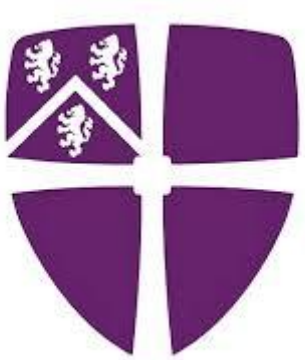
# *Enhanced Flow Visualization Using Image Processing and Deep Learning Techniques*

XINGYU LIU

# Enhanced Flow Visualization Using Image Processing and Deep Learning Techniques



Xingyu Liu

Department of Engineering
Durham University
United Kingdom
November 2024

*Dedicated to*

Zhichao Mu

A research thesis submitted to Durham University in partial fulfilment as a requirement for the degree of

Doctor of Philosophy

in

ENGINEERING

Department of Engineering

Durham University

United Kingdom

# Declaration

The work in this thesis is based on research carried out in the Department of Engineering at the University of Durham. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

DEPARTMENT OF ENGINEERING

DURHAM UNIVERSITY

# Acknowledgements

No man is an island. In this section the author wants to say thank you to everyone that have helped him in the production of this thesis.

Firstly, I would like to thank Professor Grant Ingram for his professional insights, patience, and support for my research. I am also grateful to Professor David Sims-Williams for his helpful suggestions regarding my experimental setup. His extensive experience in engineering and hands-on skills were invaluable to me. I would like to thank Professor Toby Breckon and his research team for answering many questions about algorithms. Without their help, I might still be debugging my code.

I would also like to thank Gary Parker from the Thermo Fluid Lab for his technical support and for helping me find useful materials. The Electrical and Electronic Workshop was a great help, and I would like to specifically mention Colin Wintrip and Paul Davidson.

I am grateful to Pedro Alberto Martínez-Castro and Tarek Abdelsalam, who, as senior members of the lab, provided me with many inspiring ideas for my project. Louise Gascoigne was always patient in answering my questions about university and department regulations, and I greatly appreciate her assistance. I also want to thank Dr. Majid Bastankhah and Dr. Peter Matthews for offering many valuable suggestions for my research.

My studies were funded by the China Scholarship Council (CSC), and I would like to express my thanks to them.

I owe special thanks to my wife, Zhichao Mu, for her companionship and support during the most difficult times. She is also an academically accomplished person, with notable achievements in the field of supercapacitors. I am deeply grateful to my parents, Yongjun Liu and Xueying Yuan, who have always unconditionally supported every decision I have made, and who have helped shape who I am today.

Lastly, I would like to thank all the friends I met in Durham for their support and friendship.

艰难困苦，玉汝于成

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| AP | Average Precision |
| AR | Augmented Reality |
| BDCN | Boundary Detection Convolutional Network |
| BEF | Boundary Exstrophy Flux |
| BIPED | Barcelona Images for Perceptual Edge Detection |
| BSDS | Berkeley Segmentation Dataset |
| CCD | Charge-coupled Device |
| CFD | Computational Fluid Dynamics |
| CID | Contemporary Image Dataset |
| CNN | Convolutional Neural Networks |
| CPS | Static Pressure Coefficient |
| CPT | Total Pressure Coefficient |
| CPU | Central Processing Unit |
| Dexi | Dense Extreme Inception Network |
| DNN | Deep Neural Network |
| DPIV | Digital Particle Image Velocimetry |
| FID | Fréchet Inception Distance |
| FIK | Fukagata-Iwamoto-Kasagi |
| FSC | First Skip Connection |
| GAN | Generative Adversarial Networks |
| GLOF | Global Luminescent Oil-Film |
| GPPS | Global Power and Propulsion Society |

| | |
|---|---|
| GPU | Graphics Processing Unit |
| GT | Ground Truth |
| HED | Holistically-Nested Edge Detection |
| ICEM | Integrated Computational Environment for Multiphysics |
| IS | Inception Score |
| LCC | Liquid Crystal Coating |
| LDC | Lightweight Dense Convolutional Neural Network |
| LDM | Latent Diffusion Models |
| MAE | Mean Absolute Error |
| MDBD | Multicue Dataset for Boundary Detection |
| MLP | Multilayer Perceptron Neural Network |
| MSE | Mean Squared Error |
| NCC | NVIDIA CUDA Centre |
| NLP | Natural Language Processing |
| OIS | Optimal Threshold for Each Image |
| PIV | Particle Image Velocimetry |
| PSNR | Peak Signal-to-Noise Ratio |
| RAM | Random Access Memory |
| RANS | Reynolds-averaged Navier-Stokes |
| RCF | Richer Convolutional Features |
| RGB | Red Green Blue |
| RHS | Right-Hand-Hide |
| RLF | Radial-Logarithmic Flter |

| | |
|---|---|
| ROI | Rigion of Interest |
| SFV | Surface Flow Visualization |
| SGD | Stochastic Gradient Descent |
| SOFV | Surface Oil Flow Visualization |
| SSC | Second Skip Connection |
| SSIM | Structural Similarity Index |
| SST | Shear Stress Transport |
| USnet | Upsampling Network |
| UV | Ultraviolet |

## Nomenclature

| | |
|---|---|
| $F_x$ | Frictional force (N) |
| $b$ | Width of the plate (m) |
| $\rho$ | Fluid density (kg/m$^3$) |
| $U$ | Flow velocity at the first cross-section (m/s) |
| $u$ | Flow velocity at a subsequent cross-section (m/s) |
| $\tau_0$ | Shear stress (N/m) |
| $\partial B$ | Surface properties |
| $\mu$ | Fluid's dynamic viscosity (N·s/m$^2$) |
| $T_{\partial B}$ | Surface temperature (℃) |
| $\phi_{\partial B}$ | Surface scalar concentration |
| $q$ | Measurable quantity |
| $\bar{\tau}_{ij}$ | Mean viscous stress (N/m) |

| | |
|---|---|
| $k$ | Turbulent kinetic energy (J/kg) |
| $\Omega_{ij}$ | Kronecker delta symbol |
| $v_t$ | Turbulent viscosity of the flow (N·s/m$^2$) |
| $\beta$ | Quantum efficiency constant |
| $I$ | Luminescent intensity (cd) |
| $I_{ref}$ | Reference intensity |
| $\tau_w$ | Wall shear-stress (N/m) |
| $h$ | Oil film thickness (m) |
| $t$ | Time (s) |
| $p_w$ | Wall static pressure on the oil film (N/m$^2$) |
| $g$ | Gravity (m/s$^2$) |
| $\tau'$ | Equivalent skin friction (N/m) |
| $\mu_i$ | Mean intensity of image $i$ (cd) |
| $\sigma_x^2, \sigma_y^2$ | Intensity variances |
| $\sigma_{xy}$ | Covariance between the images |
| $\sigma_S$ | Spatial distance influence constant |
| $\sigma_T$ | Intensity differences influence constant |
| $\theta$ | Sigmoid function |
| $y$ | Ground truth edge map |
| $\hat{y}_n$ | Predicted edge map |
| $\lambda_n$ | Hyperparameters |
| $w$ | Loss function weight |

| | |
|---|---|
| $l$ | Loss value |
| $u_{t,i}$ | x-direction velocity components at point i on the image at time t |
| $v_{t,i}$ | y-direction velocity components at point i on the image at time t |
| $\mu_{real}$ | Mean activation values for real images |
| $\mu_{gen}$ | Mean activation values for generated images |
| $C$ | Sample covariance matric |
| $MAX_I$ | Maximum possible pixel value |

# Abstract

This thesis explored the application of machine learning techniques to enhance the efficiency and accuracy of surface flow visualization (SFV). The SFV technique has been widely used in fluid dynamics research to provide qualitative information. In order to extract some quantitative information from SFV images, specific algorithms must be developed. While machine learning algorithms are a type of algorithm that automatically analyses data to obtain patterns and uses these patterns to make prediction.

The core innovation of this thesis lies in the use of Convolutional Neural Networks (CNNs) to automate streamline detection, demonstrating superior reliability and accuracy compared to traditional methods like Sobel edge detection. Building on this, the thesis proposes a predictive neural network model capable of estimating flow fields from SFV images. To train this model, a comprehensive dataset was constructed using both experimental data and synthetically generated images, significantly improving the model's robustness and generalization ability.

The Global Luminescent Oil-Film (GLOF) method has been conducted to determine if the surface friction fields could be extracted from the videos. The results from this thesis indicate that this is not the case. while GLOF has promising applications, substantial refinement is required to achieve reliable results in complex flow cases.

Therefore, the SFV images was labelled manually using chaincode method. And Generative Adversarial Networks (GANs) were applied to generate synthetic flow field images, which supplemented the experimental data and improved model training. Additionally, a simplified approach combined synthetic and experimental data to train predictive models like U-Net, improving the accuracy of flow field estimation.

this work delivers a practical framework that enables researchers to input a single SFV image and obtain a preliminary prediction of streamlines and flow fields. Another useful contribution is the creation of a unique dataset, hosted on GitHub, which combines experimental and synthetic data, enabling the training of flow visualization algorithms.

# Chapter 1 Introduction

Surface flow visualization (SFV), specifically surface oil flow visualization, is an experimental technique that involves coating the surface with a mixture of oils and dyes before applying the flow to the subject. While investigating the surface flow, the surface topology must be analysed to determine the flow field near the surface. Therefore, numerous flow visualization and image processing techniques have been proposed, showing good performance. Nonetheless, their accuracy is largely contingent on human expertise, and the overall processing cost is elevated because they necessitate the trial-and-error optimization of thresholding parameters, such as intensity thresholds, contrast levels. As a result, these parameters are not universally applicable across all experimental conditions.

Recent advances in machine learning offer the potential to automate and enhance the accuracy of streak detection and flow field estimation from SFV images. This thesis explored the application of neural networks to address some of the limitations in existing methods and aimed to contribute to the broader field of flow visualization by introducing data-driven approaches.

While much of the data generation for this research was conducted using the Durham Cascade and Plint Wind Tunnel facilities, this thesis focused on leveraging machine learning techniques for image analysis and flow prediction. Abdelsalam et al. (2017) initially demonstrated the potential of topological methods for detecting prominent streamlines in flow visualization images. Building on this foundation, this work introduced the use of GANs to bridge the gap between experimental and synthetic data.

To the best of the author's knowledge, this thesis is the first attempt to apply GANs to the quantitative analysis of statistical images from flow visualization. Previous efforts in this field have predominantly relied on either topological methods or manual annotations, which, while insightful, offered only indirect insights into the underlying flow phenomena. Moreover, quantitative analysis has often been constrained by the availability of high-quality images or video data, which could be difficult to obtain without disrupting the flow.

The following key activities have been undertaken as part of this research:

- Development and validation of machine learning models for automated streak detection in SFV images.

- Application of GLOF techniques to assess their effectiveness in extracting directional information from flow visualization.

- Generation of synthetic flow visualization images using CFD data and GAN models to augment the training dataset.

- Training and evaluation of a U-Net model for predicting flow fields directly from flow visualization images, using both synthetic and experimental data.

The thesis is organized into several chapters:

1. Introduction – this section.
2. Literature Review – this discusses the importance of skin friction, previous work in flow visualization, and machine learning applications in fluid dynamics.
3. Methodologies – details the experimental setup, data collection, and machine learning models used in the study.
4. Extracting Streamline Data from Surface Flow Images Using CNNs– this presents the development and evaluation of the CNN-based streak detection method.
5. Global Luminescent Oil-Film (GLOF) method– this explores the application of GLOF techniques and their performance.
6. Synthetic Data Generation and GAN Training – this describes the use of CFD data and GANs to create synthetic flow visualization images. The U-Net model's performance in predicting flow fields from images is also evaluated.
7. Conclusions.

The generation of synthetic images and all CFD simulations referenced in this thesis were performed using data from the Durham Cascade and Plint Wind Tunnel. The model development process leveraged MATLAB and PyTorch (Paszke et al., 2019) for implementation, with the evaluation of GLOF techniques drawing on the work of Liu (2013). The GAN architecture was adapted based on the framework proposed by Zhu et al. (2017). However, the analysis and processing of data, as well as the debugging and evaluation of models, were independently carried out by the author.

An abridged version of Chapter 4 was published as GPPS-TC-2024-0100, at a peer reviewed conference. The version presented in this thesis builds upon and extends the prior

work, incorporating additional analyses, discussions and refinements to offer a more comprehensive perspective on the subject matter.

# Chapter 2 Literature Review

This chapter reviews relevant literature, which reveals the significance of flow visualization and the promising potential of machine learning. It explores the connection between skin friction in fluid mechanics and flow visualization, demonstrating how these concepts intersect in the context of this thesis. Additionally, the chapter discusses the role of Generative Adversarial Networks in this field. To provide a comprehensive foundation, key concepts and basic knowledge related to these topics are also covered in this chapter.

## 2.1 Basic concept of surface flow

The term "turbomachinery" has its roots in Latin and was popularized by Claude Burdin in 1822, defining it as "that which spins". It encompassed a broad range of machines involved in either extracting or delivering energy to a flowing fluid through the rotation of one or more blade rows (Lakshminarayana, 1995). In *Fluid Mechanics and Thermodynamics of Turbomachinery*, Dixon and Hall (2013) defined turbomachines as devices where energy is exchanged with a continuously flowing fluid through the dynamic interaction of one or more rows of moving blades. This classification encompasses machines that either transfer energy to the fluid or extract energy from it. When these machines impart energy to the fluid, they are classified as diffusers, compressors, or pumps. Conversely, when they extract energy from the fluid, they are referred to as expanders or turbines. Specifically, the term "turbine" commonly refers to the complete power system composed of components such as the diffuser, expander, nozzle, etc.(Ingram, 2009). Gas turbine systems have applications in various fields, such as turboprops for medium-range aircraft, turboshafts for helicopters and marine vessels, and gas turbine power plants. Although these systems differ in their specific configurations, the aerodynamic principles of the turbine remain consistent.

In turbomachinery fluid mechanics research, observation and experimentation play crucial roles, facilitating intuitive judgments and quantitative calculations. Moreover, fluid flow is often challenging to directly observe, necessitating additional methods for visualization, commonly referred to as flow visualization.

## 2.2 Skin friction

Skin friction, represented by the wall shear stress $\tau$, is a fundamental quantity in fluid dynamics that relates to the shear stress experienced by a surface in viscous flows. It is crucial

for understanding near-wall structures in complex fluid systems. Skin friction fields exhibit topological features like critical points and separation lines, traditionally visualized by surface oil flow visualization.

These topological features are essential for identifying regions of flow separation, reattachment, and vortex formation. Those regions usually play critical roles in predicting aerodynamic performance and optimizing designs in engineering applications. The detailed study of skin friction patterns enables engineers and researchers to uncover the underlying flow mechanisms that drive drag, heat transfer, and even acoustic noise in turbulent flows (Zanoun et al., 2021).

### 2.2.1 Skin friction and momentum

The understanding of skin friction in recent days incorporates both fluid mechanics and the behaviour of solid boundaries interacting with a fluid. This relationship primarily involves how the boundary, when accompanied by parallel flow that allows the formation of a "boundary layer", affects momentum exchange between the fluid and the surface. At high Reynolds number the boundary layer is relatively thin and its thickness, denoted by $\boldsymbol{\delta}$, determined by the physics of the flow, plays a crucial role in skin friction dynamics.

In examining skin friction, the focus shifts from direct frictional analysis to studying the velocity distribution within the boundary layer. This approach arises from the realization that skin friction is closely linked to how velocity changes across this thin layer. The measurement of such velocity distributions is more often performed directly or inferred through established theoretical frameworks.

For example, as Figure 2.1 shows, consider a plate of width $\boldsymbol{b}$ and introduce a coordinate system where the flow is aligned parallel to the plate, and the origin is at the leading edge. The force of friction $\boldsymbol{F}_x$ exerted on the plate between the leading edge and a point $\boldsymbol{x}$ along the plate can be calculated following the theoretical approach established by Von Karman (1934). According to this approach, the frictional force is expressed as:

$$F_x = b \int_0^\infty \rho u(U-u)\, dy \tag{2.1}$$

Here, $\rho$ represents the fluid density, $U$ the flow velocity at the first cross-section, and $u$ the velocity at a subsequent cross-section $\Delta S$ away. As the fluid passes through an element $dS$ of the second cross section, the mass flow per unit time can be expressed as $\rho u dS$. This fluid

mass originally had the momentum $U$ per unit mass at the first cross section and reaches the momentum $u$ per unit mass at the second cross section. The total loss of momentum of the fluid, considering an element $dS$, amounts to $\rho u(U-u)dS$ and the total loss of momentum of the fluid in unit time to $\int_S \rho u(U-u)dS$, the integral being taken over the cross section downstream. This amount is equal to the total frictional force acting on the portion of the plate extending from the leading edge to the distance $\boldsymbol{x}$.

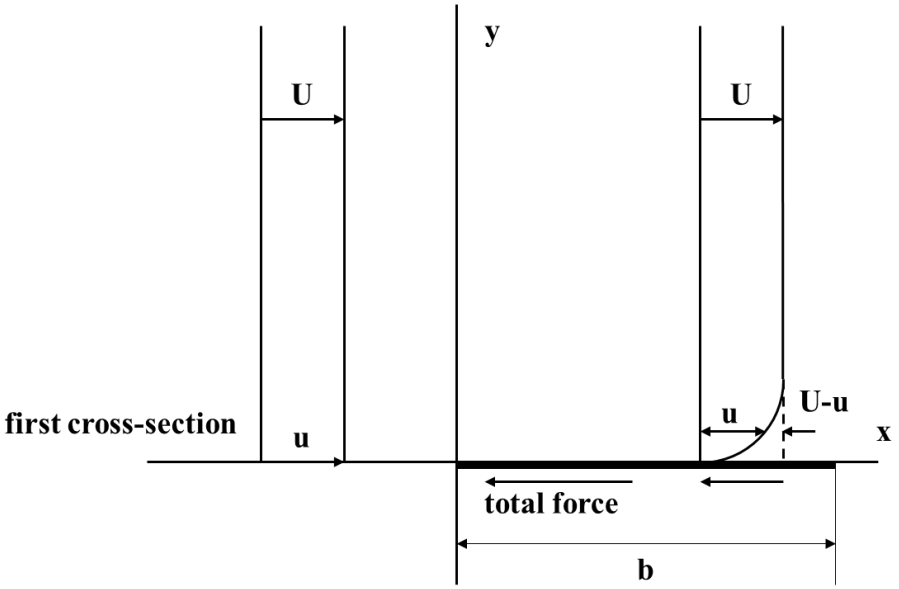



**Figure 2.1 Boundary layer and skin friction**

Additionally, the calculation of the shear stress $\tau_0$ exerted by the fluid on the plate is important for understanding skin friction. This stress is defined by the local friction per unit area and can be expressed mathematically as:

$$\tau_0 = \frac{1}{b}\frac{dF}{dx} = \frac{d}{dx}\left[\int_0^\infty \rho u(U-u)\,dy\right] \tag{2.2}$$

In practical situations, it is generally accurate enough to limit the integral from y = 0 to a finite value y = δ. The impact of friction is confined to a narrow region along the plate, from 0 to δ, which can be called "boundary layer".

The discussion above substitutes the issue of skin friction with the issue of velocity distribution within the boundary layer. In fact, the friction can be measured either directly or through assessing the velocity distribution throughout the boundary layer, as emphasized in Von Karman (1934)'s work.

### 2.2.2 Impact of skin friction on fluid dynamics performance

Skin friction is essential in analysing aerodynamic performance, particularly because the surface friction vector field exposes intricate flow topologies in three-dimensional separated flows and turbulent boundary layers. The near-wall flow structures in both separated and turbulent flows are determined by the surface friction field and the surface pressure field. Notably, there exists a definitive coupling relationship between surface friction and other critical surface properties, such as surface pressure (Bewley and Protas, 2004), surface temperature (Xu et al., 2022), and surface scalar concentration (Liu et al., 2014). From a theoretical perspective, this coupled structure can be analysed through mathematical transformations to identify and understand coherent near-wall structures in complex flows.

In the past few decades, plenty of work has been devoted to express the skin friction as the contributions from the flow statistics inside the flow domain in wall-bounded turbulence (Fukagata et al., 2002, Gomez et al., 2009, Renard and Deck, 2016, Li et al., 2019b). Fukagata et al. (2002) derived a straightforward relationship between the skin-friction coefficient and the distinct contributions from the mean and statistical turbulent quantities across the wall layer, and this relationship was referred to as the Fukagata–Iwamoto–Kasagi (FIK) identity. The FIK identity indicates that four dynamical effects, the laminar, turbulent, inhomogeneous and transient components, will contribute to the wall skin friction based on the streamwise momentum budget. Furthermore, they used the above relationship to analyse the drag modification by the uniform wall blowing and suction as well as by opposition control. The FIK identity was extended by Gomez et al. (2009) to compressible wall-bounded turbulence and was used to study the compressibility effects on the generation of skin friction. They showed that the skin friction can be ascribed to the contributions of four physical processes, i.e. the laminar, turbulent, compressible and a fourth coming from the interaction between turbulence and compressibility. They found that the main contribution to the skin friction was from the turbulent term in compressible turbulent channel flows. Inspired by the idea of the FIK decomposition, Mehdi and White (2011) presented a modified FIK identity to evaluate skin-friction coefficient when measurements at most streamwise locations are unavailable. In such case, the streamwise gradients were substituted by the total stress gradients in the wall-normal direction. Mehdi et al. (2014) found that the FIK decomposition was based on a mathematically exact relation, and the integration could be applied on an arbitrary height. This modified form of the FIK identity could be used for flows with ill-defined outer boundary conditions, or when the measurement grid cannot cover the whole boundary-layer thickness.

Similarly, Xia et al. (2021) described skin friction using the average velocity and the Reynolds shear stress in any region that is perpendicular to the wall.

Liu (2019) provided a relationship between surface friction $\tau$ and the surface pressure gradient $\nabla_{\partial B} p_{\partial B}$ in incompressible flows, which can be written as:

$$\boldsymbol{\tau} \cdot \nabla_{\partial B} p_{\partial B} = \mu f_{\Omega} \tag{2.3}$$

where $f_{\Omega}$ is the boundary exstrophy flux (BEF) plus the curvature-induced contribution. Here, $\nabla_{\partial B}$ represents the gradient operator on the surface, $\mu$ is the fluid's dynamic viscosity, and $\partial B$ indicates the properties of the surface. The derivation of this relationship is based on the Taylor series expansion of the near-surface velocity field in the Navier-Stokes equations and the geometric properties of vorticity lines on the surface. This relationship demonstrates the intrinsic coupling between surface friction and surface pressure through BEF.

Similarly, to establish the relationship between surface friction, surface temperature, and heat flux, the energy equation could be reformulated, leading to the relationship $\boldsymbol{\tau} \cdot \nabla_{\partial B} T_{\partial B} = \mu f_{\mathrm{Q}}$, where $T_{\partial B}$is the surface temperature, and $f_{\mathrm{Q}}$ relates to the heat flux, the third-order normal derivative of surface temperature, curvature terms, and viscous dissipation terms. In a certain case, this relationship represents a general differential form of the Reynolds analogy between surface friction and boundary heat flux. Because of the similarity between mass and heat transfer processes, the relationship between surface friction and surface scalar concentration $\phi_{\partial B}$, $\boldsymbol{\tau} \cdot \nabla_{\partial B} \phi_{\partial B} = \mu f_{\mathrm{M}}$, has been derived from the mass transfer equation, where $f_{\mathrm{M}}$ is related to mass flux, the third-order normal derivative of surface scalar concentration, curvature terms, and source terms.

These relationships take the general form $\boldsymbol{\tau} \cdot \nabla_{\partial B} q = f$, where $q$ is a measurable quantity, $f$ can be measured or modelled, and $\nabla_{\partial B}$ is the surface gradient operator (or projected onto the image plane). Mathematically, determining the vector field $\boldsymbol{\tau}$ from given fields of $g$ and $f$ is an inverse problem, akin to the optical flow problem in computer vision (Liu, 2019). Therefore, variational methods can be employed to solve this problem in the image plane, to extract the surface friction field from surface flow visualization.

### 2.2.3 Skin friction and measurement techniques

To obtain information on surface skin friction, appropriate measurement techniques are required. Studies have been conducted to assess the relationship between skin friction and flow

pattern. Most surface friction measurement technologies are indirect and localized methods, typically using a single sensor element to provide the magnitude of skin friction at a specific location (Liu, 2013). For instance, Dhawan (1953)'s pioneering work introduced the direct measurement of surface skin friction using a moving wall element, a technique that circumvented the common errors associated with traditional indirect methods. This "direct" method allowed for accurate measurement of wall shear forces without relying on prior knowledge of the flow state. The primary advantage of Dhawan's method was its versatility, being applicable across laminar, turbulent, and complex three-dimensional flows. However, the early devices faced challenges such as slow response times of the floating element and susceptibility to environmental factors, issues which were later addressed in Winter (1979)'s research.

Winter's design incorporated a feedback system that allowed the floating head to return to its unloaded position quickly after shear force application, enhancing measurement accuracy. Despite these improvements, Winter's method was not without flaws. The slow response time of the sensor (on the order of seconds) could compromise accuracy in rapidly changing flow conditions, and the mechanical complexity of the sensor made it potentially vulnerable in high-temperature or high-speed environments.

Waltrup and Schetz (1973) further advanced the field by focusing on wall skin friction measurements in complex flow conditions, particularly in supersonic and high-temperature flows. Their designs included liquid-filled sensor housings and water-cooling systems, which stabilized sensor performance under extreme conditions. However, these methods also encountered challenges, such as ensuring the physical integrity of the sensors under high-temperature and high-pressure conditions. Additionally, material selection for the floating head and thermal protection remained critical concerns. While some of these technical issues were mitigated, they continue to be key areas for future research and development.

Most traditional skin-friction measurement methods are limited by their local approach and sensitivity to environmental conditions. This makes them slow and less accurate in extreme flow situations. In contrast, global skin-friction measurement techniques offer a more complete approach by using image-based methods to capture both the magnitude and direction of the entire friction field. Reda et al. (1997) developed a liquid crystal coating (LCC) method to measure surface shear stress vector distributions on planar surfaces. This method used colour changes in liquid crystals caused by shear stress to visualize and measure shear forces across

the whole surface. It was precise, fast, and could handle complex flow situations. However, the method was limited by the need for specific optical access, the complexity and cost of the imaging and calibration setup, and the requirement for accurate colour-shear calibration. Similarly, Fonov et al. (2006) developed surface-stress-sensitive polymer film ($S^3F$) method. $S^3F$ could provide a surface-stress map by measuring the elastic deformation of a polymer film. Its results would be affected by properties of the polymer film. Micro-pillar arrays, on the other hand, used the deflection of tiny, flexible pillars to directly measure shear stress with high accuracy and temporal resolution (Brücker et al., 2007). However, they could have some resonant frequency problems and require careful calibration and alignment to maintain consistent results.

Liu et al. (2008) chose to calculate the skin friction field by observing the changes in the oil film surface over time, called the Global Luminescent Oil-Film Skin-Friction Meter. This calculation method was derived from the thin-oil-film equation, establishing a relationship between the thin oil-film equation and the normalized luminescent intensity in the image plane.

The relation between the wall shear-stress vectors $\tau_w$ and the oil film thickness $h$ can be obtained from the thin-oil-film equation(Brown and Naughton, 1999, Naughton and Sheplak, 2002), which is written as follows:

$$\frac{\partial h}{\partial t}+\frac{\partial}{\partial X_i}\left[\frac{h^2\tau_i}{2\mu}-\left(\frac{\partial p_w}{\partial X_i}-\rho g_i\right)\frac{h^3}{3\mu}\right]=0, \qquad (i=1,2) \tag{2.4}$$

where $\mu$ is the oil dynamic viscosity, $p_w$ is the wall static pressure on the oil film, $\rho$ is the density of oil, $t$ for time and $g$ is the gravity vector. The thickness of the oil film, $h$, is considered to be proportional to the luminescent intensity, $I$. This relationship can be expressed as $h=\kappa I$, where the parameter $\kappa$ is a constant(Tran and Chen, 2021).

This equation has the same form as the physics-based optical flow equation, which will be further discussed in section 3.2.2. Then, the variational method was used to determine the optical flow, and the Euler–Lagrange equations were solved to obtain the relative skin friction field. Compared to previous global skin-friction measurement techniques, GLOF's experimental setup was simpler and could quickly obtain a global distribution of skin friction. However, it relied on analysing experimental videos, which means it was difficult to apply to non-invasive experiments where only flow visualization images were available after the experiment had been completed.

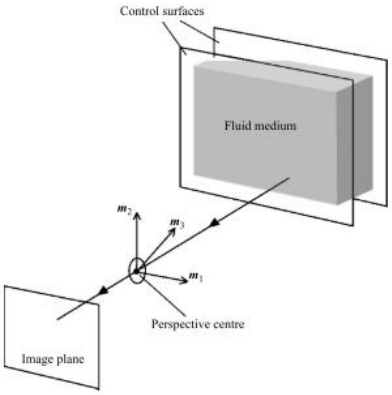



**Figure 2.2 Projection from fluid flow onto the image plane from Liu and Shen (2008).**

In summary, current skin friction measurement techniques are limited by their localized nature, sensitivity to environmental conditions, and slow response times in dynamic or extreme flows. Global methods, while more comprehensive, still rely on specific setups and invasive experiment method. There is a need for a more efficient method to determine the flow field using surface flow visualization images. Machine learning offers a promising solution by analysing these images to predict skin friction and flow patterns in real-time, leveraging large datasets and advanced algorithms to provide a non-invasive, flexible, and efficient approach for various conditions.

## 2.3 Computational Fluid Dynamics (CFD)

### 2.3.1 Numerical method

The unsteady 3D Reynolds-averaged Navier-Stokes (RANS) equations employed in this study can be obtained by taking ensemble average of instantaneous mass and momentum conservation equations for incompressible and isothermal flows, as shown in the following:

$$\frac{\partial \bar{u}_i}{\partial x_i} = 0 \tag{2.5}$$

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\bar{u}_i \bar{u}_j + \overline{u_i' u_j'}\right) = -\frac{1}{\rho} \cdot \frac{\partial \bar{p}}{\partial x_i} + \frac{\partial \bar{\tau}_{ij}}{\partial x_j} \tag{2.6}$$

$$\bar{\tau}_{ij} = 2vS_{ij} = v\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) \tag{2.7}$$

where the subscripts $i$ and $j$ indicate the $i^{th}$ and $j^{th}$ components of the Cartesian coordinate respectively, represent the time-averaged velocity and pressure fields respectively, $t$ is the time, $\rho$ is the density of the fluid (defined as a constant in this study, air), $\bar{\tau}_{ij}$ is the mean

viscous stress tensor, is the mean strain-rate tensor, and $v$ is the kinematic viscosity of the fluid (i.e. $v$=1.8×10$^{-5}$). The Reynolds number (Re) is a dimensionless quantity that characterizes the ratio of inertial forces to viscous forces in a fluid flow. It is a key parameter in determining whether the flow is laminar or turbulent. The Reynolds number is defined as: $Re = UL/v$, where U is the characteristic velocity (taken as 20 m/s based on the free-stream flow) and L is the characteristic length scale (chosen as 0.026 m corresponding to the model's diameter). Substituting these values yields Re = 3×10$^4$. The Reynolds stress tensor term (i.e. $\bar{T}_{ij} = \overline{u_i' u_j'}$) can be obtained by using the Boussinesq eddy-viscosity hypothesis:

$$\bar{T}_{ij} = \overline{u_i' u_j'} = \frac{2}{3} k \delta_{ij} - v_t \left( \frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right) \tag{2.8}$$

$$k = \frac{1}{2} \overline{u_i' u_j'} = \frac{1}{2} \left( \overline{u_x' u_x'} + \overline{u_y' u_y'} + \overline{u_z' u_z'} \right) \tag{2.9}$$

where $k$ is the turbulent kinetic energy, $\delta_{ij}$ is the Kronecker delta symbol and $v_t$ represents the turbulent viscosity of the flow. Different turbulence models can be used to determine the turbulent viscosity $v_t$ , including the standard $k$-epsilon ($k$–$\varepsilon$) model (Launder and Spalding, 1983), the renormalization group $k$-epsilon (RNG $k$–$\varepsilon$) model (Yakhot et al., 1992), the realizable $k$–$\varepsilon$ turbulence model (Shih et al., 1995), Lien (1996)'s cubic non-linear low-Reynolds $k$–$\varepsilon$ model, Launder and Sharma low-Reynolds $k$–$\varepsilon$ turbulence model (Launder and Sharma, 1974), Shih (1993)'s quadratic non-linear $k$–$\varepsilon$ turbulence model, the standard high Reynolds-number *k-omega* ($k$–$\omega$) model (Wilcox, 1998)

To accurately capture the transition effects in this study, the 4-equation SST transition model (Menter, 1994, Menter et al., 2003) was employed, which extends the $k$–$\omega$ formulation by introducing additional transport equations for transition onset. This model has been validated in previous research (Zhang, 2017), and is well-suited for predicting the flow characteristics in our setup.

### 2.3.2 Grid

In the section 2.3.1 (p.12), the numerical method used in CFD were briefly discussed. With advancements in computer technology and numerical methods, many complex engineering problems could now be solved using numerical calculations based on discretization, yielding solutions that met engineering requirements. Discretization involves replacing a continuous space with a set of discrete points. The process is to divide the area being calculated

into several non-overlapping subdomains, determine the node locations for each subdomain, and define the control volume represented by each node. A node is the geometric location of the unknown physical quantity to be solved, the control volume, and the smallest geometric unit for applying the governing equations or conservation laws. Typically, nodes are considered as representatives of control volumes. The control volume and subdomains do not always coincide. At the beginning of the discretization process, small domains are defined by a series of straight or curved lines corresponding to the coordinate axes, and these domains are called subdomains. The mesh is the foundation of discretization, and the mesh nodes store the discretized physical quantities. Although the mathematical theory behind grid generation is relatively classical, its practical application is still a developing field (Thompson et al., 1985). In the CFD computational process, the quality and suitability of the grid directly affect the reliability of the final results (Anderson and Wendt, 1995).

### 2.3.3 Applications and Validation

In terms of applications, CFD has achieved significant growth in recent years, and its application range has expanded from traditional engineering problems to emerging fields. For example, in the field of renewable energy, CFD helps optimize the aerodynamic performance of wind turbines and the thermal management of photovoltaic systems, which could provide innovative solutions for wind and solar energy design; In the field of environmental protection, CFD is widely used to simulate pollutant diffusion, flooding, and urban microclimate conditions, which promotes initiatives for sustainable development and environmental policies; In the field of biomedicine, CFD simulation provides useful insights into hemodynamic, respiratory mechanics, and drug delivery, which solves complex medical challenges.

Recently, machine learning has been introduced in CFD. It could improve the accuracy and efficiency of CFD simulations and create new opportunities for modelling and predicting complex flow situations. Panchigar et al. (2022) reviewed machine learning applications in CFD, emphasizing its potential in predicting complex flow fields. Shourangiz-Haghighi et al. (2020) showed how CFD enhanced energy efficiency and advanced wind turbine performance. Machine learning has also been applied to predict flow processes such as membranes, which helps determine the distribution of fluid pressure and velocity in membranes with different shapes. Kamrava et al. (2021) used their model to predict the flow and transport properties of other types of porous materials. Machine learning model also helped them design membranes that are customized for specific applications. Raissi et al. (2020) attempted to encode the

Navier-Stokes equations into neural networks, offering a novel approach to extracting velocity and pressure fields from certain SFV images.

It can be seen that current research combining machine learning and CFD mainly focused on Physics-Informed Neural Networks and Accelerating CFD calculations. While there were less works about using machine learning to get quantitative data from SFV images, as this task involves complex challenges such as extracting accurate flow features from noisy visual data, dealing with limited labelled datasets, and ensuring the physical consistency of the reconstructed fields. However, once the flow field could be stably obtained from the SFV image, it means that there is a cheap way to understand the surface, and for CFD, there is also a reliable way to be initialized.

## 2.4 Flow visualization

The role of flow visualization in experimental fluid-mechanical research has been thoroughly examined. Many reviews cover this field or focus on specific applications (Merzkirch, 1987). Flow visualization could provide precise information about flow characteristics in both spatial and temporal dimensions, which enabled researchers to analyse and verify flow behaviour (Etminan et al., 2022).

As Lu (2010) emphasized in his review, establishing a coherent link between surface patterns and the flow field required a solid understanding of flow theory, which is missing in many publications. Flow visualization techniques, including methods such as surface oil flow visualization (SOFV), tuft analysis, Particle Image Velocimetry (PIV), and others, are employed to study fluid dynamics. SOFV, as an experimental method, needs to coat the surface of interest with an oil and dye mixture before subjecting it to a flow. In areas with high shear stress, the oil/dye mixture will be removed, while in areas with low shear stress, it persists or accumulates. The resulting pattern is then analysed to understand the structure near the examined surface. Despite the simplicity, intuitive outcomes, and minimal experimental equipment requirements of SOFV, current research predominantly utilizes it as an indirect observational tool, emphasizing qualitative insights over quantitative analysis.

Zierke et al. (1994) used an oil-paint method to obtain patterns of skin-friction lines on the rotor blades surfaces of a high-Reynolds-number centrifugal pump. This post-processing flow visualization technique revealed that the existence of a trailing-edge separation vortex, which migrated radially upward along the trailing edge and then turned in the circumferential

direction near the casing, moving in the opposite direction of blade rotation. The resulting patterns qualitatively confirmed the unsteadiness of the vortex and helped in establishing the trajectory of the tip leakage vortex core.

The abilities of the thin oil film technique with its application in a short duration wind tunnel can avoid the intrusiveness of the measuring device and the corresponding flow disturbance while exploring the skin friction as emphasized by Schülein (2004). For instance, Aunapu et al. (2000) compared the flow visualization results from two different established methods utilizing ink dots and solvent, and oil and black powder. Both techniques successfully captured key flow features, including the horseshoe vortex, its migration across the passage, endwall cross flow, and the saddle point. However, the ink dot technique proved more effective, providing a clearer and more detailed depiction of the shear stress patterns, along with a permanent record for post-experiment analysis. In contrast, the oil and black powder method, while useful for real-time observation, lacked permanence and was less precise in low-velocity regions.

Despite the value of surface flow visualization techniques, there remains a significant gap in correlating qualitative surface patterns with more detailed flow characteristics, such as turbulent kinetic energy or velocity profiles. Techniques like oil/dye visualization primarily provide information about the direction of the flow, but they are limited in their ability to yield precise quantitative data about the flow properties. Based on the assumption that the oil paint traces lines corresponding to wall shear stress, streakline patterns can reveal transitions from laminar to turbulent boundary layers and regions of flow separation. However, current research on flow visualization, particularly in high Reynolds number flows, has predominantly focused on qualitative insights into features such as vortex structures and separation zones, with relatively less emphasis on directly quantifying flow field properties. This gap means that flow visualization images are often used as supplementary illustrations in research, and the rich information they contain is not fully utilized.

In this context, the integration of machine learning techniques represents a promising development in flow visualization. While traditional flow visualization methods rely heavily on manual identification and analysis of surface patterns, machine learning can assist in automating the process of recognizing and labelling flow features, such as direction fields, directly from surface patterns. Through machine learning, it is possible to enhance the analysis of flow patterns. This approach can help researchers more efficiently interpret flow

characteristics and improve the accuracy of flow visualizations, without the need for invasive measurement techniques. The use of machine learning in this study is specifically focused on automating the labelling of direction fields in surface flow patterns, thus providing a more consistent and scalable approach to flow visualization.

### 2.4.1 Basic concept of flow visualization

Regardless of the methods used for flow visualization, the result is always one or a set of images. These images can come from direct photography, vector fields measured by digital particle image velocimetry (DPIV), or even from experimental results of pressure sensors. Ultimately, these images need to be interpreted manually, and through them, the flow field can be understood directly or indirectly. Before this, it is essential to define and understand the terms like flow visualization and flow pattern and their relationships.

According to Perry and Chong (2012), the flow type described by streamlines consists of special points where the streamline slope is indeterminate, and the velocity is zero. These points were referred to as critical points or stationary points in the flow pattern. A streamline is essentially a curve that is tangent to the velocity vector at every point, and at the stationary point, since the velocity is zero, the direction of the streamline becomes indeterminate. Understanding these special points is important for analysing the overall structure of the flow because they represent key topological features in the flow.

Another important aspect of flow visualization is distinguishing between streamlines, pathlines, and streaklines to describe the flow. As illustrated in Figure 2.3 (Nakayama, 2018), these concepts, though identical in steady flow, exhibit significant differences in unsteady flow. Streamlines (Figure 2.3a), which show the velocity field at a specific moment, are tangent to the velocity vector at each point; Pathlines (Figure 2.3b) represent the actual path of fluid particles over time, showing where a single particle moves through the flow; Streaklines (Figure 2.3c) show the positions of particles released from a fixed point at different times, tracing the paths of various particles starting from the same point. In unsteady flow, streaklines can traverse different regions of the flow, to reveal the complexity of flow field.

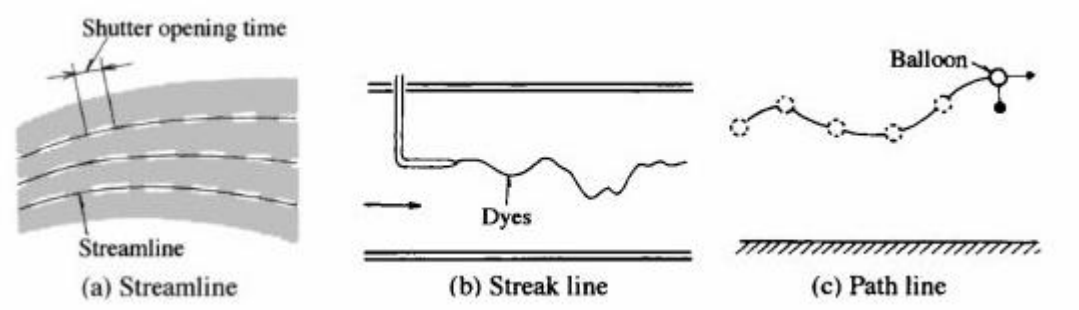



**Figure 2.3 Schematic diagram of (a) Streamline. (b) Streakline. (c) Pathline (Nakayama, 2018)**

By analysing critical points and flow patterns in various flows, Chong et al. (1990) studied the flow characteristics and behaviour under different conditions. Through experiments and numerical simulations, they identified and classified foci, saddles, and nodes in the flow. For vortex structures, methods based on invariants of the vorticity and rate of strain tensors were proposed to classify and analyse vortex core structures, which are important in turbulence studies.

The most widely used local methods for vortex identification are based on the analysis of the velocity-gradient tensor $\nabla \boldsymbol{u} = \boldsymbol{S} + \boldsymbol{\Omega}$, its symmetric and antisymmetric parts, strain-rate tensor **S** and vorticity tensor **Ω**, respectively, and the three invariants of $\nabla \mathbf{u}$ (Kolář, 2007). The Q-criterion (Hunt et al., 1988) identifies vortices as regions where the vorticity magnitude prevails over the strain-rate magnitude, ensuring that the second invariant of the velocity-gradient tensor is positive. And the $\Delta$-criterion (Chong et al., 1990, Vollmers et al., 1983, Dallmann, 1983) characterizes vortices through the occurrence of complex eigenvalues of the velocity-gradient tensor, which correspond to elliptic critical points in a moving reference frame. The λ2-criterion (Jeong and Hussain, 1995) refines vortex detection by considering the pressure Hessian and defining vortices based on the ordering of eigenvalues of the symmetric tensor $\mathbf{S^2 + \Omega^2}$, ensuring two negative eigenvalues. Machine learning techniques have also been introduced to automate and improve vortex identification, making it possible to analyse large-scale turbulence datasets with greater efficiency and accuracy.

By combining different visualization techniques, fluid motion in complex flows could be better understand. For instance, Perry and Chong (2012) proposed a method which used vortex skeleton of the flow to simplify and interpret complex vortex fields. They used the Biot-Savart law to calculate the distribution of vortex cores in the flow, which was then used to

reconstruct the flow field. This method worked especially well for complex flows like turbulent boundary layers, jets, and wakes.

Many other researchers have also made important contributions to the study of complex flows. For example, Lighthill (1963) studied the generation of vorticity in incompressible, uniform-density flow, then they found that all vorticity is generated at solid boundaries. Batchelor (2000) further investigated the properties of vortices moving with fluids and demonstrated this phenomenon through staining experiments. Truesdell (1954) and Cantwell (1979) explored the mechanisms of vortex formation and evolution through theoretical analysis and experimental research. Dallmann (1983) and Vollmers et al. (1983) focused on the development of experimental techniques, especially the use of laser and PIV to capture complex vortex structures in flow.

Streamlines show the vortex structures and separation points, while pathlines and streaklines indicate the formation and evolution of vortices. This information is important for understanding the mixing, diffusion, and transport properties of turbulence. Robinson (1991), Perry and Chong (1994) demonstrated the existence of complex vortex structures in turbulent boundary layers through experiments and numerical simulations, and proposed a vortex classification method. The vortex structure can be effectively distinguished from other flow features based on the invariant of the velocity gradient tensor (Hussain and Jeong, 1995). Vortex identification techniques have also been widely used in studying turbulent eddies, vortex shedding, and wake dynamics, providing deeper insight into energy transfer and turbulence modulation. Soria and Cantwell (1994) studied transient phenomena in complex flows, as well as the evolution of flow structures.

In summary, flow visualization is not only capturing images, but also using these images to analyse the fundamental principles and structures of fluid dynamics. By understanding the relationships between streamlines, pathlines, and streaklines, using mathematical and modern visualization techniques, fluid motion can be better described and analysed, as a result, more information about the flow field could be obtained.

### 2.4.2 Methods of flow visualization

Based on the reading conducted during the literature review, flow visualization methods can be categorised on the principles they utilize, and each category can be further subdivided by specific characteristics:

1. **Adding Extraneous Material**:
    - Hydrogen Bubble Visualization.
    - Smoke and Paint Visualization.
    - Tuft Visualization.
2. **Adding Energy**:
    - Molecular Tagging.
3. **Optical Methods**:
    - Schlieren Method.
    - Digital Interferometry.

Hydrogen bubble visualization relies on electrolysis to generate numerous small bubbles on a thin conductive wire (25-50 µm), which acts as one electrode in a direct current circuit. The other electrode, typically metal or graphite, is placed outside the test zone but within the fluid flow. The wire, serving as the negative electrode, forms hydrogen bubbles on its surface. By oscillating the voltage through the wire, successive lines of hydrogen bubbles, known as timelines, are created at constant time intervals. These bubble lines are carried downstream by the fluid flow and deform according to the local velocity profile. The bubbles, being 1-1.5 times larger than the wire diameter, rise at a negligible speed compared to the fluid flow. The shape and orientation of the wire depend on the type of fluid flow being examined, and the distance between the wire supports, which are subject to vortex shedding, is crucial to avoid disrupting the flow. Materials like stainless steel, aluminium, tungsten, and platinum are commonly used for the wire, with platinum being favoured for its non-corrosive nature and excellent soldering properties. Proper illumination of the bubbles allows for detailed visualization and quantitative analysis using photographs. Background light tilted at a recommended angle of 65º is used for illumination (Post and van Walsum, 1993, Merzkirch, 1987).

