEEE International Conference on Automation/XXIII Congress of the Chilean Association of Automatic Control (ICA-ACCA)*, pp. 1–7, IEEE, 2018.
- [51] J. Krüger and T. D. Nguyen, “Automated vision-based live ergonomics analysis in assembly operations,” *CIRP Annals*, vol. 64, no. 1, pp. 9–12, 2015.

- [52] H. Golnabi and A. Asadpour, “Design and application of industrial machine vision systems,” *Robotics and Computer-Integrated Manufacturing*, vol. 23, no. 6, pp. 630–637, 2007.
- [53] E. Zancul, H. O. Martins, F. P. Lopes, and F. A. da Silva Neto, “Machine vision applications in a learning factory,” *Procedia Manufacturing*, vol. 45, pp. 516–521, 2020. Learning Factories across the value chain â from innovation to service â The 10th Conference on Learning Factories 2020.
- [54] R. Samtaney, D. Silver, N. Zabusky, and J. Cao, “Visualizing features and tracking their evolution,” *Computer*, vol. 27, no. 7, pp. 20–27, 1994.
- [55] I. Carlbom, I. Chakravarty, and W. M. Hsu, “SIGGRAPH’91 workshop report integrating computer graphics, computer vision, and image processing in scientific applications,” *ACM SIGGRAPH Computer Graphics*, vol. 26, no. 1, pp. 8–17, 1992.
- [56] S. Abdurakipov, M. Tokarev, E. Butakov, and V. Dulin, “Application of computer vision and neural network analysis to study the structure and dynamics of turbulent jets,” *Journal of Physics: Conference Series*, vol. 1421, p. 012018, dec 2019.
- [57] M. A. Mohammed, M. K. Abd Ghani, N. a. Arunkumar, R. I. Hamed, M. K. Abdullah, and M. Burhanuddin, “A real time computer aided object detection of nasopharyngeal carcinoma using genetic algorithm and artificial neural network based on haar feature fear,” *Future Generation Computer Systems*, vol. 89, pp. 539–547, 2018.
- [58] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, “Multi-view 3d object detection network for autonomous driving,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1907–1915, 2017.
- [59] D. Feng, A. Harakeh, S. L. Waslander, and K. Dietmayer, “A review and

- comparative study on probabilistic object detection in autonomous driving,” *IEEE Transactions on Intelligent Transportation Systems*, 2021.
- [60] X. Li, S. Lai, and X. Qian, “Dbcface: Towards pure convolutional neural network face detection,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 4, pp. 1792–1804, 2021.
- [61] A. Kumar, A. Kaur, and M. Kumar, “Face detection techniques: a review,” *Artificial Intelligence Review*, vol. 52, no. 2, pp. 927–948, 2019.
- [62] Q. Bai, S. Li, J. Yang, Q. Song, Z. Li, and X. Zhang, “Object detection recognition and robot grasping based on machine learning: A survey,” *IEEE access*, vol. 8, pp. 181855–181879, 2020.
- [63] G. Du, K. Wang, S. Lian, and K. Zhao, “Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review,” *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1677–1734, 2021.
- [64] I. Marković, F. Chaumette, and I. Petrović, “Moving object detection, tracking and following using an omnidirectional camera on a mobile robot,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5630–5635, IEEE, 2014.
- [65] Z. Zhou, L. Li, A. Fürsterling, H. J. Durocher, J. Mouridsen, and X. Zhang, “Learning-based object detection and localization for a mobile robot manipulator in sme production,” *Robotics and Computer-Integrated Manufacturing*, vol. 73, p. 102229, 2022.
- [66] S. Hossain and D.-j. Lee, “Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with gpu-based embedded devices,” *Sensors*, vol. 19, no. 15, p. 3371, 2019.