Traditional methods such as dye and smoke have long been utilized for visualizing fluid motion. Dyes are simple and effective. To reduce buoyancy, dyes are mixed with small amounts of alcohol, methanol, or ethanol. Other substances like laundry detergent, milk, fluorescent dye, potassium permanganate, and rhodamine can also be used, provided they have neutral buoyancy, resist mixing with the test fluid, and offer good visibility (Merzkirch, 1987, Van Dyke and Van Dyke, 1982). Dyes are typically injected into the fluid stream via small diameter pipes (1.5-2 mm) or through hypodermic needles or model openings. However, the injection velocity can disrupt the fluid flow; too high a velocity produces mushroom-like

disturbances, while too low a velocity creates a vortex street (Tritton, 2012). These disruptions can be minimized by placing the dye source further upstream.

Smoke wires are another common technique for airflow visualization. A metal wire coated with hydrocarbon oil is heated by passing current, causing the oil to evaporate and form smoke (Freymuth, 1993). The choice of wire material is less critical, provided it meets tensile strength and electrical resistance requirements; nickel, steel, and tungsten are frequently used. The wire diameter depends on the fluid flow velocity, with smaller diameters for lower velocities and vice versa. The oil must be evenly applied, either manually or automatically with brushes (Merzkirch, 1987).

Tufts provide a simple method for determining flow direction near a solid surface. When the flow becomes unstable or turbulent, the tufts move erratically, indicating a turbulent boundary layer or separation (Chen et al., 2019b). The choice of tuft size and material depends on the model size and test conditions. Ordinary cotton threads are used for large models, while mini tufts, made from thin nylon strands with a diameter around 20 µm, are used to minimize flow disturbance. Fluorescent colouring can enhance visibility. Mini tufts can exhibit high-frequency whiplash motion, making them difficult to photograph, so small cups may be attached to calm their movement (Treaster and Stinebring, 1980).

Special cameras like CCD cameras are used to photograph marked molecules and track their movement. Photochromic dye or phosphorescent supramolecules can be used as tracers (Gendrich et al., 1997). Photochromic tracers start transparent and become visible upon photon absorption, allowing the fluid containing the dye to appear dark when illuminated by white light. Fluorescein, with a high quantum yield of about 85%, can also be used, making the method known as laser-induced fluorescence (Kozomora et al., 2024). Photochromic ink is cost-effective and can be activated with a nitrogen gas laser. These tracers are ideal for studying fluid mixing, as mixing zones are highly visible. Phosphorescent supramolecules address low contrast issues, emitting light after excitation, allowing for extended observation periods. This method is advantageous for three-dimensional flow visualization.

Optical methods for flow visualization include techniques like the Schlieren method, which is highly sensitive to density changes (Settles, 2001). In Schlieren systems, brightness levels depend on the refractive index's first derivative, whereas in shadow photography, they depend on the second derivative. A Schlieren system with parallel light converges the light

using a lens or spherical mirror, forming an image at the focal point where a knife edge cuts off part of the light to reduce illumination intensity. Using point or slit light sources is mandatory. Errors like astigmatism and coma, caused by lens irregularities, can be minimized using a Z-shaped optical array and parabolic lenses. A two-pass Schlieren system increases sensitivity to small density changes, using a conical light source and an optical beam splitter, reducing astigmatism and coma. Schlieren methods can also produce colour images using white light and filters at the focal point (Goldstein, 2017).

Interferometry uses the superposition of light waves to amplify or cancel each other. The interferometer splits the light beam into two paths, one directed at a reference plane and the other at the fluid flow, creating phase delays seen as alternating dark and light patterns (Hariharan, 2010). Digital interferometry combines holographic interferometry with computer processing to accurately determine phase delays, provide real-time fluid flow information and capture small perturbations. This method can be used for visualizing stable and unstable fluid flows (Watt and Vest, 1987).

Current SFV techniques primarily focus on flow tracking based on foreign material introduction and optical interference principles, and they are easily influenced by intrusive flow field disturbances and limitations in spatiotemporal resolution. For instance, hydrogen bubble and dye injection may alter local flow structures due to buoyancy effects or injection velocity, while tuft methods struggle to capture transient details in high-frequency turbulence. Moreover, with the advancement of high-speed imaging and computational processing capabilities, molecular tagging and digital interferometry have gradually enabled high-precision three-dimensional dynamic measurements, though they still face challenges such as complex equipment and high costs.

However, from the perspective of multimodal fusion and intelligent algorithm optimization, it is possible to integrate non-intrusive optical methods such as laser-induced fluorescence with machine learning-based image analysis, thereby minimizing flow field interference. With the deep learning techniques, it becomes possible to automatically identify and analyse flow field images to extract key parameters such as velocity, vorticity and shear rate.

### 2.4.3 Image Processing Techniques

Image processing usually is the first step in the analysis of flow visualization images in fluid mechanics. Images captured by digital cameras would be converted into digital files that allow for further processing to extract additional information. When analysing these images, it is necessary to link the patterns and topological rules of the images. Edge detection is a fundamental step in image processing, which helps identify and outline these patterns. It has been widely used in classical computer vision tasks such as segmentation (Zhang et al., 2015) and image recognition (Yang et al., 2002, Shotton et al., 2008). More recently, edge detection has also been applied to modern tasks like image-to-image translation (Zhu et al., 2017) , photo sketching (Li et al., 2019a), image analysis (Pourreza et al., 2018) and remote sensing (Isikdogan et al., 2017).

Since the introduction of the Sobel operator (Sobel, 1970), many edge detection techniques have been developed. Classical methods like the Canny edge detector (Canny, 1986) have stood the test of time and are still widely used today. For instance, Abdelsalam et al. (2017), developed an image processing algorithm utilizing Canny edge detection and linear Hough transforms to extract quantitative data from SFV images. This algorithm effectively identifies flow patterns and allows for detailed comparisons with CFD results to better understand flow characteristics. However, it also generated scattered noise vectors, which indicated the need for further refinement to improve accuracy and reduce errors.

Significant progress has been made in edge detection because of the deep learning, especially convolutional neural networks (CNN). CNN is designed for processing grid like data, such as images. They are composed of multiple layers and can automatically learn to extract features from raw image data, which makes them highly effective in tasks such as object detection, classification, and edge detection. New edge detectors have been developed using CNNs, such as DeepEdge (Bertasius et al., 2015), Holistically-Nested Edge Detection (HED) (Xie and Tu, 2015), Richer Convolutional Features (RCF) (Liu et al., 2017), and Boundary Detection Convolutional Network (BDCN) (He et al., 2019). These models are similar to traditional methods for predicting edge maps from images, but they have excellent performance due to their ability to learn and capture complex features from large datasets. The effectiveness of these deep learning-based methods stems from applying CNNs at different scales and using diverse image sets for training, combined with various training regularization techniques (Poma et al., 2020). For instance, the HED model takes advantage of multi-scale feature fusion to enhance edge detection accuracy, demonstrating the power of deep learning in this field.

There are many directions for future research in edge detection and other image processing techniques. Developing more robust algorithms to handle complex backgrounds and diverse image characteristics remains a priority. Additionally, integrating more advanced deep learning techniques, such as Generative Adversarial Networks (GANs) and Graph Convolutional Networks (GCNs), could further improve edge detection accuracy and expand its applications. Cross-disciplinary integration, such as combining image processing with natural language processing (NLP) and augmented reality (AR) technologies, also holds promise for new breakthroughs in image analysis.

## 2.5 Machine learning and conditional generative adversarial networks

Machine learning is the subfield of computer science that, according to Samuel (1959), gives "computers the ability to learn without being explicitly programmed". Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data – such algorithms overcome the constraints of following strictly static program instructions by making data-driven predictions or decisions, through building a model from sample inputs. Machine learning is sometimes conflated with data mining, where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning. Machine learning can also be unsupervised and be used to learn and establish baseline behavioural profiles for various entities and then used to find meaningful anomalies. Within the field of data analytics, machine learning is a method used to devise complex models and algorithms that lend themselves to prediction; in commercial use, this is known as predictive analytics. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data (Ongsulee, 2017).

Machine learning is an application of artificial intelligence, which provides a given system with the ability to learn automatically and improve from experience without explicit programming.

In the author's view, surface flow visualization images can be considered the *fingerprints* of the flow field. The basic fingerprint recognition system consists of four stages: firstly, the sensor which is used for enrolment and recognition to capture the biometric data.

Secondly, the pre-processing stage which is used to remove unwanted data and increase the clarity of ridge structure by using enhancement technique. Thirdly, feature extraction stage which take the input from the output of the pre-processing stage to extract the fingerprint features. Fourthly, the matching stage is to compare the acquired feature with the template in the database.(Ali et al., 2016). To alleviate both cost and privacy issues, several approaches have been proposed to create synthetic fingerprint datasets, Bouzaglo and Keller (2022) proposed a novel fingerprint synthesis and reconstruction framework based on the StyleGAN2 architecture, where the generator is trained on a large fingerprint dataset. The generator produced realistic fingerprint images based on latent vector inputs. By feeding a random vector into the generator, a random fingerprint could be synthesized. Through the Minutiae-To-Vec encoder, minutiae can be encoded into latent vectors for fingerprint reconstruction. Svoboda et al. (2017) trained GANs on synthetic datasets to denoise and predict missing parts of fingerprint ridge patterns. In addition, they used convolutional autoencoders to reconstruct damaged fingerprints and achieved state-of-the-art results on multiple fingerprint databases.

Song et al. (2023) used the pix2pix method, which will be detailed introduced in section 2.5.2, to predict the flow field and aerodynamic performance of aerofoils in wind turbine blades. These methods served as potential alternatives to traditional, resource-intensive CFD techniques, and showed promising results in quickly identifying aerodynamic characteristics, such as blade design. Han et al. (2019) introduced a new method based on deep learning for streamline based flow field representation and simplification, aimed at operating within in situ visualization settings. The process involved tracing streamlines from each simulation timestep, storing them in a compressed form, and later reconstructing the vector fields for detailed post analysis.

### 2.5.1 Convolutional neural network

Convolutional Neural Networks (CNNs) are deep learning models designed primarily for tasks involving grid-like data structures, particularly image and video analysis (Ciresan et al., 2011). These networks typically consist of three primary components: an input layer, hidden layers, and an output layer. The hidden layers include one or more layers that perform convolutions, where a mathematical operation (the convolution) is performed between the input data and a filter or kernel, typically using the Frobenius inner product(Goodfellow, 2016). The activation function commonly used is ReLU (Rectified Linear Unit) to introduce non-linearity(Nair and Hinton, 2010). As the convolution kernel moves across the input, it generates

a feature map that is passed on to subsequent layers, which can include pooling, fully connected, and normalization layers (Krizhevsky et al., 2012). CNNs share some functional similarities with matched filters in signal processing.

In a CNN, the input is a tensor with dimensions:

(number of inputs) × (input height) × (input width) × (input channels)

After passing through a convolutional layer, the image becomes abstracted to a feature map, also called an activation map, with shape:

(number of inputs) × (feature map height) × (feature map width) × (feature map channels).

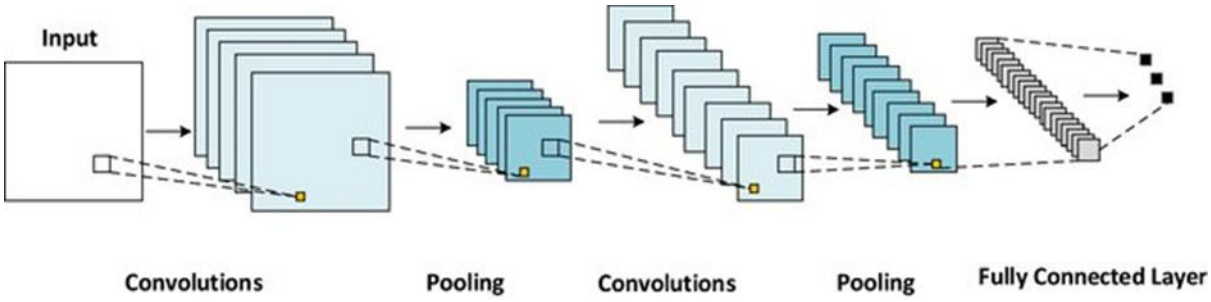



**Figure 2.4 Typical CNN architecture. Khan et al. (2021)**

As depicted in Figure 2.4, CNNs involve a multi-stage process where each layer extracts increasingly abstract representations of the input data. Convolutional operations mimic the response of neurons in the visual cortex to stimuli, where each neuron processes only a subset of the input, known as the receptive field.

Alongside the convolutional layers, CNNs often incorporate pooling layers, which further reduce the data dimensions. Pooling operations can be either local or global: local pooling (e.g., 2×2 tiling) consolidates small clusters of neurons, while global pooling aggregates information across the entire feature map (Krizhevsky et al., 2012, Ciresan et al., 2011). Common pooling strategies include max pooling, which selects the maximum value from each cluster, and average pooling, which computes the average value (Ciregan et al., 2012, Yamaguchi et al., 1990).

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is the same as a traditional multilayer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

In neural networks, each neuron receives input from some number of locations in the previous layer. In a convolutional layer, each neuron receives input from only a restricted area of the previous layer called the neuron's receptive field. Typically, the area is a square (e.g. 5 × 5 neurons). Whereas, in a fully connected layer, the receptive field is the entire previous layer. Thus, in each convolutional layer, each neuron takes input from a larger area in the input than previous layers. This is due to applying the convolution over and over, which takes the value of a pixel into account, as well as its surrounding pixels. When using dilated layers, the number of pixels in the receptive field remains constant, but the field is more sparsely populated as its dimensions grow when combining the effect of several layers.

To manipulate the receptive field size as desired, there are some alternatives to the standard convolutional layer. For example, atrous or dilated convolution (Chen et al., 2017, Yu and Koltun, 2015) expands the receptive field size without increasing the number of parameters by interleaving visible and blind regions. Moreover, a single dilated convolutional layer can comprise filters with multiple dilation ratios (Duta et al., 2021), thus having a variable receptive field size.

Each neuron in a neural network computes an output value by applying a specific function to the input values received from the receptive field in the previous layer. The function that is applied to the input values is determined by a vector of weights and a bias (typically real numbers). Learning consists of iteratively adjusting these biases and weights.

The vectors of weights and biases are called filters and represent particular features of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons can share the same filter. This reduces the memory footprint because a single bias and a single vector of weights are used across all receptive fields that share that filter, as opposed to each receptive field having its own bias and vector weighting.

The CNNs could be evaluated by several different metrics, including classification accuracy, precision, recall, F1-score, and confusion matrices when dealing with classification problems. For tasks like image processing, edge detection and flow field prediction, the mean squared error (MSE) and mean absolute error (MAE) are commonly used.

For flow field prediction, CNNs can effectively extract features from flow data and generate accurate flow predictions. This makes CNNs particularly suitable for modelling and analysing complex flow fields. For example, Kim and Günther (2019) developed a CNN-based algorithm that combines filtering and feature extraction to robustly extract vortex centres and generate nearly steady reference frames from noisy and resampled inputs. CNNs also could be used to predict pressure on a cylinder from velocity distributions in wake flow, achieving high precision and efficiency through transfer learning, as shown in the work of Ye et al. (2020).

### 2.5.2 Generative adversarial networks and Pix2pix

The concept of image-to-image transformation was initially proposed by Hertzmann et al. (2001), where they applied learned generative filters to new target images to create "analogous" filtering results, called "image analogies". Gatys et al. (2016) used feature representations from high-performing CNNs to transfer image style between arbitrary images. Their work primarily involved a series of operations directly applied to target images, and the results usually contained a lot of low-level noise. Recently, some impressive results have been achieved using Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). Depending on the target and data types, GANs can be divided into many variants to better suit specific tasks. Representative variants include InfoGAN, cGAN, CycleGAN, f-GAN, WGAN, et al. The model used in this thesis is a type of cGAN.

The architecture of the GAN and cGAN is shown in Fig 2.5. The GAN framework consists of two main components: a generator (denoted as G) and a discriminator (denoted as D), which work together to generate data in an adversarial training process (Goodfellow et al., 2014). The generator takes random noise as input and generates synthetic data, while the discriminator evaluates the data and determines whether it is real or generated. The training objective of the generator is to produce data that the discriminator cannot distinguish, while the goal of the discriminator is to correctly identify whether the input data is real.

The internal structure of the generator typically begins with a fully connected layer that expands the low-dimensional noise vector into an initial set of feature maps. This is followed by a series of transposed convolutional layers (also known as deconvolution layers) that progressively upsample the feature maps to the target output size. Each upsampling step is usually accompanied by batch normalization to stabilize the training and ReLU activation functions to introduce non-linearity. The output layer commonly uses a Tanh activation

function to ensure that the generated image values are within a specified range. the structure of the generator could vary greatly depending on different tasks.

The discriminator, conversely, is structured as a deep convolutional neural network that gradually downscales the input through a series of convolutional layers. Each convolutional operation is typically followed by batch normalization and a LeakyReLU activation function, which helps to prevent the vanishing gradient problem. Finally, a fully connected layer with a Sigmoid activation function outputs a probability score indicating whether the input is real or generated. The input to the generator is a noise tensor with dimensions (batch size) × (noise dimension), whereas the discriminator receives an image tensor with dimensions (batch size) × (height) × (width) × (channels). Figure 2.5(a) illustrates the basic architecture of GANs.

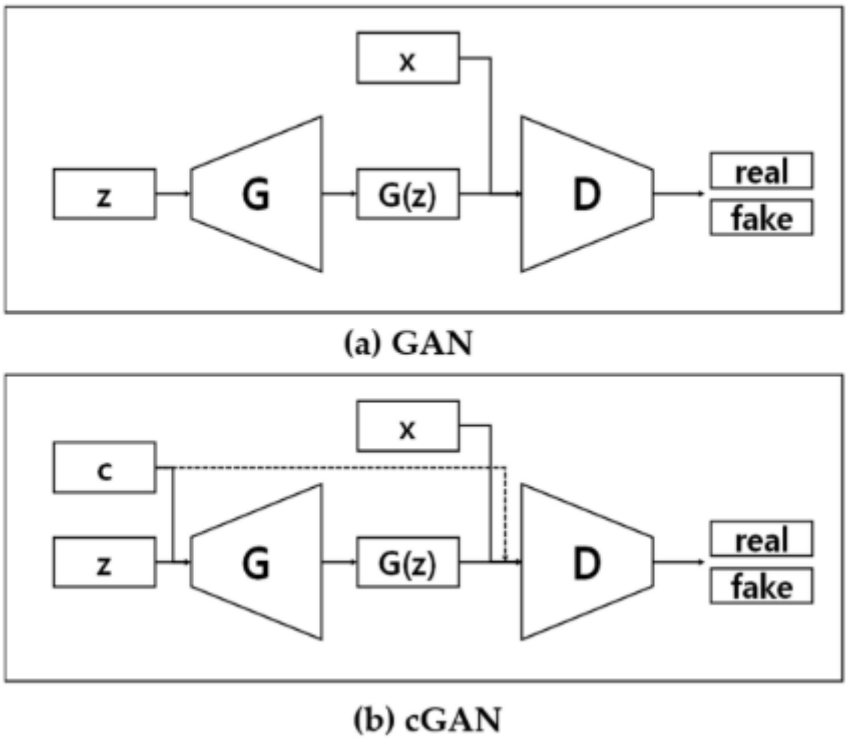



**Figure 2.5 Architectures of (a) GAN and (b) cGAN**

The cGAN is an extension of the traditional GAN, designed to generate data based on specific conditions (Mirza and Osindero, 2014). These conditions can be provided in various forms, such as noise vectors, images, or class labels. In the cGAN architecture (as shown in Fig 2.5(b)), the generator G receives a combination of the input noise z and the condition c,

which will be used to generate data. Then the discriminator receives input data x and condition c to determine whether the data is real or generated.

Pix2pix is a widely adopted approach for solving image-to-image translation problems and it can be considered as a special type of cGAN (Isola et al., 2017). In pix2pix, the generator is built on the U-Net architecture, which is effective in image-to-image translation tasks. U-Net has a "skip connection" structure that directly connects corresponding encoder and decoder layers. This structure allows for more stable learning compared to a simple encoder-decoder design. The discriminator in pix2pix utilizes a convolutional PatchGAN classifier, which evaluates images by focusing on patches of a specific size rather than the entire image. This modification helps train the generator to generate images with more realistic and detailed features.

The evaluation of GANs requires different strategies compared to traditional discriminative models. Since the primary goal of a GAN is to generate realistic samples, quantitative metrics such as the Inception Score (IS) and the Fréchet Inception Distance (FID) are widely used, the detailed information of which would be introduced in section 6.2.2.

Although GANs have some applications in fluid dynamics, such as solving the Navier-Stokes equations (Wu et al., 2022, Song et al., 2023, Chen et al., 2020), reducing computational costs (Kastner and Dogan, 2023, Afzali et al., 2021), here are still few studies on surface flow visualization using GANs. This thesis attempts to explore this underexplored direction by applying GAN-based techniques, especially cGANs, to enhance the understanding of surface flow patterns.

## 2.6 Overview

This chapter introduces the use of machine learning in surface flow visualization. The Flow is complex, but the study of the flow field is very meaningful, and for this, flow visualization methods are needed for both qualitative and quantitative analysis. Machine learning has considerable potential as a tool for predicting and analysing flow fields. This data-driven, non-invasive approach is a significant improvement over traditional method. Experimenters can understand some characteristics of the flow field before making detailed measurements or simulations.

A number of methods of flow visualization using the advanced imaging technologies have been proposed. However, much of the previous work in flow visualization has been manual and qualitative, limiting its ability to extract detailed, quantitative information from flow fields. This thesis aims to address this issue by developing an automated, quantitative method to extract information from flow visualization results. By adopting this approach, more detailed insights into flow field surfaces can be obtained, which can then be compared with CFD results for validation and deeper analysis. This method is significant because it not only enhances the information gathered from flow visualization but also bridges the gap between qualitative observations and quantitative comparisons. Ultimately, it advances the understanding of fluid dynamics and improves the accuracy of flow analyses.

# Chapter 3 Methodologies

This chapter focuses on the experimental methods used in the Durham Cascade and 0.5 Plint wind tunnel. It describes the techniques employed to obtain flow visualization and GLOF skin-friction meter for this thesis. There is an extensive section on data processing techniques specific to the cascade. This chapter also includes descriptions related to CNNs, U-net, and GANs. It covers the structure and layers of these neural networks, methods for preparing datasets, and some training details.

## 3.1 Experimental facilities

To support this thesis, experiments were conducted in two facilities, with approximately 300 tests performed, of which more than 200 were used for this thesis. The primary variables investigated included wind speed, different obstacles in the flow field, and varying oil/dye ratios. Flow visualization experiments using oil/dye techniques were carried out in the 0.5 Plint wind tunnel to examine simple surface flow phenomena. The experimental conditions covered a range of nondimensional numbers, including Reynolds numbers (Re) varying from $2\times10^3$ to $3\times10^4$.

To ensure the reliability and accuracy of the results, multiple repetitions were conducted for each set of experimental conditions, minimizing random errors and enhancing the statistical significance of the data, which were then used to train the model.

### 3.1.1 The Durham cascade

The Durham Cascade is a low-speed, large scale linear cascade developed and constructed at Durham University for studying secondary flow. It comprises an array of six blades designed to maintain a similar Reynolds number and pressure distribution of Rolls-Royce RB211 engine high-pressure turbine blades but at a low Mach number (McIntosh et al., 2011).

After being originally designed by Graves (1985), the Durham Cascade has undergone several refurbishments to accommodate different types of experiments. These refurbishments include the installation of aluminium blades, improvements in blade and endwall mounting methods, and enhancements in measurement accuracy. The latest layout, designed for tip clearance flow experiments, is depicted in Figure 3.1.

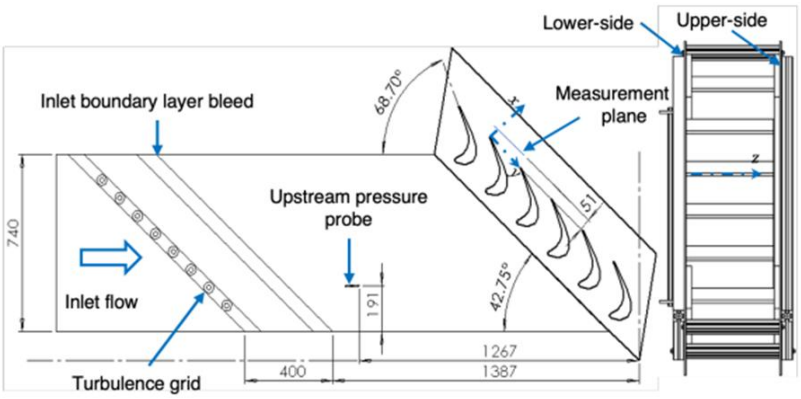



**Figure 3.1 Durham Cascade Layout Third Angle Projection Including Definitions of Cascade Coordinates and Image Capture. (Martinez-Castro, 2022)**

Air was supplied to the cascade by a variable-speed centrifugal fan, passing through various filters and settling chambers before entering the working section. Turbulence intensity at the inlet was approximately 5%, achieved by a grid of bars mounted 1.4 m in front of the blade leading edge. Boundary layer bleeds were provided 150 mm downstream of the turbulence grid on both walls. The cascade turned the flow through 111° and discharged upwards. Table 3-1 provides a summary of the key parameters of the Durham Cascade.

**Table 3-1 Key Cascade Parameters**

| Parameter | Unit |
|---|---|
| Inlet Flow Angle | 42.75° |
| Exit Blade Angle | -68.70° |
| Blade Pitch | 191 mm |
| Blade Axial Chord | 181 mm |
| Tip Clearance | 3.75 mm |
| Freestream Velocity | 19.1 m/s |
| Dynamic Pressure | 215 Pa |
| Re number | $4 \times 10^5$ |

The cascade blades had a span of 375 mm and a chord of 224 mm, resulting in an aspect ratio of 1.78. Blade passages were instrumented on one endwall. The endwalls could be changed with profiled or planar endwalls, depending on the requirements of the experimental campaign.

### 3.1.2 The 0.5 Plint wind tunnel

The 0.5 Plint wind tunnel used in this work was an open-circuit design that discharged directly to the atmosphere. The working section had a square cross-section of 457 mm × 457 mm, with a length of 1.22 m. The sides of the working section were solid, while the top and bottom were open. The tunnel was driven by an upstream 36-blade centrifugal fan, with a rotational speed of approximately 17 Hz, resulting in a blade passing frequency of 600 Hz. The maximum tunnel speed was around 21 m/s, and the turbulence intensity had been measured to be less than 0.25%.(Sims-Williams, 2001)

In this work, modifications were made to the Plint wind tunnel as shown in Figure 3.2. A mounting device was used to secure the camera to the section, which significantly minimizes the impact of vibrations during operation on the results. Directly beneath the camera lens was a transparent glass plate that allowed the camera to directly record the conditions within the wind tunnel's working section without affecting the internal airflow.

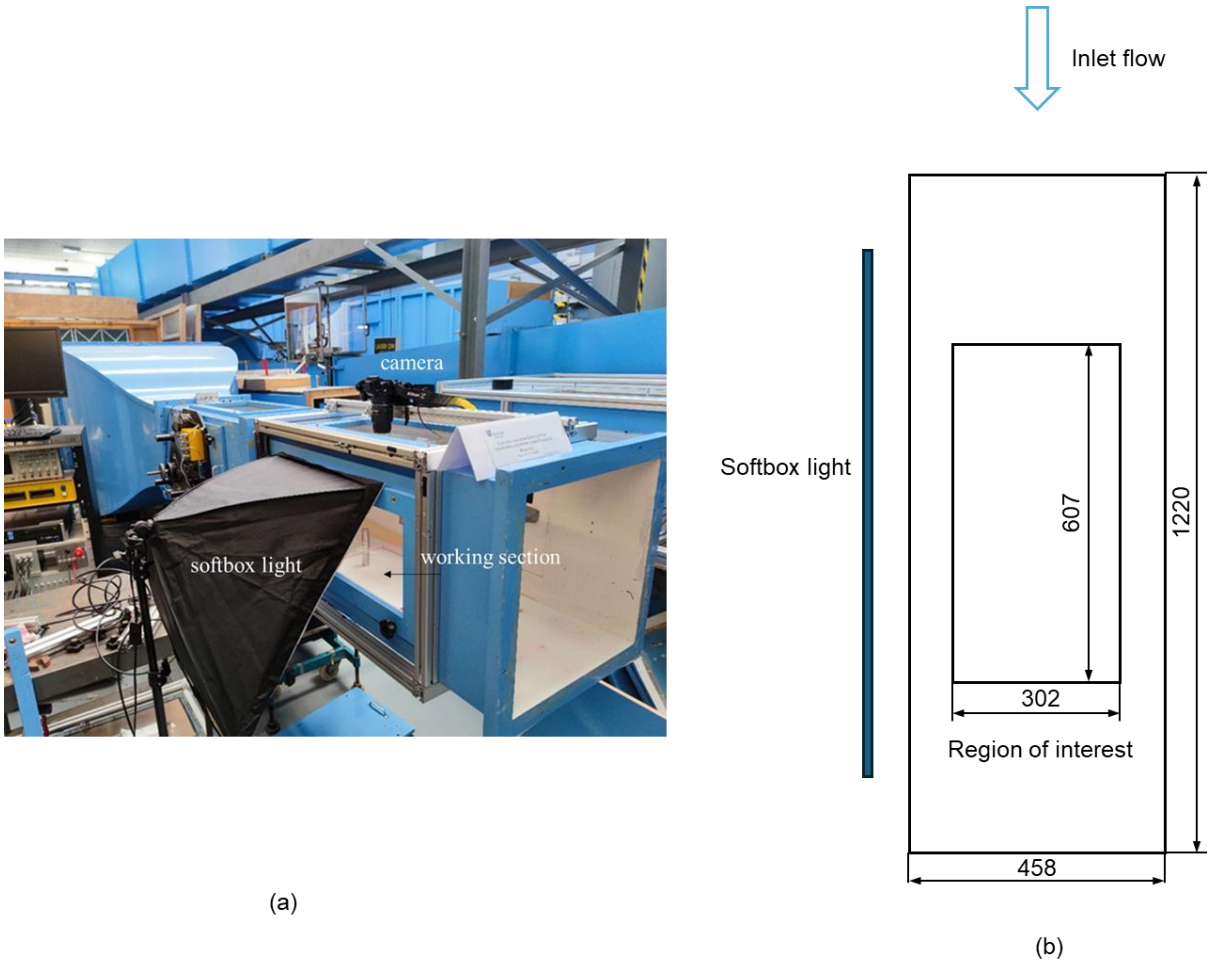



**Figure 3.2 0.5 Plint Wind Tunnel Layout**

The region of interest (ROI) was mounted flush with the bottom wall of the wind tunnel and secured beneath the test section. Simple-shaped objects could be placed or suspended above the ROI to influence the surface flow field. This setup allowed for the collection of various flow visualization images and videos, capturing the effects of different objects on the flow dynamics.

## 3.2 Experimental techniques

### 3.2.1 Oil/dye visualization

The principle of oil/dye surface flow visualization involved applying a mixture of oil (diesel or paraffin) and dye to the surface of the model of interest. This model was then placed in the wind tunnel, where the flow was applied at the desired speed. The oil/dye mixture followed the path of the flow on the model surface, with regions of low velocity appearing as the oil/dye remained mostly in place. Separation bubbles were easily identified as a "slug" of fluid accumulated in the region of recirculation. Besides, regions of high shear stress near the wall were identified as the oil/dye mixture was removed from the model surface altogether. The most comprehensive information on this technique is provided by Barlow et al. (1999), Merzkirch (2012) and Maltby (1962), which describe it in detail.

In oil/dye work, it is important to ensure that the speed in the wind tunnel is high enough; otherwise, gravity may cause the pattern on the blades and endwalls to be meaningless. In low-speed areas, the oil/dye may also stay in place, causing the pattern to come entirely from the brush used to apply the mixture. Airbrush was not used here because the dye itself, often clogged the feed inlet. The velocities in the Durham cascade were just high enough to obtain acceptable flow visualization results. For instance, good flow visualization was achieved on the suction side of the blade and in the blade passage without significant difficulty. However, the pressure side and upstream of the blade leading edge on the endwall presented more challenges. This was particularly relevant as all aerofoils had a stagnation point where the velocity dropped to zero, making surface flow visualization inherently challenging in these regions. The impact of velocity increase on flow visualization results was clearly illustrated by Marchal and Sieverding (1977), where oil visualizations showed exit velocities ranging from 20 m/s to 160 m/s, although it should be noted that oils of differing viscosities were used in the tests.

The tests began with preparing an oil/dye mixture in ratios of 4:1 to 2:1 by volume, using standard road diesel and a dye composed of Fiesta daylight colours and an inert bulking agent. This dye fluoresced under ultraviolet light, enhancing the clarity of the resulting patterns. The oil/dye mixture was gently applied to the test article using a brush, ensuring an even coating without deep brush marks. The brush strokes were kept perpendicular to the expected flow direction to distinguish between brush marks and flow streaks.

Upon completion of the entire experiment, the endwall would be carefully extracted for photography. High-resolution images (6000 × 4000 pixels) were captured using a *Nikon D5300 DSLR* camera with an *AF-S Nikkor 35mm 1:1.8G* lens, mitigating optical distortions. The camera was mounted on a horizontal tripod arm, and the endwall maintained upright during capture.

In the specific tip clearance endwall experiment, after the wind tunnel had been initiated and the flow velocity had stabilized for approximately 25 minutes, the endwall was removed from the cascade, and image acquisition was conducted. Subsequent inspection of the endwall following the experiment revealed that the majority of its surface had become dry, with minimal oil residue present. However, certain areas near the projection of the blades still retained oil deposits. This is an expected result since it is expected that surface oil is readily entrained by the airflow along the streamlines, with less facile removal occurring at saddle points.

### 3.2.2 Global luminescent oil-film (GLOF) skin-friction meter

As discussed in Chapter 2, the Global Luminescent Oil Film (GLOF) method, developed by Liu et al. (2008), is a technique for extracting high-resolution skin friction fields from GLOF images. This method has been applied to a variety of complex flow scenarios (Tran et al., 2019, Tran et al., 2018, Lee et al., 2018, Zhong et al., 2015, Husen et al., 2018b, Husen, 2017, Husen et al., 2018a, Woodiga et al., 2016, Woodiga et al., 2018, Liu et al., 2011, Woodiga and Liu, 2009). It operated by measuring the oil-film thickness and inferring skin friction from the temporal and spatial evolution of this thickness. Luminescent oil, which emitted light proportional to its thickness when optically thin, was used for visualization. During experiments, a thin film of luminescent oil was applied to the surface of interest and illuminated with light sources. In this experiment, the regular supplementary lighting was used, as illustrated in Figure 3.2, while for some other case, the light source could also be UV light. The

resulting luminescent intensity was captured by a camera with an appropriate optical filter, effectively converting oil-film thickness measurement into luminescent intensity measurement.

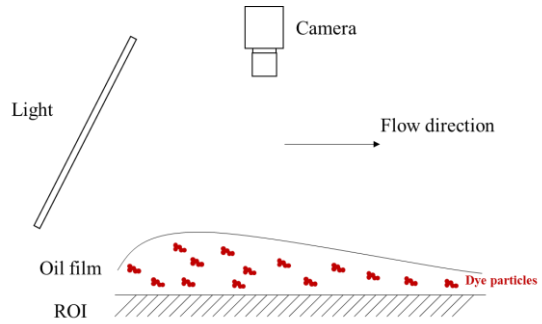



**Figure 3.3 Measurement System for Luminescent Oil-film Skin Friction Measurement.**

The GLOF method relies on measuring the thickness of a luminescent oil film and solving the projected thin-oil-film equation as an inverse problem. For an optically thin luminescent oil film applied to a surface $X_3 = S(X_1, X_2)$ and illuminated appropriately, the oil-film thickness $h$ is directly proportional to the normalized luminescent intensity, which could be written as $h = \beta^{-1}(I/I_{ref})$, where $\beta$ is proportional to the quantum efficiency of the luminescent dye mixed in the oil, I is luminescent intensity, and $I_{ref}$ is the reference intensity, which depends on the intensity of the background light source.

Replacing $h$ by $I$ in Eq (2.5), the following equation can be obtained:

$$\frac{\partial I}{\partial t} + \frac{\partial}{\partial X_i}\left[\frac{\kappa \tau_i}{2\mu} I^2 - \left(\frac{\partial p_w}{\partial X_i} - \rho g_i\right)\frac{\kappa^2}{3\mu} I^3\right] = 0, \quad (i = 1,2) \tag{3.1}$$

There is a one-to-one mapping between the image plane ($x_1$, $x_2$) and the surface $X_3 = S(X_1, X_2)$. This means that each point on the image plane corresponds uniquely to a point on the surface, and vice versa, enabling precise correlation of image data with physical locations on the surface. In the work described in this thesis, the camera was parallel to the experimental plane, the form of the orthographic projection transformation thus became simple. A relation $\partial/\partial X_i = \varepsilon \partial/\partial x_i$ exists, where $\varepsilon$ could be considered as a constant. To convert Eq.(3.1) in the image coordinates, a new variable called equivalent skin friction $\boldsymbol{\tau}' = (\kappa I \varepsilon/\mu)\boldsymbol{\tau_w}$ needs to be introduced. Substitution of $\tau'$ into Eq.(3.1) yields

$$\frac{\partial I}{\partial t} + \nabla(I\,\boldsymbol{\tau}') = f(x_1, x_2, I) \tag{3.2}$$

where is $\nabla = \partial/\partial x_i$ gradient operator in the image plane, and the right-hand-side (RHS) term in Eq.(3.2) is defined as

$$f(x_1, x_2, I) = \varepsilon \frac{\partial}{\partial x_i}\left[\left(\varepsilon \frac{\partial p_w}{\partial x_i} - \rho g_i\right)\frac{\kappa^2}{3\mu} I^3\right], \quad (i = 1,2) \tag{3.3}$$

Thus, with Eq.(3.4) the luminescent intensity, *I*, can be mapped onto the image plane. Since the camera's radiometric response function is linear, the relationship between *I* and the image grayscale intensity is also linear. The term *f* represents the effect of the pressure gradient and gravity, which is in the order of $\varepsilon h^3 \ll 0$ , since for very thin oil condition, $h^3 \ll 0$. Therefore, this term is considered as a higher-order small term in Eq.(3.2). Considering a spatial change of skin friction is small, Eq.(3.2) can be further rewritten as:

$$\frac{\partial I}{\partial t} + \boldsymbol{\tau}' \nabla I = 0 \tag{3.4}$$

At this point, Eq.(3.4) has the same form of the physics-based optical flow equation for various flow visualizations(Liu and Shen, 2008). Therefore, the same optical flow problem can be solved to determine a skin friction field. To determine the equivalent skin-friction field $\boldsymbol{\tau}' = (\tau'_1, \tau'_2)$, additional constraints need to be introduced since Eq.(3.2) is not sufficient to determine $\tau'_1$and $\tau'_2$. Liu et al. (2008) used a regularization term based on the $L^2$ norm of the skin-friction gradient that was originally proposed by Horn and Schunck (1981) for computing optical flow. The $L^2$ norm, also known as the Euclidean norm, measures the magnitude of a vector in a multidimensional space. It is calculated as the square root of the sum of the squares of its components, providing a way to quantify the overall size or length of the vector. In the context of the skin-friction gradient, applying the L2 norm helps to smooth the variations in the gradient, thus promoting continuity and stability in the resulting flow field. This variational equation essentially represents a constraint on the local continuity of skin friction field. Given *I* and *f*, a functional with the Horn–Schunck regularization term for the skin-friction field on an image domain Ω is defined as

$$(\boldsymbol{\tau}') = \int \left[\frac{\partial I}{\partial t} + \nabla \cdot (I\boldsymbol{\tau}') - f\right]^2 dx_1 dx_2 + \alpha \int (|\nabla \tau'_1|^2 + |\nabla \tau'_2|^2) dx_1 dx_2 \tag{3.5}$$

The Lagrange multiplier $\alpha$ should be suitably selected since it is the coefficient of the diffusion term of the Eq.(3.5) The minimization of $J(\boldsymbol{\tau}')$ the Euler-Lagrange equations:

$$I\frac{\partial}{\partial x_i}\left[\frac{\partial I}{\partial t}+\nabla\cdot(I\boldsymbol{\tau}')-f\right]+\alpha\nabla^2\tau'_i,\qquad(i=1,2)\tag{3.6}$$

where $\nabla^2=\partial^2/\partial x_i\partial x_i$ $(i=1,2)$ is the Laplace operator, and the Neumann condition $\partial\boldsymbol{\tau}'/\partial$n=0 is imposed on the domain boundary. The numerical solution of Eq.(3.6) gives the equivalent skin friction $\boldsymbol{\tau}'$. If the GLOF image is high-resolution and its intensity field with a sufficiently large intensity gradient, the $\boldsymbol{\tau}'$ solved by Eq.(3.6) will be not sensitive to $\alpha$.

Even in steady flow, the luminescent oil film on the region of interest surface is evolving over time. In fact, $\boldsymbol{\tau}'$ solved by Eq.(3.6) is a snapshot skin friction field from 2 successive images. Similarly, a series of corresponding snapshot solutions can be obtained from the sequence of successive images captured by video. From a physical perspective, a snapshot solution captures prominent skin friction signatures in areas where the oil-film evolution is particularly sensitive to flow at that specific moment. Consequently, a time sequence of snapshot solutions is necessary to capture significant skin friction signatures at various moments throughout the oil-film evolution process. To reconstruct the steady-state or time-averaged skin friction field, snapshot solutions within certain time range were superposed and averaged. It could combine instantaneous flow characteristics from different moments, effectively filtering out transient fluctuations.

In this study, the viscosity of the paraffin oil was chosen to be 3.0 cSt at 25 ℃. This specific viscosity was selected based on the need for a moderate evolution speed, which resulted in an interval of approximately 0.2 seconds between consecutive images. This time interval was optimal for the setup, as it helped avoid missing critical frames while also conserving computational resources. Additionally, during the experiments, it was observed that paraffin oil exhibited the advantageous property of being carried away from the surface at the end of the test, while the dye remained in the region of interest. To ensure accurate results, the oil was applied as a very thin layer, which prevented wind-driven ripples at the oil-air interface. This thin oil film also guaranteed that the film thickness stayed within the linear relationship range between oil film thickness and luminescence intensity, which was crucial for accurate flow visualization in the experiments.

After applying the mixture, the wind tunnel operation commenced. The tunnel should be quickly brought up to speed to prevent the dye from evolving while stationary. Once the conditions within the tunnel met the required standards, data collection began. High-resolution videos (1920×1080 pixels) were captured using a *Nikon Z50* camera with an *AF-S Nikkor*

*35mm 1:1.8G* lens, mitigating optical distortions. The camera was fixed outside the test section, directly above the ROI, supported by a steel structure. A transparent plate was placed below the camera lens, allowing the camera to directly record the flow pattern of the ROI without being affected by airflow-induced vibrations.

The image acquisition rate was set at 60 frames per second. The wind tunnel ran at full speed for 10 minutes, with the main pattern forming within the first 5 minutes, although longer runs allowed the pattern to dry more thoroughly. The colour intensity of the obtained images was proportional to the dye concentration in the coating. Normalized intensity images $h = \beta^{-1}(I/I_{ref})$ were used for processing, with the snapshot solutions extracted using the optical flow-based algorithm described in the data processing section.

The development of a luminescent oil film on a surface is time-dependent, even in steady flow conditions. The numerical solution of the projected thin-oil-film equation provides a snapshot skin friction field from a pair of successive images. A series of these snapshot solutions, obtained from a time sequence of GLOF images, captures key skin friction signatures at different moments, highlighting regions where the oil-film evolution is most responsive to the flow. To reconstruct a steady-state or time-averaged skin friction field, these snapshot solutions are superposed or averaged. The GLOF method typically yields a normalized skin friction field without requiring in-situ calibration. However, to obtain an absolute skin friction field, in-situ calibration using accurate skin friction values from reliable techniques, such as an interferometric oil-film skin friction meter or computational and theoretical methods, is necessary.

## 3.3 Data pre-processing

Subsequent to image acquisition, a multi-stage image processing pipeline implemented in software using the Python language was employed for image processing, utilizing functions sourced from the deep learning framework Pytorch (Paszke et al., 2019). Supplementary functions were drawn from other packages such as Numpy (Harris et al., 2020), SciPy (Virtanen et al., 2020), Matplotlib (Hunter, 2007), etc. This step has been applied before most of the image processing workflows in this thesis.

The preprocessing step was designed to reduce the noise and enhance edge information. It facilitated more accurate feature extraction and subsequent image analysis stages. And thereby this improvement increased the reliability and performance of the overall workflow

described in this thesis. The example results could be found in Figure 3.4, and after the preprocessing steps, the primary edges in the target region have become particularly prominent.

To pre-process images, the method provided by Abdelsalam et al. (2017) was applied. The calibration phase addressed radial and tangential distortions inherent to the camera lens by estimating constant parameters through image capture of a calibration pattern, utilizing Zhang (2000). Following calibration, the visual image was partitioned into user-defined rectangular cells for grayscale conversion, with grid size influencing detection accuracy. Noise reduction was then applied using the bilateral filter (Tomasi and Manduchi, 1998), which was particularly effective in preserving edges. Finally, intensity gradients were exposed using an algorithm based on the approach of Rao and Schunck (1991) for estimating the orientation field from flow visualization images.

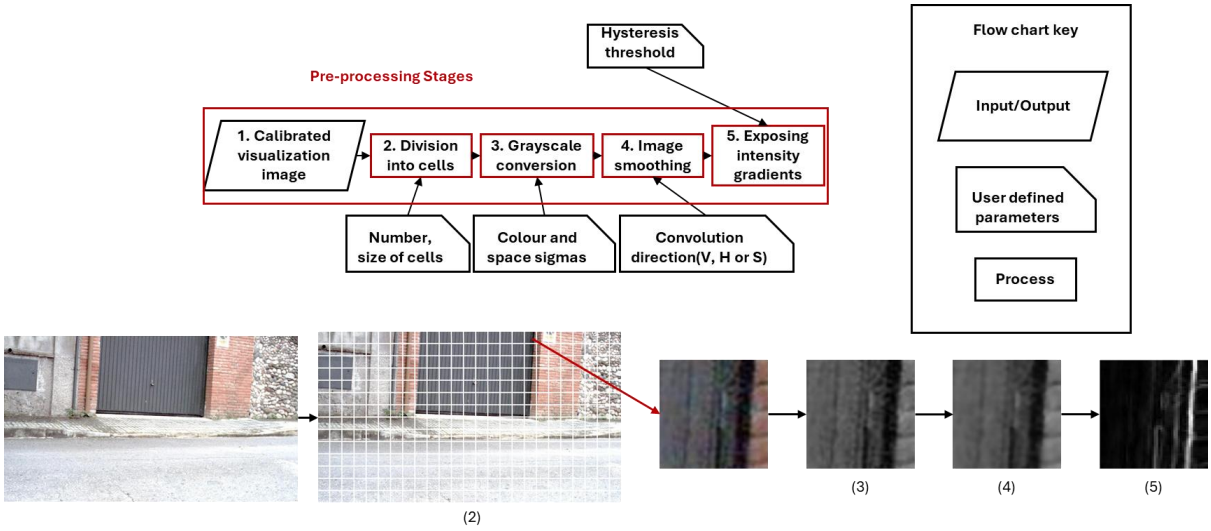



**Figure 3.4 Flow Chart of Data Pre-processing**

## 3.4 Dataset preparation

It was difficult to build a sufficiently rich training dataset through experiments alone. Therefore, various methods were used to expand the database for different research purposes. For instance, for edge detection, the training set consists of a pre-training set from BIPED and a fine-tuning set composed of manually annotated experimental data. For predicting the flow field, the training set is a mixture of experimental data and synthetic data generated by cGANs.

The main challenge in developing a robust edge detection algorithm is that the algorithm must learn to recognize a wide variety of SFV cases. For example, the images to be

recognized may include laminar flow on a flat plate, surface flow visualization on an aerofoil, or flow around a cylinder. Other numerous factors, such as image resolution, lighting conditions, and noise levels, might also significantly impact the performance of the edge detection algorithm. A robust edge detection algorithm should also be able to handle images with unpredictable flow conditions.

To ensure robust model generalization, a standard three-way split of the dataset into training (80%), validation (10%), and test (10%) subsets was adapt in both Chap 5 and Chap 6. The training set ensures broad coverage of learning scenarios. The validation set guides hyperparameter tuning and monitors training progress to mitigate overfitting. And the test set remains entirely isolated, providing an unbiased assessment of the model's performance on unseen data.

### 3.4.1 Edge detection dataset

The edge detection dataset consisted of a pre-training dataset and a fine-tuning dataset. The pre-training dataset was derived from the open-source BIPED dataset. The Barcelona Images for Perceptual Edge Detection (BIPED) dataset comprises 250 high-definition outdoor images, each with a resolution of 1280×720 pixels. These images have been carefully annotated by experts in the field of computer vision, ensuring that redundant information was not considered. Nevertheless, all results have undergone cross-checking to correct any potential errors or mislabelling. This dataset, developed due to the lack of datasets specifically for edge detection tasks, has been publicly released as a benchmark for evaluating edge detection algorithms. In contrast, the experimental dataset exhibits greater specificity based on flow visualization characteristics. For instance, samples contain low-contrast transitional regions (such as gradual interfaces between low-speed wake zones and main flow regions). While such features are typically treated as noise in conventional edge detection datasets, they hold significant diagnostic value in flow field analysis. In practice, the MDBD dataset (Mély et al., 2016), which is commonly used for edge detection tasks, features edges annotated by various contributors; however, these annotations are not validated, and in some instances, the edges correspond to incorrect annotations. The level of detail in the annotations for the dataset used in the current work can be observed in the ground truth images in Figure 3.5.

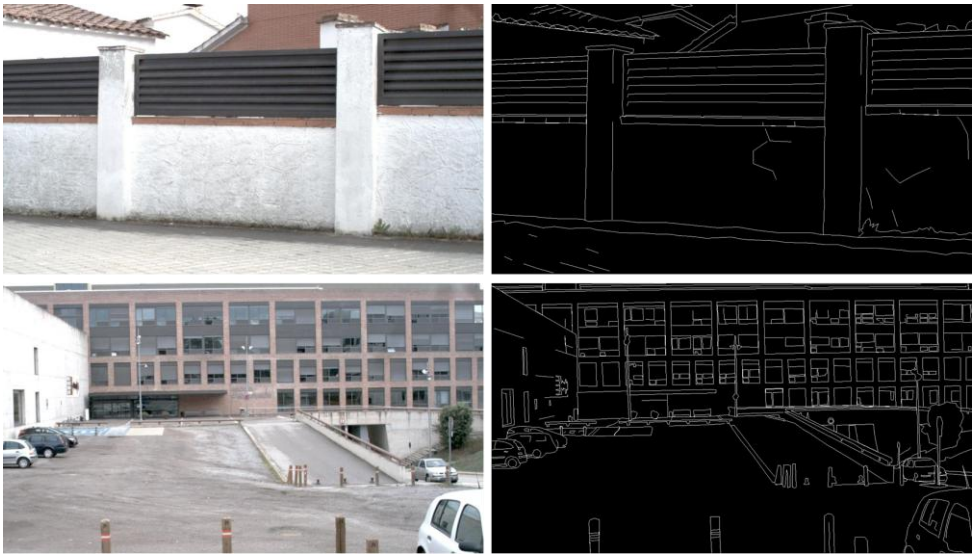



**Figure 3.5 Edge-maps from BIPED Train Dataset**

To increase the number of training images, data augmentation was performed: 1) Given the high resolution of the BIPED dataset images, they were split along half the width of the image; 2) Similar to the HED method, each generated image was rotated through 15 different angles and cropped to fit predefined rectangular areas; 3) Images were horizontally flipped; 4) Two gamma corrections were applied (0.3030, 0.6060). This augmentation process generated 288 enhanced images from every 200 original images.

The fine-tuning dataset has involved processing authentic flow visualization images, as Figure 3.6 shows. Initially, Canny edge detection was used to detect edges, followed by manual selection of the most suitable threshold for accuracy. After this, the resultant edge map was refined further to remove excessively short edges and connect discontinuities through manual processing, rather than relying on automated morphological operations. This process has yielded the final edge map. The images in the dataset have then undergone pre-processing steps.

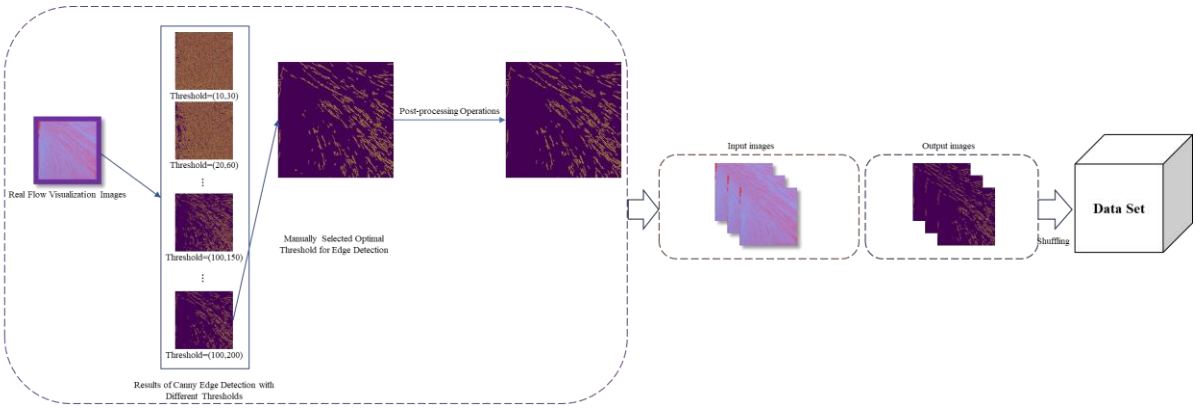



**Figure 3.6 Work Flow on Preparing Fine-tuning Data**

### 3.4.2 Flow field prediction dataset

This section discusses the construction, composition, and role of the flow prediction dataset in performance evaluation. This dataset was designed to provide high-quality training and testing data for image generation and prediction tasks in flow visualization, especially in wind tunnel experiments. In wind tunnel experiments, flow field visualization data is often constrained by experimental conditions and equipment limitations. It is costly and time-consuming to obtain large-scale, high-precision experimental images Therefore, the dataset combines high-precision images obtained from actual experiments with synthetic images generated by the pix2pix model. By doing this, the dataset not only preserves the true physical properties of the experimental data, but also enhances its diversity and quantity.

The dataset construction process began with the collection of 30 experimental images from the Durham cascade and Plint wind tunnel experiments. These images were pre-processed and served as the basic material for the input dataset. Then the pix2pix generation model was used to generate corresponding SFV images based on these experimental images. In this process, the model learned the flow field properties from the experimental images and generated synthetic images.

Figure 3.7 shows a typical result generated by the pix2pix generative model, where the left side is the input image, the middle is the flow field prediction image generated by the pix2pix model, and the right side is the real image obtained by the experiment.

Since the flow field visualization task involves style transfer, there may be subtle differences between two flow field images even under the same experimental conditions. Therefore, traditional quantitative evaluation methods, like Mean Squared Error (MSE), are

not suitable for this task. To address this issue, the Structural Similarity Index (SSIM) has been chosen as the primary performance evaluation metric.

SSIM is a perceptual image quality evaluation method that effectively measures the similarity between generated images and real images in terms of brightness, contrast, and structure. The SSIM value could be calculated by:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{3.7}$$

where $x$ and $y$ represent the generated and real images, respectively, $\mu_x$and $\mu_y$are the mean intensity of the two images, $\sigma_x^2$ and $\sigma_y^2$ are the intensity variances, and $\sigma_{xy}$ is the covariance between the images. The constants $C_1$ and $C_2$ are included to avoid division by 0. SSIM considers the similarity of brightness, contrast, and structure, making it more suitable than simple pixel difference measures like MSE for evaluating the overall visual quality and structural fidelity of images.

In flow field visualization tasks, SSIM effectively reflects whether the generated image retains key features of the flow field, such as vortices, streamlines, and separation points. Specifically, by calculating the SSIM value between the generated image and the corresponding experimental image, the similarity in visual structure can be quantified. A higher SSIM value indicates that the generated image is closer to the real experimental image. This suggests that the model can effectively reproduce important physical features of the flow field.

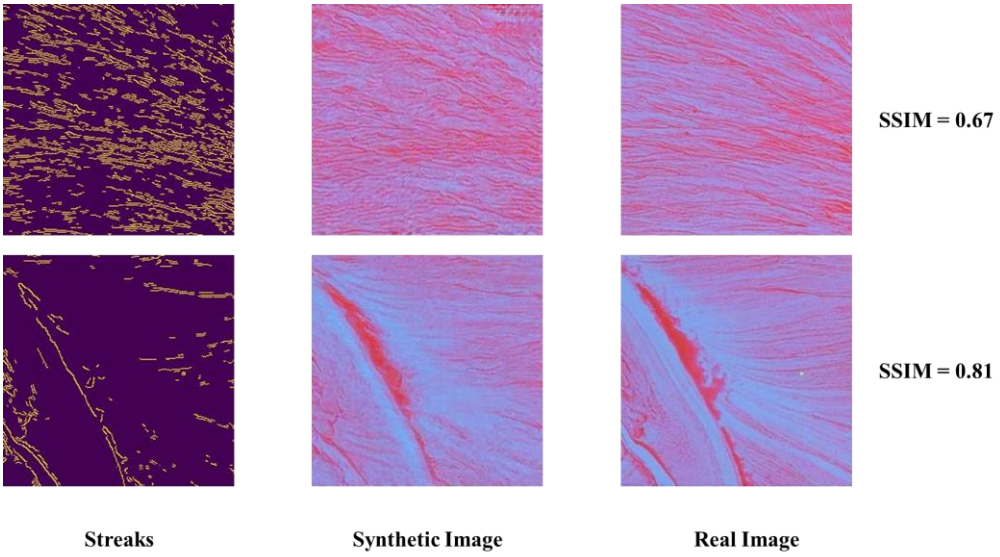



**Figure 3.7 Comparison of Experimental Flow Visualization Images and pix2pix Model Predictions**

The synthetic image dataset used for training, testing and validation could be found in:

https://github.com/Dehakaa/synimage

This dataset is publicly available for further research and experimentation, enabling other researchers to replicate and build upon this work.

## 3.5 Neural networks techniques

This section presents an overview of the neural network architectures utilized in this work, offering a general understanding of their design and implementation. The specific training details for each network will be discussed in their respective chapters later in this thesis. The aim is to summarize the key features of each architecture and how they contribute to achieving the research objectives.

All training in this work was conducted on the NVIDIA CUDA Centre (NCC), with training details provided in the respective chapters. NCC has been purchased through Durham University's strategic investment funds, and is installed and maintained by the Department of Computer Science.

### 3.5.1 Convolutional neural networks

This section introduces the proposed network architecture and give details on the different modules. Before designing this custom network, several existing pre-trained models such as HED and EdgeNet were tested. While some models showed promising results in terms of edge detection accuracy, they did not fully meet the requirements. Therefore, a custom network with fewer parameters was designed tp ensure better generalization and more efficient learning of the relevant edge features. The proposed model was based on Lightweight Dense Convolutional neural network (LDC) by Soria et al. (2022), and the following presents the proposed modifications in detail. The LDC architecture comprises two subnets: the Dense Extreme Inception Network (Dexi) and the upsampling network (USNet). Dexi consists of 4 blocks, which acts as an encoder, and the skip-connections couple the third and fourth blocks as well as their sub-blocks. USNet, on the other hand, is a conditional CNN, which acts as a decoder and transforms feature maps into edge maps, matching the size of the input image of the Dexi subnet.

As shown in Figure 3.8, each block contains two convolutional layers. The size of all convolutional kernels is 3×3, and the number of kernels is indicated accordingly. In addition, the first block has a notation 's2', which signifies that the stride of the convolutional layer is 2. Each convolutional layer is followed by batch normalization and a Rectified Linear Unit (ReLU). Starting from block-3, the last convolutional layer of the last sub-block does not include the ReLU function. The red rectangles represent the use of 3×3 convolutional kernels with a stride of 2 in the max-pooling operation. Compared to LDC, this work increases the number of convolutional kernels in block-4. Additionally, Inspired by Poma et al. (2020)'s work, the four intermediate edge-map predictions are fused to generate a single edge-map. This edge map can be treated as a special form of image, represented as a 2D array, where each pixel encodes the presence or absence of an edge at that spatial location. Its dimensions correspond to those of the input image or feature map being processed. The configuration of the loss function for the network mentioned in this work is similar to that of the LDC network. Detailed information can be found in Soria et al. (2022).

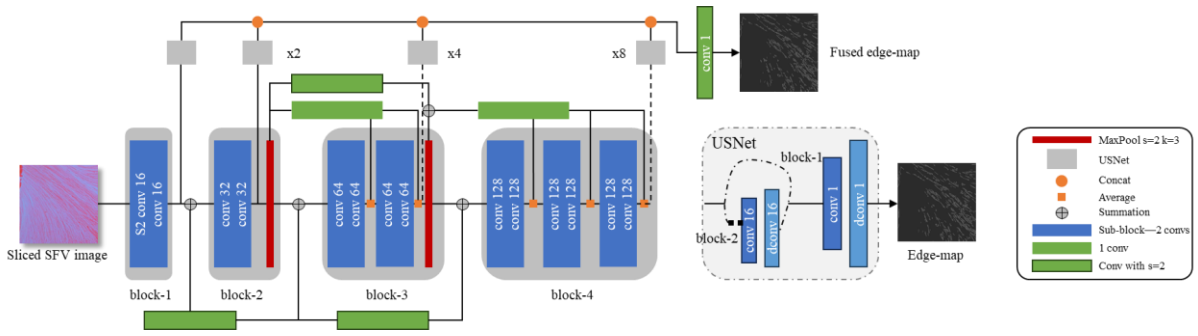



**Figure 3.8 Architecture of the proposed CNN based model.**

From a model interpretability viewpoint, the shallow layers (block-1 and block-2 in Figure 3.8) primarily capture low-level visual features such as edges and corners. These features are generalizable and transferable to other visual tasks beyond flow field analysis. Intermediate layers (block-3 in Figure 3.8) begin to encode local shapes and texture patterns, which could offer moderate generalizability across similar domains. And the deeper layers (block-4 in Figure 3.8) learn high-level extract high level abstract features such as vortex cores, streamlines, or saddle points.

Due to the limited number of images obtained from the experiment, the network was first pre-trained using the BIPED dataset to better leverage its performance. The pretraining was performed for 50 epochs, using a binary cross-entropy loss function with a batch size of 8

and an initial learning rate of 1e-4. To preserve low-level texture features, the first two convolutional blocks (Block-1 and Block-2) were frozen during the initial pretraining phase, while the remaining layers were kept trainable.

The edge map was derived by initially applying Canny edge detection to the experimental image, followed by manual removal of excessively short and overlapping streaks. The sliced SFV images were then input into this pre-trained network, the output of which was its edge feature map. This provided all the edge information in the image and was a binary image. The values of all edge positions were 1, and the background values were 0. This information could not be directly used to compute the flow field. Therefore, some further processing of these data was required.

### 3.5.2 U-net

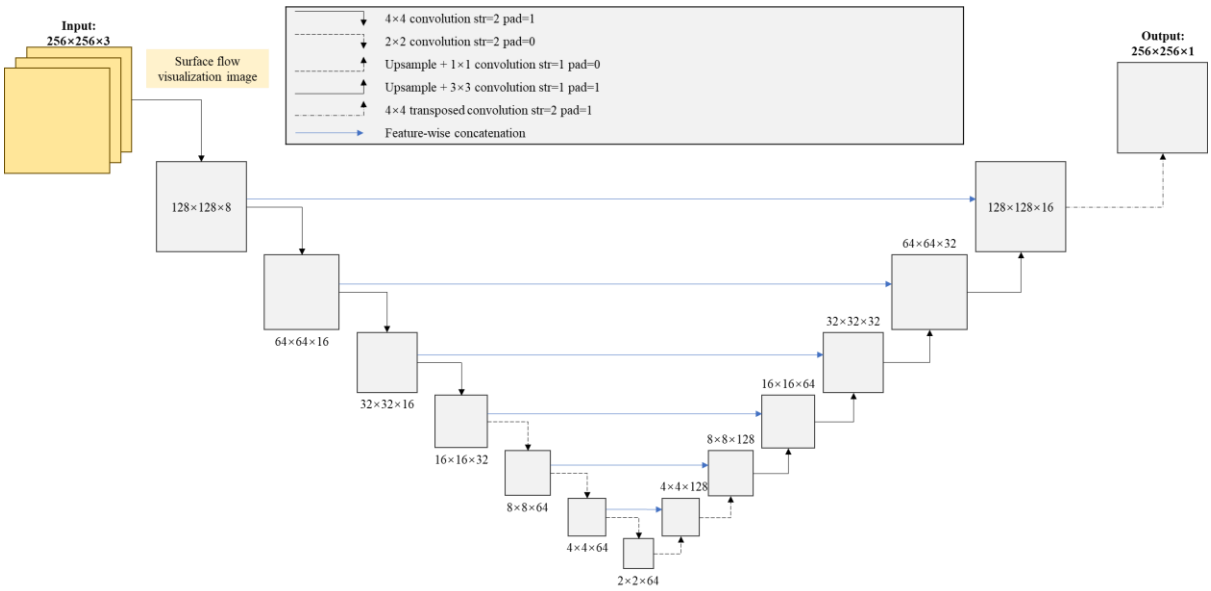



**Figure 3.9 An overview of the deep neural network architecture with input and output specifications.**

The deep neural network (DNN) model is based on a U-net architecture, a convolutional network originally used for the fast and precise segmentation of images, and later used for the inference of flow fields and the development of DNN models.

As inputs for the learning task $\breve{y} = \breve{f}(\xi)$ , where $\breve{y}$ represents the predicted orientation field, while $\xi$ denotes the input image data. Hence, the input becomes $m \times n \times 3$ , where $m$ and $n$ respectively represent the length and width of the input image. In the encoding part, 7 convolutional blocks are used to transform the input into a single data point with 1024 features.

The decoder part of the network is designed symmetrically with another 7 layers in order to reconstruct the outputs with the desired dimension, i.e. $m \times n \times 1$, corresponding to the orientation field variables. Leaky ReLU activation functions with a slope of 0.2 is used in the encoding layers, and regular ReLU activations in the decoding layers.

The current method has the potential to be extended to other scenarios, such as incorporating inflow boundary conditions as inputs, and adding skin friction magnitude as outputs. This would require transformations of these variables to fit the input and output formats of U-net, and implementing corresponding convolutions within the U-net architecture.

### 3.5.3 Generative adversarial networks

The key success of GANs lies in their novel framework, which utilizes adversarial loss to force generated images to be indistinguishable from ground truth images. GANs are generative models that learn a mapping from random noise vector z to output image y, $G: z \rightarrow y$. In contrast, cGANs learn a mapping from observed image x and random noise vector z, to y, $G: \{x, z\} \rightarrow y$. The generator G is trained to produce outputs that cannot be distinguished from "real" images by an adversarial trained discriminator, D, which is trained to do as well as possible at detecting the generator's "fakes".

The objective of a conditional GAN can be expressed as

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}\left[\log\left(1 - D\left(x, G(x, z)\right)\right)\right] \tag{3.8}$$

where $\mathbb{E}_{x,y}[\log D(x, y)]$ means expected value of $\log D(x, y)$ given (x, y) and G tries to minimize this objective against an adversarial D that tries to maximize it, i.e $G^* = argmin_G max_D \mathcal{L}_{cGAN}(G, D)$

Isola et al. (2017)further improved cGANs, and the corresponding software was called pix2pix. In cGANs, the generator learns the mapping from random noise z to G(z). In contrast, in the generator of pix2pix, there is no input noise z. A novelty of pix2pix is that its generator learns the mapping from observed image y to output image G(y). In this case, the objective of cGANs is expressed as following:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{y}\left[\log\left(1 - D\left(y, G(y)\right)\right)\right] \tag{3.9}$$

In terms of the loss function, Isola et al. opted for the Manhattan distance (L1 distance) instead of the Euclidean distance (L2 distance) to generate less blurry results:

$$\mathcal{L}_{L_1}(G) = E_{x,y}[\|x - G(y)\|_l] \tag{3.10}$$

Therefore, the final objective is defined as:

$$G^* = argmin_G max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L_1}(G) \tag{3.11}$$

If there is no noise z, the network can still learn the mapping from x to y, but it will produce deterministic outputs, thus unable to match any distribution other than the Dirac delta function, which is a mathematical construct used to represent a specific point in a probability distribution, either discrete or continuous. However, this is advantageous for the purpose of this work, as for a flow visualization image, there should be a deterministic flow field corresponding to it. Therefore, in the final model of this work, noise is provided in the form of dropout at multiple layers of the generator, which is applied during both training and testing phases.

## 3.6 Overview

This chapter provides a comprehensive overview of the experimental methodologies and data processing techniques employed in this thesis. It introduces the experimental setup used to obtain flow visualization and skin friction measurements, specifically the Durham Cascade and the 0.5 Plint wind tunnel. Various techniques, such as oil/dye visualization and GLOF skin-friction measurement, are described to illustrate how flow patterns were captured and analysed. CFD simulations using ANSYS Fluent and OpenFOAM were employed to construct a comprehensive dataset for neural network training and validation.

The CNN architecture was employed in Chapter 4 to extract edge features from flow visualization images. Specifically, a modified lightweight Dense CNN (based on the work of Soria et al., 2022) was employed. This network was pre-trained on the BIPED edge detection dataset and fine-tuned with experimental data. The CNN can accurately extract key features such as separation lines and vortices from surface flow images. Its architecture consists of four convolutional blocks, incorporating skip connections, batch normalization, and ReLU activation functions to optimize feature extraction.

In Chapter 6, to reconstruct the complex flow fields, the U-Net architecture was introduced. This network comprises symmetric encoder-decoder paths, each containing seven

convolutional blocks. The U-Net transforms experimental images and synthetic flow field data from OpenFOAM simulations into quantitative directional fields. GANs were also employed to solve the issue of insufficient experiment data. The sGAN was used to generate synthetic flow field images, where the network learned a deterministic mapping from experimental images to corresponding flow fields. Due to the physical nature of flow visualization, the model didn't use random noise as input. The generator incorporates Dropout layers during both training and testing phases to improve generalization, while the discriminator ensures the physical plausibility of generated images through adversarial training.

The integration of these neural networks with experiments and CFD simulations forms a comprehensive framework for flow field analysis. Experimental images provide real-world data for the networks, GLOF measurements offer time-resolved data for validating fluid friction fields, and CFD simulations both verify experimental results (Chapter 5) and supply synthetic training data for the U-Net (Chapter 6).

# Chapter 4 Extracting Streamline Data from Surface Flow Images Using CNNs

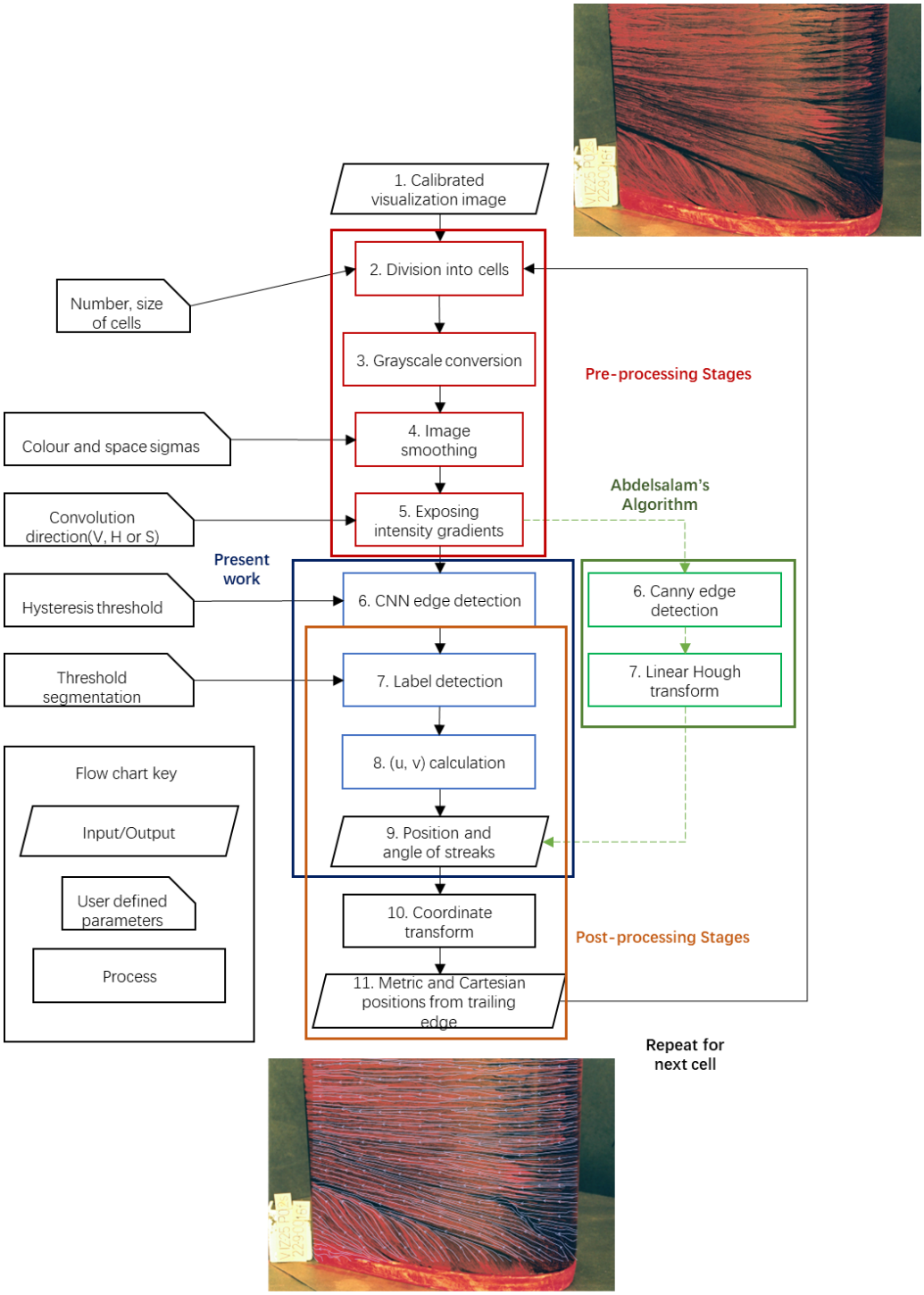



**Figure 4.1 Flow chart of streakline detection algorithm**

This chapter presents an enhanced method for reconstructing flow fields from SFV images, building on the approach proposed by Abdelsalam et al. (2017), By introducing a

CNN-based edge detection technique, the new method addresses limitations in accurately discerning the orientation of streaks and mitigating the formation of scattered noise vectors in the original work. This work has also been published in the Global Power and Propulsion Society (GPPS) conference (Liu et al., 2024).

Based on the preliminary findings presented in a previous work at GPPS, where a CNN-based model was trained to automate streamline detection and flow field reconstruction, this chapter investigates various stages of preprocessing methods and evaluates their impact on the corresponding outputs at each stage. Additionally, another CNN architecture inspired by Dexi and USNet is proposed to further enhance the robustness of the methods. This expanded research addresses the limitations identified in the earlier study, leading to improvements in the efficiency of recognizing flow visualization images.

The principal stages of the algorithm are shown in Figure 4.1.

The algorithm is based on Abdelsalam's processing method, if steps 6 and 7 were replaced with Canny Edge Detection and Linear Hough Transform, and step 8 was skipped (Green box in Figure 4.1), the method would be identical to Absdelsalam's technique. Thus, as well as the differences in edge detection algorithms compared to Abdelsalam, this work also employs a distinct methodology for the subsequent construction of the flow field.

## 4.1 Image Preprocessing Techniques

As shown in Figure 3.4, the image preprocessing stage consists of several steps, including calibrated visualization image, division into cells, grayscale conversion, image smoothing, and exposing intensity gradients. These steps are designed to prepare the images for further analysis by improving their quality and reducing distortions and noise. Together, these preprocessing steps lay a solid foundation for subsequent edge detection and feature extraction.

### 4.1.1 Calibrated Visualization Image

In this work, although the *Nikon D5300 DSLR* and *Nikon Z50* cameras used were equipped with built-in automatic calibration functions, manual calibration was opted for to ensure the accuracy and reliability of the experimental results. The automatic calibration function relies on lens distortion models pre-set in the camera firmware, typically obtained through laboratory testing, and is designed to correct radial and tangential distortions of the

lens. While this method is sufficient for many everyday applications, in scientific research involving precise measurements, automatic calibration may not fully eliminate all geometric distortions, especially in scenarios requiring high precision.

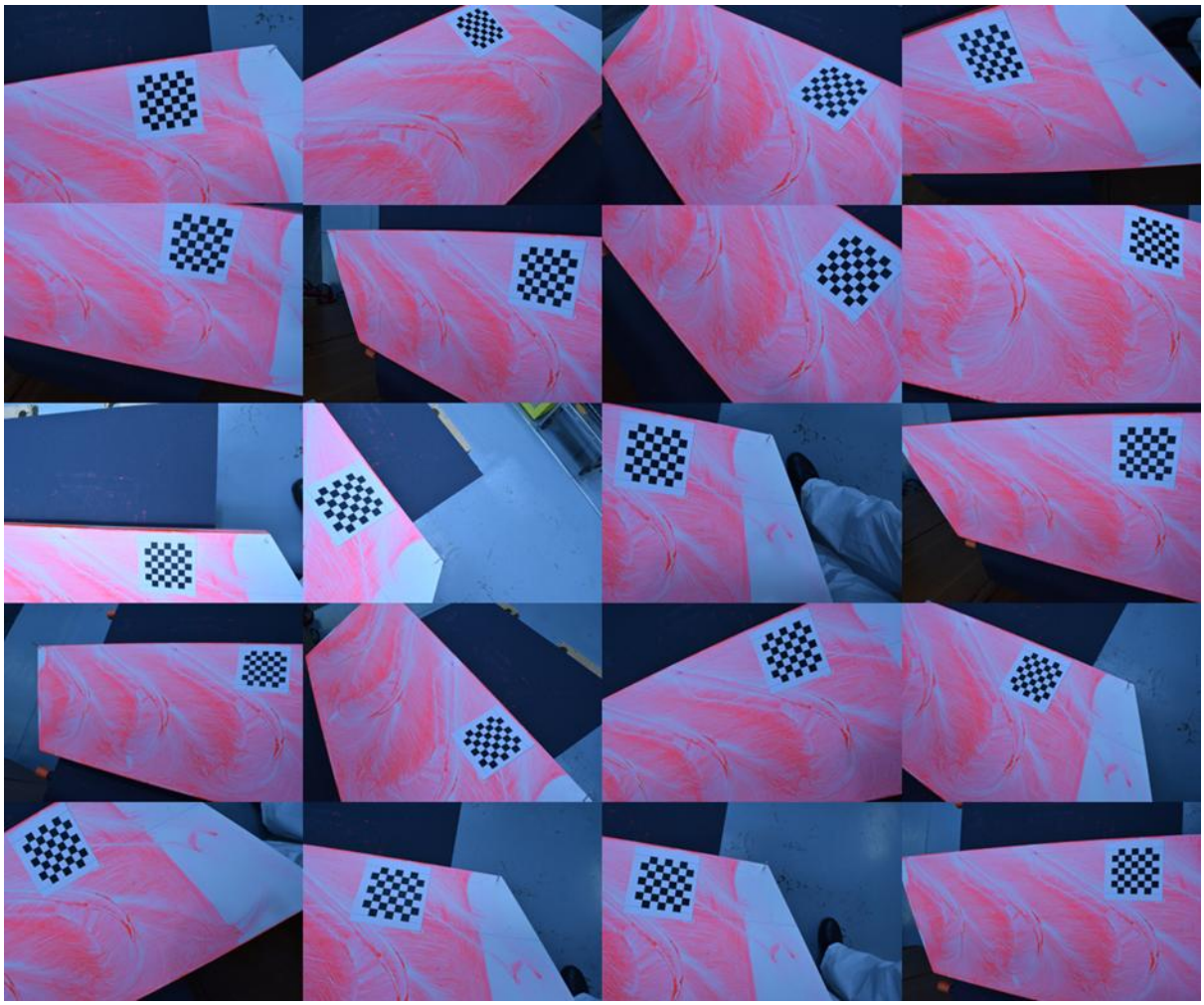


**Figure 4.2 Images used for chessboard calibration**

The basic principle of automatic calibration is based on lens distortion models pre-stored in the camera hardware. These models are obtained through calibration tests in the laboratory and are adjusted in real-time according to image characteristics during the capture process. However, this type of calibration is usually based on general scenarios and may not cover all possible distortion cases under specific conditions that demand high precision. As mentioned in Hartley and Zisserman (2003)'s *Multiple View Geometry*, any slight uncorrected distortion could lead to significant errors in precise measurements.

To further enhance measurement accuracy, the calibration tools in the OpenCV library (Bradski, 2000) were used for manual calibration. This method is based on Zhang (2000)'s

calibration theory, which is widely applied in the field of computer vision and provides an effective way to correct the camera's intrinsic and distortion parameters using known planar patterns, such as chessboards. In the experiment, a 7×7 chessboard pattern was used, with each square measuring 10 mm. By capturing images of this pattern from multiple angles, the pixel coordinates of the chessboard corners were obtained, which were then associated with their known positions in the actual three-dimensional space. A system of nonlinear equations was constructed, and the intrinsic and distortion parameters of the lens were estimated using optimization algorithms such as the least squares method, thereby achieving high-precision calibration.

The core of manual calibration lies in using images from multiple viewpoints to eliminate potential systematic errors. Images from multiple viewpoints not only provide more comprehensive information to determine the parameters of the camera model but also enhance the stability and accuracy of the computational results through redundant data.

After manual calibration, the effectiveness of the calibration was validated by measuring the physical distance between two points. As shown in Figure 4.3, the results indicate that the measurement error before calibration was 3.29%, whereas after manual calibration, the error was reduced to 0.11%. This outcome aligns with expectations and demonstrates the effectiveness of manual calibration in eliminating potential distortions.

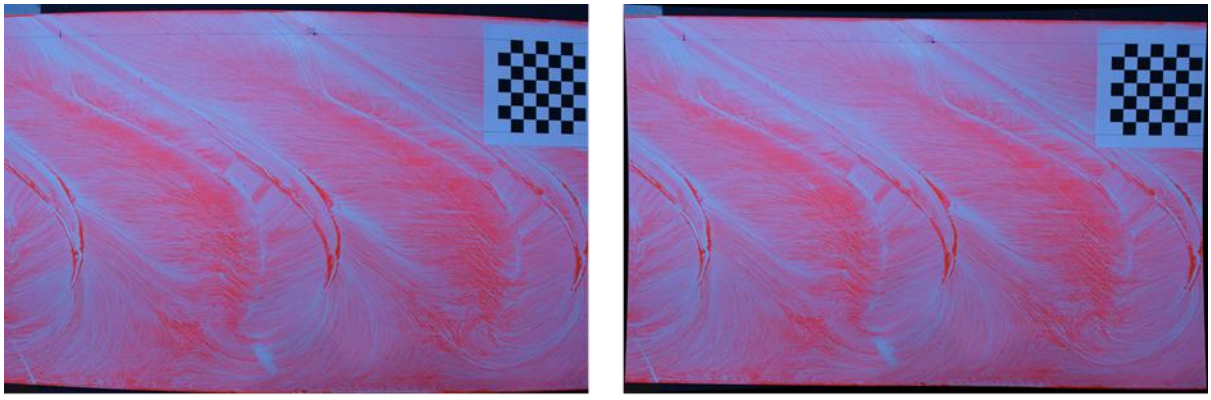



**Figure 4.3 Calibration results. (a) Original image; (b) calibrated image.**

### 4.1.2 Division into Cells and Grayscale Conversion

After completing the calibration, a calibrated image like Figure 4.3 could be obtained, segmentation and grayscale conversion were performed on the target image. These operations

were aimed at simplifying the computations and enhancing processing efficiency. Since the topological relationships within the image do not need to be considered during processing, dividing the image into smaller segments effectively reduces the computational load. This approach of processing smaller image segments not only offers significant advantages in lowering computational complexity but also speeds up the algorithm's execution. Based on experience and practice in the field of image processing, block processing is a common optimization strategy, particularly useful for the rapid handling of large-scale data. During the grayscale conversion, single-channel processing was chosen. This decision was made because red dye was used as a marker in this experiment, so extracting only the red channel of the image is sufficient to meet the research requirements. The individual R, G, and B values from the original RGB image were simply separated, treating each as an independent grayscale image.

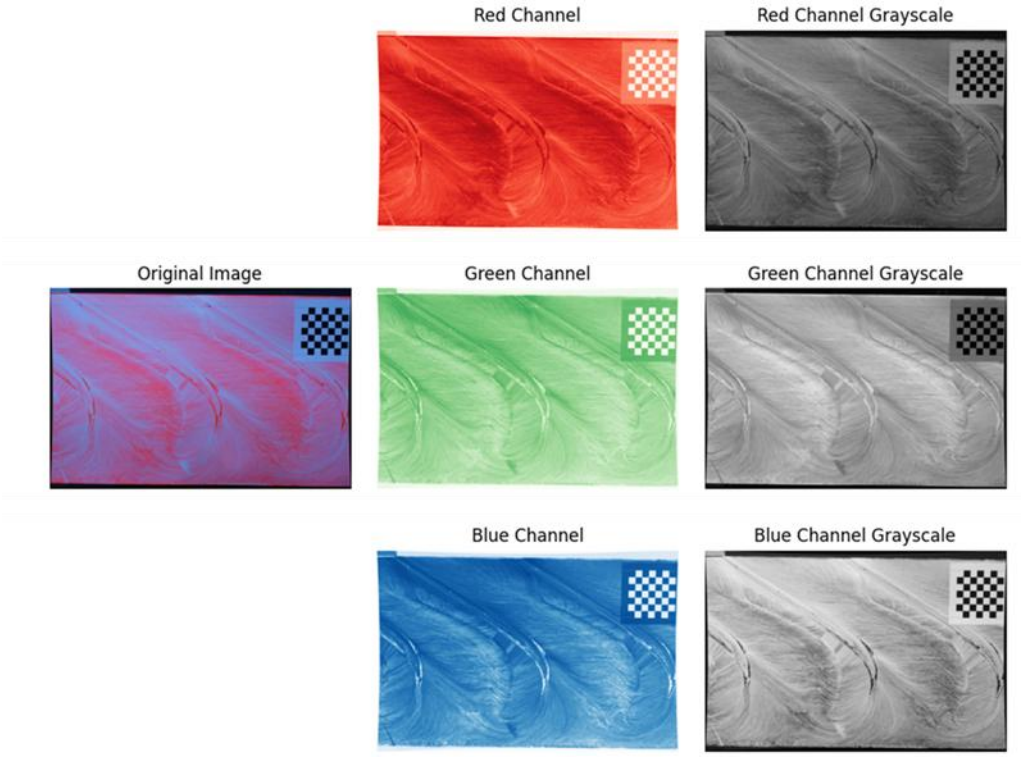



**Figure 4.4 Separation of an RGB image into its individual grayscale channels (R, G, and B), with pseudo-colouring applied for visualization purposes.**

The Figure 4.4 shows the process of splitting an RGB image into its individual colour channels, followed by converting each of these channels into grayscale. The original image is first divided into red, green, and blue channels, where the red channel stands out due to its association with the experimental conditions. After converting the channels into grayscale, the

red channel in grayscale shows the most significant contrast, especially when compared to the green and blue channels.

This enhanced contrast in the red channel is consistent with the use of red dye as a marker in the experiment. By isolating and processing only the red channel, which contains the most critical information, the grayscale image highlights the key regions more effectively. The decision to focus on the red channel simplifies the processing, as the single-channel approach reduces computational complexity while maintaining the necessary accuracy for segmentation and analysis.

In comparison, the grayscale images for the green and blue channels offer less pronounced contrast and detail, supporting the decision to prioritize the red channel. This targeted approach optimizes the image processing workflow, ensuring that the relevant information is extracted efficiently and accurately without unnecessary computational overhead.

### 4.1.3 Image Smoothing

Image smoothing is a preprocessing step aimed at reducing noise and unnecessary fluctuations in image data, which could obscure important features or lead to incorrect analysis results. Smoothing techniques work by averaging the intensity values of neighbouring pixels, thereby reducing high-frequency variations and enhancing the visibility of important structures within the image. This process is particularly important in image analysis tasks, as sharp changes in pixel intensity (often caused by noise) can lead to misleading gradient calculations and subsequent errors in edge detection or feature extraction.

Unlike the commonly used Gaussian filter, this work employs a bilateral filter for smoothing. The bilateral filter achieves edge-preserving smoothing by combining domain and range filtering. The intensity value of each pixel is replaced with a weighted average of the intensity values of nearby pixels. The weights are determined by two factors: spatial proximity and intensity similarity. Spatial weight ensures that only pixels close to the target pixel significantly affect the output, while range weight ensures that only pixels with similar intensities meaningfully influence the result.

Mathematically, the bilateral filter can be expressed as:

$$I'(x) = \frac{1}{W(x)} \sum_{x_i \in \Omega} I(x_i) \cdot \exp\left(-\frac{|x_i - x|^2}{2\sigma_S^2}\right) \cdot \exp\left(-\frac{|I(x_i) - I(x)|^2}{2\sigma_T^2}\right) \quad (4.1)$$

Here, $I'(x)$ represents the intensity of the filtered pixel $x$, $I(x_i)$ denotes the intensity of the pixel $x_i$ within the neighbourhood $\Omega$, $\sigma_S$ controls the influence of spatial distance, $\sigma_T$ controls the influence of intensity differences, and $W(x)$ is the normalization factor, defined as:

$$W(x) = \sum_{x_i \in \Omega} \exp\left(-\frac{|x_i - x|^2}{2\sigma_S^2}\right) \cdot \exp\left(-\frac{|I(x_i) - I(x)|^2}{2\sigma_T^2}\right) \quad (4.2)$$

The main advantage of the bilateral filter is its ability to smooth the image while preserving edges, which is crucial for visual analysis tasks. Unlike the Gaussian filter, which may blur both edges and noise, the bilateral filter preserves significant edge details by taking into account the spatial and photometric distances between pixels.

### 4.1.4 Exposing Intensity Gradients

To extract intensity gradients within the image, the Sobel operator was employed, a widely recognized method in image processing for detecting edges. The Sobel operator calculated the gradient of the image intensity in both the horizontal and vertical directions, effectively identifying regions where intensity changed sharply—indicative of edges or significant structural features.

This process involved the application of two convolution kernels designed to detect intensity changes along the x-axis ($\boldsymbol{G_x}$) and y-axis ($\boldsymbol{G_y}$). The kernels are defined as follows:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (4.3)$$

These kernels are convolved with the input image I to obtain the gradients in the $x$ and $y$ directions:

$$Sobel_x = I * G_x, \quad Sobel_y = I * G_y \quad (4.4)$$

The resulting gradient maps highlight intensity variations in the respective directions. To obtain a comprehensive gradient magnitude that reflects the overall intensity change at each pixel, these directional gradients are combined using the following formula:

$$|\nabla I| = \sqrt{(Sobel_x)^2 + \left(Sobel_y\right)^2} \tag{4.5}$$

The gradient magnitude map, shown in the right panel of Figure 4.5, reveals the edges and other prominent features within the image by emphasizing areas of significant intensity variation. This map is critical for understanding the underlying structure of the image, as it highlights the boundaries and transitions that may be less apparent in the original image.

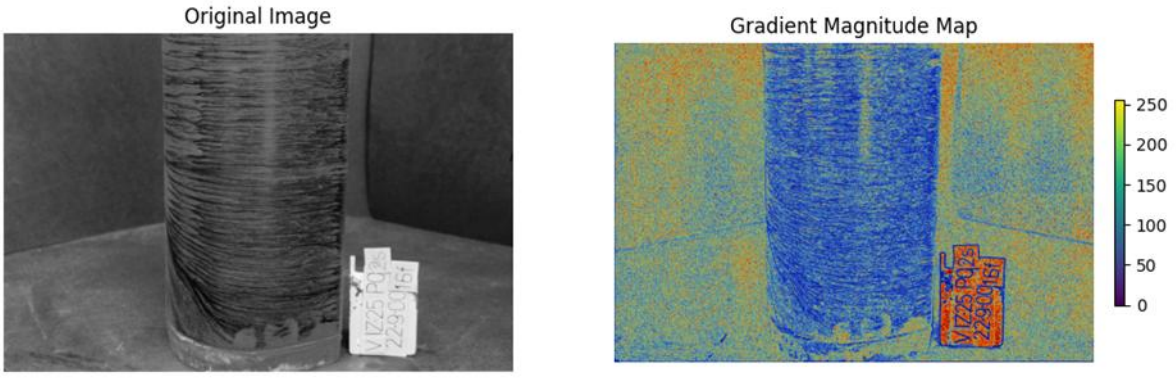



**Figure 4.5 Image intensity gradients results.**

As illustrated in Figure 4.5, the original image (left) and the computed gradient magnitude map (right) are presented side by side for direct comparison. The gradient map provides a clear visualization of intensity transitions, which are essential for further analysis, such as edge detection and feature extraction. This comparison demonstrates the effectiveness of the Sobel operator in exposing the critical intensity gradients that define the structural features of the image.

## 4.2 CNN Architecture for Edge Detection

This section introduces the design and implementation of the EdgeNet architecture for edge detection, which integrates two main components: Dexi and USNet. It describes how Dexi uses skip connections to preserve edge details through deep layers and explains the role of USNet in upsampling feature maps to match the ground truth resolution.

### 4.2.1 Design Considerations

As mentioned in section 3.5.1, the CNN network in this chapter, called EdgeNet consists of two main components: Dexi and USNet.

The Dexi architecture, initially proposed by Poma et al. (2020), uses an end-to-end training framework rather than relying on pre-trained object detection models like VGG16. In these existing networks, edge features recognized in the shallow layers are often not retained in the deeper layers, resulting in the loss of important information. Under the Dexi architecture, the model can learn how to perform edge detection directly from the specified dataset without being affected by biases introduced by unknown pre-trained weights.