- [67] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.
- [68] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, “Imagenet large scale visual recognition challenge,” *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.
- [69] J. Deng, X. Xuan, W. Wang, Z. Li, H. Yao, and Z. Wang, “A review of research on object detection based on deep learning,” in *Journal of Physics: Conference Series*, vol. 1684, p. 012028, IOP Publishing, 2020.
- [70] M. Carranza-García, J. Torres-Mateo, P. Lara-Benítez, and J. García-Gutiérrez, “On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data,” *Remote Sensing*, vol. 13, no. 1, 2021.
- [71] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.
- [72] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.
- [73] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *Advances in neural information processing systems*, vol. 28, 2015.
- [74] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

- [75] L. Cheng and Z. Yang, “Grcnn: Graph recognition convolutional neural network for synthesizing programs from flow charts,” 2020.
- [76] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pp. 21–37, Springer, 2016.
- [77] R. Samtaney, D. Silver, N. Zabusky, and J. Cao, “Visualizing Features and Tracking Their Evolution,” *Computer*, vol. 27, no. 7, pp. 20–27, 1994.
- [78] W. M. Hsu, D. E. Corporation, and N. Hunt, “SIGGRAPH ’91 Workshop Report Integrating Computer Graphics , Computer Vision , and Image Processing in Scientific Applications Ingrid Carlbom , Digital Equipment Corporation Indranil Chakravarty , Schlumberger Laboratory for Computer Science,” *Computer*, pp. 8–17, 1992.
- [79] D. Silver and X. Wang, “Tracking and visualizing turbulent 3D features,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 2, pp. 129–141, 1997.
- [80] S. Abdurakipov, M. Tokarev, E. Butakov, and V. Dulin, “Application of computer vision and neural network analysis to study the structure and dynamics of turbulent jets,” *Journal of Physics: Conference Series*, vol. 1421, no. 1, 2019.
- [81] Y. Wang, L. Deng, Z. Yang, D. Zhao, and F. Wang, “A rapid vortex identification method using fully convolutional segmentation network,” *Visual Computer*, vol. 37, no. 2, pp. 261–273, 2021.
- [82] B. Kim and T. Günther, “Robust reference frame extraction from unsteady 2D vector fields with convolutional neural networks,” *Computer Graphics Forum*, vol. 38, no. 3, pp. 285–295, 2019.

- [83] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
- [84] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, “Face recognition: A convolutional neural-network approach,” *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.
- [85] R. Lguensat, M. Sun, R. Fablet, E. Mason, P. Tandeo, and G. Chen, “EddyNet: A deep neural network for pixel-wise classification of oceanic eddies,” *International Geoscience and Remote Sensing Symposium (IGARSS)*, vol. 2018-July, no. June 2021, pp. 1764–1767, 2018.
- [86] K. Franz, R. Roscher, A. Milioto, and S. Wenzel, “OCEAN EDDY IDENTIFICATION AND TRACKING USING NEURAL NETWORKS Katharina Franz, Ribana Roscher, Andres Milioto, Susanne Wenzel, Jürgen Kusche University of Bonn, Institute of Geodesy and Geoinformation, Nussallee 15+17, 53115 Bonn, Germany,” pp. 1–4, 2018.
- [87] C. M. Ströfer, J. Wu, H. Xiao, and E. Paterson, “Data-Driven, Physics-Based Feature Extraction from Fluid Flow Fields,” 2018.
- [88] L. Deng, Y. Wang, Y. Liu, F. Wang, S. Li, and J. Liu, “A CNN-based vortex identification method,” *Journal of Visualization*, vol. 22, no. 1, pp. 65–78, 2019.
- [89] L. Zhang, Q. Deng, R. Machiraju, A. Rangarajan, D. Thompson, D. K. Walters, and H.-W. Shen, “Boosting techniques for physics-based vortex detection,” in *Computer Graphics Forum*, vol. 33, pp. 282–293, Wiley Online Library, 2014.
- [90] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of computer and system*