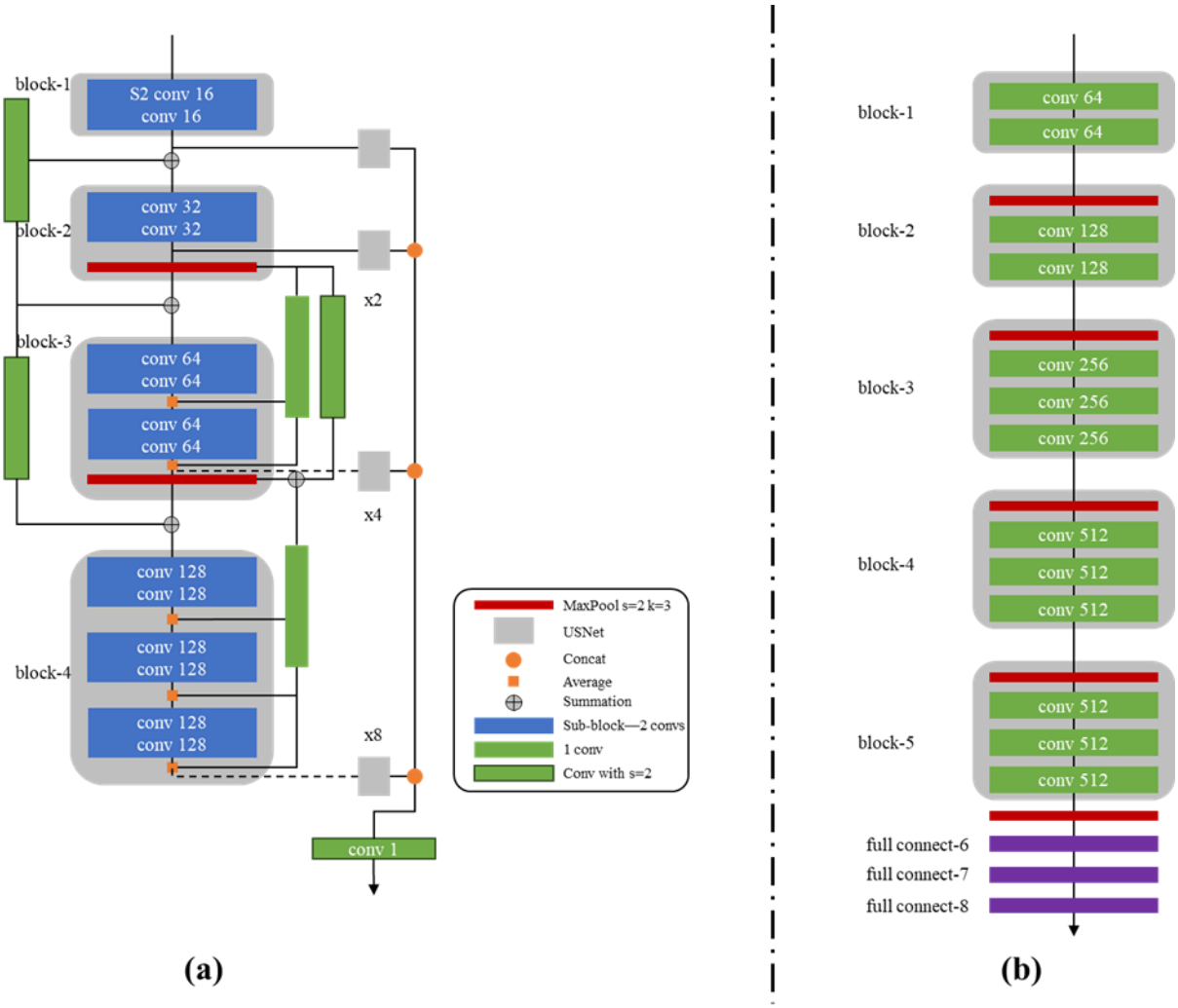



**Figure 4.6 Flowchart of (a) EdgeNed architecture and (b) normal CNN architecture**

A notable design feature of Dexi is the introduction of parallel skip connections inspired by the Xception architecture. These skip connections is critical in preserving edge-related information as it propagates through the network. By connecting the early and later layers, they help retain edge features that might otherwise be lost during deep convolutional operations. This approach ensures that the model maintains a high level of detail in its edge predictions, which is particularly important for fine-scale edge detection.

In the USNet component, the design focuses on effectively upsampling the feature maps generated by the Dexi part to match the resolution of the ground truth edge maps. USNet gradually upsamples the feature maps to the required scale by combining convolutional and deconvolutional layers.

By closely integrating the designs of Dexi and USNet, the EdgeNet model performs exceptionally well across various scenarios and datasets, providing precise and detailed edge detection results. The EdgeNet model was trained on the BIPEDv2 dataset, which is described in detail in the Section 3.4.1(p.42). The fine-scale edge annotations in BIPEDv2 allow the model to accurately detect edges even in images significantly different from the training set, giving the pre-trained EdgeNet model considerable generalization ability for new scenes.

### 4.2.2 Layers and Features for Image Analysis

In the EdgeNet architecture, image analysis relies on the close collaboration between different layers, particularly the effective integration of the Dexi and USNet components The core architecture of the Dense Extreme Inception Network (Dexi) which functions as an encoder, consisting of six blocks, as shown in Figure 4.7. These blocks are made up of multiple sub-blocks, each containing convolutional layers that generate feature maps. Skip connections are introduced to link blocks and sub-blocks, ensuring that the features computed in the early layers are preserved and integrated as data is passed through the network. Due to the regularity in the flow field's directional patterns, the absence of very complex boundary cases, and the limited size of the available training dataset, modifications have been introduced in this work to reduce computational costs and mitigate the risk of overfitting. Additionally, the output layer has been improved to produce a binarized edge map. The architecture of the EdgeNet could be found in Figure 4.6. It can be seen that the architecture on the left side of the Figure 4.6 presents a deep convolutional neural network used for edge detection, divided into multiple convolutional blocks, each containing several convolution layers (conv) and pooling layers (max-pooling). Feature maps are extracted layer by layer and downsampled through different convolution operations, with feature aggregation between layers leading to edge detection output. The architecture of Figure 4.7, also used for edge detection, places more emphasis on multi-scale feature fusion between convolutional blocks. It combines feature maps at different scales (×2, ×4, ×8, ×16) to further refine these multi-scale features, producing a more detailed edge map. Both architectures employ convolution and pooling operations, but the DexiNed architecture focuses more on multi-level feature fusion to improve edge detection accuracy,

which means higher computational costs while the EdgeNet architecture is comparatively simpler. Due to the depth and complexity of the network, feature extraction efficiency may decrease; therefore, adaptive adjustments have been made to the internal structure of each block to enhance performance and reduce redundancy. Overall, EdgeNet has less than 3% of parameters (around 1 M) comparing to the original architecture (around 35 M).

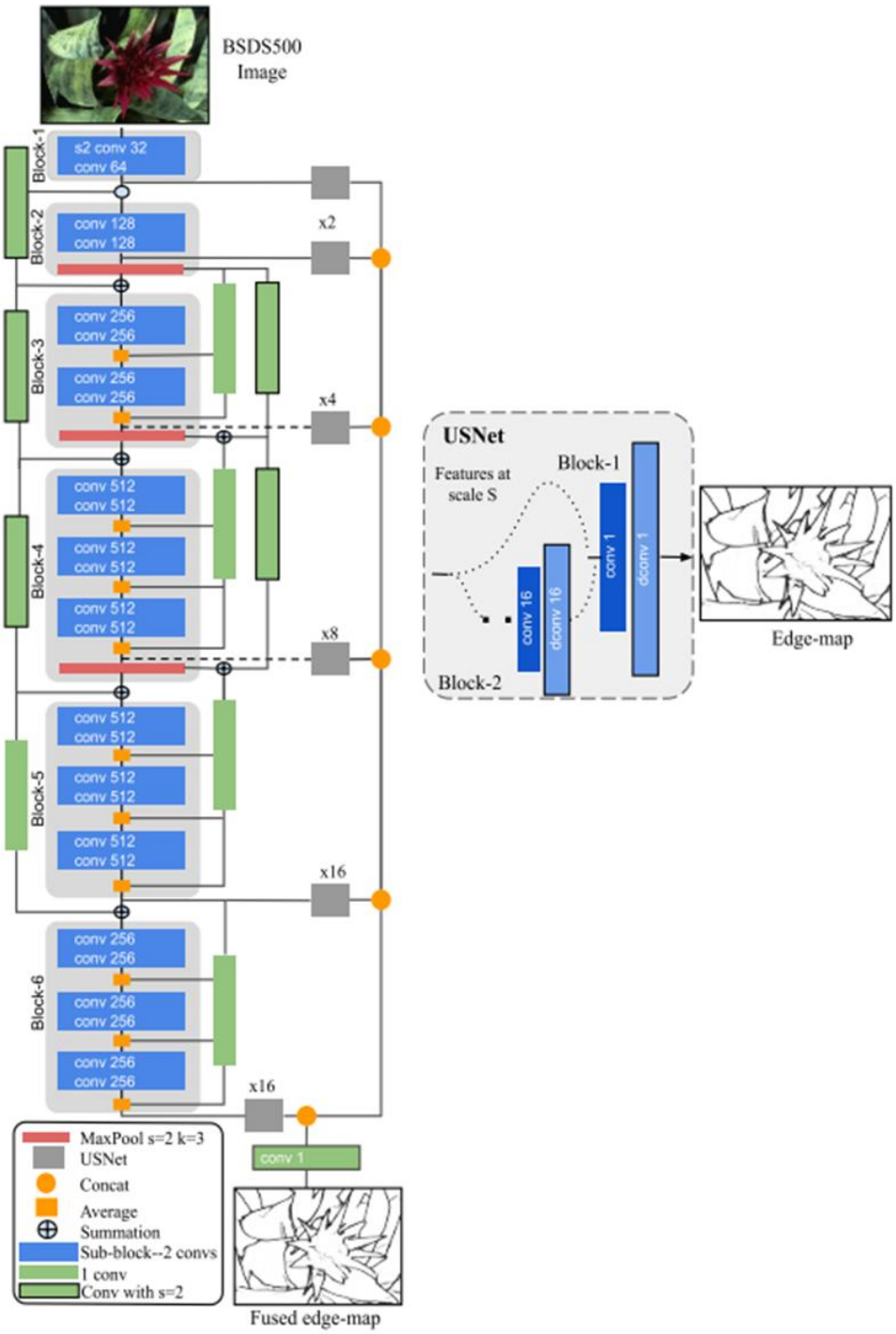



**Figure 4.7 Flowchart of DexiNed architecture from Poma et al. (2020)**

To ensure the preservation of important edge features in a deep network, the Dexi component includes two types of skip connections: the First Skip Connection (FSC) and the Second Skip Connection (SSC).

The First Skip Connection establishes links between the early and later layers of the network, ensuring that edge features extracted in the shallow layers can be directly transferred to deeper levels. For example, if the input to a block is *X*, and the output after convolutional operations is *F(X)*, then the final output of the block, *Y*, can be expressed as:

$$Y = F(X) + X \tag{4.6}$$

This design allows shallow information to be added to deep features, enabling edge features to be retained in the deeper layers of the network, thereby enhancing the model's detection capability.

To further strengthen the preservation of edge features, the Second Skip Connection transfers information between different sub-blocks within the same block. Suppose the current block has two sub-blocks with outputs $F_1(X)$ and $F_2(X)$, respectively, then the output of the Second Skip Connection, *Z*, can be expressed as:

$$Z = \frac{F_1(X) + F_2(X)}{2} \tag{4.7}$$

By combining the First Skip Connection and the Second Skip Connection, EdgeNet can retain multi-level edge features when processing complex images, ultimately producing more accurate edge maps. Overall, the function of these connections can be represented by the following formula:

$$Y_{final} = \frac{FSC(X) + SSC(X)}{2} \tag{4.8}$$

where, $FSC(X)$ and $SSC(X)$ represent the outputs of the First and Second Skip Connections, respectively. The final output, $Y_{final}$, is the average of these two results. Through this fusion, the Dexi component ensures the transmission and retention of edge information in the deep network.

It is important to distinguish these skip connections from the dense connections used in DenseNet architectures. While both techniques aim to improve feature reuse and gradient flow, they operate differently. Skip connections were used to directly link corresponding layers in

the encoder and decoder, which help preserve spatial details during upsampling. And it was wildly used in tasks like image segmentation. In contrast, DenseNet employed dense connections, where each layer receives input from all previous layers.

The USNet component is responsible for upsampling the feature maps generated by Dexi to match the target resolution. This process is achieved through a combination of convolutional and deconvolutional layers, where each iteration upsamples the feature maps to a higher resolution until they match the size of the ground truth edge map. The design of USNet focuses on accurately upsampling while preserving edge details to produce clear edge maps.

Specifically, as shown in Figure 4.8, block-b is activated only when upscaling the input feature maps from the Dexi network. This process is repeated until the size of the feature maps is doubled compared to the initial input image. The enlarged feature maps are then passed to block-a, which first processes the input feature maps using a 1 × 1 convolution kernel and applies a nonlinear transformation through the ReLU activation function. Next, block-a performs transposed convolution using a kernel of size $s \times s$, where $s$ represents the scale of the input feature map. After the final deconvolution operation in block-a, the feature map is upsampled to the same size as the initial input image. The final convolutional layer does not have an activation function.

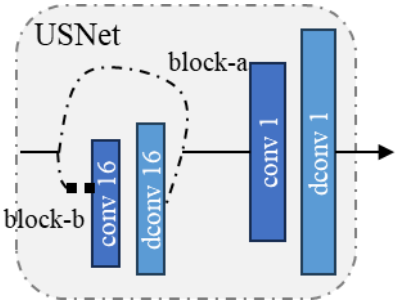



**Figure 4.8 USNet architecture**

In the implementation of USNet, three different upsampling strategies can be used: bilinear interpolation, sub-pixel convolution, and transposed convolution. Based on empirical evaluations in the literature, it was found that among these three, transposed convolution demonstrated better performance in terms of the F-measure,

$$\text{where } F = \frac{2 \times Precision \times Recall}{Precision + Recall}.$$

Therefore, this method was also adopted in this study. Transposed convolution performs upsampling through "inverse convolution", preserving spatial information and better maintaining the fine structure of edges while enlarging the image.

## 4.3 Training

This section provides a detailed description of the training process for the EdgeNet model, including a description of the dataset used, data augmentation strategies, as well as specific training strategies and epoch settings. The design and execution of these steps ensure that the model can achieve high-precision edge detection in various complex scenarios.

### 4.3.1 Dataset Description and Augmentation

As mentioned in the section 3.4.1, the BIPED dataset was used as the primary dataset to pre-train the EdgeNet model in this study.

During the dataset construction, both the original images and augmented images were used to expand the diversity of the training set. These data augmentation techniques included image rotation, flipping, scaling, and colour jittering. The application of these techniques enabled the model to better adapt to various image changes, thereby improving its generalization ability to unknown scenes. These operations also increased the size of the dataset and allowed the model to encounter a more diverse range of edge patterns during training.

Except the BIPED dataset, this thesis also used other standard edge detection datasets to further optimize and validate the model's performance. Those datasets include MDBD (Multicue Dataset for Boundary Detection), which consists of various images with multiple boundary detection cues, allowing for robust evaluation of edge detection algorithms. The BSDS500 (Berkeley Segmentation Dataset) contains 500 images with ground-truth segmentation, which can comprehensively evaluate the performance of the model in different scenes. The Contemporary Image Dataset (CID) includes a collection of contemporary images that challenge the model with both simple and complex scenes. Together, these datasets cover a wide range of scenarios, ensuring that the model can effectively adapt to various application environments.

Edge detector evaluation has been well established since the groundbreaking work cited in Ziou and Tabbone (1998)’s work. Given that BIPED provides annotated ground-truth edge maps, three commonly adopted evaluation metrics in the field have been utilized: the fixed

contour threshold (ODS), the optimal threshold for each image (OIS), and average precision (AP).

**Table 4-1 Quantitative results of EdgeNet trained on BIPED**

| Dataset | ODS | OIS | AP |
|---|---|---|---|
| MDBD | 0.859 | 0.864 | 0.917 |
| CID | 0.652 | 0.690 | 0.711 |
| BSDS300 | 0.709 | 0.726 | 0.738 |
| BSDS500 | 0.728 | 0.745 | 0.689 |

From Table 4-1, it can be seen that EdgeNet performs exceptionally well on the MDBD dataset, achieving ODS, OIS, and AP scores of 0.859, 0.864, and 0.917, respectively, indicating outstanding detection performance in high-contrast scenes. On the BSDS500 dataset, which features more complex scenes, EdgeNet also shows robust performance, with an ODS of 0.728, OIS of 0.745, and an AP of 0.689, although the AP is slightly lower. The results on the CID dataset are relatively lower, which may be due to the low contrast between the foreground and background in the images, making it difficult to detect edges amidst strong grayscale or colour variations. Overall, EdgeNet's performance across various datasets demonstrates its adaptability and effectiveness in different application scenarios.

### 4.3.2 Training Strategy and Epoch Details

To ensure that the EdgeNet model fully learns the edge features in images, a systematic training strategy was designed. First, the model was initial training on the BIPED dataset. The primary goal of this step was for the model to master the basic edge detection task, gradually learning high- and low-level features in images to establish an initial capability for edge recognition.

During the initial phase of training, a relatively high learning rate of 0.001 was set to accelerate the convergence of model parameters. As training progressed, the learning rate was gradually reduced to ensure that the model could make finer adjustments in local regions, thereby avoiding overfitting. This dynamic learning rate strategy effectively balanced the speed and accuracy of the model's training.

In each training epoch, the model went through the entire training dataset, making edge predictions for each image and calculating the loss value. For the loss function, a cross-entropy

loss function was employed, modified to meet the specific needs of the edge detection task. The specific form of the loss function is:

$$\mathcal{L} = \sum_{n=1}^{N} \lambda_n \cdot l_n \tag{4.9}$$

$$l_n = -w\left[y \cdot log\left(\theta(\hat{y}_n)\right) + (1-y) \cdot log\left(1-\theta(\hat{y}_n)\right)\right] \tag{4.10}$$

Here, $\theta$ represents the sigmoid function, $y$ and $\hat{y}_n$ denote the ground truth edge map and the predicted edge map, respectively. $\lambda_n$ is a set of hyperparameters used to balance positive and negative samples, and w is the weight in the loss function, given by the following equation:

$$w_{(y>0)} = 1 \times \frac{y_-}{y_+ + y_-} \quad and \quad w_{(y\leq 0)} = 1.1 \times \frac{y_+}{y_+ + y_-} \tag{4.11}$$

Here, $y^+$ and $y^-$ represent the number of positive and negative samples in the given ground truth edge map (GT), respectively. Additionally, the loss value $l$ is obtained by averaging the loss values $l(i,j)$ over all pixel positions $(i, j)$:

$$l = mean(l^{I \times J}) \tag{4.12}$$

where, $l_{(i,\ j)} = [l_1, \ldots, l_N]$ represents the set of loss values obtained for each pixel position $(i,j)$ across all predictions. This average loss value is used to compute the final overall loss $\mathcal{L}$. This loss function ensures accurate detection of different types of edges in the predictions by reasonably balancing positive and negative samples.

In each training round, the model output multiple edge prediction maps, which were then combined using different fusion strategies to produce the final edge detection result. In this step, the fusion strategy derived the result through a fusion process at the end of the network, rather than averaging all prediction maps. According to Poma et al. (2020)'s work, these two strategies perform similarly in terms of quantitative results, so the former results will be primarily presented in subsequent analyses.

The training process included multiple epochs to ensure that the model fully learned the features in the training data. In the early stages, as the number of epochs increased, the model's loss decreased rapidly, indicating that the model was gradually mastering the edge detection task through continuous optimization. However, in the later stages of training, the rate of loss reduction tended to slow down, at which point performance on the validation set

was monitored to determine whether further adjustment of the training strategy or stopping of the training was necessary.

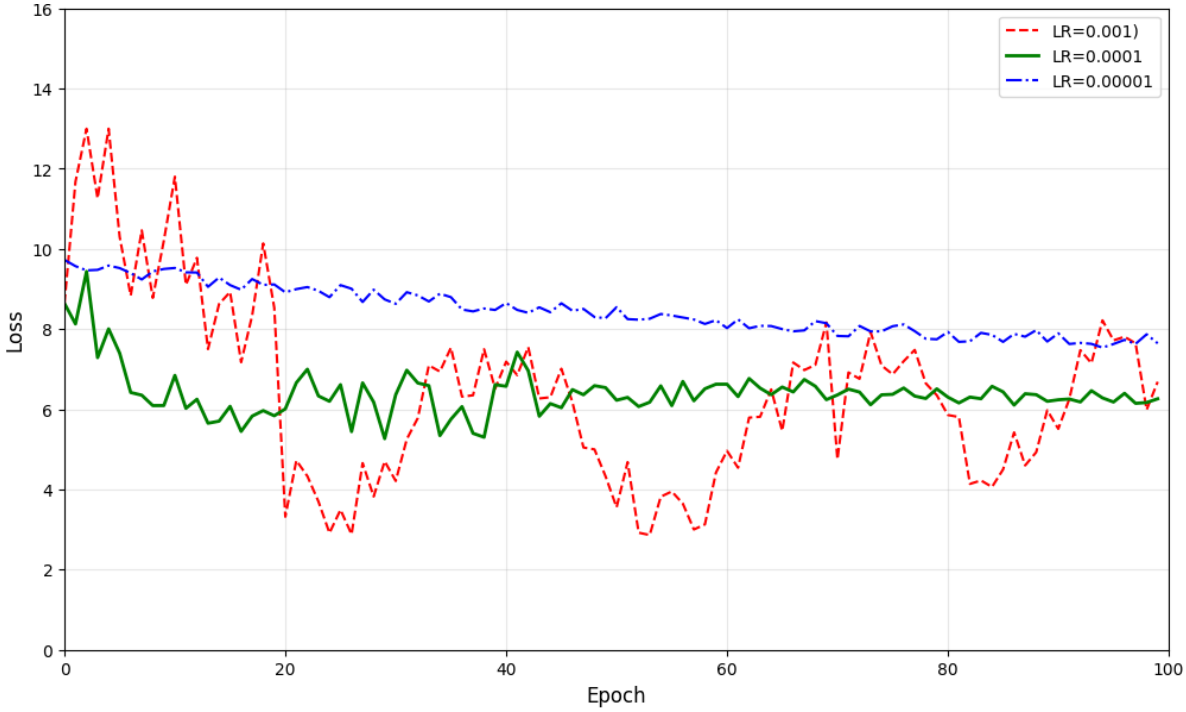



**Figure 4.9 Training Curve of each epoch**

Figure 4.9 illustrated the loss curves across epochs under different learning rate settings. As shown, a moderate learning rate (0.0001) enabled stable and effective convergence. In contrast, a higher learning rate (0.001) caused oscillations and instability, while a lower learning rate (0.00001) led to slower convergence.

During the fine-tuning phase, the model's detailed performance was further optimized while maintaining the generalization ability obtained in the pre-training phase. Since the dataset used for fine-tuning is smaller and more specialized, the learning rate was fixed at 0.0001 to ensure that the model could converge stably and make fine adjustments during the fine-tuning process. The main architecture and strategies from the pre-training phase were retained, but the focus shifted to enhancing the model's performance in flow visualization-related tasks through a lower learning rate and smaller parameter updates.

With a reasonable epoch setting and dynamic learning rate adjustment, the EdgeNet model demonstrated excellent performance across different datasets and application scenarios. Additionally, cross-validation strategies were introduced on the validation set during the

training process to ensure that the model could broadly adapt to different test datasets after training and achieve high-precision edge detection in practical applications.

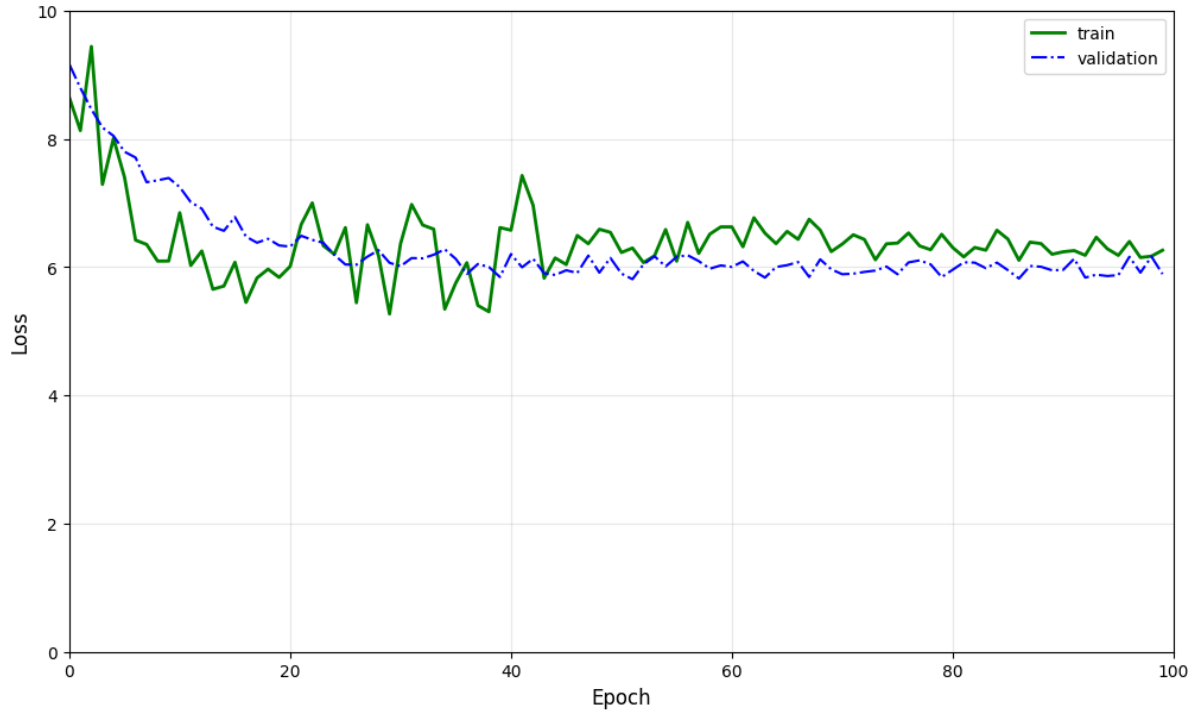



**Figure 4.10 Learning Curve of each epoch at LR = 0.0001**

The training and validation loss curves was shown in Figure 4.10. Both curves exhibit minor fluctuations, particularly in the earlier epochs. These are likely due to a relatively small batch size and the use of a fixed learning rate, but overall convergence is observed. And the validation curve closely follows the training curve without significant divergence, indicating that the model generalizes well and is not overfitting. Through systematic training and adjustment, the EdgeNet model has acquired strong edge detection capabilities, allowing it to accurately identify and extract edge information in various complex scenarios.

## 4.4 Post-processing

This section outlines the post-processing steps for analysing flow visualization images, focusing on label detection, obtaining orientation fields, and determining direction.

### 4.4.1 Label Detection

Labelling (also known as Connected component labelling, connected component analysis, or region labelling) represents an algorithmic implementation of graph theory employed for ascertaining the connectivity of regions resembling "blobs" within a binary image. In a binary image, such as a grayscale image, if two adjacent pixels have the same value (both 0 or 1), these two pixels are considered to be part of a mutually connected region. This relationship is transitive, and the "labelling" process involves assigning the same value to all pixels within a connected region.

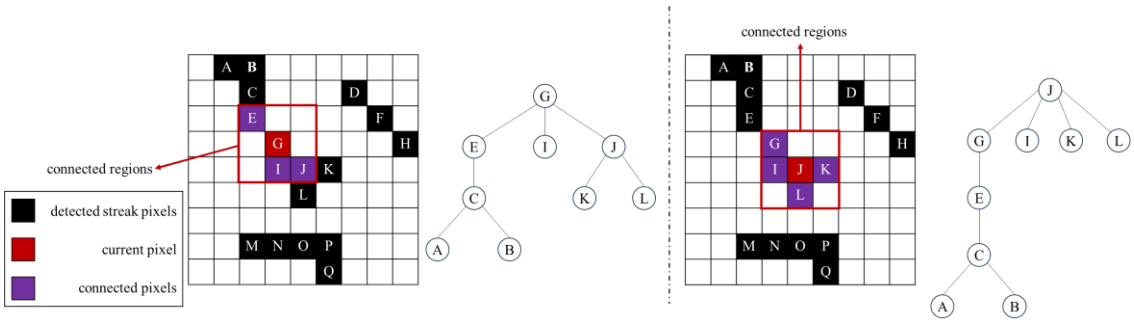



**Figure 4.11 Label and store the detected streaks. (left) start with G; (right) start with J.**

Specifically, as illustrated in Figure 4.11 (for clarity, a simplified lettering scheme is used in the figure), for each pixel, the presence of other pixels within its surrounding eight cells is examined. This evaluation is recursively applied to the detected pixels until no suitable candidates remain. Throughout this process, all pixels involved are labelled as part of the same streak. The whole process can be started from any point on the streak, such as G (Figure 4.11left) or J (Figure 4.11 right). Initiating the labelling process for one of the streaks, starting from G (Figure 4.11 left), reveals that E, I and J are adjacent to it. Subsequently, C is adjacent to point E, while K and L is adjacent to J. Continuing in this sequential manner, a tree-like structure can be employed to record the results, ultimately resulting in the "labelling" of A, B, C, E, G, I, J, K as a cohesive entity. Suppose the process starts from J (Figure 4.11 right), similarly, it would reveal that G, I, K and L are adjacent to J. Finally, a different tree-like structure with

same content could be obtained. In fact, starting from any point on the streak will eventually result in the entire streak being labelled.

Figure 4.12 provides a visual summary of Steps 6 and 7 applied to a single cell. The segmented grayscale image cell is shown in Figure 4.12(a) and underwent edge detection to transform into the binary image shown in Figure 4.12(b). Labelling was then performed on this edge feature map. Although the actual map was more intricate than the example in Figure 4.11, the underlying principles remained consistent. Ultimately, each "independent" edge was labelled, and for ease of distinction, different colours are used in Figure 4.12(c) to show the detected streaks.

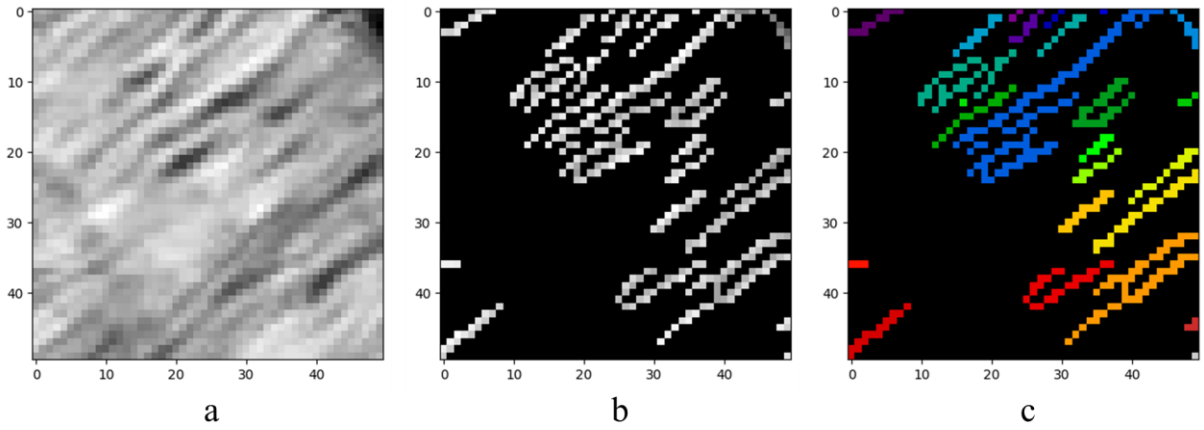



**Figure 4.12 The label detection result from a cell of flow visualization image: (a) grey scale image; (b) image after edge detection; (c) image after labelled**

### 4.4.2 Obtaining the Orientation Field

In Step 7, despite the extraction of all streaks, which constitutes the most direct observational data, this information cannot be directly used to generate the directional field. This limitation stems from the heterogeneous distribution and clutter of streaks throughout the image, which obscures relevant information. Additionally, on the scale of the entire image, the identified streaks form only a sparse matrix. Therefore, alternative approaches are required.

The algorithm used to estimate the orientation field from a surface oil flow visualization image relies on the gradients of the resulting images after post-processing, which is described in detail by Rao and Schunck (1991). Specifically, it targets the gradients obtained from detected streaks, aiming to reduce interference from background noise.

The gradient at a point (i, j) is represented in polar coordinates as $G_{ij}e^{i\theta_{ij}}$. In Step 7, employing label detection, one can obtain the set $\boldsymbol{L}$ of streaks present on a given cell. Therefore, for an N × N cell, the dominant orientation, $\hat{\theta}_{mn}$ of its centre point (m, n) can be represented as below:

$$\hat{\theta}_{mn} = \tan^{-1}\left(\frac{\sum_{(i,j)\in \boldsymbol{L}} G_{ij}^2 \sin 2\theta_{ij}}{\sum_{(i,j)\in \boldsymbol{L}} G_{ij}^2 \cos 2\theta_{ij}}\right)/2 \tag{4.13}$$

The estimated orientation angle is denoted as $\hat{\theta}_{mn} + \pi/2$, as the gradient vector is perpendicular to the direction of anisotropy.

A key point is that the orientation estimation algorithm yields an orientation field rather than a vector field. The flow at a specific point could correspond to either direction: $\theta$ or $\theta + \pi$. As a result, any method depending on an accurate vector field as input will inevitably fail to achieve the intended outcomes.

### 4.4.3 Direction Decision

Additional measures were required to aid in determining the specific flow direction of cells. In this work, the flow direction assessment of cells is based on the following assumptions: at the inlet of the flow field, the flow direction of cells should align with the direction of the wind tunnel's outflow; for any cell at arbitrary positions, due to the sufficiently small size of the cell, its flow direction should not exhibit significant "discontinuities" compared to neighbouring cells. For instance, in Figure 4.13(a), a cell with an undetermined orientation exhibits two potential directions, denoted as (b) and (c). Clearly, the predominant flow direction at this juncture is upward. Hence, direction b appears more plausible, while direction c is disregarded due to the presence of a significant "discontinuity". To quantitatively describe the coherence in the flow direction at the centre (m, n) of a given cell, consider another cell whose centre is (i, j), where i and j are chosen so that this cell resides within a defined region $\boldsymbol{W}$ surrounding the target cell. In practice, the selected entities comprise the eight neighbouring cells surrounding the target cell. The directional relationship between two vectors can be represented by the cosine of the angular displacement difference. The smaller the absolute difference in angular displacements, the smaller the angle between the vectors, indicating a closer alignment in direction. The measure of coherence could be defined by

$$Score_{mn} = \sum_{(i,j)\in W} \cos\left|\theta_{ij} - \left(\varepsilon_{mn} + \hat{\theta}_{mn}\right)\right| \tag{4.14}$$

$$\text{where } \varepsilon_{mn} = \begin{cases} 0 \\ \pi \end{cases} \tag{4.15}$$

Thus, the problem shifts to maximizing the overall coherence across all cells, whose centre points fall falls within the set ***F***, with the boundary condition being the predetermined direction at the inlet position. By solving for the maximum of Eq.(4.16) the distribution of the vector field can be determined.

$$\varepsilon_{mn}{}^{*} = \arg\max_{\varepsilon_{mn}} \sum_{(m,n)\in F} \sum_{(i,j)\in W} \cos\left|\theta_{ij} - \left(\varepsilon_{mn} + \hat{\theta}_{mn}\right)\right| \tag{4.16}$$

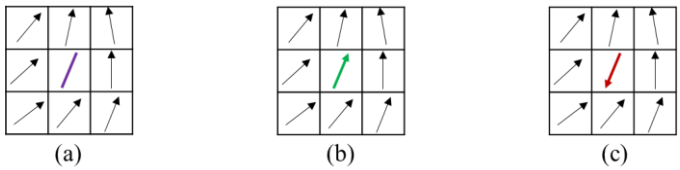



**Figure 4.13 Possible case for ambiguity. (a) Original orientation to be inferred; (b) Probable estimations (true); (c) Probable estimations (false)**

## 4.5 Performance Evaluation

To illustrate the advantages of the proposed algorithm, a comparison is made between the results obtained using the traditional Hough transform and the CNN-based algorithm developed in this chapter. This comparison provides a clear visual representation of the differences in performance. The lines detected in the flow visualisation images with a 3.75 mm tip clearance using CNN-based algorithm is shown in Figure 4.14 (top). The result of using linear Hough transform on the detected streaklines is exhibited in Figure 4.14(bottom). The algorithm proposed in this work significantly mitigates the formation of noise vectors, which often manifest as seemingly vertical, horizontal, or diagonally oriented at 45° (Figure 4.14 (bottom)). Moreover, the algorithm in this work calculates the average information of each labelled streak using Eq. (4.16) to obtain directional information, which is then redrawn as streaks. This approach differs from fitting the maximum possible line segments within cells (as shown in the right panel of Figure 4.14). Therefore, it is evident that the distribution of streaks in the top panel of Figure 4.14 is more uniform, with minimal intersection between streaks.

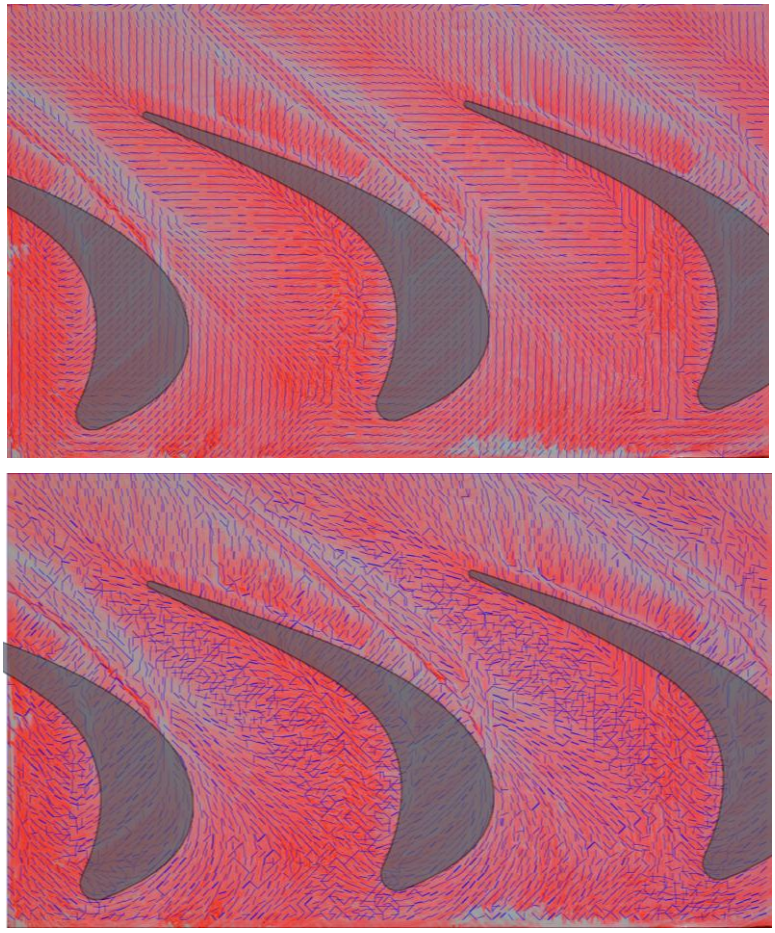


**Figure 4.14 Image processing results; (top) CNN-based algorithm result in this work; (bottom) image processing solution from Abdelsalam et al. (2017)**

In the field of image processing, evaluating the robustness of algorithms against blurring is crucial for applications. This study aims to assess and compare the robustness of a CNN-based algorithm, and a traditional image processing solution proposed by Abdelsalam et al., focusing on their performance under varying levels of image blurring.

A systematic approach was employed, utilizing the Mean Squared Error (MSE) metric to quantify image quality degradation. The MSE measures the disparity between the original and blurred orientation fields of images processed by both algorithms. A lower MSE value indicates better preservation of the original image's directional features despite the blurring, reflecting greater robustness against degradation.

By applying this methodology to both the CNN-based algorithm and the image processing solution, significant insights into their performance and robustness under different blurring conditions are revealed.

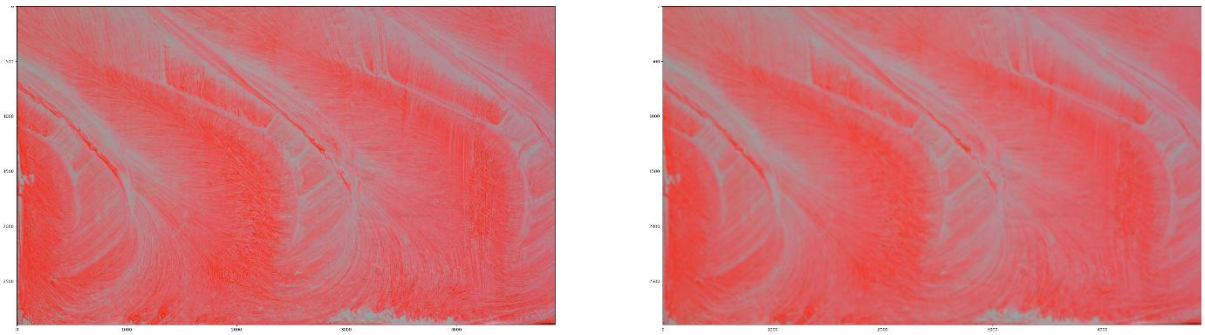


**Figure 4.15 Gaussian blur results; (left) original image; (right) blur image with kernel size=10 and sigma=25.**

As Figure 4.15 shows, the blurred images are created using Gaussian blur, where the parameters kernel size and sigma determine the extent of blurring. A larger kernel size increases the filter's spatial coverage, while a higher sigma value results in a stronger blur effect. Blurring reduces image details, which can alter the directional information contained in the image.

To compute MSE between original and blurred orientation fields, the difference in orientation angles at each pixel is evaluated. Orientation matrices are derived from the Hough transform, which detects the primary direction of lines in the image. The original orientation matrix represents the unblurred state of the image, while the blurred orientation matrix reflects the directions detected after applying Gaussian blur. MSE quantifies how much the blurring process distorts or alters the directional features of the image.

The MSE is calculated by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 \tag{4.17}$$

where $x_i$ represents the orientation angle at pixel $i$ in the original orientation field, $y_i$ represents the orientation angle at pixel $i$ in the blurred orientation field, $n$ is the total number of pixels.

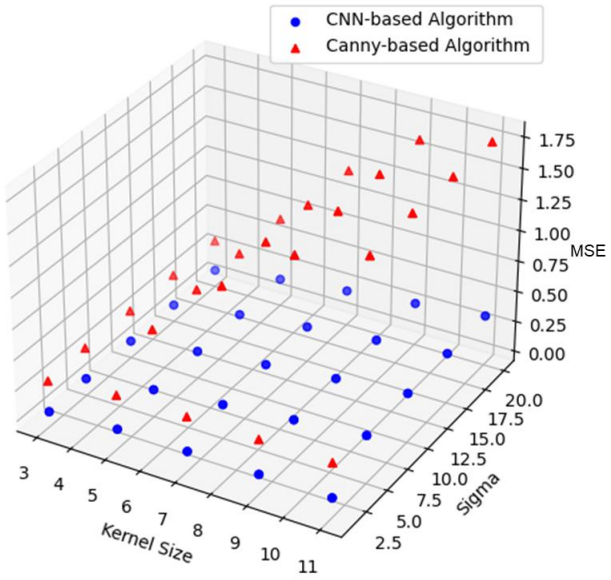



**Figure 4.16 MSE performance in relation to kernel size and sigma for CNN-based algorithm and Abdelsalam's algorithm.**

For CNN-based algorithm, as kernel size and sigma increase, the MSE also increases, but the overall rate of increase is relatively small. For example, with a kernel size of 3, MSE rises from 0.044 at sigma = 1 to 0.059 at sigma = 20. Similarly, with a kernel size of 11, MSE increases from 0.082 at sigma = 1 to 0.331 at sigma = 20. This steady but moderate rise in MSE suggests that CNN-based algorithm maintains a good level of robustness across different levels of blurring, with directional matrices showing limited deviation.

In contrast, Abdelsalam's algorithm. exhibits a much sharper rise in MSE as kernel size and sigma increase. For example, with a kernel size of 3, MSE rises from 0.297 at sigma = 1 to 0.314 at sigma = 20, while with a kernel size of 11, MSE escalates rapidly from 0.364 at sigma = 1 to 1.741 at sigma = 20. This indicates that Abdelsalam's algorithm is less robust to blurring, with significant changes in the orientation matrices as the level of blurring intensifies.

From this robustness analysis, CNN-based algorithm appears to be more stable than Abdelsalam's algorithm when exposed to different levels of blurring, especially at larger kernel sizes and higher sigma values. The relatively smaller increase in MSE for CNN-based algorithm suggests that it better preserves the geometric features of the image, even after significant blurring. On the other hand, Abdelsalam's algorithm shows a more pronounced increase in MSE, indicating that its performance is more susceptible to blurring, with a greater loss or alteration of directional information.

Robustness analysis is crucial in understanding an algorithm's stability and reliability in real-world applications. For tasks such as image matching, edge detection, or object tracking,

CNN-based algorithm may be more adaptive to varying levels of blur and noise, whereas Abdelsalam's algorithm might require further optimization to improve its robustness in more complex or degraded environments.

Figure 4.17 shows the calculated streamlines from streaks using ***streamslice*** function in MATLAB. Precisely locating the visualization image facilitates direct plotting of blade profiles onto the image. The obtained quantitative flow field results from the input of a flow visualization photograph are notably encouraging. At the inlet, streamlines exhibit approximate uniform flow. However, in proximity to the cascade blades, the streamlines undergo separation following the expected pattern of the tip leakage vortex. A close examination of the results shows that regions within the original image displayed limited clarity which may introduce distortions into the results. In the dotted box area of Figure 4.17, a number of vectors point in apparent non-physical directions at certain points. Some of these were attributed to surface irregularities or contaminants. A few anomalies were also attributed to deficiencies in the processing algorithms. In regions with complex flow conditions, especially near saddle points and areas with nearly horizontal flow, the existing direction recognition algorithm struggled to provide accurate results.

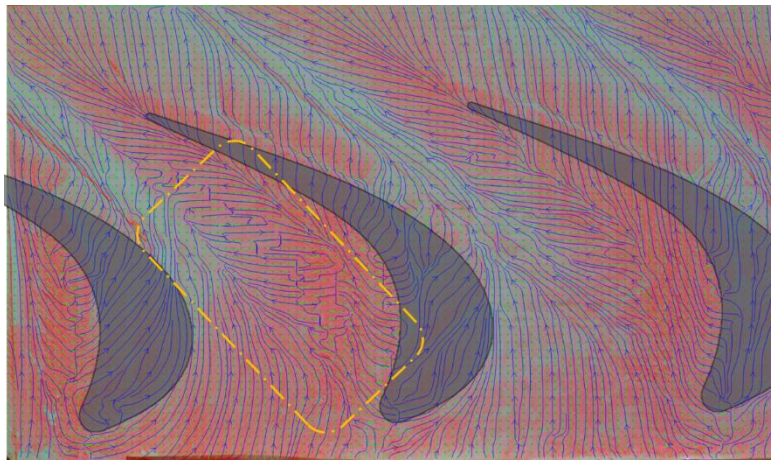


**Figure 4.17 Calculated streamlines from streaks.**

This approach was not exclusively applicable to the experimental results obtained from the cascade; in fact, it yielded promising qualitative outcomes for surface flow visualization images sourced from other sources as well.

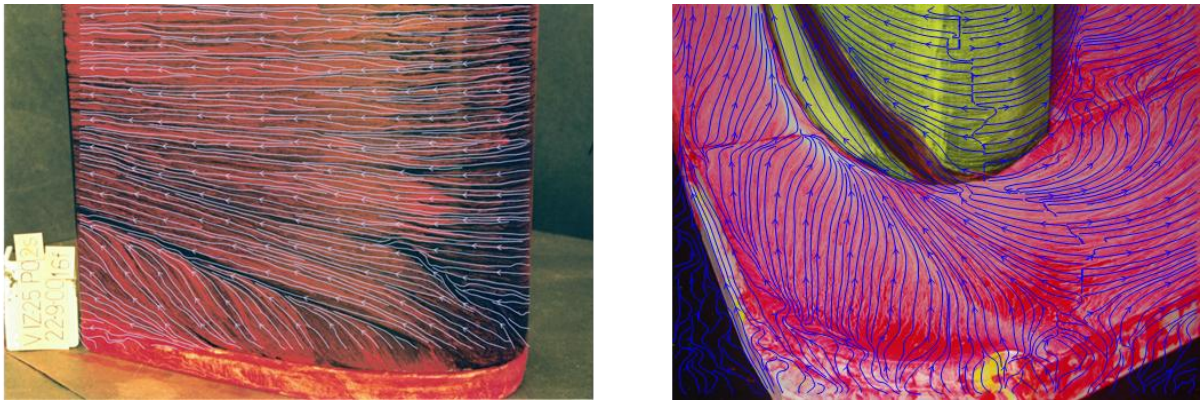


**Figure 4.18 Calculated streamlines from streaks from other surface flow visualization images.**

As shown in Figure 4.18, the application of this approach to surface flow visualization images sourced from other experiments resulted in flow patterns that are consistent with expert interpretation of the image in the literature(Ingram et al., 2005) and with other measurements (pressure probe, wool tufts etc) in the cascade. As well as a qualitative description of the image that aids interpretation of the flow field the technique offers quantitative data from surface oil flow visualisation.

## 4.6 Conclusion

This chapter builds on the foundational work by Abdelsalam et al. (2017), enhancing their edge detection approach with CNNs to improve the accuracy and reliability of extracting streamline information from surface flow visualization images. By integrating modern image processing techniques, a comprehensive algorithm has been developed that surpasses previous methods in effectively discerning the direction and structure of flow patterns.

In pre-processing stage, the images were manually calibrated using advanced techniques from the OpenCV library, which solves the limitations of automatic camera calibration and minimizes geometric distortion. Subsequently, the images were divided into smaller cells, converted to grayscale, smoothed with a bilateral filter to preserve edges, and processed to reveal intensity gradients. These steps enhanced the visibility of important structures.

The most important part of this method is to apply a customized CNN architecture for edge detection. This architecture is adapted from the EdgeNet model and incorporates novel design elements such as parallel skip connections and USNet. These innovations enable the model to preserve fine edge features and achieve high-resolution edge detection. Training the

model on various datasets, including BIPED and other standard edge detection datasets, ensures its robustness and adaptability to various scenes.

After edge detection, several post-processing steps are used to improve the results. Label detection identifies and marks streaks in the processed image, while orientation estimation algorithms are used to calculate the dominant flow direction. Then, a coherence-based approach is adopted to address directional ambiguity, ensuring consistency in the vector field of the entire image. Even in complex flow areas that are difficult to achieve with traditional methods, this method can accurately detect and quantify flow patterns. The calculated streamlines are highly consistent with the expected flow behaviour and expert interpretations.

All datasets and code are available for download at:

https://github.com/Dehakaa/Streamline_detector

# Chapter 5 Assessment of the GLOF Method for Surface Friction Visualization

This chapter examines the application and performance evaluation of the Global Luminescent Oil-Film (GLOF) method for measuring surface friction fields in fluid dynamics experiments While this method offers a non-invasive optical alternative to traditional measurement techniques, its practical effectiveness remains uncertain. It can be considered an optical alternative to traditional measurement techniques, with which detailed information about boundary layer interactions, surface shear stresses and other fundamental flow properties could be captured non-invasively.

However, while the GLOF method provides valuable insights, it also has limitations in practical use. This chapter evaluates the accuracy of surface friction result obtained with GLOF, with a particular focus on its agreement compared to established experimental data. For this purpose, the findings obtained using the GLOF method are compared with the experiment results of Eckerle and Langston (1987), who investigated the flow field around a cylinder. CFD results are also compared with Eckerle's experimental data, showing better alignment with the observed flow characteristics. As a result, the GLOF method did not work effectively in this thesis.

## 5.1 Global luminescent oil-film (GLOF) method

In this section, inspired by the work of Liu et al. (2008), the application of the Global Luminescent Oil Film (GLOF) method is explored to assess the relationship between surface friction and flow patterns. Utilizing oil film flow visualization techniques, particularly the analysis of surface oil flow, critical insights are derived from the residual flow traces. Based on the thin oil film equation established in prior studies, which has been discussed in section 2.2.3 and section 3.2.2, the relationship between oil film thickness and wall shear stress is examined, revealing the connection between oil film characteristics and flow behaviour. Furthermore, a formula linking the luminous intensity of the oil film to its thickness, as proposed in Liu (2013)'s work, serves as a foundation for this analysis. With the camera aligned parallel to the experimental plane, the image data can be accurately correlated with the actual surface position, enabling precise and reliable evaluation.

However, despite the rigorous theoretical framework, GLOF faces several challenges in practical applications, particularly with regard to accuracy and stability. This analysis evaluates the effects of equipment vibration, inter-frame time intervals and temporal stability on the estimation of the surface friction field. In a high-vibration environment, the clarity of the images is compromised, affecting subsequent data processing and analysis, which leads to incorrect identification of flow patterns. The variations in inter-frame time intervals significantly influence the stability of the skin friction field, demonstrating a high dependency on flow conditions. The results in the following sections include cases with no obstacles in the flow field and cases with different obstacles: a rectangular prism, a streamlined object, and a cylinder.

### 5.1.1 Discrepancies Between Experimental Challenges and Theoretical Applications

In the previous chapter, the theoretical framework of the GLOF method was established, detailing how surface friction is mapped from oil film thickness and luminescent intensity using optical flow equations. One of the primary challenges in implementing GLOF lies in the accuracy of the estimated surface friction field. This method depends on precise image data and a well-defined relationship between oil film luminescent intensity and the underlying flow dynamics. Although the theoretical basis is robust, discrepancies can occur between the predicted surface friction field and observed flow patterns. To illustrate these challenges, the performance of GLOF was evaluated using a specific video dataset.

To extract flow field information from video frames and generate corresponding vector data, this work utilizes videos recordings obtained from surface flow visualization experiments. In these experiments, a luminescent oil film was applied to the surface, and the flow patterns were captured in real-time using high-resolution cameras.

The flow visualization experiments were conducted in a low-speed wind tunnel at 0.5 plint wind tunnel, with a test section measuring 0.46 m × 0.46 m × 1.22 m. The freestream velocity was set to 19.1 m/s and the Reynolds number varying from $2\times10^3$ to $3\times10^4$ due to the different characteristic lengths of the experimental obstacle. The model surface was coated with a thin luminescent oil film to visualize the surface friction distribution. The wind tunnel was operated under steady-state conditions to ensure consistent flow characteristics throughout the experiment.

The video data was then processed using the GLOF method, implemented using a MATLAB code developed based on Liu (2019)'s work, with appropriate adjustments tailored to specific requirements. The video was processed frame by frame, with each pair of consecutive frames serving as input for optical flow computation. For instance, frame $I_t$ and frame $I_{t+1}$ form an image pair used to calculate motion information for that time step, ensuring the continuity of the temporal sequence.

Prior to optical flow computation, each frame underwent preprocessing to optimize computational performance. This included scaling the images to reduce resolution (using the scaling factor *scale_im*), thereby lowering computational complexity. Additionally, Gaussian filtering (with a filter kernel size of *size_filter*) was applied to smooth the images and reduce noise. These preprocessing steps enhanced image quality, contributing to improved accuracy and robustness of the subsequent optical flow algorithm.

The optical flow computation was performed using a two-step estimation method. First, the Horn-Schunck method was applied for an initial estimation of the optical flow field. This step balanced the smoothness of the flow field and brightness consistency using the Lagrange multiplier $\lambda_1$, producing an initial optical flow vector field. Next, the Liu-Shen method was employed to refine the initial results. By applying a higher weighting factor $\lambda_2$ to enforce stronger constraints, this step yielded an optical flow vector field with enhanced physical relevance. The optical flow field consisted of displacement components $u_x$ and $u_y$ for each pixel, providing a spatial representation of pixel movement between consecutive frames.

The optical flow vector field was further processed through brightness normalization to generate the skin friction field. The specific formulas used were $\tau_x = u_x / g$ and $\tau_y = u_y / g$, where gg represented the mean brightness value of the two frames. This normalization step reduced deviations caused by brightness inconsistencies, ensuring that the results were more stable and aligned with physical reality. By processing multiple image pairs, the skin friction fields generated for each frame were accumulated and averaged, resulting in the final vector field corresponding to the entire video.

The experiments were conducted on a system with an AMD Ryzen 5 5600G, 16GB RAM, and an NVIDIA GTX 3060 GPU. And the video was recorded by Nikon Z50 camera with an AF-S Nikkor 35mm 1:1.8G lens. The video used for flow visualization experiments had a total duration of 120 seconds, captured at a frame rate of 60 frames per second (fps).

Each frame was a high-resolution image with a resolution of 1920×1080 pixels. This video would be analysed from the following aspects.

#### 5.1.1.1 Vibration

One significant challenge encountered during the implementation of GLOF is the vibration of the equipment. The wind tunnel generates intense vibrations during operation, and even though the camera is fixed externally to the tunnel, the recorded images and data can still be significantly affected. This vibration not only disrupts the clarity of the images but also directly impacts subsequent image analysis and data processing. Specifically, when the algorithm analyses the movement of dye streaks, it becomes difficult to accurately distinguish whether the relative displacement of the dye streaks in the test article results from their actual movement or from false variations caused by equipment vibrations. This uncertainty greatly reduces the reliability of the experimental results.

**Figure 5.1 In the unobstructed test article area, evenly apply the oil/dye mixture using a brush**

Vibration is not limited to a specific direction, which leads to severe distortion in the analysis results. To visually illustrate this effect, Figure 5.1 shows the experimental conditions where no obstacles are placed within the wind field. Under these conditions, the test article should exhibit a stable laminar flow. In this scenario, theoretically, most vector directions should remain consistent and be parallel to the main flow direction—i.e., the inlet flow direction. However, under vibrating conditions, the vectors in the images display noticeable deviations and scatter. This phenomenon clearly indicates that vibrations disrupt the local flow, causing what should be a smooth flow field to become turbulent and uneven.

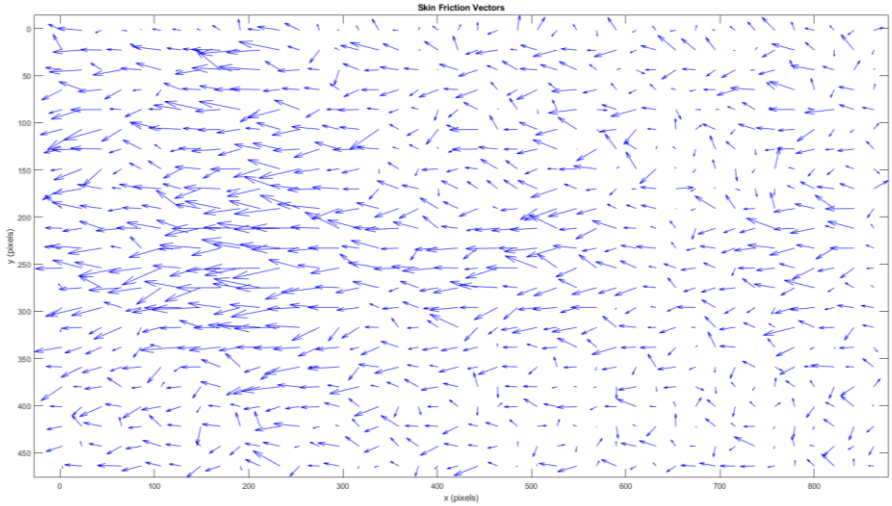



**Figure 5.2 skin-friction vectors on the image surface with vibration**

To address this issue, a sturdy structure can be added externally to the equipment to reduce the impact of vibrations. Specifically, the camera was securely fixed to a steel frame, allowing it to vibrate synchronously with the wind tunnel, maintaining the same frequency and pattern. This design can significantly reduce image jitter caused by equipment vibrations, thereby greatly enhancing the stability and accuracy of the data. As shown in Figure 5.3, the improved experimental results demonstrate that the directions of the skin friction vectors maintain a high degree of consistency, closely aligning with the expected laminar state. This proves that the measures successfully mitigated the interference caused by vibrations, ensuring more precise experimental data.

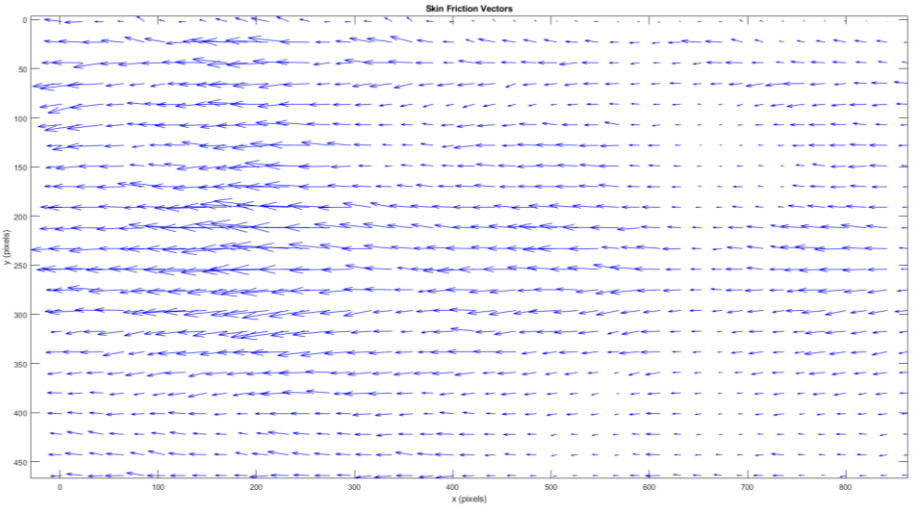



**Figure 5.3 skin-friction vectors on the image surface without vibration**

#### 5.1.1.2 Time Interval Between Frames

An in-depth analysis of the GLOF algorithm revealed significant variations in the surface friction field between adjacent frames. This variability suggests that GLOF may be sensitive to the time intervals between frames, which requires careful consideration. To quantify this deviation, the SSIM method was used to evaluate the similarity of the surface friction field under different time intervals, as discussed in the previous chapter.

A specific comparison was made using a frame interval of 30 frames, which balances sufficient sample size and manageable data density. As shown in Figure 5.4, three experimental scenarios with varying levels of disturbance are presented alongside their corresponding SSIM curves. In the graphs on the right, the trends of SSIM values with changing frame counts are depicted. The horizontal axis represents the "starting frame index," while the vertical axis denotes the "SSIM score." Each point in the graphs indicates the SSIM value calculated every 30 frames.

For the first SSIM curve (Figure 5.4b): As the frame count increases from 0, the SSIM value initially remains relatively high, around 0.96, indicating a high degree of similarity between frames in the early stages. Around frame 2000, the SSIM value begins to decline significantly, dropping sharply from its earlier stable state until it levels off around frame 4000. After frame 4000, the SSIM value continues to decrease, with the distribution of data points becoming more scattered, suggesting a reduction in similarity between adjacent frames.

For the second SSIM curve (Figure 5.4d): The trend resembles that of the Figure 5.4b. In the early stages of frame count increase, the SSIM value fluctuates around 0.96, indicating a high similarity. At approximately frame 1000, the SSIM score experiences a slight increase and remains above 0.96. However, around frame 2000, the SSIM value starts to show a clear downward trend, becoming more pronounced after frame 5000, ultimately decreasing below 0.93, with a sparser distribution of points reflecting a further weakening of similarity between images.

For the third SSIM curve (Figure 5.4f): This curve initially mirrors the trends of the Figure 5.4b and Figure 5.4d, with the SSIM value remaining above 0.965 before frame 1000, indicating a high degree of similarity. However, after frame 1000, the SSIM score gradually declines, with a notable downward trend around frame 2000. Unlike the previous two curves, this one exhibit more fluctuations after frame 3000, with more frequent and severe variations

in the SSIM value, especially after frame 5000, where the amplitude of fluctuation significantly increases. Ultimately, the SSIM value drops to around 0.94 at approximately frame 7000, with substantial volatility in image similarity persisting.

The observations from the above images can be summarized as follows:

1. Temporal Change in Image Similarity: The SSIM curves of the three experimental groups show that in the initial frame segment (between 1000 and 2000 frames), the similarity between adjacent frames is high, indicating minimal changes in the object and background. However, around frame 2000, the SSIM values begin to decrease, signifying a significant reduction in image similarity.

2. Sustained Decrease and Variability: As the frame count increases, the magnitude of the SSIM value's decline gradually levels off after frame 4000; however, the fluctuation patterns differ across experiments. For example, the third experimental group exhibits greater SSIM variability, while the first group shows a smoother declining trend, reflecting differences in experimental scenarios.

3. Overall SSIM Value Changes: In all three experimental groups, SSIM values remain high initially, reflecting strong similarity between adjacent frames. However, as the frame count increases, SSIM values exhibit an overall downward trend, particularly noticeable between frames 2000 and 5000.

These phenomena can be interpreted as follows: in the initial phase, the flow remains relatively stable, leading to minor changes in the surface friction field, which results in high similarity between adjacent frames. However, around frame 2000, the SSIM value begins to decline sharply, with a non-linear distribution emerging. This change can be attributed to the startup of the wind tunnel, where alterations in airflow conditions lead to marked differences between frames.

Additionally, scenarios with frame intervals of 60, 90, and 120 frames were compared, all exhibiting similar trends. This suggests that regardless of the chosen frame interval, SSIM changes are primarily dependent on surface flow conditions. When frame intervals are larger, the results appear "sparser" due to a reduction in sampled points, which may result in accuracy declines from insufficient computational data. Notably, smaller frame intervals are not necessarily better. If set too small, the camera may fail to effectively capture surface changes induced by airflow. In those experiments, the boundary layer velocity is approximately 1 m/s,

meaning that within 1/60 seconds, the surface oil film moves less than 0.0167 meters in the direction of airflow. Such a short distance results in minimal visible surface changes, leading to accumulated errors that could negatively impact the results. Therefore, setting a frame interval of 30 frames offers an optimal balance between these concerns.

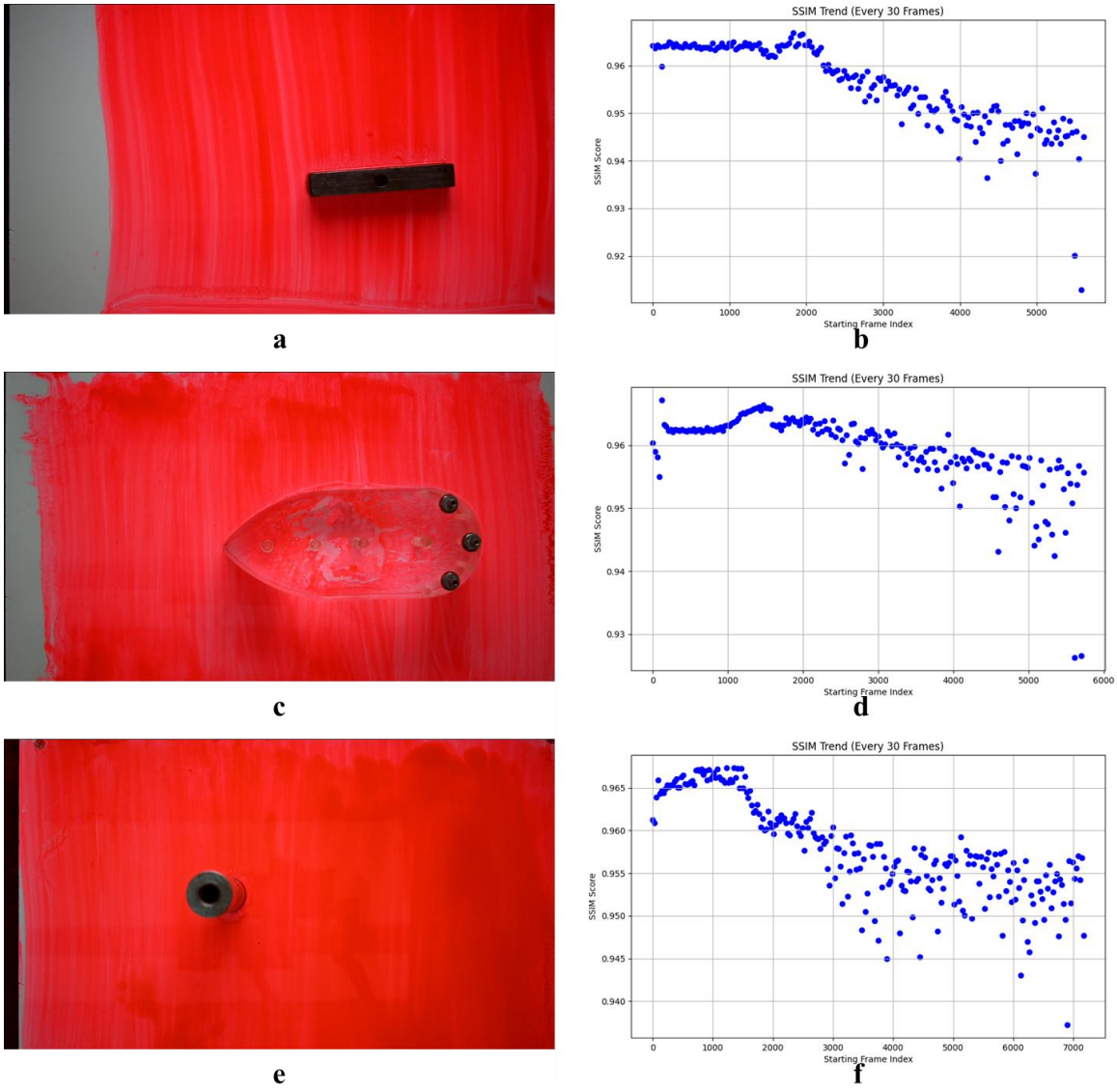



**Figure 5.4 SSIM Under Different Obstacle Conditions. (a) flow visualization experiment under rectangular obstacle condition; (b) SSIM trend for the rectangular obstacle condition;(c) flow visualization experiment under streamlined obstacle condition; (d) SSIM trend for the streamlined obstacle condition; (e) flow visualization experiment under cylindrical obstacle condition; (f) SSIM trend for the cylindrical obstacle condition.**

In conclusion, the choice of time interval directly affects the performance of the GLOF algorithm under different flow conditions. In practical applications, appropriately configuring camera parameters and flow conditions will help enhance the accuracy of GLOF results.

#### 5.1.1.3 Temporal stability

To evaluate the temporal stability of GLOF results, the impact of varying durations and different time "slices" on the outcomes is considered. This interval enables the capture of transient effects or changes in surface friction over time. Key parameters influencing this stability include the time interval between successive images, the selection of calculation regions, and the duration of computation.

For the same video, the calculated skin friction field exhibits significant variations with changes in these factors. For instance, as illustrated in Figure 5.5, all three images represent GLOF calculations derived from the same video, with the only difference being the selection of different time segments. In Figure 5.5a, which focuses on an early segment post-wind tunnel activation, only a few prominent vectors are visible near the edges. In contrast, Figure 5.5b and Figure 5.5c analysed later time intervals, revealing distinct differences in the skin friction field. The discrepancies are so pronounced that even qualitative observations yield clear variations, indicating a high dependence of GLOF results on temporal factors.

This dependency complicates the selection of which time segment to present as the final result, and extracting a standardized flow field from the video becomes challenging. During quantitative comparisons, variance calculations show that no specific moment achieves complete stability.

The situation in Figure 5.5 can also be reflected by the variation in variance. Selecting the time period between the wind tunnel starting and stopping, and setting a frame interval of 30 frames, the skin friction field is calculated using the GLOF method and used as the ground truth (GT). Within this time period, dividing it into several smaller time segments with an interval of 20 seconds, and similarly setting a frame interval of 30 frames, the corresponding friction field is calculated. To compare the temporal stability of the directional field changes, the following formula is proposed:

$$S_t = \frac{1}{N}\sum_{i\in I}\left(\tan^{-1}\left(\frac{u_{t,i}}{v_{t,i}}\right)+\tan^{-1}\left(\frac{u_{GT,i}}{v_{GT,i}}\right)\right)^2 \tag{5.1}$$

where $u_{t,i}$, $v_{t,i}$ are the x-direction and y-direction velocity components at point *i* on the image at time *t*, calculated using the GLOF method. While $u_{GT,i}$, $v_{GT,i}$ are the ground truth velocity components in the x-direction and y-direction at point *i* on the image, calculated using the GLOF method.

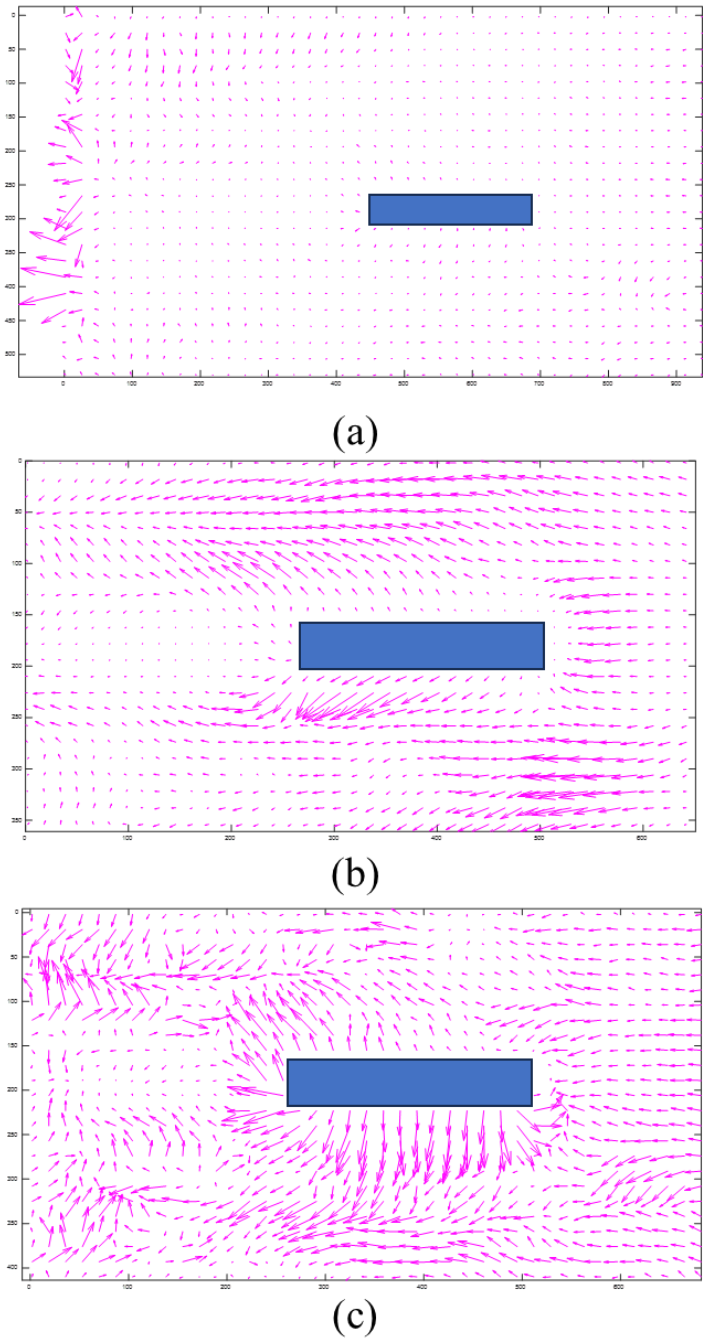



**Figure 5.5 Examples of different calculated results of skin friction field for the proposed video using GLOF method.**

Figure 5.7 shows the trend of $S_t$ over time, whose experiment condition could be found in Figure 5.6. The fluctuations in $S_t$ under different conditions are quite significant, displaying noticeable rises and falls. This "non-stationary" characteristic makes it difficult to identify a representative moment that can describe the typical state of $S_t$.

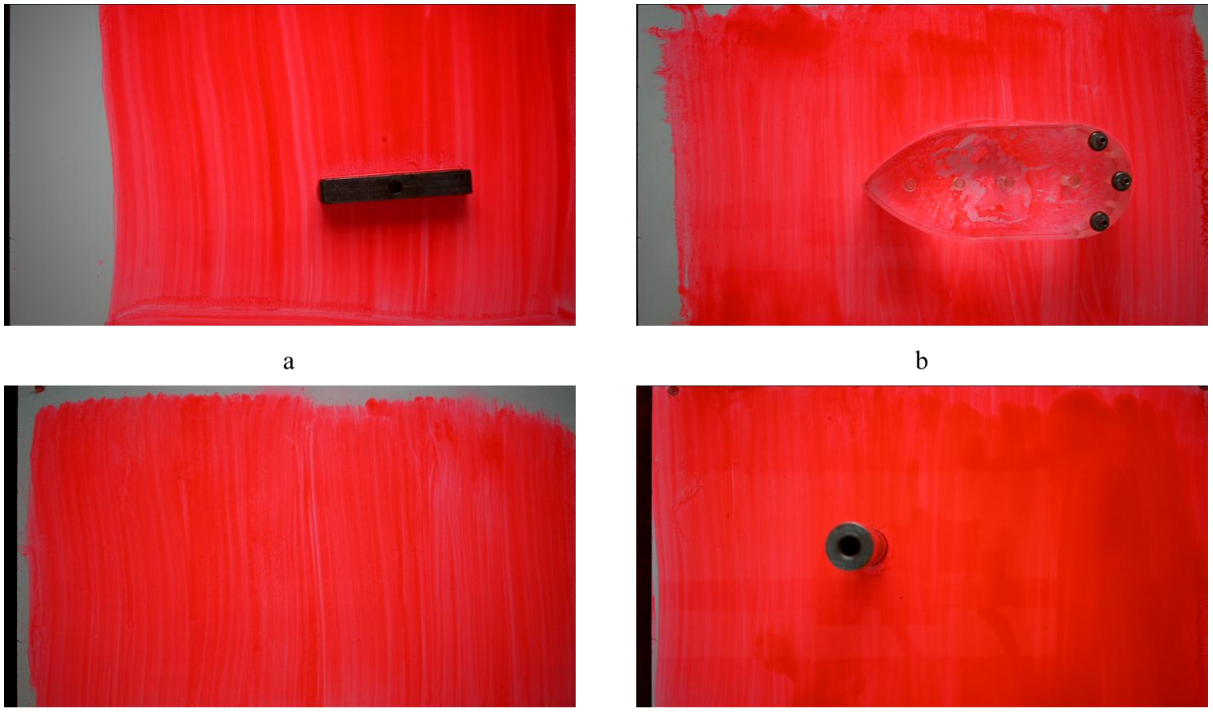



**Figure 5.6 Experiments conducted under different obstacle conditions and their corresponding name. (a) rectangular obstacle; (b) streamlined obstacle;(c) empty obstacle; (d) cylindrical obstacle.**

In scenarios without obstacles, such as the empty space shown in Figure 5.7, the variations in St are relatively stable, with only slight fluctuations, indicating a more stable system under these conditions. However, in other scenarios, when obstacles are present, the fluctuations in $S_t$ increase significantly, with the curve showing much more pronounced undulations. This reflects a greater susceptibility of the system to external disturbances, leading to a more unstable state under these conditions.

For the case represented by the black squares (rectangular obstacle), $S_t$ shows large fluctuations in the early stage, particularly around 30 seconds, where the value rapidly rises from 2 to nearly 4. After this, the fluctuations become more stable. During the middle period from 60 to 120 seconds, while $S_t$ still experiences minor oscillations, the magnitude is small, staying between 2 and 3, indicating that the system gradually stabilizes under this condition. However, after 120 seconds, there is a slight rise in St again, suggesting that the system is not entirely stable and still exhibits a tendency to fluctuate.

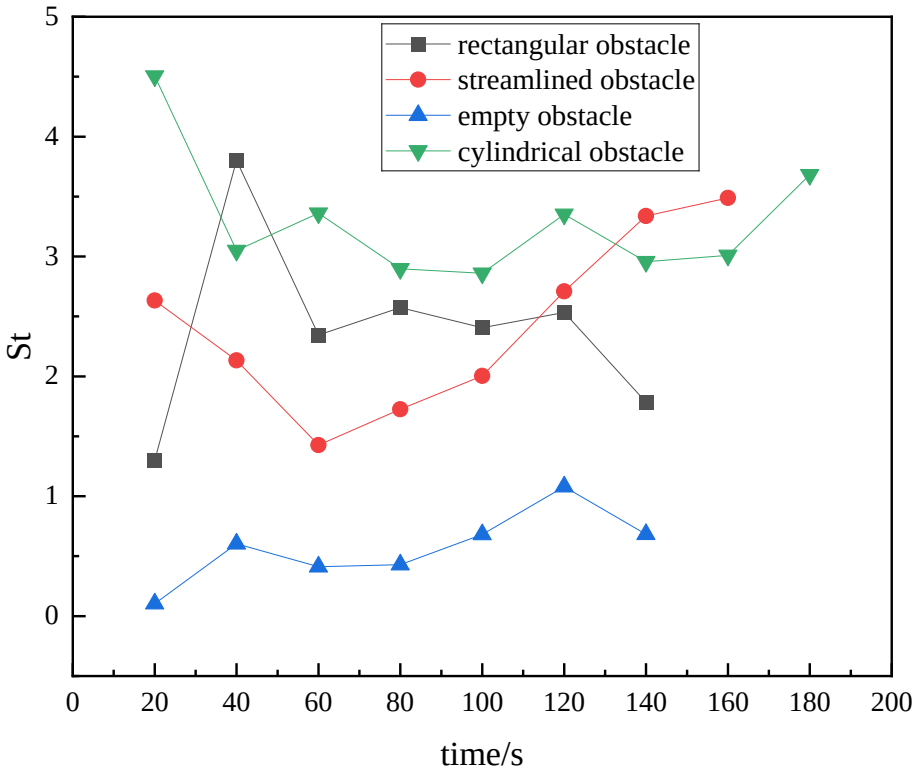



**Figure 5.7 Temporal Variation of St Under Different Obstacle Conditions**

The curve represented by the red dots (streamlined obstacle) shows an overall downward trend. At the beginning, $S_t$ is around 2.5, and as time progresses, the value gradually decreases to about 1.5 at 120 seconds, followed by a slight rebound and stabilization around 2. Compared to the other curves, the fluctuations in $S_t$ under condition streamlined obstacle are smaller, showing that the system is relatively more stable, though still displaying a mild downward trend.

The curve represented by the blue triangles (empty obstacle) is the most stable. The $S_t$ value consistently remains between 0.5 and 1.5, with almost no significant fluctuations. This suggests that under this condition, the system exhibits very high stability, with minimal impact from external disturbances, maintaining a low operating range. This performance might indicate that the system is hardly affected by external interference in this condition, displaying marked stability.

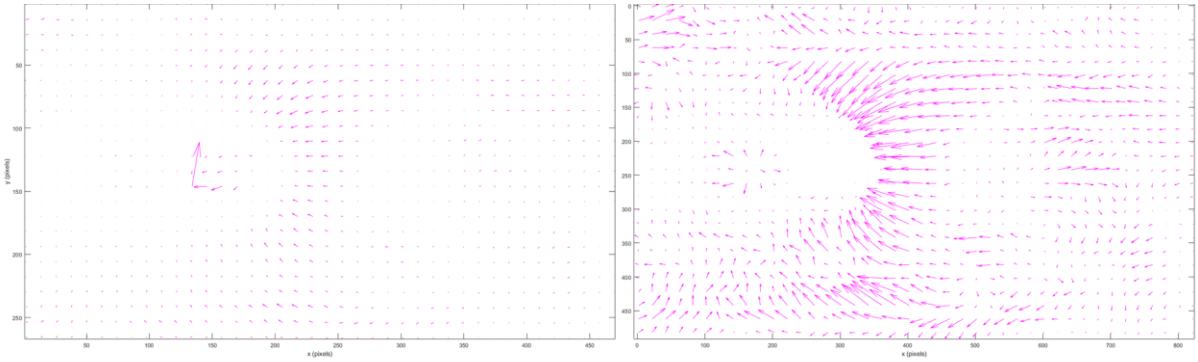
In contrast, the curve represented by the green inverted triangles (cylindrical obstacle) shows noticeable fluctuations. Particularly in the early stage, from 0 to 40 seconds, $S_t$ drops from 4 to around 3.5, followed by some fluctuations. Throughout the time period, the $S_t$ value stays between 3 and 4, reflecting significant system volatility under this condition, indicating a strong instability. Then after 140 seconds, the curve exhibits a clear upward trend, suggesting that the system may have experienced stronger disturbances later on, resulting in a more unstable state.

By analysing this data, it becomes apparent that system stability varies significantly under different conditions. The blue curve exhibits the most stability, with the smallest amplitude of fluctuations, remaining at a low level, which corresponds to a scenario without obstacles. Meanwhile, the green curve shows dramatic fluctuations, with higher $S_t$ values, reflecting a sharp increase in instability under conditions of strong external interference. The red and black curves show moderate fluctuations, with a downward trend and a trend toward stabilization, respectively.

In conclusion, the influence of obstacles or external disturbances on system stability is significant. The data indicates that under conditions with minimal external interference, the system maintains high stability, with small $S_t$ fluctuations. However, when obstacles or strong external disturbances are present, system instability increases markedly, with larger $S_t$ fluctuations.

#### 5.1.1.4 Resolution and Accuracy

The spatial resolution of video frames significantly affects the precision of surface friction measurements. To assess this, skin friction fields calculated from different resolutions: full resolution (1920×1080) and reduced resolution (960×540). The results revealed notable differences. This finding underscores the importance of high-resolution images for accurately determining the surface friction field. High spatial resolution ensures more detailed flow visualization and enhances the reliability of the GLOF results, ultimately leading to more robust experimental conclusions.



(a) (b)

**Figure 5.8 Comparison of Surface Friction Fields at Different Resolutions. (a) reduced resolution (960×540); (b) full resolution (1920×1080)**

These findings discuss several factors that impact the accuracy of results when applying GLOF. The next section will delve deeper into the performance evaluation of GLOF, utilizing experimental results and comparisons with other techniques to validate its effectiveness.

### 5.1.2 Performance evaluation

Figure 5.9a shows the last oil-film image taken during the experimental process. In the experiment, a cylinder was placed on the test article surface, and the corresponding flow visualization was recorded by camera. Since the flow field distribution around a cylinder has been extensively studied, using this example can better illustrate the results and highlight any existing issues. The result image has been preprocessing with techniques described in Section 3.3. A saddle point and the separation lines can be seen in the Figure 5.9a. As shown in Figure 5.9a and Figure 5.9b, the experimental observations are consistent with the theoretical framework presented in Eckerle and Langston (1987)'s work. An accumulation of the oil was also observed behind the cylinder, which is attributed to the formation of vortices in this region.

Figure 5.9c, d present the skin-friction vectors and skin-friction lines, averaged from 80 image pairs. However, a significant discrepancy is observed in the results. According to flow visualization and experimental measurements, the skin friction vector around the cylinder is expected to remain predominantly horizontal before reaching the cylinder. Upon encountering the cylinder, the flow should bifurcate and subsequently reconverge downstream. In contrast, the skin-friction distribution obtained from GLOF deviates substantially from this expected pattern. Instead of exhibiting the anticipated flow separation and reconvergence, the GLOF results indicate a trend of convergence near the cylinder, which contradicts the observed

oil flow pattern. Moreover, the GLOF results introduce numerous artificial vortices that are not physically expected in this flow configuration.

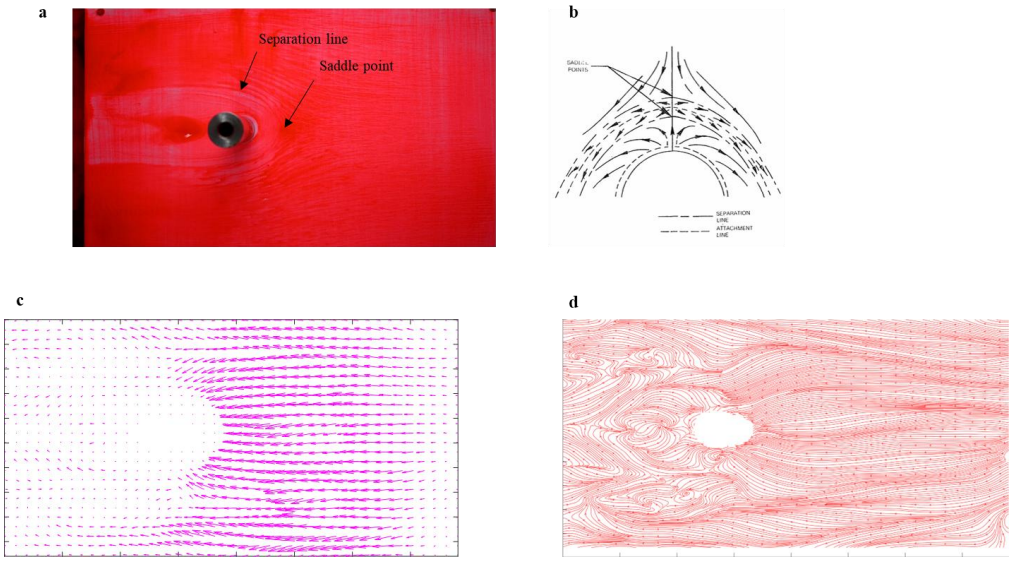



**Figure 5.9 Skin-friction topology by the proposed optical-flow algorithm: a) typical luminescent oil-film image, b) cylinder vortex model developed by Eckerle and Langston (1987), c) skin-friction vectors, d) skin-friction line on the image surface.**

For the purpose of quantifying comparisons, this section presents a structured three-way comparison between the reference experimental data from Eckerle (1985)'s doctoral dissertation, the current GLOF experimental results, and corresponding CFD simulations. By examining agreements and discrepancies across all three results, the performance limitations in the GLOF could be found. The experimental data presented in Eckerle (1985)'s doctoral dissertation served as the baseline for validating both the GLOF measurements and CFD results.

Figure 5.10 illustrates the schematic of the test section, and the coordinate system used in this study. The origin of the coordinate system coincides with the centre of the cylinder and the circular disk at the lower end wall. The angular and radial positions are represented by β and R, respectively. The Cartesian coordinates indicating flow direction, vertical, and lateral positions are x, y, and z. The experiment aims to create vertical and horizontal symmetry planes along the centreline and mid-height of the test section. Given the focus on surface visualization of the end wall, particular attention is placed on the bottom surface of the selected quadrant.

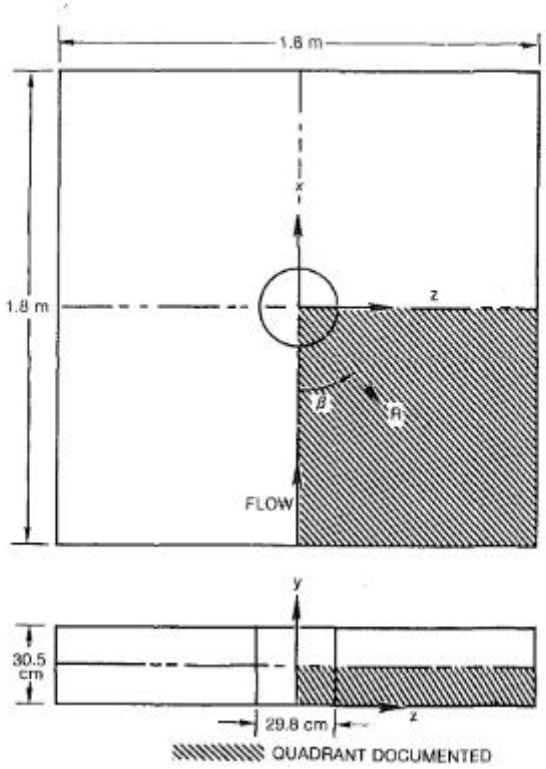



**Figure 5.10 Schematic showing test section with cylinder and coordinate systems from Eckerle (1985)**

Eckerle measured the pressure distribution around a large cylinder using an instrumented floor disk. This disk is mounted on a support plate with multiple static pressure holes, allowing for pressure data acquisition at various radial and angular positions. The setup of the pressure holes enables minimal interference when measuring the static pressure distribution on the end wall and the cylinder surface. Furthermore, the design of the disk allows it to rotate, facilitating precise measurements at different locations and ensuring data symmetry and accuracy. The internal flow field data were obtained using a movable five-hole probe, with data collected in the angular planes defined by $\beta$ = -5, 0, 5, 25, 45, and 90 degrees. The probe was traversed at 81 radial locations to define the flow field adjacent to and within the separation region. A layer at a height of y/D $\approx$ 0.0083 was used for flow visualization as comparison experimental data, representing the lowest point for which data could be collected without end wall interference. The test conditions included a free stream velocity of 30.5 m/s, with a Reynolds number based on the cylinder diameter of $5.5 \times 10^5$, and a boundary layer thickness occupying 13% of the cylinder diameter. Some of the data are summarized in the Table 5-1, reflecting detailed measurements of the flow field through the five-hole probe at different angular planes.

Firstly, the velocity ratio $U/U_0$ shows significant fluctuations with changes in radial position R/D. Near smaller radial positions, such as R/D = 0.29, the velocity ratio drops to approximately 0.4, suggesting significant flow separation in that region. However, as the radial distance increases, the velocity ratio gradually recovers and approaches higher values (around 0.8), indicating a reattachment and recovery trend of the flow away from the wall. Secondly, the static pressure coefficient CPS exhibits negative values at multiple locations, particularly around R/D = 0.4, indicating local static pressure below the reference pressure, possibly related to local pressure changes caused by flow separation in that area. The total pressure coefficient CPT mostly remains near 1.0, but shows slight declines at certain positions, such as R/D = 0.32 and R/D = 0.45, suggesting possible total pressure losses or energy dissipation in those areas. The significant variations in yaw angle Phi and pitch angle Theta, especially around R/D = 0.31 where the yaw angle reaches approximately 14 degrees, reveal the complexity of flow direction, particularly near the separation region, where flow may experience dramatic lateral deflection and turbulence. These data collectively demonstrate the asymmetry of flow in radial and angular planes, especially in the flow structure near the separation zone, showcasing significant changes in velocity, pressure, and direction, providing important experimental evidence for understanding the behaviour of separated flows.