- sciences*, vol. 55, no. 1, pp. 119–139, 1997.
- [91] E. B. Hunt, P. J. Stone, and J. Marin, *Experiments in induction / Earl B. Hunt, Janet Marin, Philip J. Stone*. Academic Press New York, 1966.
- [92] N. Nilsson, “Learning machines, new york, ny: Mcgraw-hill, chapter 1,” trainable pattern classifiers,” pp. 1-13; and chapter 2,” some important discriminant functions: their properties and their implementations,” pp. 15-42,” *Neurocomputing*, vol. 2, p. 464, 1988.
- [93] F. Rosenblatt, “The perceptron: a probabilistic model for information storage and organization in the brain.,” *Psychological review*, vol. 65, no. 6, p. 386, 1958.
- [94] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, “Machine learning, neural and statistical classification,” 1994.
- [95] J. Alzubi, A. Nayyar, and A. Kumar, “Machine learning from theory to algorithms: An overview,” *Journal of Physics: Conference Series*, vol. 1142, p. 012012, nov 2018.
- [96] J. R. Quinlan, “Induction of decision trees,” *Machine learning*, vol. 1, pp. 81–106, 1986.
- [97] J. Malone, K. McGarry, S. Wermter, and C. Bowerman, “Data mining using rule extraction from kohonen self-organising maps,” *Neural Computing & Applications*, vol. 15, pp. 9–17, 2006.
- [98] H. Zhang, Y. Hou, J. Zhao, L. Wang, T. Xi, and Y. Li, “Automatic welding quality classification for the spot welding based on the hopfield associative memory neural network and chernoff face description of the electrode displacement signal features,” *Mechanical Systems and Signal Processing*, vol. 85, pp. 1035–1043, 2017.

- [99] E. Alpaydin, *Introduction to machine learning*. MIT press, 2020.
- [100] V. Rajendran, K. Y. Kelly, E. Leonardi, and K. Menzies, “Vortex detection on unsteady cfd simulations using recurrent neural networks,” in *2018 Fluid dynamics conference*, p. 3724, 2018.
- [101] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science*, vol. 349, no. 6245, pp. 255–260, 2015.
- [102] S. Ahmad, A. Kalra, and H. Stephen, “Estimating soil moisture using remote sensing data: A machine learning approach,” *Advances in water resources*, vol. 33, no. 1, pp. 69–80, 2010.
- [103] T. M. Mitchell and T. M. Mitchell, *Machine learning*, vol. 1. McGraw-hill New York, 1997.
- [104] M. Frank, D. Drikakis, and V. Charissis, “Machine-learning methods for computational science and engineering,” *Computation*, vol. 8, no. 1, p. 15, 2020.
- [105] J. H. Lee, J. Shin, and M. J. Realff, “Machine learning: Overview of the recent progresses and implications for the process systems engineering field,” *Computers Chemical Engineering*, vol. 114, pp. 111–121, 2018. FOCAPO/CPC 2017.
- [106] V. Nasteski, “An overview of the supervised machine learning methods,” *Horizons. b*, vol. 4, pp. 51–62, 2017.
- [107] H. U. Dike, Y. Zhou, K. K. Deveerasetty, and Q. Wu, “Unsupervised learning based on artificial neural network: A review,” in *2018 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, pp. 322–327, 2018.
- [108] Y. Li, “Deep reinforcement learning: An overview,” *CoRR*, vol. abs/1701.07274, 2017.