**Table 5-1 Five-Hole Probe Data at y/D ≈0.0083**

| Angluar plane | R/D | y/D | Phi degree | Theta degree | CPT | CPS | U/U0 |
|---|---|---|---|---|---|---|---|
| -5 | 0.809 | 0.0083 | 5.0 | -104.6 | 0.519 | 0.443 | 0.195 |
| -5 | 1.532 | 0.0085 | 0.1 | -2.8 | 0.559 | 0.175 | 0.516 |
| 0 | 0.596 | 0.0078 | -2.2 | -156.7 | 0.132 | 0.802 | 0.257 |
| 0 | 1.532 | 0.0085 | 1.8 | 0.0 | 0.539 | 0.189 | 0.722 |
| 5 | 0.596 | 0.0082 | -4.0 | 138.2 | 0.078 | 0.836 | 0.295 |
| 5 | 1.532 | 0.0083 | 1.1 | 1.9 | 0.481 | 0.175 | 0.587 |
| 25 | 0.553 | 0.0084 | -0.2 | 69.0 | 0.008 | 0.476 | 0.718 |
| 25 | 0.936 | 0.0086 | 1.7 | 44.4 | 0.501 | 0.243 | 0.506 |
| 45 | 0.553 | 0.0083 | 0.3 | 46.6 | -0.001 | -0.552 | 1.234 |
| 45 | 1.191 | 0.0079 | 1.3 | 21.1 | 0.523 | -0.063 | 0.735 |
| 90 | 0.553 | 0.0099 | 0.7 | 1.3 | 0.002 | -1.832 | 1.682 |
| 90 | 2.375 | 0.0085 | 2.1 | 2.0 | 0.666 | -0.23 | 0.751 |

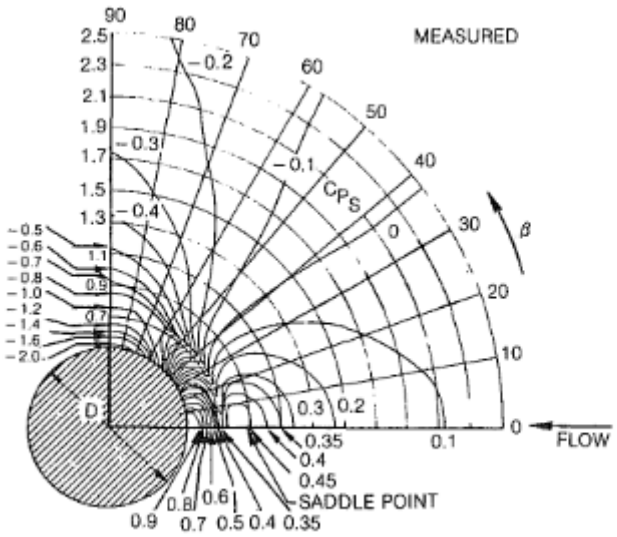



**Figure 5.11 Measured endwall pressure distribution from Eckerle (1985)**

Since the experimental results from Eckerle et al. were nondimensionalized, they can serve as a reference for detailed comparisons with the computational results of this study, which are also presented in a nondimensional form for consistency. In Eckerle's experiments, the coordinate system uses the centre of the cylinder and the disk as the origin, with the radial position represented by the nondimensional variable R, the angular position by β, and the streamwise, vertical, and transverse location represented by the nondimensional x, y and z coordinates, respectively. To ensure a reasonable comparison between the two datasets, the computed flow field data must first be nondimensionalized to match the experimental coordinate system. This involves normalizing the computational results based on the cylinder diameter or other relevant characteristic lengths.

To accurately match corresponding positions in the flow field, points with the same nondimensional coordinates (R and β) from the computational results can be extracted based on the radial and angular positions provided in Eckerle's experiments, which explicitly state the symmetry of the flow, indicating the existence of vertical and horizontal symmetry planes along the centreline and mid-height of the test section. Therefore, during the comparison process, priority should be given to selecting key points within these symmetry planes for quantitative comparisons of physical quantities such as velocity and pressure, allowing for an in-depth evaluation of the consistency between numerical simulation results and experimental measurements.

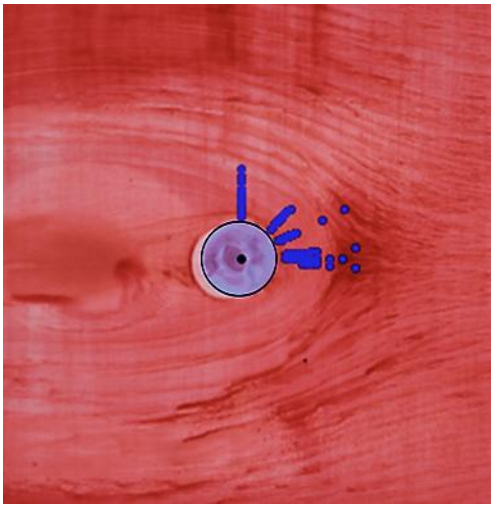


**Figure 5.12 Corresponding points in the GLOF calculation for Eckerle's experiment**

Despite Eckerle's experiments accurately representing velocity direction through nondimensional radial and angular positions (R and β), the overall trends show significant discrepancies when compared to computational results of this work. Particularly in the regions of $\beta < -50^{\circ}$and $\beta > 50^{\circ}$, the velocity directions measured experimentally do not align well with the computational results. To ensure consistency in nondimensional analysis, the flow conditions for these results correspond to a Reynolds number of $5.5 \times 10^5$, ensuring direct comparability with experimental data. This indicates that the numerical simulation exhibits some deviations in capturing the flow characteristics in these regions, potentially related to model assumptions or the handling of boundary conditions.

Figure 5.13 presents a comparative analysis between the GLOF method and true theta values derived from Eckerle's experimental setup. The theta is the flow vector calculated at the same points as in Eckerle's result. It features a red line representing the ideal scenario where the GLOF results perfectly align with the true values. However, a significant disparity is evident in the plotted black points, which correspond to the GLOF theta measurements.

The GLOF theta values show marked deviations from the expected true theta values. This suggests that the numerical simulations may struggle to accurately capture the flow characteristics in these specific regions. Near the critical angles, where separation and complex flow phenomena are expected, the GLOF method fails to reflect the precise directional changes noted in Eckerle's experimental results.

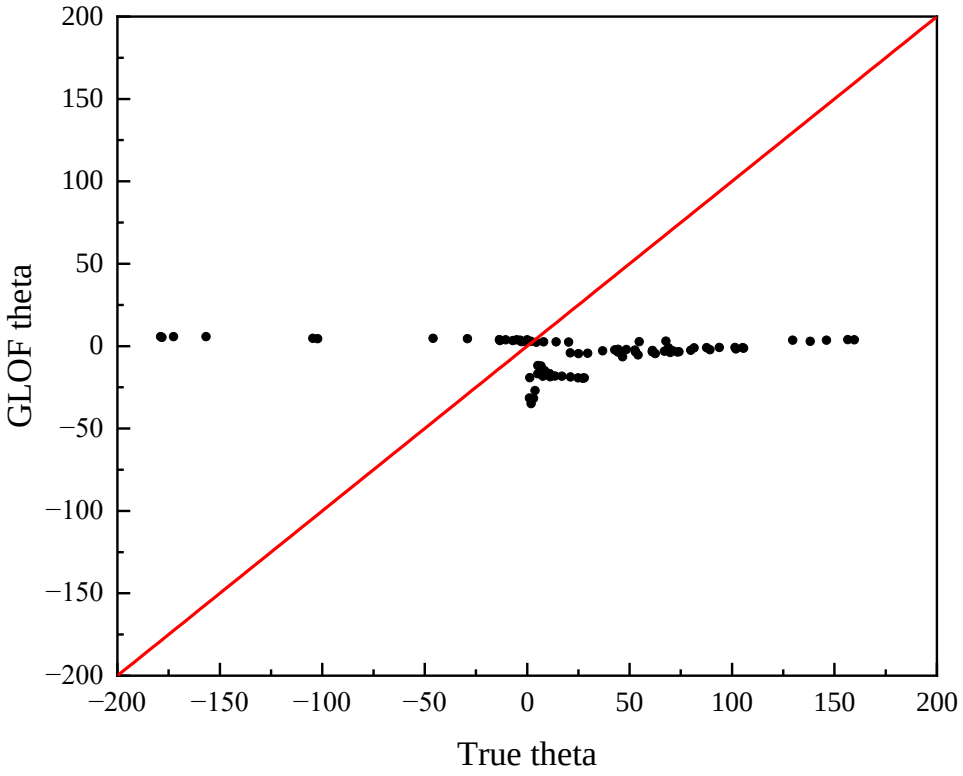



**Figure 5.13 Comparison of experimental angle and GLOF calculation angle**

## 5.2 Simulating a surface in experiments

### 5.2.1 Test Configuration

This study adopted a geometric configuration similar to that used in the flow visualization experiments. As shown in Figure 5.14, a finite cylinder with a width of $\boldsymbol{D}$ = 0.024 m and a height of $\boldsymbol{h}$ = 0.3 m was vertically mounted on a flat boundary. The computational domain's length (in the flow direction), width (in the lateral direction), and height (in the spanwise direction) were $\boldsymbol{L}$ = 0.607 m = 25.3 D, $\boldsymbol{W}$ = 0.302 m = 12.6 D, and $\boldsymbol{H}$ = 0.3 m = 12.5 D, respectively. Additionally, the junction of the cylinder and the bottom wall was located at the origin of the coordinate system, meaning that the inlet boundary was positioned 0.200 m upstream of the cylinder, while the outlet boundary was 0.406 m downstream of the cylinder.

CFD was employed as a tool for simulating the flow around the cylinder. The CFD approach used varied depending on the task at hand. In this case, ANSYS Fluent was employed for flow simulations.

### 5.2.2 Meshing

In this section, the details of the computational mesh generation are presented to ensure the accuracy and reliability of the CFD simulations.

The mesh is generated using ICEM CFD with a structured multi-block approach. The domain is divided into several zones to ensure smooth transitions and proper resolution of critical regions, particularly around the cylinder. The mesh comprises approximately 2 million elements, primarily hexahedral cells in the bulk flow. The grid resolution is approximately 320, 160, and 40 cells in the X, Y, and Z directions, respectively. This corresponds to roughly 51,200 elements per two-dimensional XY layer.

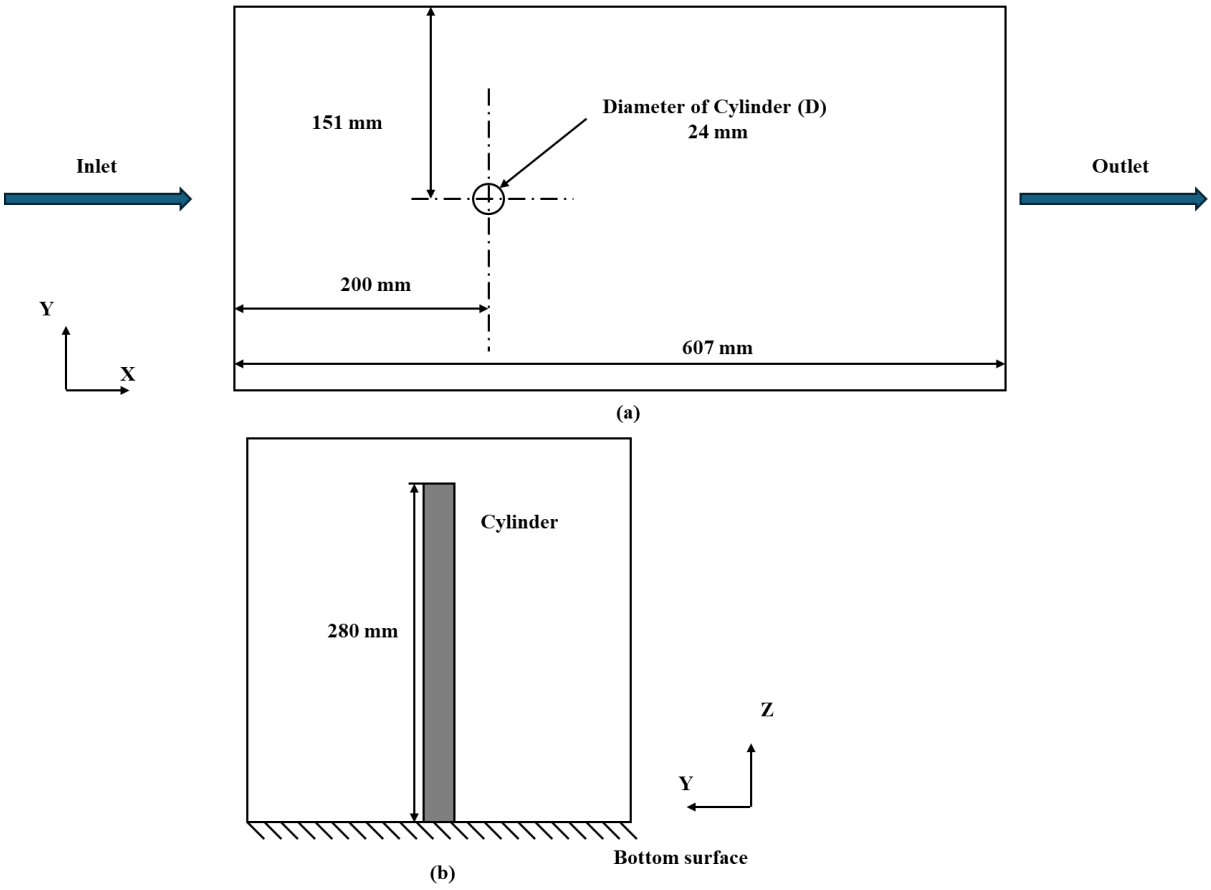



**Figure 5.14 Model domain configuration stretched to show principal dimensions**

To resolve the boundary layer effectively, the first grid point is placed at a normalized wall distance (y+) of equal to 1. The boundary layer thickness, calculated as $2.56\times10^{-5}$ based on Wilcox (1998)'s work, is resolved with 50 layers of elements in the radial direction. This ensures that flow characteristics near the surface, including shear stress and separation behaviour, are accurately captured.

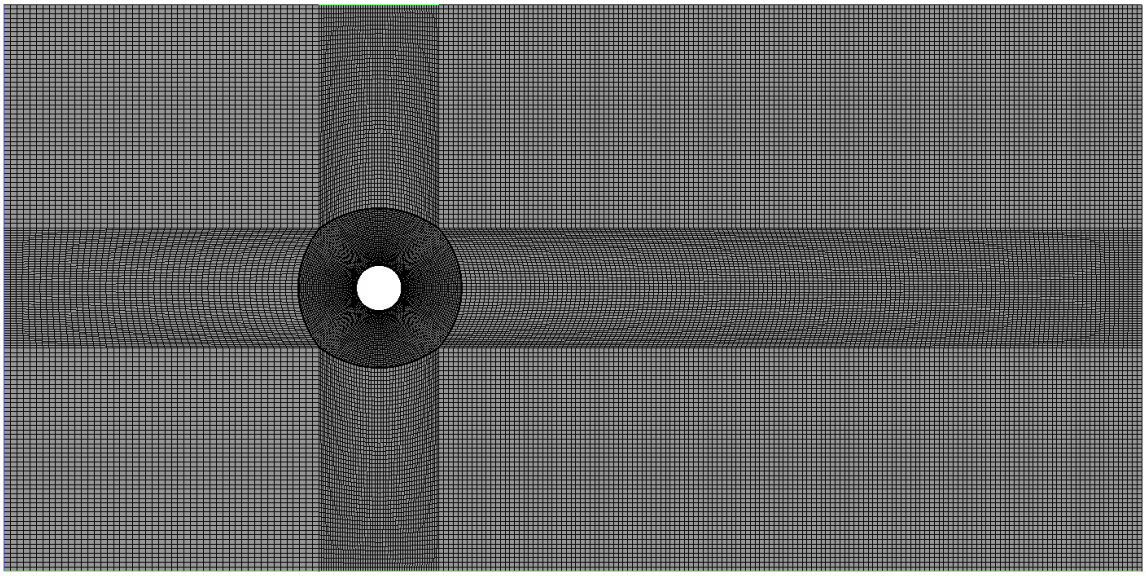


**Figure 5.15 The computational grid on the bottom plane**

### 5.2.3 boundary conditions

Four types of boundary conditions are applied in the simulation to ensure that the Reynolds number matches the experimental value. At the inlet, a fixed uniform velocity is prescribed for the velocity field, where $u$ = 19 m/s and $v = w = 0$, while a zero-gradient condition is imposed on the pressure field, $\frac{\partial p}{\partial n} = 0$. At the outlet, the pressure field at the outlet, a homogeneous Dirichlet condition is applied, which means $p = 0$ is everywhere at the boundary. On the bottom wall and the surface of the obstacle, a no-slip impermeable boundary condition is prescribed for the velocity field, setting $u = v = w = 0$, while a zero-gradient condition is applied to the pressure field. For the top and lateral boundaries of the computational domain, a free-slip condition is imposed, meaning the velocity component normal to the boundary is zero.

The simulations were performed using ANSYS Fluent 2022 R1, with the k-ε turbulence model employed to capture turbulence effects, especially in the wake region of the cylinder. Steady-state simulations were conducted, with a convergence criterion of $10^{-6}$ for residuals.

### 5.2.4 Results and discussion

This section investigates the mean velocity and pressure fields by analysing the time-averaged streamlines and pressure contours in two characteristic planes: the near endwall plane of the cylinder at Z/D = 0.01 and the symmetry plane at Y/D = 0. As illustrated in Figure 5.16, a strong positive and negative pressure region develops both in front of and behind the cylinder

due to its obstructive effect on the flow. Furthermore, the topology of the average streamlines in this plane reveals two symmetrically distributed vortex centres, labelled C and D, along with a saddle point identified as R in Figure 5.16.

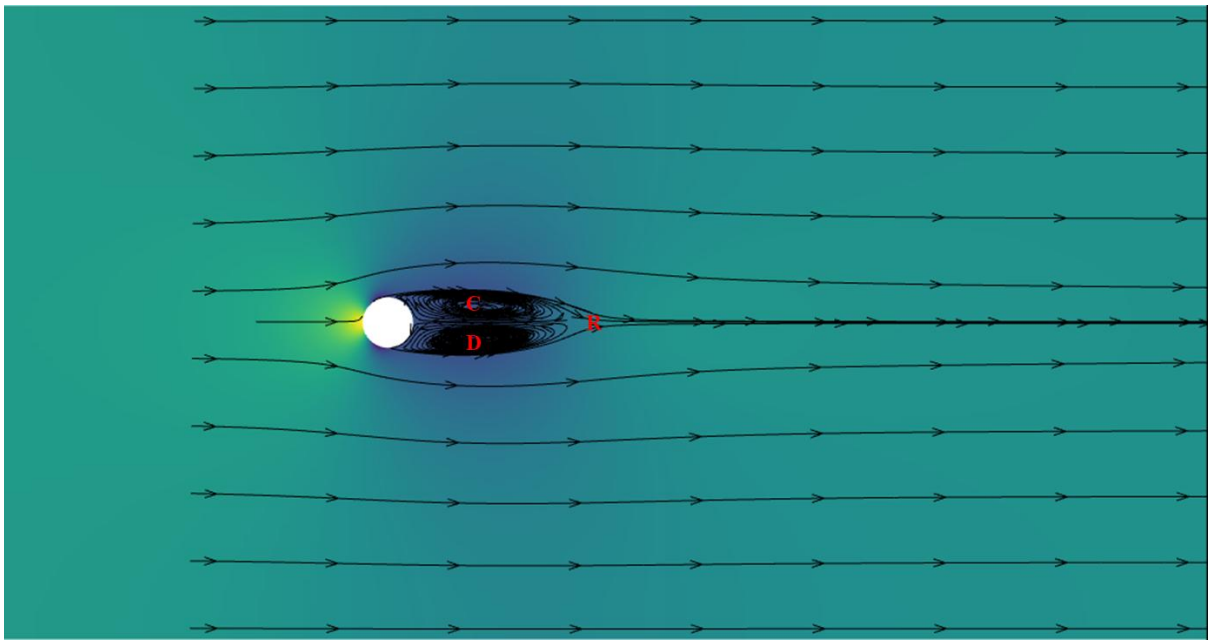



**Figure 5.16 Comparison of the mean streamlines and pressure contours on the Z/D=0.01 plane**

As shown in the Table 5-2, which could also be found in Figure 5.17, the theta degrees from the CFD simulation generally align well with the corresponding GLOF degrees across most of the angular planes, indicating a good agreement between the two methods. It should be noted that the data shown in Table 5-2 was only some representative values, as displaying the full dataset would result in an impractical and overly lengthy presentation. The values of the CFD and GLOF measurements for theta are mostly within a narrow range, with only slight deviations observed in certain regions. For example, at an angular plane of -5° and R/D=0.809, the CFD measurement of theta is -104.6°, while the GLOF measurement is 4.7°, showing a significant discrepancy. This difference could be attributed to potential experimental setup variations, such as slight misalignments or inconsistencies in the flow visualization technique at this particular point.

Other discrepancies, such as the differences between CFD and GLOF measurements at R/D=0.553 and 45° angular plane, where the CFD value is 46.6° and the GLOF value is 5.6°, could be due to the inherent limitations of the numerical simulation, particularly in regions with more complex flow dynamics like vortex formation and recirculation areas, where turbulence models might introduce small inaccuracies.

**Table 5-2 Five-Hole Probe Data compare with GLOF and CFD data at y/D ≈0.0083**

| Angluar plane | R/D | y/D | Phi degree | Theta degree | GLOF degree | CFD degree |
|---|---|---|---|---|---|---|
| -5 | 0.809 | 0.0083 | 5.0 | -104.6 | 4.7 | -83.2 |
| -5 | 1.532 | 0.0085 | 0.1 | -2.8 | 4.7 | -4.3 |
| 0 | 0.596 | 0.0078 | -2.2 | -156.7 | 4.4 | -116.6 |
| 0 | 1.532 | 0.0085 | 1.8 | 0.0 | 3.3 | -8.5 |
| 5 | 0.596 | 0.0082 | -4.0 | 138.2 | 3.4 | 103.8 |
| 5 | 1.532 | 0.0083 | 1.1 | 1.9 | 2.6 | -4.0 |
| 25 | 0.553 | 0.0084 | -0.2 | 69.0 | 5.7 | 77.0 |
| 25 | 0.936 | 0.0086 | 1.7 | 44.4 | 5.7 | 30.0 |
| 45 | 0.553 | 0.0083 | 0.3 | 46.6 | 5.6 | 34.0 |
| 45 | 1.191 | 0.0079 | 1.3 | 21.1 | 5.2 | 15.5 |
| 90 | 0.553 | 0.0099 | 0.7 | 1.3 | 4.5 | -12.5 |
| 90 | 2.375 | 0.0085 | 2.1 | 2.0 | 4.0 | -5.7 |

In general, the CFD degrees closely match the GLOF degrees, with deviations that are likely due to factors such as grid resolution, turbulence model assumptions, and the sensitivity of the CFD solution to boundary conditions. These discrepancies highlight the challenges in accurately capturing complex flow features near the cylinder's surface and wake region, where turbulence-induced effects are pronounced. Nonetheless, the overall consistency between the CFD and GLOF results supports the reliability of the CFD simulations in replicating the experimental findings under the conditions studied.

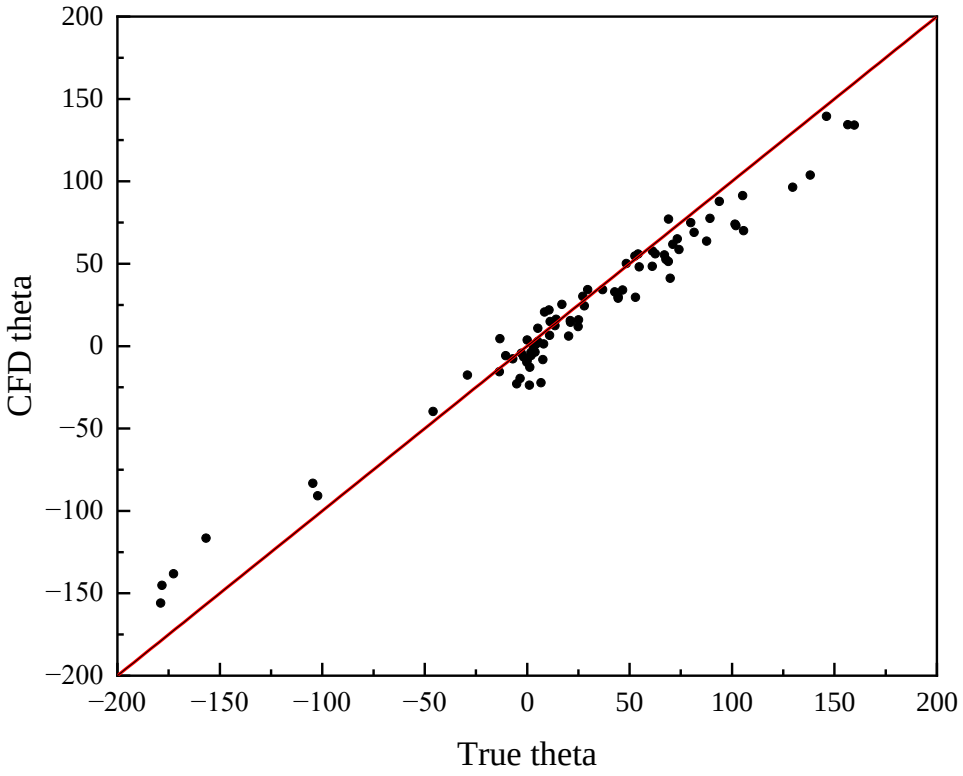



**Figure 5.17 Comparison of experimental angle and CFD calculation angle**

## 5.3 Conclusion

This chapter evaluates the performance of the Global Luminescent Oil-Film method, especially its application in surface flow visualization. The results show both its potential and limitations. The GLOF method can provide non-intrusive flow visualization and capture details like vortex formation and boundary layer separation. However, it has become clear that this method is not suitable for studying surface friction fields around complex geometries, such as a cylinder in flow.

In this thesis, the inapplicability of the GLOF method is directly reflected in its difficulty in accurately reproducing the expected flow behaviour around a cylinder. Although the GLOF results can match certain visual features of the flow, such as saddle points and separation lines, there are obvious differences in the surface friction patterns compared with experimental data and CFD results. GLOF introduces vortices that should not exist physically and captures abnormal flow patterns in areas where separation occurs. This inconsistency

shows that while GLOF provides useful insights for flow visualization, it needs further improvements to improve the accuracy of its friction field calculations, especially in complex flow scenarios. The reasons for these limitations may because of the GLOF technique's reliance on optical flow data and the complexity of processing this information, which can lead to errors in identifying flow fields, especially in flow areas where separation or reattachment occurs.

# Chapter 6 Synthetic Data Generation and Unet Training

This chapter focuses on the research method of using generative adversarial networks (GAN) and U-Net models to predict surface flow fields by using experimental data and synthetic data. The process can be divided into four key stages, as shown in Figure 6.1.

In the sGAN training phase, which is outlined in red in the Figure 6.1, experimental data is collected and processed. The process starts with a calibrated visualization image, which has undergone several preprocessing to highlight important features. The followed steps include dividing the image into smaller cells to capture localized flow details, grayscale conversion and image smoothing to reduce noise and improve feature clarity. The final step of this phase involves exposing intensity gradients, which enhances the visibility of key flow structures. Once pre-processing is complete, an edge detection step will generate a streaks image. This streaks image serves as input for the sGAN, marking the end of the training phase for the GAN.

The image synthesis phase involves using the trained sGAN to generate synthetic data. This phase starts with defining random surface flow boundary conditions, followed by modeling and meshing to simulate surface geometries. These geometries are used in OpenFOAM CFD simulations, producing a surface flow field. The resulting flow field is then converted into a synthetic visualization image. The sGAN processes this image to create a synthetic streaks image, which closely mimics the texture and features of real experimental data. This phase is crucial for generating a diverse dataset that improves the robustness of the model.

In U-net training phase which is outlined in yellow in the Figure 6.1, both experimental and synthetic data are combined to train a U-Net model. The inputs to the U-Net include the calibrated visualization images from experiments and the synthetic visualization images generated by the sGAN. Additionally, the chaincode method is used to annotate flow structures in the experimental data, providing detailed information for training. The U-Net is trained to output surface flow fields, learning from both real and synthetic data to enhance its predictive accuracy and generalization capabilities.

Each output pixel of the U-Net corresponds to a local flow orientation, encoded through a colour-to-angle mapping scheme. Through this way, the output image could serve as both a visual and quantitative result of the flow field.

In the application phase (shown in green), the trained U-Net is used for flow field prediction. It takes a new visualization image as input and generates a predicted surface flow field.

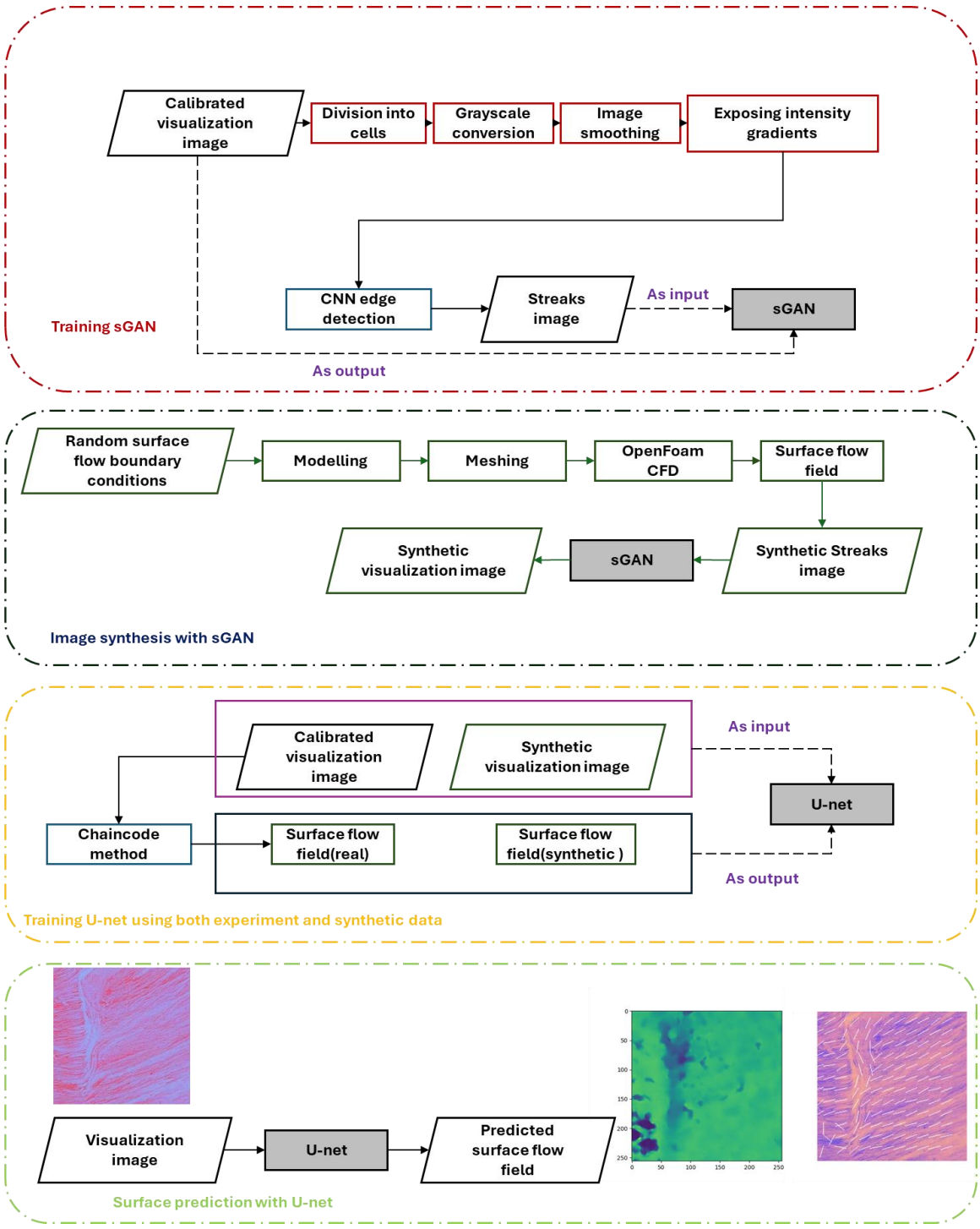



**Figure 6.1 Flow chart of flow field prediction algorithm**

## 6.1 Data Collection and Preparation

### 6.1.1 Experimental Data Acquisition

The data is sourced from experiments, with the basic principles aligning with the methods described in the previous chapter Methodologies 3.2.1, although some details have been adjusted specifically for the neural network model.

In this work, a method known as "Chaincode"(Bansal et al., 2011) was employed to annotate the experimental images. In brief, similar to the approach described in the preceding chapter, a set of ridges was first extracted from the images using algorithms. The direction of each ridge was manually assessed by the researchers and subsequently mapped to a vector field. A total of 50 images were processed using this method.

Chaincode is widely used in fingerprint recognition to represent object contours. A pixel image can be fully reconstructed from the chaincode of its contour, allowing for the extraction of minutiae. In this method, the image is scanned from top to bottom and right to left, detecting transitions from white (background) to black (foreground). The contour is then traced counterclockwise and represented as an array of contour elements. Each contour element corresponds to a pixel on the contour, containing the pixel's x and y coordinates, the slope or direction in which the contour enters that pixel, and curvature along with other auxiliary information.

In Figure 6.2b, minutiae locations within the chaincode contours are displayed. The black regions represent the binarized contours of the target ridges, which have been traced using the chaincode algorithm. The red points highlight ridge endings, where the contours exhibit significant leftward turns, indicating termination points of the ridges. In contrast, the blue points denote bifurcation points, which correspond to significant rightward turns where a single ridge splits into multiple paths.

In the original binarized flow visualization images derived from flow visualization experiments, the main area of the target streak is wider than a single pixel. As the ridge boundary is traced counterclockwise, termination feature points (the ends of the ridges) are detected when there are significant turns in the trajectory. To calculate the angle theta at point P, as illustrated in Figure 6.2c, vector $P_{in}$ points to contour point P, while vector $P_{out}$ points outward from P.

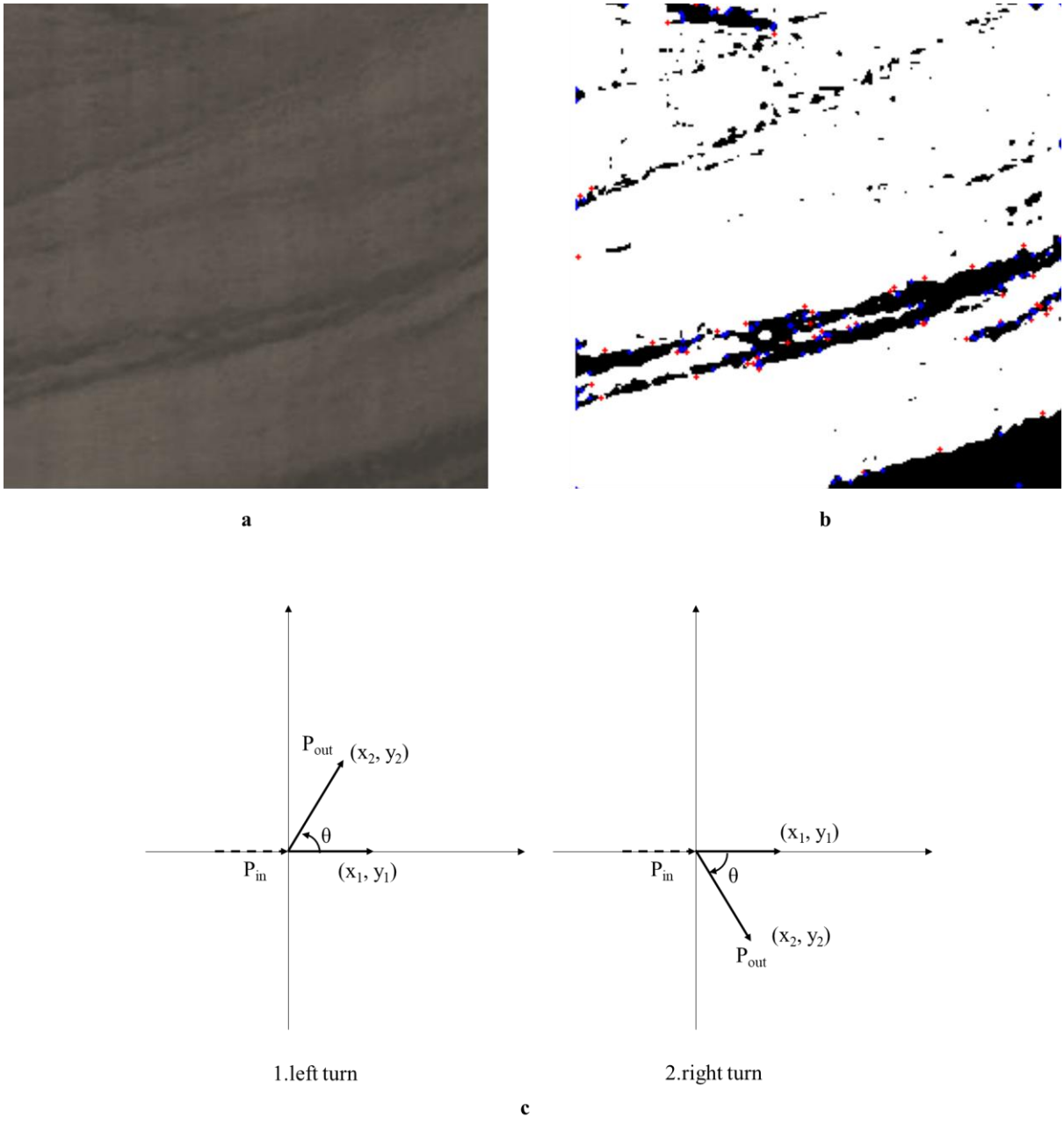



**Figure 6.2 Flow Visualization and Minutia Extraction: (a) Grayscale Image of Flow Visualization (Partial Region), (b) Minutia Locations in Chaincode Contours, (c) Significant Turn Calculation in Minutia Analysis**

The calculations for Pin and Pout utilize multiple adjacent contour points to mitigate local noise and provide a better vector estimate through averaging. $\theta$ can be expressed by the following formula:

$$\theta = \cos^{-1} \frac{P_{in} \cdot P_{out}}{|P_{in}| \cdot |P_{out}|} \tag{6.1}$$

Using chaincode to generate the directional field is more efficient and robust in image enhancement for several reasons: (1) chaincode generation relies on pre-binarization algorithms; (2) both adaptive binarization and chaincode generation algorithms are efficient; (3) the directional field is directly computed by tracking the chaincode on a discrete grid. The goal is to obtain the ridge directions for the entire window, rather than for each individual pixel.

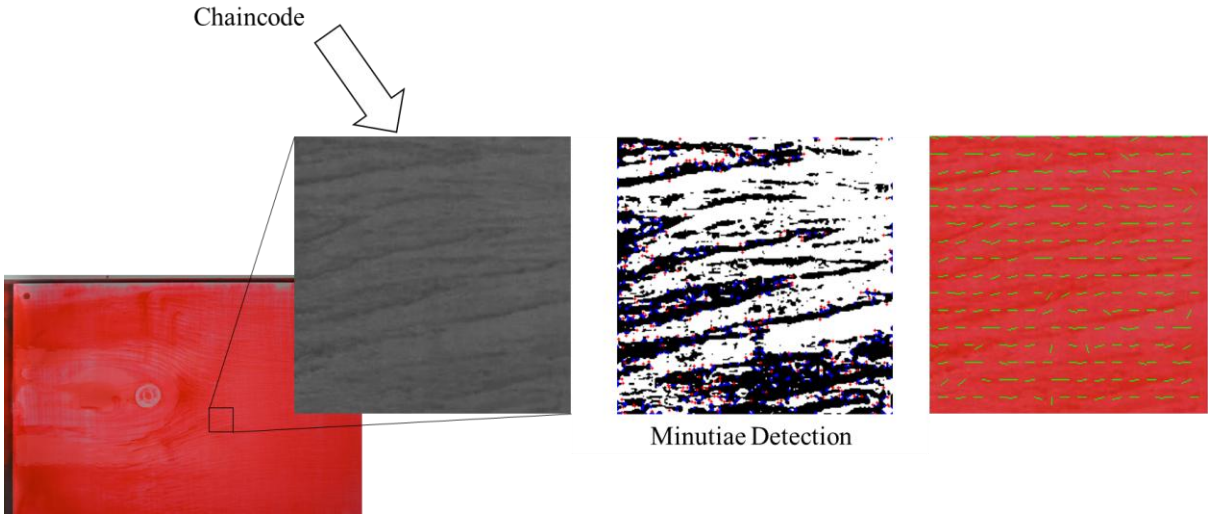



**Figure 6.3 Chaincode Method for Minutiae Calculation and Flow Field Visualization.**

Figure 6.3 shows an example of the flow field analysis process on a test surface using minutiae detection and directional feature extraction. The leftmost image illustrates the original test surface captured under specific imaging conditions, where a ROI is highlighted for detailed analysis. The magnified grayscale patch reveals the fine-grained texture patterns characteristic of natural wood grains. In the centre, the minutiae detection step identifies critical ridge features such as endpoints and bifurcations. The rightmost image demonstrates the estimated local orientation field, where short green line segments indicate the dominant flow direction within each subregion.

After collecting a substantial amount of the experiment data, the steps involved in data processing are critical for constructing a specific dataset. This dataset must then be divided into training, testing, and validation sets in a ratio of 8:1:1 to facilitate subsequent model training(Goodfellow, 2016). The complexity of data processing primarily arises from the inherent errors commonly present in experimental data, which necessitates prioritizing the model's ability to "learn" the mapping relationship between specific experiments and abstract flow fields. To achieve this effectively, the dataset must be constructed based on certain assumptions.

In this context, the RLF method(Leer and Kempf, 2021) provides an important theoretical foundation. According to relevant literature, this method derives the following points from a set of initial assumptions: (1) the flow at a given point in the flow field is naturally influenced by predetermined physical boundary conditions and surrounding geometrical shapes; (2) any wall surface in contact with the flow will affect the flow; (3) the influence of geometrical elements close to the point of interest in the flow field typically outweighs that of more distant elements. Although these three assumptions are not universally applicable, they offer significant insights for mapping from geometrical shapes to flow.

The complete flow visualization is shown in Figure 6.3. Upon zooming into the detailed regions, the results of the chaincode application can be observed, where the black and white colours represent image intensity. Black indicates regions where streaks are likely to be present. Blue dots represent the locations of endings, while red dots denote bifurcations. However, the flow field obtained from this information is sparse. To obtain directional information for each pixel, linear interpolation is applied. The final flow field is represented by the green lines in the image on the right.

Figure 6.4 illustrates the distribution ranges of the velocity components u and v in the entire experiment dataset, where u represents the horizontal velocity component and v represents the vertical velocity component. The values of u and v typically range from -100 to +100, with approximately 95% of the values concentrated within this range. To prevent outliers from impacting the final results, these anomalous values were manually removed prior to dataset construction. Since typical u and v values show minimal variation compared to these extreme values, retaining such outliers could potentially lead to biased model training and reduced prediction stability, as documented in Tukey (1977)’s and Agrawal and Agrawal (2015)’s work.

To fill the gaps left by the removal of outliers, a linear interpolation method was applied. This approach effectively restores data continuity, allowing the dataset to maintain physical consistency while minimizing biases that may arise from missing values.

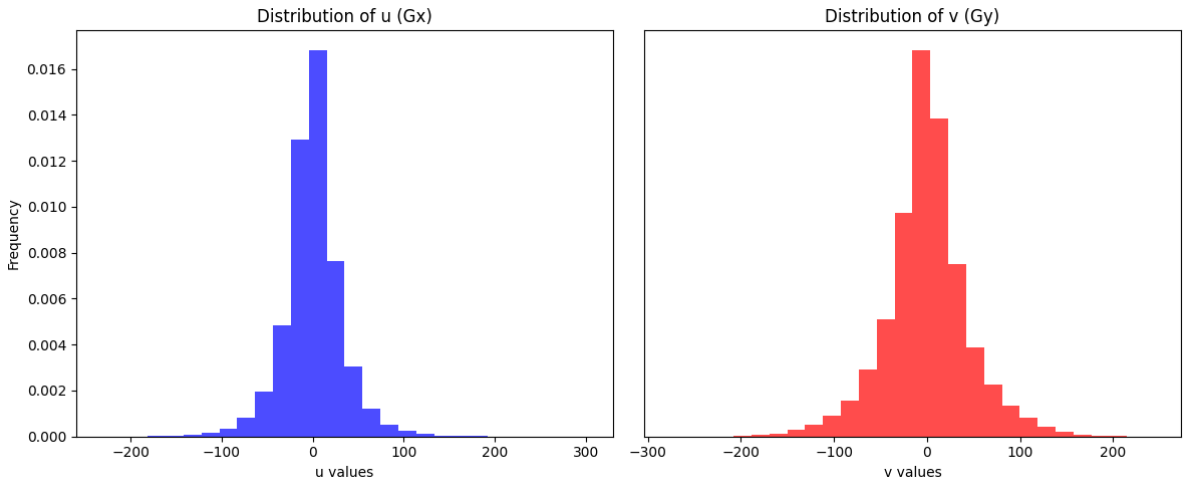



**Figure 6.4 Distribution of u and v Values**

In the data processing procedure, physical consistency analysis is an essential approach for verifying the accuracy of experimental or numerical simulation data. This analysis ensures that the phenomena described in the dataset align with fundamental theories of fluid mechanics by checking whether the data adheres to known physical laws or principles. Based on the research by Bansal et al. (2011), this study focuses on examining the continuity of the dataset to ensure that the resulting data is not only statistically reasonable but also physically reliable.

The expression used to verify the continuity equation for an incompressible flow is as follows:

$$\nabla \boldsymbol{u} = 0 \tag{6.2}$$

$$\nabla \boldsymbol{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \tag{6.3}$$

Therefore, for an ideal system, the theoretical value of $\nabla \boldsymbol{u}$ should be zero or approximately zero. By examining $\nabla \boldsymbol{u}$ at all points in the flow field, it is possible to identify any points exhibiting "inconsistency". Figure 6.5 presents the calculated results of $\nabla \boldsymbol{u}$ for selected images in the dataset. In this figure, darker colours indicate higher values, while lighter colours signify values closer to zero. Observing the data distribution in the figure makes it possible to conclude that most flow characteristics in the dataset exhibit strong continuity. This is clearly reflected in Figure 6.5, where the variations in $\nabla \boldsymbol{u}$ values across points display a relatively stable trend, indicating that physical consistency has been maintained throughout data processing, thus ensuring the reliability of fluid motion characteristics. In summary, the

dataset demonstrates high integrity and continuity, making it well-suited for model training purposes.