- [109] I. N. da Silva, D. Hernane Spatti, R. Andrade Flauzino, L. H. B. Liboni, and S. F. dos Reis Alves, *Introduction*, pp. 3–19. Cham: Springer International Publishing, 2017.
- [110] S. Agatonovic-Kustrin and R. Beresford, “Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research,” *Journal of Pharmaceutical and Biomedical Analysis*, vol. 22, no. 5, pp. 717–727, 2000.
- [111] E. Yilmaz and B. J. German, “A convolutional neural network approach to training predictors for airfoil performance,” *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2017*, no. June, pp. 1–19, 2017.
- [112] J. N. Kutz, “Deep learning in fluid dynamics,” *Journal of Fluid Mechanics*, vol. 814, pp. 1–4, 2017.
- [113] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “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, pp. 686–707, 2019.
- [114] S. Lee and D. You, “Data-driven prediction of unsteady flow over a circular cylinder using deep learning,” *Journal of Fluid Mechanics*, vol. 879, pp. 217–254, 2019.
- [115] V. Sekar, Q. Jiang, C. Shu, and B. C. Khoo, “Fast flow field prediction over airfoils using deep learning approach,” *Physics of Fluids*, vol. 31, no. 5, p. 057103, 2019.
- [116] J. Ling, A. Kurzawski, and J. Templeton, “Reynolds averaged turbulence modelling using deep neural networks with embedded invariance,” *Journal of Fluid Mechanics*, vol. 807, pp. 155–166, 2016.

- [117] B. D. Tracey, K. Duraisamy, and J. J. Alonso, “A machine learning strategy to assist turbulence model development,” in *53rd AIAA aerospace sciences meeting*, p. 1287, 2015.
- [118] Z. J. Zhang and K. Duraisamy, “Machine learning methods for data-driven turbulence modeling,” in *22nd AIAA computational fluid dynamics conference*, p. 2460, 2015.
- [119] G. G. Stokes, *On the Effect of the Internal Friction of Fluids on the Motion of Pendulums*, vol. 3 of *Cambridge Library Collection - Mathematics*, p. 1â10. Cambridge University Press, 2009.
- [120] F. M. White and J. Majdalani, *Viscous fluid flow*, vol. 3. McGraw-Hill New York, 2006.
- [121] H. ÖZMEN and A. Alipaşa, “Studentsâdifficulties in understanding of the conservation of matter in open and closed-system chemical reactions,” *Chemistry Education Research and Practice*, vol. 4, no. 3, pp. 279–290, 2003.
- [122] S. Ganesan and D. Poirier, “Conservation of mass and momentum for the flow of interdendritic liquid during solidification,” *Metallurgical Transactions B*, vol. 21, pp. 173–181, 1990.
- [123] M. Brunt and G. Brunt, “Introducing conservation of momentum,” *Physics Education*, vol. 48, p. 664, sep 2013.
- [124] J. W. Swegle, “Conservation of momentum and tensile instability in particle methods,” tech. rep., Sandia National Labs., 2000.
- [125] R. Eymard, T. Gallouët, and R. Herbin, “Finite volume methods,” *Handbook of numerical analysis*, vol. 7, pp. 713–1018, 2000.
- [126] M. Jadidi, S. Moghtadernejad, and A. Dolatabadi, “A comprehensive review on fluid dynamics and transport of suspension/liquid droplets and particles

- in high-velocity oxygen-fuel (hvof) thermal spray,” *Coatings*, vol. 5, no. 4, pp. 576–645, 2015.
- [127] J. H. Ferziger, M. Perić, and R. L. Street, *Computational methods for fluid dynamics*, vol. 3. Springer, 2002.
- [128] S. Patankar, “Numerical heat transfer and fluid flow: Computational methods in mechanics and thermal science,” 1980.
- [129] J. R. DeBonis, “A high-resolution capability for large-eddy simulation of jet flows,” *40th AIAA Fluid Dynamics Conference*, vol. 1, no. July, pp. 1–22, 2010.
- [130] M. Lesieur and O. Métais, “New trends in large-eddy simulations of turbulence,” *Annual Review of Fluid Mechanics*, vol. 28, pp. 45–82, 1996.
- [131] G. Lindner, Y. Devaux, and S. Miskovic, “VortexFitting: A post-processing fluid mechanics tool for vortex identification,” *SoftwareX*, vol. 12, p. 100604, 2020.
- [132] C. Liu, Y.-s. Gao, X.-r. Dong, Y.-q. Wang, J.-m. Liu, Y.-n. Zhang, X.-s. Cai, and N. Gui, “Third generation of vortex identification methods: Omega and liutex/rortex based systems,” *Journal of Hydrodynamics*, vol. 31, pp. 205–223, 2019.
- [133] B. Cabral and L. Leedom, “Imaging vector fields using line integral convolution,” *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1993*, pp. 263–270, 1993.
- [134] A. Farhadi and J. Redmon, “Yolov3: An incremental improvement,” in *Computer Vision and Pattern Recognition*, pp. 1804–2767, Springer Berlin/Heidelberg, Germany, 2018.

- [135] Z. Ni, J. Chen, N. Sang, C. Gao, and L. Liu, “LIGHT YOLO FOR HIGH-SPEED GESTURE RECOGNITION Key Laboratory of Ministry of Education for Image Processing and Intelligent Control , School of Automation , Huazhong University of Science and Technology , Wuhan 430074 , China . NERCEL , Central China Norma,” *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 3099–3103, 2018.
- [136] M. Jiang, T. Hai, Z. Pan, H. Wang, Y. Jia, and C. Deng, “Multi-Agent Deep Reinforcement Learning for Multi-Object Tracker,” *IEEE Access*, vol. 7, pp. 32400–32407, 2019.
- [137] S. Duman, A. Elewi, and Z. Yetgin, “Distance estimation from a monocular camera using face and body features,” *Arabian Journal for Science and Engineering*, pp. 1–11, 2022.
- [138] Q. Wang, S. Bi, M. Sun, Y. Wang, D. Wang, and S. Yang, “Deep learning approach to peripheral leukocyte recognition,” *PLoS ONE*, vol. 14, no. 6, pp. 1–18, 2018.
- [139] U. Nepal and H. Eslamiat, “Comparing yolov3, yolov4 and yolov5 for autonomous landing spot detection in faulty uavs,” *Sensors*, vol. 22, no. 2, 2022.
- [140] Z. Yi, S. Yongliang, and Z. Jun, “An improved tiny-yolov3 pedestrian detection algorithm,” *Optik*, vol. 183, pp. 17–23, 2019.
- [141] N. Lv, J. Xiao, and Y. Qiao, “Object detection algorithm for surface defects based on a novel yolov3 model,” *Processes*, vol. 10, no. 4, 2022.
- [142] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13*, pp. 740–755, Springer, 2014.

- [143] J. Davis and M. Goadrich, “The relationship between precision-recall and roc curves,” in *Proceedings of the 23rd international conference on Machine learning*, pp. 233–240, 2006.
- [144] S. A. Khan and Z. Ali Rana, “Evaluating performance of software defect prediction models using area under precision-recall curve (auc-pr),” in *2019 2nd International Conference on Advancements in Computational Sciences (ICACS)*, pp. 1–6, 2019.
- [145] S. Edström, “Object detection for autonomous trash and litter collection,” 2022.
- [146] Y. Xue and Y. Li, “A fast detection method via region-based fully convolutional neural networks for shield tunnel lining defects,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 8, pp. 638–654, 2018.
- [147] A. G. Calvet, M. Dave, and J. A. Franck, “Unsupervised clustering and performance prediction of vortex wakes from bio-inspired propulsors,” *Bioinspiration Biomimetics*, vol. 16, no. 4, p. 046015, 2021.
- [148] T. Kansal, S. Bahuguna, V. Singh, and T. Choudhury, “Customer segmentation using k-means clustering,” in *2018 international conference on computational techniques, electronics and mechanical systems (CTEMS)*, pp. 135–139, IEEE, 2018.
- [149] P. D. Hung, “Detection of central sleep apnea based on a single-lead ecg,” in *Proceedings of the 2018 5th International Conference on Bioinformatics Research and Applications*, pp. 78–83, 2018.
- [150] P. D. Hung, “Central sleep apnea detection using an accelerometer,” in *Proceedings of the 1st International Conference on Control and Computer Vision*, pp. 106–111, 2018.