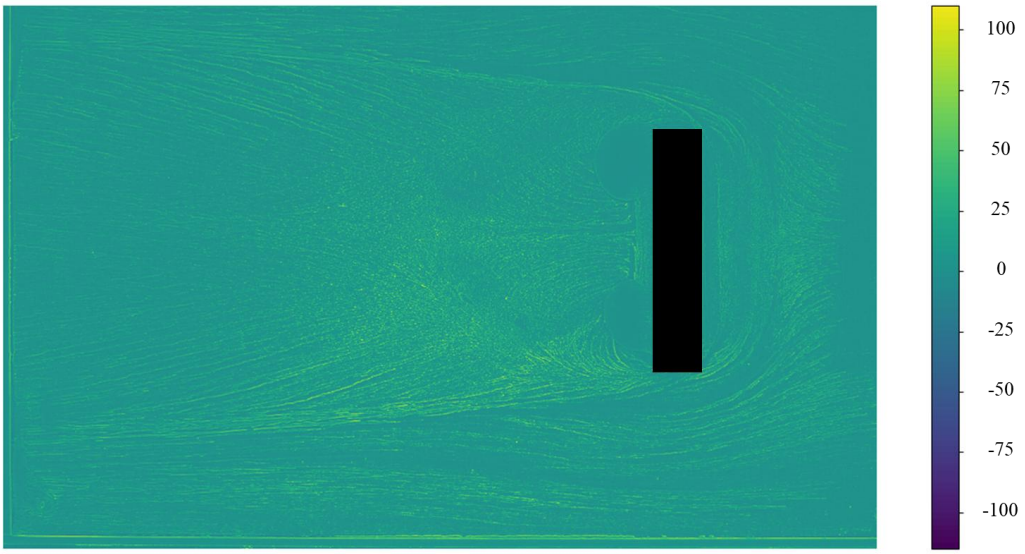



**Figure 6.5 ∇u Distribution for Flow Field Consistency**

## 6.2 Synthetic Data Generation

As noted in the previous chapter, despite being a relatively novel technique in recent literature, the GLOF method exhibits poor robustness and does not align well with experimental results, rendering it unsuitable for applications such as generating training data. However, due to the necessity of having a reliable data source, the focus is shifted to GANs (Generative Adversarial Networks), which can generate synthetic images to serve as a training set. This approach has been documented in works by authors such as Frid-Adar et al. (2018) and Antoniou (2017). In simple terms, this involves utilizing techniques like style transfer to enhance the data generation process.

The basic structure of GANs consists of a generator and a discriminator, with most existing research being based on the original GAN framework proposed by Goodfellow et al. (2014). Therefore, improvements and applications based on this structure are of significant reference value and can serve as benchmark models for comparative experiments. This section compares four models: the proposed sGAN, the baseline model pix2pix(Isola et al., 2017), GauGAN(Park et al., 2019), and CycleGAN(Zhu et al., 2017).

First, a brief overview of these models is provided: pix2pix, based on Conditional Generative Adversarial Networks (cGANs), is used for image-to-image translation tasks and

has been widely applied in tasks such as image inpainting and style transfer, as shown in Figure 6.6a. GauGAN generates high-quality landscape images from user-provided sketches or semantic segmentation maps, demonstrating strong detail generation capabilities, as shown in Figure 6.6b. CycleGAN, on the other hand, performs image-to-image translation through unsupervised learning, enabling style transfer or image synthesis without paired data, and is particularly suited for domain adaptation tasks, as shown in Figure 6.6c.

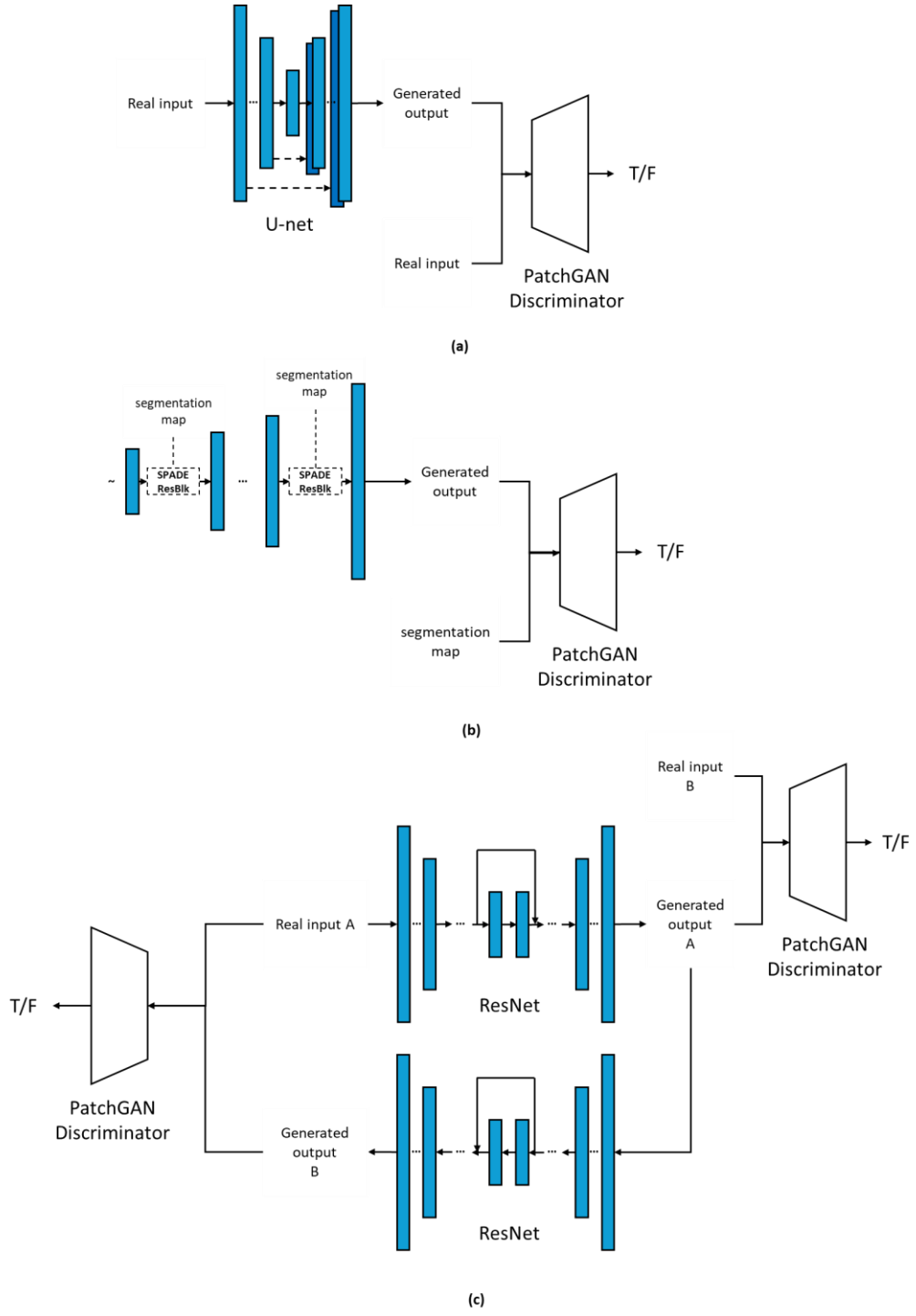



**Figure 6.6 GANs Architectures. (a) pix2pix; (b) GauGAN; (c)CycleGAN.**

### 6.2.1 Training process

In this work, the experimental data from Section 6.1 were used to train the GAN neural network to generate synthetic images. The GAN consists of two main components: a discriminator and a generator, as described in the following paragraph. The GAN employed in this study is based on the CycleGAN architecture(Zhu et al., 2017). However, due to the specific requirement for streak detection in the task, the generator is replaced with a U-Net architecture, and an attention mechanism(Vaswani et al., 2017) is incorporated to improve streak recognition. This modified network is referred to as SGAN.

In the GAN architecture, the generator network consists of five blocks, each of which includes a Convolution Transpose layer, Batch Normalization, and ReLU activation. Only the last Convolution Transpose layer is followed by a Tanh activation layer, replacing Batch Normalization and ReLU. The detailed parameters for each layer in the generator network are shown in Table 6-1.

The discriminator network is composed of five consecutive convolutional blocks. Each block includes a Convolution layer, Batch Normalization, and a LeakyReLU activation layer. Unlike the generator, the discriminator uses LeakyReLU as the activation function in all blocks except the last one. The last block substitutes the Batch Normalization and LeakyReLU layers with a Sigmoid and Flatten layer to output a single prediction score. The detailed parameters of the discriminator network are provided in Table 6-2.

Both networks were trained using the Adam optimizer with a learning rate of 0.0002. The decay factors for the Adam optimizer were set as follows: $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

**Table 6-1 The Parameter of the Generator Architecture**

| No. | Names of the Layers | Number of Convolutional Layer Filters | Convolutional Layer Filter Size/Stride/Padding |
|---|---|---|---|
| 1 | Input + Reshape | 3 | - |
| 2 | Conv2d + LeakyReLU + BatchNorm | 64 | 4/2/1 |
| 3 | Conv2d + LeakyReLU + BatchNorm | 128 | 4/2/1 |
| 4 | Conv2d + LeakyReLU + BatchNorm | 256 | 4/2/1 |
| 5 | Conv2d + LeakyReLU + BatchNorm | 512 | 4/2/1 |
| 6 | Conv2d + LeakyReLU + BatchNorm | 512 | 4/2/1 |
| 7 | Conv2d + LeakyReLU + BatchNorm | 512 | 4/2/1 |
| 8 | Conv2d + LeakyReLU + BatchNorm | 512 | 4/2/1 |
| 9 | ConvTranspose2d + ReLU + BatchNorm | 512 | 4/2/1 |
| 10 | ConvTranspose2d + ReLU + BatchNorm | 512 | 4/2/1 |
| 11 | ConvTranspose2d + ReLU + BatchNorm | 256 | 4/2/1 |
| 12 | ConvTranspose2d + ReLU + BatchNorm | 128 | 4/2/1 |
| 13 | ConvTranspose2d + ReLU + BatchNorm | 64 | 4/2/1 |
| 14 | ConvTranspose2d + Tanh | 3 | 4/2/1 |

**Table 6-2 The Parameter of the Discriminator Architecture**

| No. | Names of the Layers | Number of Convolutional Layer Filters | Convolutional Layer Filter Size/Stride/Padding |
|---|---|---|---|
| 1 | Conv2d + LeakyReLU | 64 | 4 / 2 / 1 |
| 2 | Conv2d + BatchNorm + LeakyReLU | 128 | 4 / 2 / 1 |
| 3 | Conv2d + BatchNorm + LeakyReLU | 256 | 4 / 2 / 1 |
| 4 | Conv2d + BatchNorm + LeakyReLU | 512 | 4 / 1 / 1 |
| 5 | Conv2d + Sigmoid + Flatten | 1 | 4 / 1 / 1 |

Training a GAN network is computationally intensive and requires extensive training time and resources. For this, a RTX A6000 GPU was used, and the network was implemented using PyTorch(Paszke et al., 2019). The training process was carried out on NVIDIA CUDA Centre (NCC) GPU system. The code for this experiment is available on GitHub at https://github.com/Dehakaa/sGAN, allowing for reproducibility of the results.

The objective is to generate a corresponding synthetic flow visualization image based on a given flow field map. To ensure high fidelity and minimal artifacts, the GAN model should effectively learn the mapping between flow field data and visualization images.

The loss function is less critical for GAN-based neural networks, mainly because the GAN architecture exploits an adversarial process between the generator (G) and discriminator (D) to dynamically improve output quality (Goodfellow et al., 2014). The discriminator, due to its robust ability to differentiate between real and synthetic data, provides relatively accurate feedback, guiding the generator more effectively.

Unlike in supervised learning models, where validation loss directly reflects the model's generalization capability, GANs do not exhibit a straightforward validation loss. This is because the discriminator is trained to fit the current state of the generator, which may lead to overfitting. Therefore, the loss values are not reliable indicators of the model's ability to generalize to unseen data.

Under these conditions, although the generator's loss (G loss) may appear suboptimal, the model is still able to achieve high-quality results, as demonstrated in Figure 6.7. In the figure, the D loss refers to the discriminator's loss, which indicates how well the discriminator is distinguishing between real and generated data.

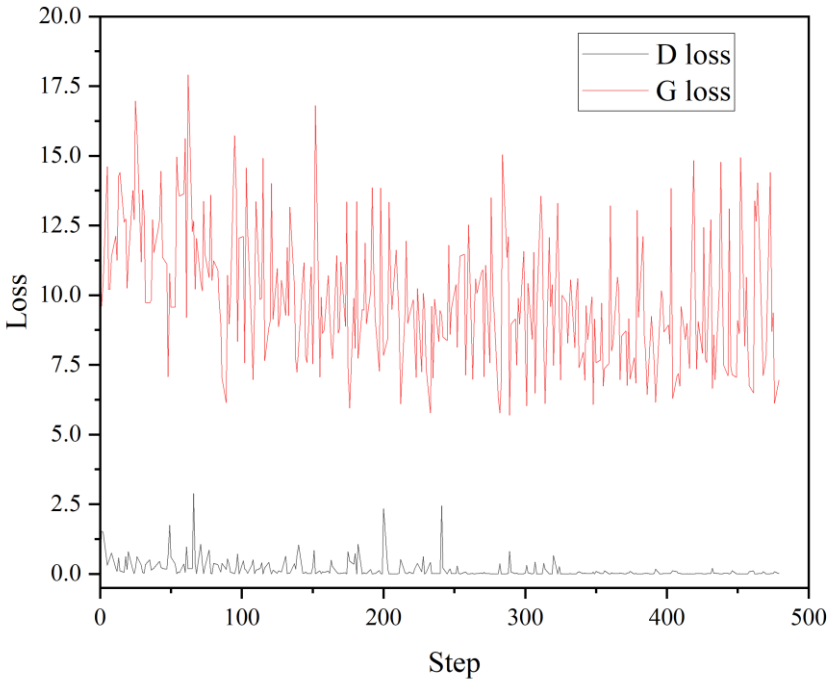



**Figure 6.7 GAN Training Results for Synthetic Flow Visualization Generation**

### 6.2.2 Synthetic Image Results

This section presents extensive experimental results to validate the effectiveness and superiority of the proposed method.

Figure 6.8 illustrates the sGAN results using experimental images from the Durham cascade. The left image shows the input data, which were obtained from streaks generated through CNN-based edge detection applied to flow visualization images, as described earlier, the middle image displays the real experimental images, and the right image presents a flow visualization prediction generated by the sGAN. While Mean Squared Error (MSE) is commonly used to quantitatively assess predictive performance, it is not particularly suitable for the present task. The task at hand is essentially style transfer, where even under identical

experimental conditions, two flow visualization images will not be exactly the same. However, from a visual perspective, realistic synthetic "experimental images" can be generated convincingly, especially when considering streaks in the images.

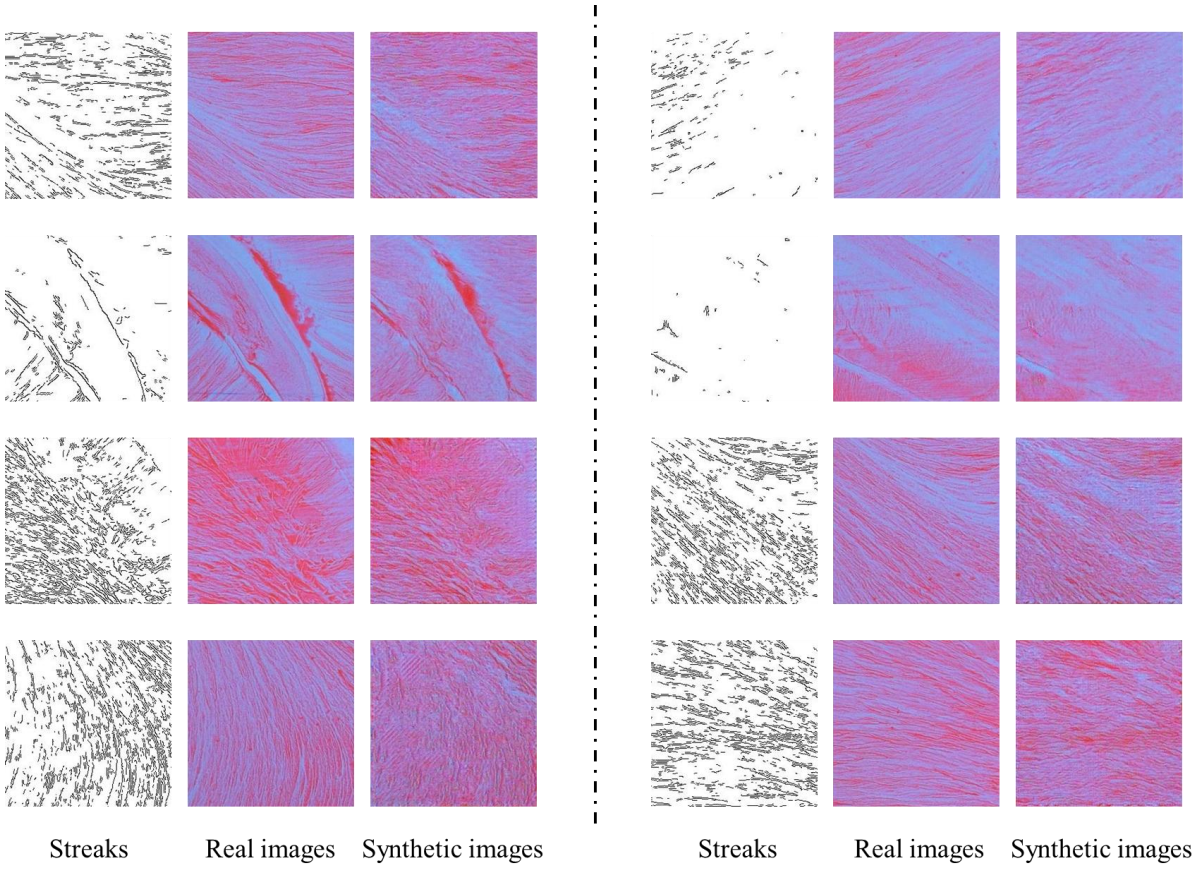



**Figure 6.8 sGAN Results for Flow Visualization Synthetic images**

Quantitatively comparing the quality of generated images has long posed a challenge in establishing suitable evaluation standards. For GAN architectures, the most widely accepted evaluation metrics are the Inception Score (IS) and the Fréchet Inception Distance (FID) scores.

The IS method leverages a pre-trained InceptionV3 model to assess the quality and diversity of generated images. The InceptionV3 model, trained on the ImageNet database, which contains over a million images, achieves 78.8% accuracy in classifying images into 1,000 categories. However, IS does not evaluate the similarity between generated images and real images. Therefore, in this work, the FID method was introduced to assess image quality by comparing the distribution of generated images with the distribution of real images used to train the generator.

The core idea of FID is to map both the generated images and real images into a feature space and then compare their distributions in this space. The specific steps are as follows:

1. **Feature Extraction**: Use the InceptionV3 model (typically using the feature vector from its final layer) to extract features from both the generated and real images.
2. **Feature Distribution Calculation**: For both the generated and real image sets, the extracted feature vectors form a multi-dimensional normal distribution, which can be characterized by a mean vector (μ) and a covariance matrix (Σ).
3. **Fréchet Distance Calculation**: The distribution of generated images and real images is then substituted into the Fréchet Distance formula:

$$\mathrm{FID} = \left\|\mu_{real} - \mu_{gen}\right\|^2 + \mathrm{Tr}\left(C_x + C_y - 2\left(C_x C_y\right)^{1/2}\right) \tag{6.4}$$

Here, Tr represents the trace of a matrix, which is the sum of all diagonal elements. $\mu_{real}$ and $\mu_{gen}$ are the mean activation values generated from the final average pooling layer for real and generated images, respectively. Cx and Cy are the sample covariance matrices of these activation values. The trace is determined using the following formula:

$$\mathrm{Tr}\left(\left(C_x C_y\right)^{1/2}\right) = \sum_{i=1}^{m} \sigma_i\left(C_x^T C_y C_y^T C_x\right) \tag{6.5}$$

In this expression, $\sigma_i$ represents the singular values of CxCy, as the eigenvalues and singular values are identical in the context of covariance matrices.

The trained GAN was used to generate the required synthetic images, and their FID values are determined and summarized in

Table 6-3. Initially, the GAN's FID score on the experimental dataset decreased gradually from 24.5 to 14.8, indicating that the model progressively learned the characteristics of the data distribution, thereby significantly improving the quality of generated images. Notably, the model reached its optimal state at 150 epochs, demonstrating the effectiveness of the training process, showing the corresponding changes in the GAN model as training epochs increased.

**Table 6-3 Training performance of GANs**

| Model Name | Number of Generated Images | Number of Real Images | FID Score | Training Epochs | Notes |
|---|---|---|---|---|---|
| GAN | 500 | 500 | 24.5 | 50 | Initial training |
| GAN | 500 | 500 | 16.7 | 100 | Mid-term training |
| GAN | 500 | 500 | 14.8 | 150 | Optimal model |

### 6.2.3 Comparison of generation performance with respect to different models

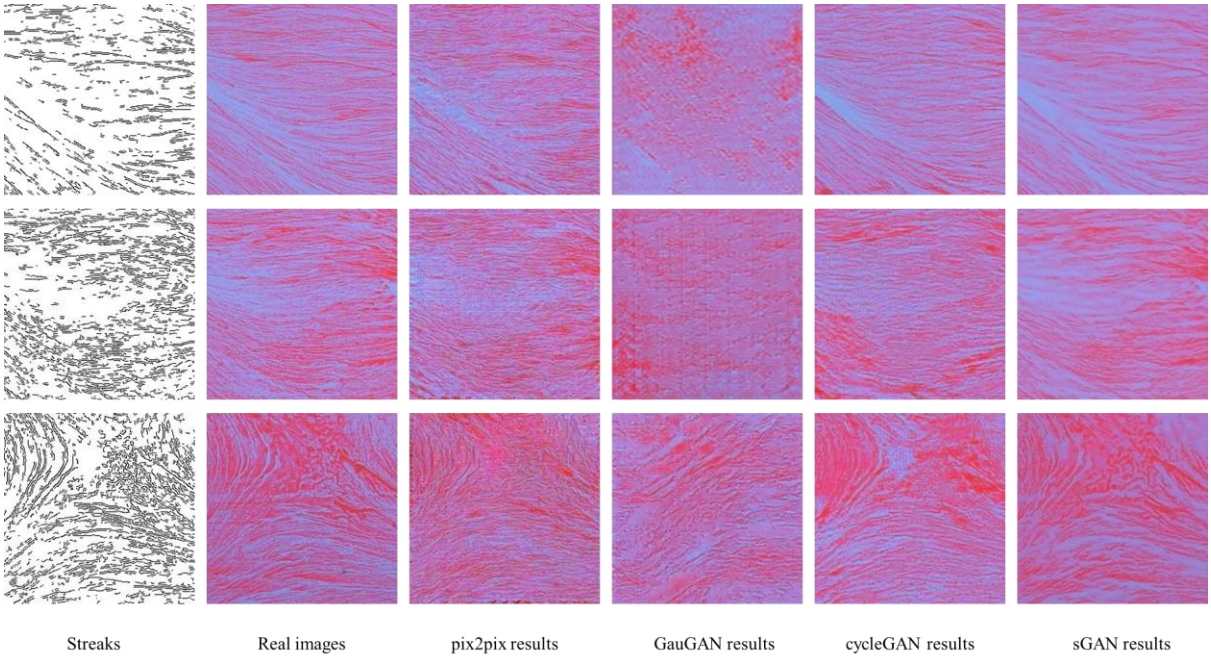



**Figure 6.9 Comparison of Synthetic Images Performance with Different GAN Models**

From the result images in Figure 6.9, it can be seen that the surface flow visualization images generated by sGAN and CycleGAN are almost identical to the real images. However, when it comes to GauGAN, which focuses solely on style transfer, it fails to generate convincing images. It only produces streaks with corresponding colors near the streak regions, while other areas exhibit lower resolution. This is likely because GauGAN is primarily designed for generating the overall style of an image rather than precise detail accuracy. As a result, its generative capacity is limited when dealing with tasks that require fine structural

details or physical features. Specifically, GauGAN focuses on converting sketches or semantic segmentation maps into high-quality images but may overlook the accurate reproduction of complex physical phenomena or details. Studies suggest that style transfer models like GauGAN often perform poorly in detail reconstruction, especially when fine structures or high-frequency information are involved, leading to a loss of resolution or distortion in the generated images.(Gui et al., 2023)

By comparing the generated results with the real flow visualization images and calculating the differences using formula

$$\mathcal{L}_{L_1}(G) = E_{x,y}[\|x - G(y)\|_l]$$

, previously introduced as Eq.(3.10), the absolute error map of the generated images is shown in Figure 6.10. In the map, lighter colours indicate larger errors, while darker (blue) colours represent smaller errors. The flow field images generated by the pix2pix model are structurally close to the real flow fields but exhibit some deviations in detail and texture. The absolute error map reveals that pix2pix shows a higher concentration of warm colours in certain areas (particularly in the streak regions), indicating lower prediction accuracy in key areas. Pix2pix mainly relies on pixel-wise loss computation, which enables it to generate relatively sharp edges but causes detail loss in complex textured regions. The images generated by GauGAN show relatively small errors in the streak regions, with a broader distribution of cooler colours. However, white noise points are evenly spread across the entire image, suggesting that the model may need improvement in generating local consistency in fine details. CycleGAN performs well in predicting the overall structure of the flow field but shows reduced accuracy in regions with large directional changes (as shown in the bottom row of Figure 6.10, where CycleGAN's predictions exhibit large white areas at points of curvature). The sGAN model demonstrates a more balanced performance in generating both overall structure and key regions. Compared to the other models, sGAN maintains global consistency while also capturing finer details more effectively. This is likely due to sGAN's focus on improving the accuracy of image content reconstruction, allowing it to achieve a better balance between structural integrity and detailed texture accuracy in the generated results.

To investigate the efficiency of generating "effective information" in synthetic images, this study also performs a Peak Signal-to-Noise Ratio (PSNR) analysis for each method. The peak signal-to-noise ratio is a metric that quantifies the ratio between the power of the signal and the power of the noise. PSNR is defined through the Mean Squared Error (MSE). For two

$m \times n$ grayscale images $I$ and $K$, where one is the ground truth and the other is the predicted data, the MSE is defined as:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \tag{6.6}$$

The PSNR value is then determined by the following equation:

$$PSNR = 10 \cdot \log_{10} \left( \frac{MAX_I^2}{MSE} \right) \tag{6.7}$$

here $MAX_I$ represents the maximum possible pixel value of the image. If each sample point is represented using 8 bits, then $MAX_I = 255$. higher PSNR values indicates less distortion and better preservation of the original image details. This is performed by reducing the noise in the input image. (Sethi et al., 2022)

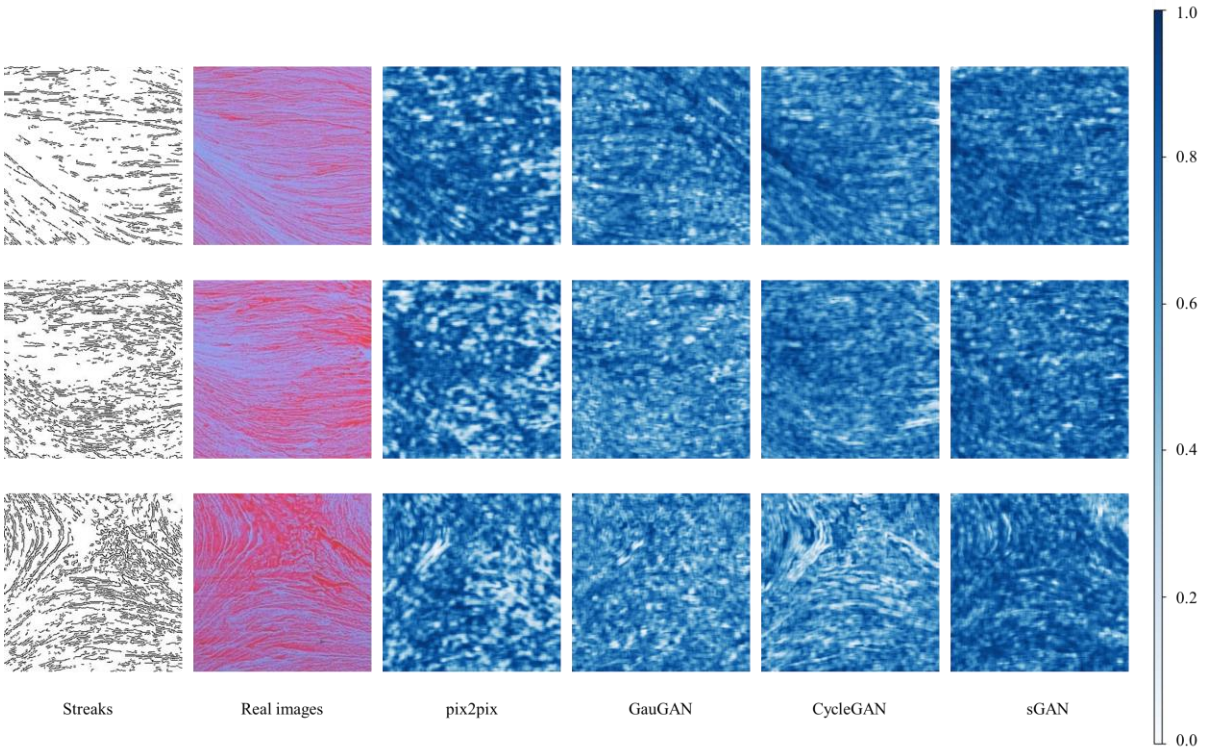



**Figure 6.10 Error Cloud Pictures of Synthetic Images Performance from Different GAN Models**

Table 6.4 lists PSNR and SSIM for four different generative adversarial networks (GANs)—Pix2pix, GauGAN, CycleGAN, and sGAN—evaluated on three different real

images. The evaluation metrics include SSIM, MSE, and PSNR, which provide a comprehensive measure of image quality in terms of similarity to the real image.

**Table 6-4 PSNR and SSIM Comparison of Different GAN Models**

| Real image | Network Model | SSIM | MSE | PSNR (dB) |
|---|---|---|---|---|
| 0 | Pix2pix | 0.6017 | 0.0037 | 24.31 |
| | GauGAN | 0.6194 | 0.0037 | 24.30 |
| | CycleGAN | 0.6434 | 0.0029 | 25.31 |
| | sGAN | 0.8686 | 0.0008 | 30.79 |
| 1 | Pix2pix | 0.4617 | 0.0056 | 22.50 |
| | GauGAN | 0.5452 | 0.0063 | 21.99 |
| | CycleGAN | 0.5217 | 0.0051 | 22.93 |
| | sGAN | 0.8503 | 0.0011 | 29.67 |
| 2 | Pix2pix | 0.4439 | 0.0060 | 22.20 |
| | GauGAN | 0.4742 | 0.0068 | 21.69 |
| | CycleGAN | 0.4294 | 0.0077 | 21.13 |
| | sGAN | 0.8540 | 0.0012 | 29.36 |

For Image 0, sGAN achieved the highest SSIM value of 0.8686, significantly outperforming the other models. This indicates that sGAN generates images most similar to the real image in terms of structural quality, with the highest PSNR value of 30.79 dB and the lowest MSE of 0.0008. In contrast, Pix2pix, GauGAN, and CycleGAN had lower SSIM scores, with Pix2pix showing the lowest PSNR (24.31 dB) and the highest MSE (0.0037).

For Image 1, sGAN again led with an SSIM of 0.8503, a PSNR of 29.67 dB, and an MSE of 0.0011, showcasing its ability to preserve image quality across different inputs. On the other hand, Pix2pix produced the lowest SSIM of 0.4617 and a PSNR of 22.50 dB, indicating a relatively poor reconstruction of the real image. GauGAN and CycleGAN perform moderately, with CycleGAN having a slightly better SSIM and PSNR than GauGAN.

In Image 2, sGAN continued to outperform the other models, achieving an SSIM of 0.8540, a PSNR of 29.36 dB, and an MSE of 0.0012. The other models, particularly Pix2pix and CycleGAN, had the lowest performance with SSIM values of 0.4439 and 0.4294, respectively. Pix2pix's PSNR is the lowest at 22.20 dB, further highlighting sGAN’s superior image quality in this case.

Overall, sGAN consistently produces the highest SSIM, PSNR, and lowest MSE across all three images, demonstrating its effectiveness in generating high-quality images with better structural and visual fidelity compared to Pix2pix, GauGAN, and CycleGAN.

## 6.3 Test data generation

This section describes the data generation process used throughout the remainder of this chapter, primarily focusing on simple flow fields, such as the flow around a cylindrical body with a random base shape, and the processing of GAN-generated images. The goal of this data generation is to prepare training materials for the U-Net model, as illustrated in Figure 6.1. This data will be used for pre-training in Section 6.4. The procedure was as followed.

Arbitrary two-dimensional shapes were generated using B-spline curves, where control points were randomly sampled within a specified range. These shapes simulate simplified flow obstacles. Each shape was then meshed for simulating 3D laminar flow fields using OpenFOAM. The Navier-Stokes equations were solved under steady-state conditions with uniform inlet velocity and zero-gradient outlet conditions. After convergence, the resulting velocity components were extracted and normalized. These flow fields serve as the input to the U-Net network. Corresponding synthetic visualization images were generated from the simulation results. Each sample thus consisted of a paired set: [visualization image (as input), flow field (as output)]. The data was resized to 256×256 pixels to ensure consistency and efficiency during training. To improve the generalization capability, additional flow field data from the Durham cascade experiments Martinez-Castro (2022) was included and processed similarly. This pre-training dataset enabled the U-Net model to first learn basic flow structures and visualization mappings, before later fine-tuning with experimental datasets in Section 6.4.

### 6.3.1 Generation and Preprocessing of Simple Flow Field Data

#### 6.3.1.1 Mesh Generation for Simple Geometries

The first step in generating random shapes was to plot *n* random points within the range $[0,1]^2$, and then translated them so that their centroid could be located at (0, 0). These points were then sorted in ascending order based on their polar angles (see Fig 6.11a).

Unlike the work by Chen et al. (2019a), which generated random shapes using Bezier curves, this work employed B-spline curves to generate random shapes. Specifically, the `scipy.interpolate.splprep` and `scipy.interpolate.splev` functions in Python 3.8 were used.

In the process of constructing the B-spline curve, a set of discrete points $p_i$ was first defined as the basis for generating the curve. To ensure smooth interpolation of these points, the points were first sorted by their polar angles. Then, the point set is parameterized using the `splprep` function to create a smooth B-spline curve.

To control the smoothness of the curve, a parameter *s* was introduced. The range of *s* is [0, ∞), and in this thesis, $s = 0.5$. When *s* is close to 0, the resulting curve tightly fits the input discrete points, capturing sharp features of the point set. When *s* takes on a larger value, the curve becomes smoother and deviates more from the original point set, resulting in a softer shape. Therefore, by adjusting *s*, the global smoothness of the curve could be controlled.

The shape of the B-spline curve is primarily defined by the combination of knots and control points. The shape of each curve segment is determined by its control point set, which is similar to the intermediate control points in Bezier curves, influencing the tangent direction and the curvature of the segment, as shown in Figure 6.11b. Unlike Bezier curves, B-splines do not pass directly through the control points at the start and end of each segment. Instead, the curve is interpolated between these control points based on the knot sequence. This method ensures that the shape of each segment is defined by a set of adjacent control points, which maintains continuity and smoothness between segments.

During the spline construction process, the curve's degree can be further adjusted using the parameter *k*. By setting $k = 3$, a cubic B-spline curve is generated, and the curve has $C^2$-smoothness at the connection points between segments. This means that the curve transitions smoothly at the knots without noticeable angular discontinuities. This property is well-suited for most smooth connection requirements, and produces visually pleasing curves.

In terms of tangent control, B-splines differ from Bezier curves in that they do not explicitly set the tangent directions. Instead, the density of the knots controls the tangent shape in different regions of the curve. A denser knot sequence leads to sharper bends in the curve, while a sparser knot sequence makes the curve appear more linear. Although tangent control is not as precise as in Bezier curves, B-splines offer good smoothness and can generate fluid curves.

To create a closed spline, the `splev` function is used to sample the curve, obtaining a smooth set of points that form a closed loop. This loop can then be used as the outline of a geometric shape or as smooth mesh lines for further rendering and visualization. This method,

similar to Bezier curves, achieves smooth curves through global smoothness control and the influence of local control points.

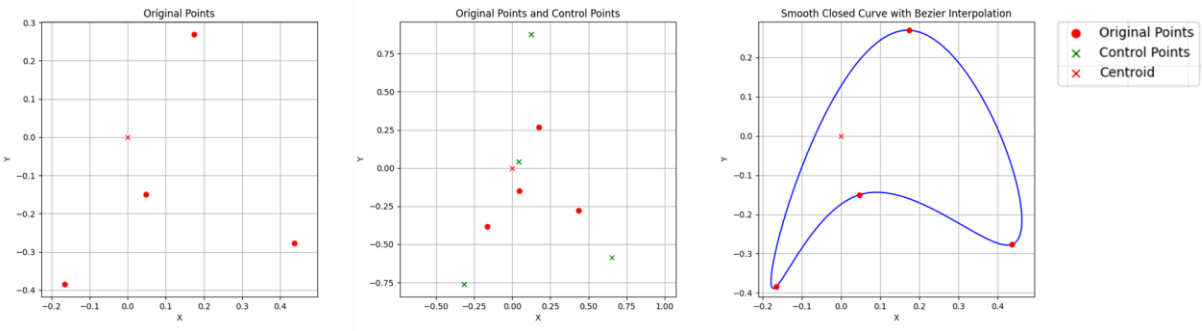



**Figure 6.11 Random Shape Generation with B-spline Curves. (a) Random original points generation; (b) Control points generation; (c) Related Bezier curves.**

By specifying different values of *n*, a variety of shapes can be generated, as shown in Figure 6.12. Although most shapes are valid, some unrealistic shapes occasionally occurred with low probability and were manually removed. Additionally, since the goal of this section is to model near-surface flow conditions, the distance between the generated shape and the surface, known as the "tip clearance," must be specified.

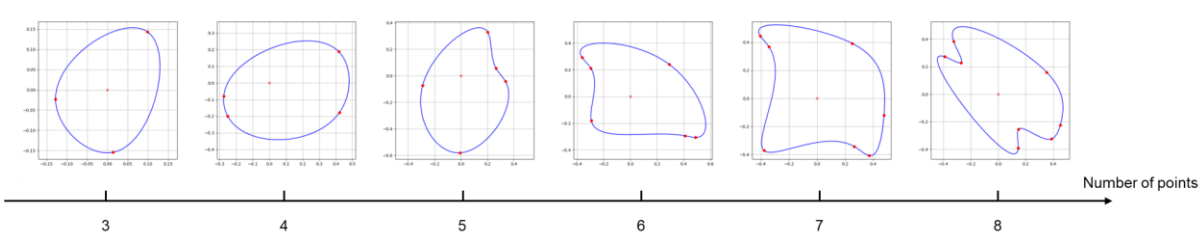



**Figure 6.12 Random Shape Generation Results of Different n Number**

#### 6.3.1.1 CFD Simulation Setup

In this section, CFD was used to generate and process simple flow field data for training a U-Net model. OpenFOAM was selected for this task due to its suitability for batch processing and its open-source flexibility in large-scale simulations. The goal was to simulate various flow conditions around arbitrary shapes generated by B-spline curves, which would later be used for training a deep learning model. A non-structured mesh was created for the fluid domain using FEATool's mesh generation tools. The fluid domain was divided into two regions: one close to the cylinder wall and the other farther from it. In the region near the cylinder, where flow gradients are expected to be steep and detailed resolution is required, the mesh elements were

set to a very fine size of 0.0002. The top view of the generated mesh is shown in the Figure 6.13. This fine grid allowed for the accurate capture of boundary layer effects, vortex shedding, and other complex flow features typically observed in such proximity to solid surfaces. Conversely, in the outer region of the fluid domain, where flow gradients are expected to be less pronounced, larger mesh elements of size 0.002 were employed. This helped reduce computational costs while still ensuring an adequate resolution for capturing global flow behaviour. This refinement captured the detailed flow behaviour near the cylinder while reducing computational costs in the outer region. The average number of cells was approximately 200,000, providing a balance between resolution and computational efficiency.

Boundary conditions were set with no-slip conditions at the cylinder surface and far-field conditions at the outer boundaries. The inlet and outlet boundaries were defined to ensure realistic flow behaviour. A uniform inlet velocity of 19 m/s ($u = 19$ m/s, $v = w = 0$) was applied at the inlet. The outlet had a homogeneous Dirichlet condition for the pressure field ($p = 0$). The top and side boundaries of the domain were assigned free-slip conditions.

The simulations were conducted using OpenFOAM v2212 with the simpleFoam solver, which is suitable for steady-state, incompressible flow simulations. The k-ε turbulence model was used to simulate the turbulent flow behaviour. A convergence criterion of $10^{-5}$ for residuals was set, with the pressure and velocity fields monitored during the simulation to ensure stability.

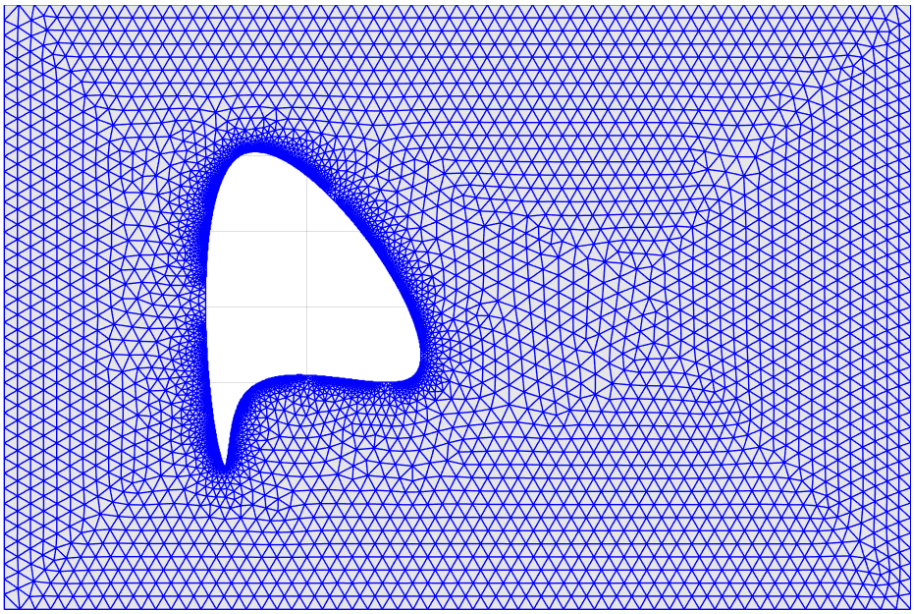


**Figure 6.13Top View of the Unstructured Mesh for CFD Simulation**

#### 6.3.1.2 Results and Dataset Generation

The dataset consists of 5,000 shapes along with their steady-state velocity and pressure fields computed at a Reynolds number of 3000. This Reynolds number was selected to ensure turbulent flow characteristics while maintaining numerical stability, and was calculated based on a characteristic length scale defined by the characteristic diameter of cylinder (L = 0.02 m), and a fixed inlet velocity (U = 19 m/s). Figure 6.14 illustrates the pressure field corresponding to the shape shown in Figure 6.13. Naturally, these results cannot be directly used for model training; they require preprocessing as described in Section 6.3.3. The dataset was systematically divided into three parts: 80% for the training set, 10% for the validation set, and 10% for the test set.

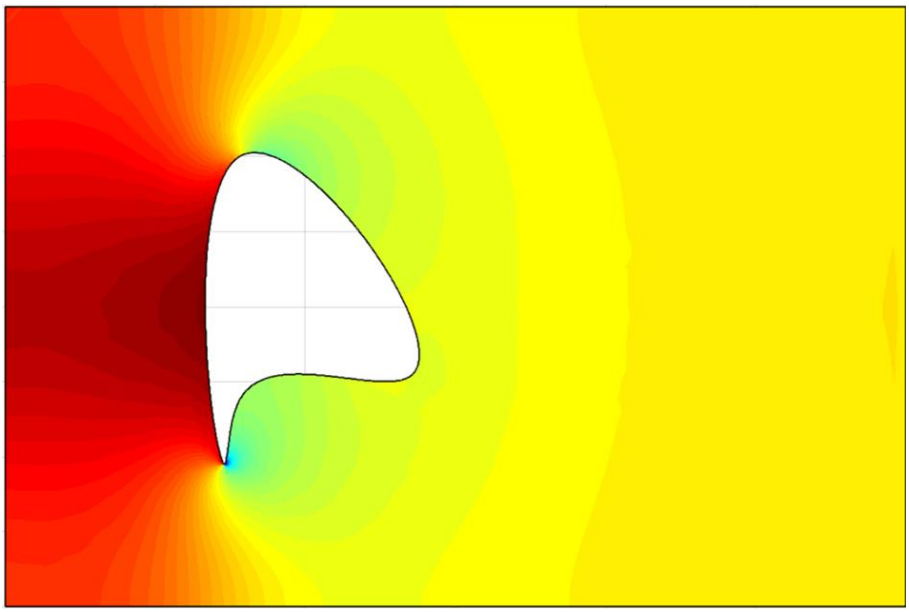


**Figure 6.14 Pressure field of a sample shape shown in Figure 6.13**

### 6.3.2 Durham Cascade Flow Field Simulation Based on Specific Tip Clearance

The analytical approach and model selection in this section are informed by the work of Martinez-Castro (2022), utilizing the Transition SST turbulence model to solve the RANS equations. This model, based on the $k$-$\varepsilon$ turbulence framework, was initially proposed by Wilcox (1988) and primarily addresses the transport equations for $k$ (turbulent kinetic energy) and $\varepsilon$ (dissipation rate). The Transition SST model incorporates modifications to account for the effects of primary turbulent shear stresses, using a limiter function to adjust the eddy viscosity (Menter, 1992). Additionally, this model solves transport equations for intermittency

($\gamma$) and the Reynolds number ($Re_\theta$) and is thus also referred to as the ($\gamma$ - $Re_\theta$) turbulence model. After post-processing, these outputs would also be used as training data/for validation purposes.

### 6.3.3 GAN Synthetic Image Generation Process

In both Sections 6.3.1 and 6.3.2, the resulting data is structured in a "coordinate-data" format, where the data may include u-velocity, v-velocity, pressure fields, or other flow field information. Typically, a mesh contains numerous nodes; however, using such high-resolution data directly in machine learning would drastically increase the number of training parameters, leading to an excessive computational cost. To address this, the study resamples the data to a 256 × 256 Cartesian grid, aligning with the input image size.