- [151] P. D. Hung, T. D. Hanh, and V. T. Diep, “Breast cancer prediction using spark mllib and ml packages,” in *Proceedings of the 2018 5th International Conference on Bioinformatics Research and Applications*, pp. 52–59, 2018.
- [152] S. Yin and J. Liu, “A k-means approach for map-reduce model and social network privacy protection,” *J. Inf. Hiding Multim. Signal Process.*, vol. 7, no. 6, pp. 1215–1221, 2016.
- [153] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.
- [154] R. A. Dunne and N. A. Campbell, “On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function,” in *Proc. 8th Aust. Conf. on the Neural Networks, Melbourne*, vol. 181, p. 185, Citeseer, 1997.
- [155] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014.
- [156] Y. Li, H. Zhang, Q. Guo, and X. Li, “Machine learning methods for ship detection in satellite images,” 2013.
- [157] G. I. Taylor and A. E. Green, “Mechanism of the production of small eddies from large ones,” *Proceedings of the Royal Society of London. Series A - Mathematical and Physical Sciences*, vol. 158, no. 895, pp. 499–521, 1937.
- [158] L. Diosady and S. Murman, “Case 3.3: Taylor green vortex evolution,” in *Case Summary for 3rd International Workshop on Higher-Order CFD Methods*, 2015.
- [159] A. D. Beck and G. J. Gassner, “Numerical Simulation of the Taylor-Green Vortex at $Re=1600$ with the Discontinuous Galerkin Spectral Element Method

- for well-resolved and underresolved scenarios,” *50th AIAA Aerospace Sciences Meeting*, pp. 1–4, 2012.
- [160] J. R. DeBonis, “Solutions of the Taylor-Green vortex problem using high-resolution explicit finite difference methods,” in *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition 2013*, no. January, pp. 1–20, 2013.
- [161] T. L. Bergman, T. L. Bergman, F. P. Incropera, D. P. Dewitt, and A. S. Lavine, *Fundamentals of heat and mass transfer*. John Wiley & Sons, 2011.
- [162] D. Ganji, H. Babazadeh, F. Noori, M. Pirouz, and M. Janipour, “An application of homotopy perturbation method for non-linear blasius equation to boundary layer flow over a flat plate,” *International Journal of Nonlinear Science*, vol. 7, no. 4, pp. 399–404, 2009.
- [163] M. Szoke, N. Nurani Hari, W. J. Devenport, S. A. Glegg, and T.-R. Teschner, “Flow Field Analysis Around Pressure Shielding Structures,” pp. 1–25, 2021.
- [164] H. Liu, C. P. Ellington, K. Kawachi, C. Van Den Berg, and A. P. Willmott, “A computational fluid dynamic study of hawkmoth hovering,” *The journal of experimental biology*, vol. 201, no. 4, pp. 461–477, 1998.
- [165] F. X. Caradonna and C. Tung, “Experimental and Analytical Studies of a Model Helicopter Rotor in Hover.,” *Vertica*, vol. 5, no. 2, pp. 149–161, 1981.
- [166] F. Ricci, P. A. Silva, P. Tsoutsanis, and A. F. Antoniadis, “Hovering rotor solutions by high-order methods on unstructured grids,” *Aerospace Science and Technology*, vol. 97, p. 105648, 2020.
- [167] J. Li, J. Gu, Z. Huang, and J. Wen, “Application research of improved YOLO V3 algorithm in PCB electronic component detection,” *Applied Sciences (Switzerland)*, vol. 9, no. 18, 2019.

- [168] S. Osher, “Convergence of generalized muscl schemes,” *SIAM Journal on Numerical Analysis*, vol. 22, no. 5, pp. 947–961, 1985.
- [169] C.-W. Shu, “High-order finite difference and finite volume weno schemes and discontinuous galerkin methods for cfd,” *International Journal of Computational Fluid Dynamics*, vol. 17, no. 2, pp. 107–118, 2003.