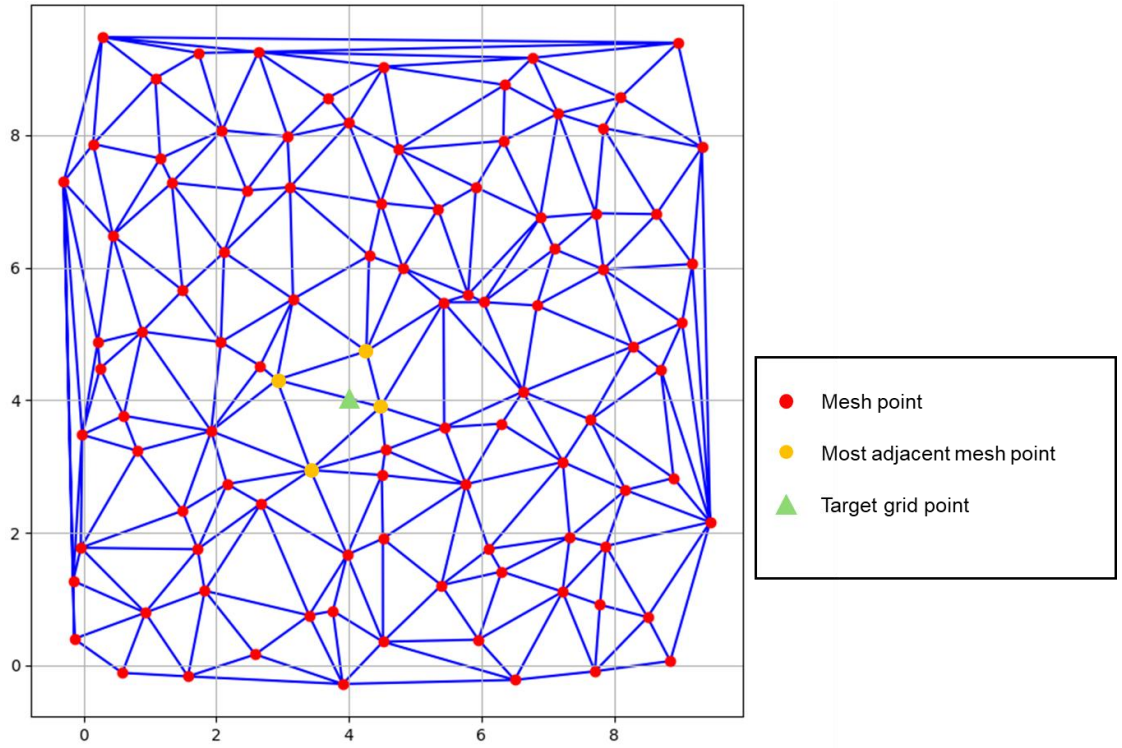



**Figure 6.15 Mesh Data Projection onto Cartesian Grid**

The first step is to project the mesh data onto a 256 × 256 Cartesian grid while preserving accuracy as much as possible. The method is illustrated in the Figure 6.15. For instance, to obtain flow field data at a target coordinate, consider the point $\boldsymbol{T}$ at (4,4) (shown as a green triangle in the Figure 6.15). The algorithm identifies the four nearest points $A_1$, $A_2$, $A_3$ and $A_4$ (marked in orange), with coordinates ($x_1$, $y_1$), ($x_2$, $y_2$), ($x_3$, $y_3$) and ($x_4$, $y_4$) and corresponding flow field values $F_1$, $F_2$, $F_3$ and $F_4$. The flow field value $\boldsymbol{F_T}$ at $\boldsymbol{T}$ is then calculated as a distance-weighted average of these four points, as follows:

$$F_T = \frac{\sum_{i=1}^{4} \frac{F_i}{d_i}}{\sum_{i=1}^{4} \frac{1}{d_i}} \tag{6.8}$$

where

$$d_i = \sqrt{(x_T - x_i)^2 + (y_T - y_i)^2} \tag{6.9}$$

This equation represents the Euclidean distance from point $\boldsymbol{T}$ to point $A_i$. The numerator of Eq.(6.9) calculates the weighted sum of the flow field values $F_i$ of each neighbouring point, with the weights being the inverse of their respective distances $d_i$; the closer points contribute more to $F_T$. The denominator normalizes the result, ensuring that the weighted average remains within a reasonable range. This process can be applied iteratively to obtain the coordinates of all points on the grid.

## 6.4 Training U-net to Predict Flow Field

The U-Net architecture is described in detail in the methods section; here, a brief overview of its working principle is provided. U-Net was first proposed by Ronneberger et al. (2015) as an improvement to the traditional sliding window approach used in convolutional neural networks (CNNs)(Ciresan et al., 2012). In conventional CNNs, a contracting path, similar to the first branch of U-Net, is used to extract high-level features from the input image. However, as the features become more abstract, the localization information is often lost, making it difficult for the network to directly generate meaningful reconstructed images. Walker et al. (2015) proposed a CNN-based method that predicts the future motion of every pixel as optical flow from a single static image without any human supervision, achieving state-of-the-art performance across diverse scenes. The core idea of U-Net is to follow the contracting path with an expanding (or upscaling) path. This path uses deconvolution layers to progressively restore the resolution of the feature maps, with the restoration process being symmetric to the contracting path. After each deconvolution operation, the expanding path reintroduces localization information through shortcut connections with the corresponding layers in the contracting path. In this process, the convolution operations in the expanding path concatenate features from the previous expanding layer with the localization information passed through the shortcut connections, effectively merging the information. A schematic

overview of the U-Net input and output structure is briefly illustrated in Figure 3.9, with further architectural details already provided in section 3.5.2.

As previously mentioned, pretraining is designed to train the network's ability to identify edges and directions. However, since synthetic images are based on simplified assumptions, real-world scenarios are often more complex, requiring additional fine-tuning with real experimental images and annotations.

The learning process in neural networks involves adjusting the network's biases and weights to minimize the value of a carefully chosen loss function. In regression problems, the most commonly used loss functions are L1 loss function and L2 loss function, which are also referred to as Mean Absolute Error (MAE) and Mean Squared Error (MSE), respectively. Once a loss function is selected, the network optimization is carried out through backpropagation and Stochastic Gradient Descent (SGD), with gradient updates relying on the partial derivatives of the loss function with respect to the network's weights.

The formula for MSE is:

$$\varepsilon_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{6.10}$$

where $y_i$ is the true value of the $i$-th sample, $\hat{y}_i$ is the predicted value, and $n$ is the number of samples. MSE amplifies the impact of large errors by squaring each error term and averaging. When using MSE as the loss function, the gradient update rule is:

$$\frac{\partial \varepsilon_{MSE}}{\partial w} = \frac{2}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)\cdot\frac{\partial \hat{y}_i}{\partial w} \tag{6.11}$$

Because of the squared error term, larger errors contribute more to the gradient, which leads to the model prioritizing adjustments to larger errors. Although MSE allows for faster reduction of large errors during the early stages of training, it can also lead to overfitting due to its sensitivity to outliers.

In contrast, the formula for MAE is:

$$\varepsilon_{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{6.12}$$

where $|y_i - \hat{y}_i|$ represents the absolute error of the $i$-th sample. MAE penalizes each error term linearly, unlike MSE, which amplifies the effect of larger errors. When using MAE as the loss function, the gradient update rule becomes:

$$\frac{\partial \varepsilon_{MAE}}{\partial w} = \frac{1}{n}\sum_{i=1}^{n} \text{sign}(y_i - \hat{y}_i) \cdot \frac{\partial \hat{y}_i}{\partial w} \tag{6.13}$$

where $\text{sign}(y_i - \hat{y}_i)$ indicates the sign of the error (positive or negative), and the gradient direction matches the sign of the error. Since MAE does not involve squaring the error term, each error contributes equally to the gradient update, which results in a more gradual gradient descent process, especially minimizing the effect of outliers.

Once the loss function is selected, backpropagation and SGD are used to optimize the weights and biases of the network. Gradient descent is typically performed on small random subsets of data, known as mini-batches, rather than calculating the gradient on the entire dataset at once. This method reduces computational cost and prevents excessive memory consumption. In this study, the effect of different mini-batch sizes (1, 8, 16, 32) on the training process was investigated. The dataset is divided into multiple mini-batches, which are used multiple times in random order. One complete pass through the entire dataset is called an "epoch." This process repeats until the validation accuracy (i.e., the prediction accuracy on the validation subset) no longer improves, and the training stops when accuracy starts to decrease, indicating the network has begun overfitting by learning non-generalizable features from the training set.

In practice, the differences between L1 loss function and L2 loss function can also affect the speed and stability of convergence. L2 loss function tends to penalize large errors more heavily at the beginning of training, which typically leads to faster convergence and quicker correction of large errors. However, its high sensitivity to outliers can cause the model to overfit noise or rare samples. On the other hand, L1 loss function has less sensitivity to outliers, as it applies a linear penalty to each error. Therefore, L1 loss function leads to a more robust convergence process, especially when there is significant noise in the data, but its convergence is generally slower than L2 loss function.

In this study, input images are resized to a fixed size of 256×256 pixels to control training costs. To ensure fairness in comparisons, all random seeds used for training different networks are initialized with a fixed value at the beginning. PyTorch (Paszke et al., 2019)is

selected for constructing the U-net due to its strong integration capabilities and the ease of use provided by the Python programming language.

In general, a larger batch size can accelerate the training process for each epoch since more data is processed simultaneously, allowing parallel computation to enhance efficiency when supported by hardware. Larger batch sizes also contribute to more stable gradient updates, enabling the model to approach the optimal solution more rapidly. On the other hand, smaller batch sizes offer more frequent gradient updates. While each update involves fewer samples, potentially causing greater fluctuations in the gradients, these fluctuations can sometimes help the model escape local optima and achieve better solutions.

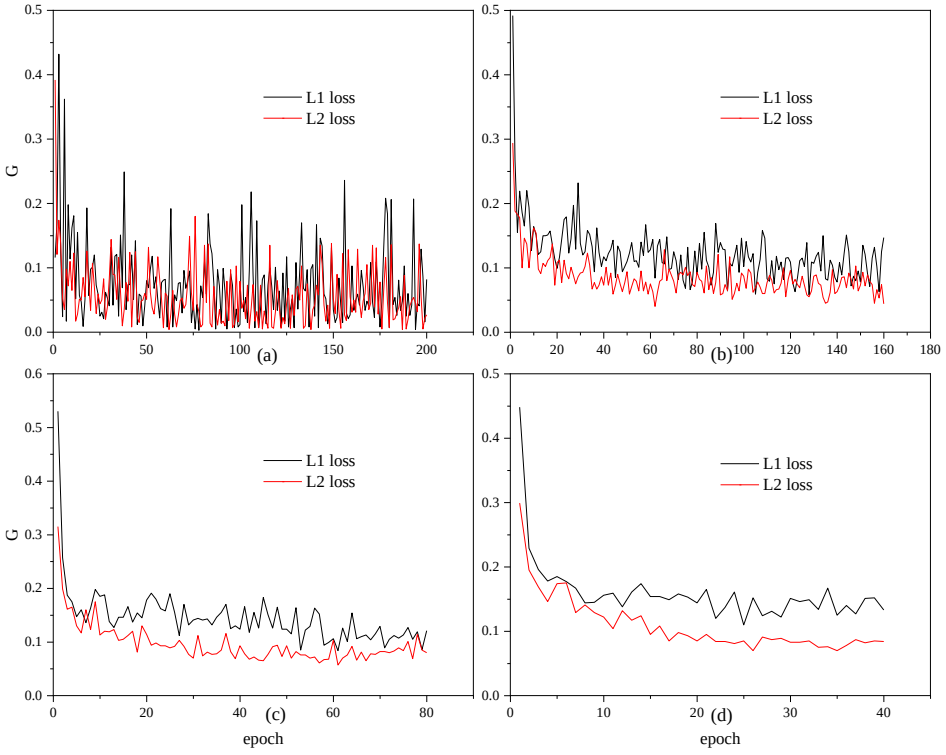



**Figure 6.16 Loss Curves Variation with Epochs for Different Batch Sizes. (a) batch size=1; (b) batch size=8; (c) batch size=16; (d) batch size=32**

Figure 6.16 illustrated the variation trends of the L1 and L2 loss function values on the training set as a function of epochs. Under both L1 and L2 loss functions, the error demonstrates distinct trends over the epochs. For smaller batch sizes (e.g., batch size = 1), the errors for both L1 and L2 losses exhibit significant fluctuations, resulting in more pronounced oscillations in

the error curves. As the batch size increases (from 8 to 32), the error fluctuations gradually decrease, convergence becomes faster, and the curves smooth out, indicating enhanced training stability. Notably, under the L2 loss function, the error curve for batch size = 32 is the smoothest, and the final error is relatively low. On the other hand, the figures also reveal that once the batch size reaches 16, further increases in batch size do not lead to significantly reduced fluctuations in the loss function. Therefore, this study adopts a batch size of 16 for model training. Additionally, the effects of the L1 and L2 loss functions on training results are compared, as shown in Figure 6.18.

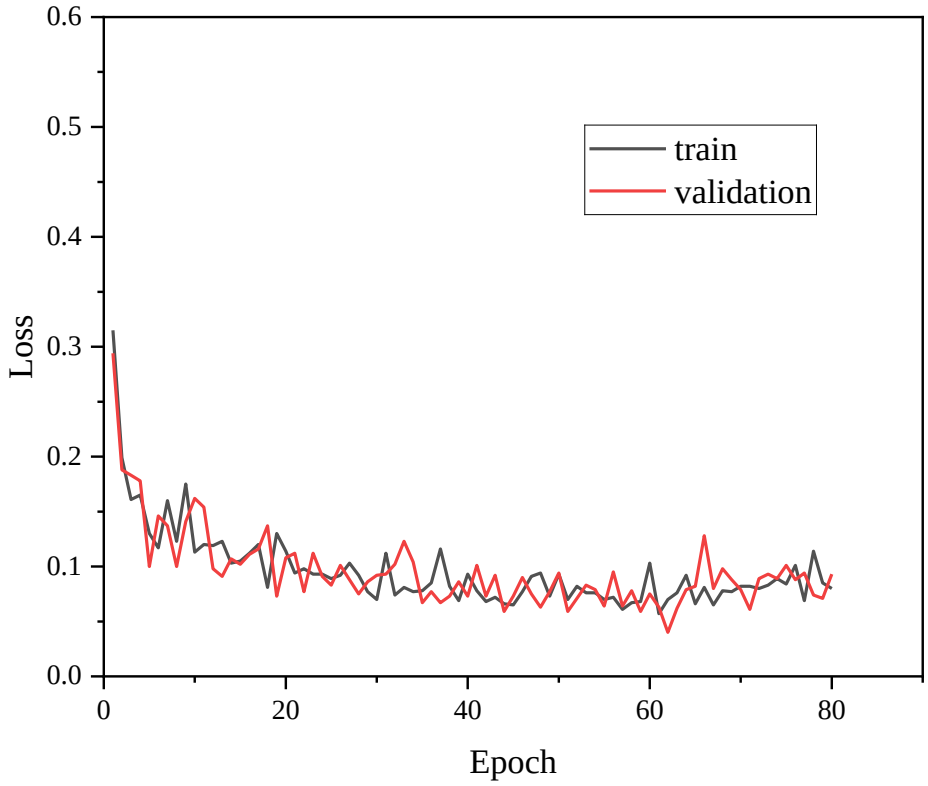



**Figure 6.17 Comparison of Loss Curves for Training and Validation with Batch Size 16, L2 Loss**

The loss curves for training and validation shown in Figure 6.17 correspond to a batch size of 16 and the use of the L2 loss function. At the early stage of training, both the training and validation losses decrease rapidly. Then at training progresses, after approximately 20 epochs, the losses stabilize and fluctuate around a low value (approximately between 0.08 and 0.12), suggesting that the model has gradually converged. Throughout the process, the training

and validation losses remain closely aligned, with no significant indications of overfitting or underfitting. In certain local regions, the validation loss is slightly lower than the training loss, which can be attributed to the application of L2 regularization, which could enhance the generalization ability of U-net. Although slight fluctuations are observed in the curves, they are likely caused by the randomness introduced by the relatively small batch size. Such fluctuations are normal and do not negatively impact the overall training stability.

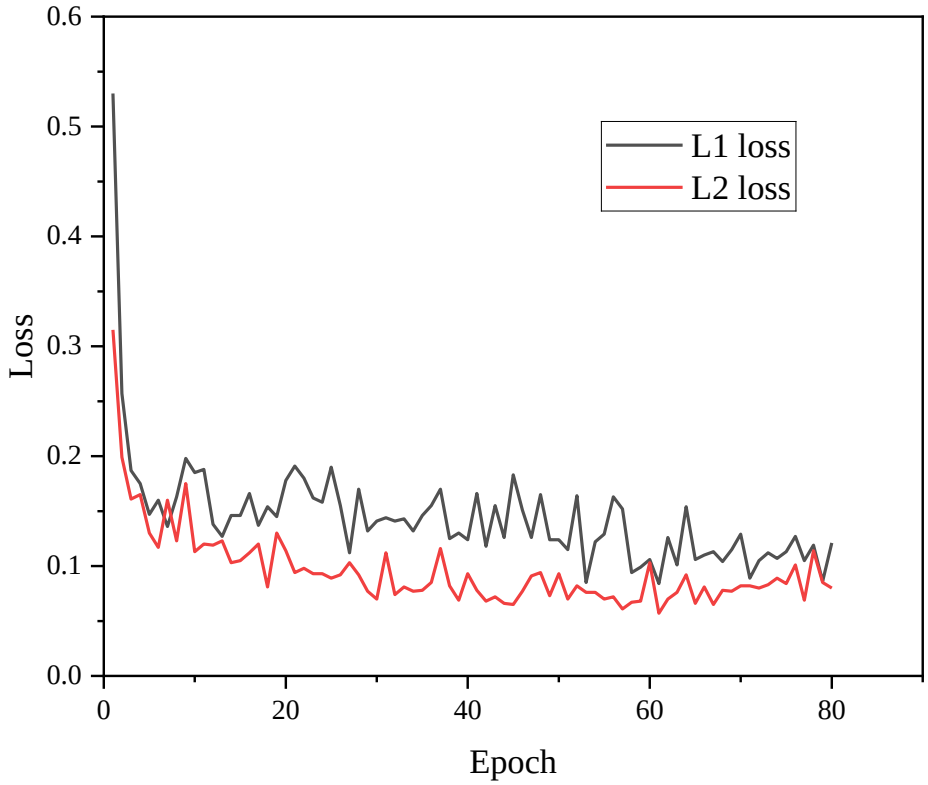



**Figure 6.18 Comparison of L1 and L2 Loss Curves with Batch Size 16**

Figure 6.18 shows the variation in model error over epochs under identical batch sizes when using the L1 and L2 loss functions. During the early stages of training (approximately the first five epochs), the errors for both L1 and L2 losses decrease rapidly. The L2 loss shows a slightly faster rate of decline and converges to a lower error value sooner. This indicates that, in the initial phase, the L2 loss is more effective in rapidly reducing errors. In the stable phase, the error curve for the L2 loss remains consistently lower and exhibits less fluctuation compared to the L1 loss. This aligns with the characteristics of the L2 loss, which imposes stronger penalties on larger error values, leading to a smoother error curve. Therefore, models

trained with the L2 loss tend to achieve a lower average error and the variability of the curve is reduced. The L1 loss curve shows greater fluctuations, likely due to its reduced sensitivity to small error values, which makes it less effective at smoothing out noise during training. In contrast, the L2 loss's squared penalty effect amplifies its sensitivity to errors. While this may theoretically increase the impact of noise, if the batch size is large enough, the impact of noise will be averaged out, resulting in a more stable L2 loss curve. Throughout the training process, the L2 loss is always lower than L1 loss, indicating that, under the same training conditions, the L2 loss is more efficient at reducing errors and enables the model to converge to a lower error value. Overall, under the current conditions, the L2 loss function is better than the L1 loss function, achieving faster convergence and maintaining a lower error level. While the L1 loss demonstrates robustness against extreme noise, it falls short in terms of smoothness and the final convergence of errors compared to the L2 loss.

## 6.5 Demonstration of the Workflow on a Sample Case

To demonstrate the end-to-end workflow of the proposed synthetic data generation and flow field prediction framework, a representative test case was constructed based on the Durham Cascade. This geometry and flow scenario have been extensively used in previous experimental studies and numerical validations, providing a reliable foundation for benchmarking the model's performance.

### 6.5.1 Test Case Setup

The test case adopts the same setup as described in the section 3.1.1, including blade geometry, domain dimensions, and boundary conditions. The corresponding flow visualization image was obtained from the experimental dataset under well-controlled wind tunnel conditions. The flow conditions in this test case correspond to a Reynolds number of approximately $Re = 4 \times 10^5$, based on chord length and inlet velocity, falling within the transitional regime.

**Figure 6.19 Surface flow visualization image of the test case**

This specific configuration has also been subject to high-fidelity numerical validation by Martinez-Castro (2022), who performed detailed CFD simulations of the Durham cascade using a RANS approach with a validated turbulence model. His results serve as a reference for evaluating flow structures such as wake development, blade surface boundary layers, and secondary flow features.

Figure 6.19 shows that the surface flow visualization result of the Durham cascade. This panel was placed on the right side of the cascade as an endwall and could be removed after the experiment. The black lines in the figure indicate the corresponding blade positions during the experiment. Since the tip clearance under the experimental conditions was 3.75 mm, flow patterns within the area enclosed by the black lines can also be observed. A specific region (highlighted by the yellow rectangle in the figure) was selected as the comparison area — this is because not all areas in the photograph represent flow regions, and alignment with the CFD results is necessary.

### 6.5.2 Generation of Reference CFD Results

The CFD simulations were conducted using a RANS approach with a k–ω SST turbulence closure. The computational domain consisted of a linear cascade of turbine blades with periodic boundary conditions applied in the pitchwise direction. A structured mesh was used, with localized refinement near blade surfaces and in the wake region to ensure accurate resolution of boundary layer separation and vortex shedding phenomena. The total cell count was approximately 3 million, and mesh independence was confirmed in the original study.

Figure 6.20 shows the 3D mesh structure of the blade and endwall. The red frame in Figure 6.20 (b) shows the tip clearance details. The first blade is fixed on the endwall, while the second blade has a small gap from the endwall. This reflects the design of the Durham cascade, where only the middle four blades have adjustable tip clearance, and the corresponding section of the endwall is removable. As shown in Figure 6.19, the removable endwall section can be detached after the experiment, whereas the remaining parts of the cascade are fixed.

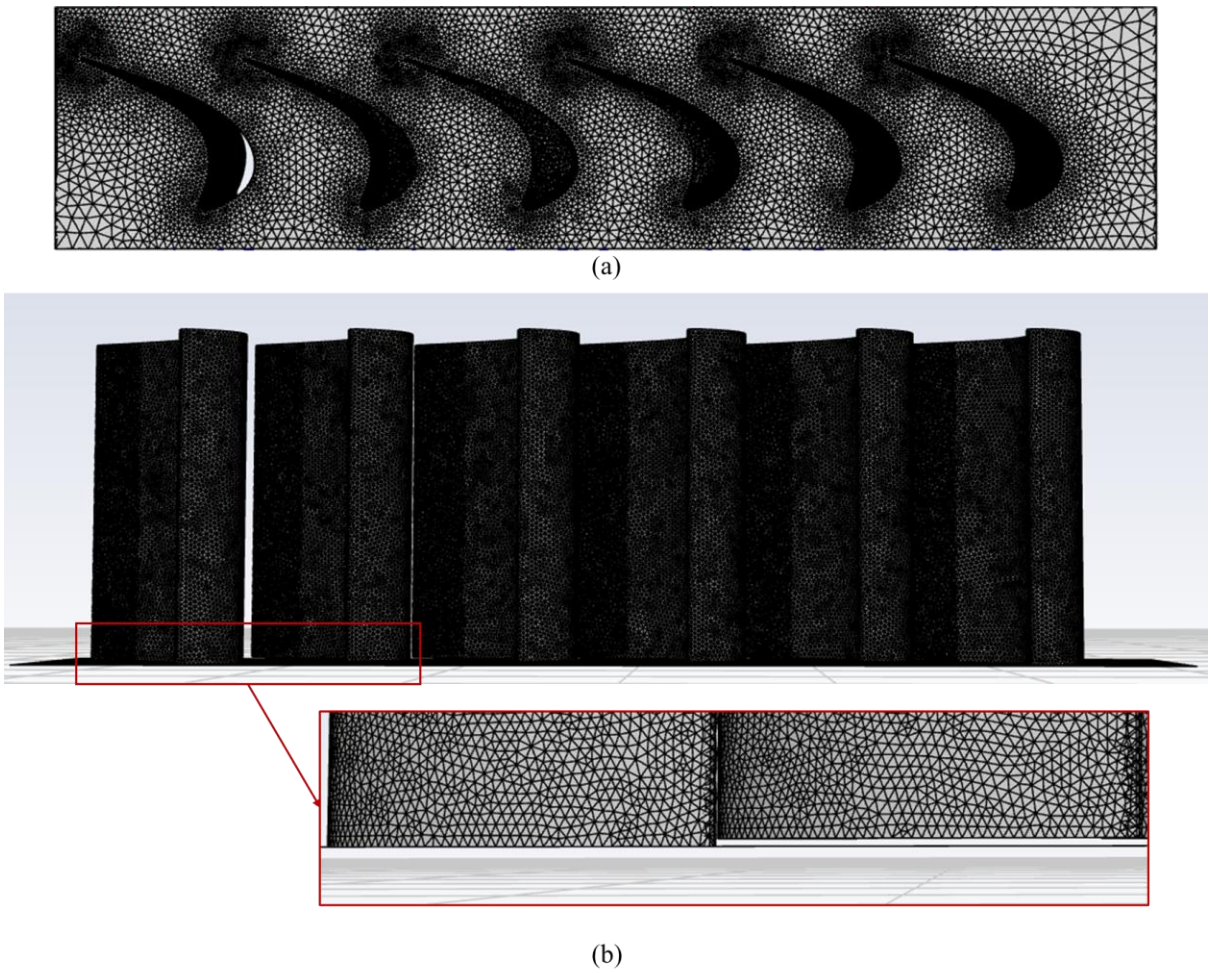



**Figure 6.20 3D Basic Mesh of blade and endwall; (a) top view; (b) front view.**

Boundary conditions were set to match those of the experimental setup. At the inlet, the free stream velocity was specified as 19 m/s, while the outlet was defined with a fixed static pressure. Blade and endwall surfaces were treated as no-slip walls. The operating point corresponds to an inflow Reynolds number of $Re = 4 \times 10^5$, based on chord length and axial velocity. Simulations were considered converged once residuals dropped below $1 \times 10^{-6}$.

The final CFD solution includes the full velocity field which was extracted and interpolated to the same spatial resolution as the U-net output. This ensures pixel-wise alignment for direct quantitative error analysis between prediction and ground truth.

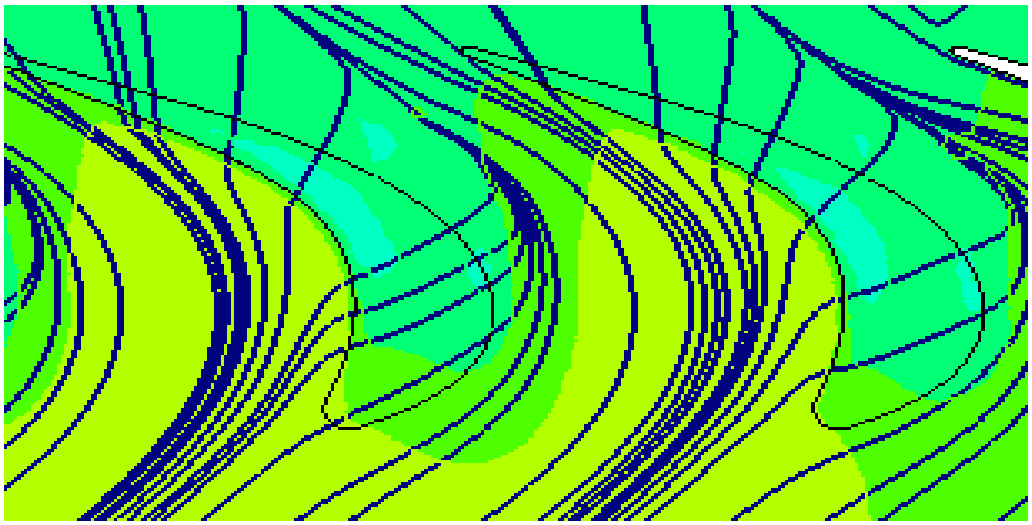


**Figure 6.21 Reference CFD streamline for the test case.**

Figure 6.21 presents the time-averaged streamlines and pressure contours in the endwall plane. At the inlet, the flow direction is relatively parallel, but after encountering the blades, most streamlines are deflected following the blade geometry. Due to the gap between the blade tips and the endwall, some streamlines are able to pass through the blade projection area (indicated by the black closed regions in the figure) and subsequently merge with other streamlines in the wake region.

### 6.5.3 Flow Field Prediction Using U-net

The experimentally obtained flow visualization image corresponding to the Durham cascade test case was used as input to the trained U-net model. No manual annotations or velocity field inputs were used — the model predicted the flow purely from the visual streamline patterns.

The U-net model had been trained on a mixed dataset comprising both synthetic and experimental flow images, enabling it to generalize across a range of flow features, including wakes, vortices, and boundary layers. For this test, the model operated in test mode and produced a two-component velocity field output, with the same grid resolution as the reference CFD data.

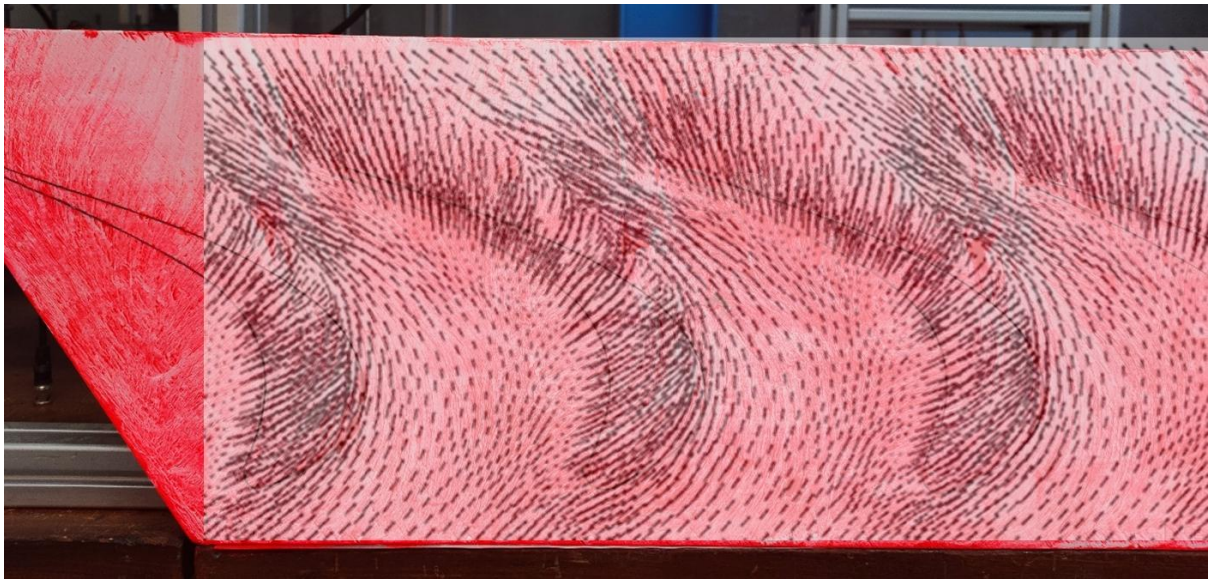


**Figure 6.22 Predicted velocity magnitude field from U-net, inferred from SFV image of the Durham cascade.**

Figure 6.22 shows the U-net prediction results for the specified region in Figure 6.19, presented in vector form. To provide a more intuitive assessment, the predicted vectors are overlaid onto the original FSV image. Black arrows represent the predicted skin friction directions from the U-net, while the background displays the experimentally captured flow field. Overall, the U-net successfully captured the main flow features, including the primary flow direction and wake development. The results demonstrated good agreement with the experimental observations. In the region near the inlet and between the blades, the predicted vectors closely match both direct visual observations, and the velocity direction fields from CFD simulations. This shows that the model has a strong ability to predict boundary layer flows. In the wake region, however, some areas exhibited more disordered vectors, which might be attributed to factors such as the limited resolution of the training data, the complexity of turbulent flow structures, and the detailed treatment within the model.

Quantitative error analysis is presented in the following section, where the U-net prediction is directly compared to the CFD reference solution using both pixel-wise and structure-aware metrics.

### 6.5.4 Error Distribution Analysis

To quantitatively assess the prediction accuracy of the U-net model, a pixel-wise comparison was performed between the predicted flow field and the reference CFD solution.

Based on the definition provided in Eq. (6.10) the mean absolute error $\varepsilon_{MAE}$ was calculated to capture the average deviation across all pixels. In addition to $\varepsilon_{MAE}$, a supplementary pixel-wise relative error function, inspired by the work of Chen et al. (2019a), is introduced to evaluate prediction quality:

$$\varepsilon_{\alpha} = \frac{1}{n(P)} \sum_{p=1}^{n(P)} \mathbb{1}\left(\varepsilon_{\alpha}^{p} > 0\right) \tag{6.14}$$

where,

$$\varepsilon_{\alpha}^{p} = \mathbb{1}\left(\left|\frac{y_p - y_p'}{y_p + \eta}\right| > \alpha\right) \tag{6.15}$$

and

$$y_p = \sum_{j=1}^{3} y_{pj} \tag{6.16}$$

where $y_p$ represents the reference velocity direction at pixel $p$, $\alpha$ is a threshold parameter, and $\eta$ is a small constant to avoid division by zero.

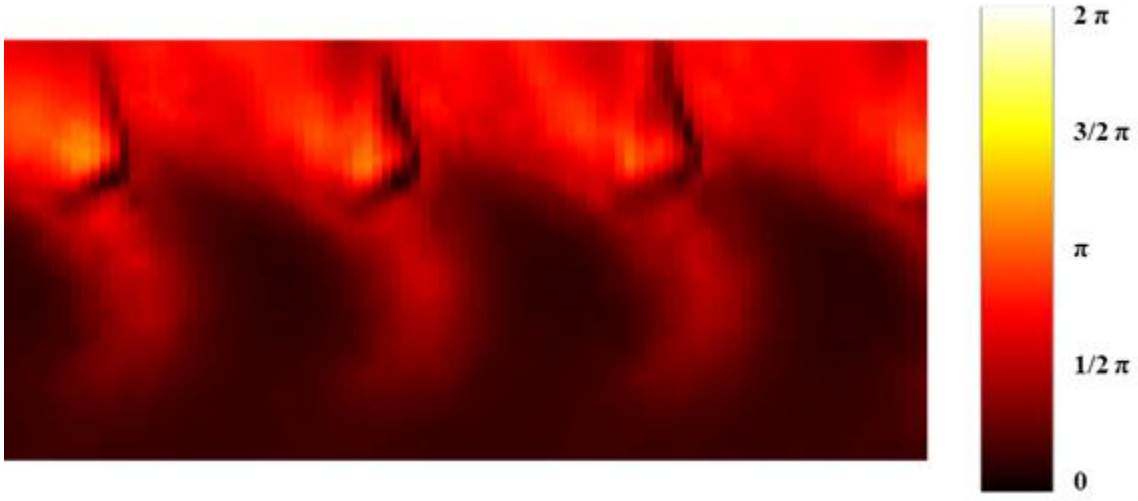



**Figure 6.23 Absolute error distribution of velocity magnitude.**

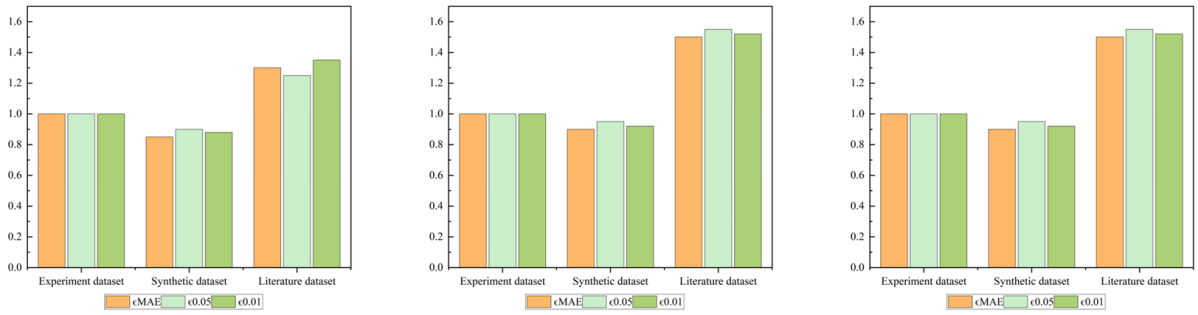
Figure 6.23 presents spatial maps of the absolute error across the prediction domain. The highest error concentrations are observed in the wake region and along blade edges, where sharp gradients and unsteady features are more difficult to reconstruct from static visualization images.

Overall, the model achieves a $\varepsilon_{MAE}$ of 0.369 across the domain, with $\varepsilon_{0.1}$yielding a value of 18.6%, indicating that fewer than one in five pixels exhibit more than 10% deviation relative to the ground truth. These results are consistent with earlier evaluations across the synthetic and literature datasets, confirming the model's capacity to recover dominant flow structures with reasonable fidelity.

Nevertheless, the error distribution reveals limitations in boundary fidelity and small-scale structure recovery, especially in regions influenced by complex three-dimensional effects or turbulent breakdown, which are not fully encoded in the 2D streamline input.

## 6.6 Results

This section presents a quantitative analysis of the U-net's predictive performance on different test subsets, using two metrics: $\varepsilon_{MAE}$ and $\varepsilon_{\alpha}$, whose definition could be found in section 6.5.4. Statistical comparisons were conducted across U-net architecture using the test subsets. Additionally, the prediction performance of the different networks was evaluated on unseen flow visualization images, primarily sourced from the literature. Figure 6.24 shows the different errors.




(a) Average error (b) Standard deviation (c) Maximum error

**Figure 6.24 Comparison of Prediction Errors Across Experiment, Synthetic, and Literature Datasets.**

The results demonstrate that predictions on experiment dataset are serving as the baseline reference for the experimental data, with all error values normalized to 1.00. synthetic dataset shows overall lower errors compared to Experiment dataset, indicating improved prediction accuracy on synthetic data. Specifically, the average errors in the $\varepsilon_{MAE}$, $\varepsilon_{0.05}$, and $\varepsilon_{0.01}$ metrics are reduced by 15%, 10%, and 12%, respectively. This suggests that incorporating synthetic data enhances the pretraining effect of the model, reducing overall error levels.

Additionally, there are significant reductions in the standard deviation and maximum error, by 20% and 10%, respectively, indicating improved stability and fewer extreme errors in predictions for Synthetic dataset.

In contrast, errors for Literature dataset are notably higher than those for Subsets A and B, which aligns with expectations. Since Literature dataset consists of data sourced from the literature, the model has no prior exposure to its distribution. The average errors in the $\varepsilon_{MAE}$, $\varepsilon_{0.05}$, and $\varepsilon_{0.01}$ metrics increase by 30%, 25%, and 35%, respectively. These elevated errors are primarily concentrated around shape boundaries and high-gradient regions, likely reflecting the model's limited generalization capacity. Standard deviation and maximum error for Literature dataset also increase significantly, with each reaching 1.40 and 1.60 times the baseline values, respectively. This indicates that predictions on new data distributions not only exhibit higher overall errors but also greater variability and extreme values.

In summary, synthetic data contributes to improved model performance to some extent, but there remains room for enhancing the model's capabilities on novel data distributions. Future efforts could focus on optimizing data diversity and distribution coverage to strengthen the model's generalization ability, while also improving resolution in prediction regions to mitigate boundary errors.

## 6.7 Conclusion

This chapter explores the application of GANs for generating synthetic flow field data and investigated their effectiveness in enhancing flow field prediction tasks. The presented method involved multiple stages, including data collection, preprocessing, GANs training, and model evaluation. Through these stages, techniques such as chaincode annotation, interpolation, and consistency checks were used to ensure the dataset's quality and physical reliability. The obtained results shows that synthetic data generated by GANs can improve the accuracy of flow prediction models.

The PSNR and SSIM scores showed that the images generated by sGAN were visually consistent with the real images. Compared with other GAN architectures such as CycleGAN, pix2pix, and GauGAN, sGAN was clearer and more consistent structures when generating flow field images. Meanwhile, although sGAN performed well in generating realistic flow visualizations, CycleGAN and pix2pix also performed well in terms of edge clarity. This shows that different GAN architectures are suitable for different aspects of flow field data generation,

and the choice of model should be based on task requirements. During training, the FID score of sGAN dropped from 24.5 to 14.8, indicating that the synthetic images gradually became similar to the real experimental data.

It was found that synthetic data generated by GAN could make it possible to train a flow field prediction models when experimental data is not enough or unavailable. However, further improvements in the synthetic data generation process are needed to improve the generalization of the model to new flow conditions. Increasing the diversity of the dataset and incorporating a wider range of flow geometries, Reynolds numbers, and turbulence models will help improve the robustness and prediction accuracy of the model.

# Chapter 7 Conclusions

## 7.1 Key Findings

Recent advances in flow visualization techniques, such as surface oil flow visualization and global luminescent oil-film method have enabled non-invasive analysis of fluid dynamics. However, there are still some challenges in quantitative analysis, automating streamline detection and addressing data scarcity for data-driven models. Machine learning, especially deep learning showed a possible way to solve these challenges.

This thesis extended the state-of-the-art by addressing these gaps through three key findings:

1. Streamline Detection Using CNNs

Based on previous work, this thesis developed a CNN-based method to enhance the streamlines detection in SFV images. The proposed preprocessing steps, including manual calibration, grayscale conversion, and bilateral filtering, ensures the quality of the input image. The proposed model significantly improves the preservation of detailed features by adding skip connections and a customized upsampling network. Then in the post processing part, label detection and flow direction estimation would help to further optimize the output results. Despite the progress made, the experimental results also show that continued optimization and cross-validation in different flow situations are still very important.

2. Evaluation of GLOF Method

The performance of the GLOF method in surface friction visualization has been evaluated in Chapter 5. Although the method provides a non-invasive means to capture flow features such as separation lines and saddle points, there are still some limitations. One of the most critical issues is its inability to accurately reproduce expected flow behaviours around a cylinder. Although some visual features of the flow were correctly identified, the method showed significant differences in the surface friction patterns compared to experimental data and CFD results. Improvements in computational algorithms and increased imaging resolution are needed in the future to improve its reliability, especially in complex flow situations.

3. Synthetic Data Generation and Unet Training

Chapter 6 explored methods to augment the dataset using synthetic images. In order to generate synthetic flow images similar to the experimental data, the sGAN model was proposed. Data preparation steps such as chain code annotation, interpolation, and consistency checks ensured that the generated images remained physically reasonable and followed the principles of fluid dynamics. The sGAN model successfully generated high-quality synthetic images, achieving good SSIM and PSNR scores. Then these synthetic images were combined with experimental data to train the U-Net model for flow field prediction. With U-net, a preliminary quantitative estimation of the flow field can be obtained by SFV images; with sGAN, the flow field data generated by CFD can be used to generate "fake" SFV images.

## 7.2 Recommendations for Future Work

- Mixing some visible tracer particles into the dye or using pressure-sensitive dye are preferable to using dye alone and should be used where possible in future. This not only allows the surface flow field to be measured with PIV, but also greatly improves the accuracy compared to manual annotation.
- Future flow visualization experiments should accurately handle the thickness of the oil film. In the current work, it was assumed that the thickness of the oil layer applied to the surface was uniform. In fact, since it was applied with a brush, although air brush and roller brush had been tried in addition, it is impossible to ensure that the thickness of the oil film is absolutely uniform. This should be taken into account in future work.
- Professional photography equipment should be used as much as possible. In the initial work, the author used a GoPro camera, which turned out to be not suitable for this job.
- Experimental techniques should be carefully recorded and published. Data and program codes for new technologies or designs should be made public when confidentiality is not required. This allows experiments to be repeated and the experimental process to be reviewed by peer review.
- There are many ways for machine learning to enhance CFD results, but due to limitation on the author's time, these methods are not included in the main body of this thesis. One particularly promising method comes from Raissi et al. (2020), which encodes the Navier-Stokes equations into neural networks. Theoretically, U-net can do the same thing to predict the flow field.

- Extending the current 2D flow visualization techniques to 3D measurements would provide a more complete understanding of the flow structures. Stereoscopic PIV or tomographic techniques could be considered for future work.
- The video results could be further analysed using recurrent neural networks (RNNs), particularly LSTM or GRU architectures, to model temporal evolution and improve flow field predictions. This approach may better capture long-term dependencies in unsteady flows.
- A more comprehensive validation of the experimental results using complementary techniques, such as pressure probe or hot-wire anemometry, could further strengthen the reliability of the findings.

## 7.3 Concluding Remarks

Surface flow visualization has long been a crucial tool in fluid dynamics research, which help researchers capture and analyse flow patterns on surfaces. Recent advancements in machine learning, like CNNs, GANs and so on, have shown promise in automating the extraction of streamlines from SFV images. However, challenges remain in improving the accuracy and applicability of these methods across different flow conditions. Building on these existing techniques, this thesis extends the state-of-the-art by developing a robust framework for streamline detection and flow field prediction. This technique enables preliminary quantitative analysis of flow fields under limited experimental conditions, while also serving to verify CFD results.

# References


ABDELSALAM, T. I., WILLIAMS, R. & INGRAM, G. 2017. Exploiting modern image processing in surface flow visualisation. Global Power and Propulsion Society (GPPS).

AFZALI, J., CASAS, C. Q. & ARCUCCI, R. Latent GAN: using a latent space-based GAN for rapid forecasting of CFD models. International Conference on Computational Science, 2021. Springer, 360-372.

AGRAWAL, S. & AGRAWAL, J. 2015. Survey on anomaly detection using data mining techniques. *Procedia Computer Science,* 60**,** 708-713.

ALI, M. M., MAHALE, V. H., YANNAWAR, P. & GAIKWAD, A. Overview of fingerprint recognition system. 2016 international conference on electrical, electronics, and optimization techniques (ICEEOT), 2016. IEEE, 1334-1338.

ANDERSON, J. D. & WENDT, J. 1995. *Computational fluid dynamics*, Springer.

ANTONIOU, A. 2017. Data Augmentation Generative Adversarial Networks. *arXiv preprint arXiv:1711.04340*.

AUNAPU, N. V., VOLINO, R. J. & FLACK, K. A. 2000. Surface flow visualization of a scaled-up turbine blade passage. *Measurement Science and Technology,* 11**,** 987.

BANSAL, R., SEHGAL, P. & BEDI, P. 2011. Minutiae extraction from fingerprint images-a review. *arXiv preprint arXiv:1201.1422*.

BARLOW, J. B., RAE, W. H. & POPE, A. 1999. *Low-speed wind tunnel testing*, John wiley & sons.

BATCHELOR, G. K. 2000. *An introduction to fluid dynamics*, Cambridge university press.

BERTASIUS, G., SHI, J. & TORRESANI, L. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. Proceedings of the IEEE conference on computer vision and pattern recognition, 2015. 4380-4389.

BEWLEY, T. R. & PROTAS, B. 2004. Skin friction and pressure: the "footprints" of turbulence. *Physica D: Nonlinear Phenomena,* 196**,** 28-44.

BOUZAGLO, R. & KELLER, Y. 2022. Synthesis and reconstruction of fingerprints using generative adversarial networks. *arXiv preprint arXiv:2201.06164*.

BRADSKI, G. 2000. The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer,* 25**,** 120-123.

BROWN, J. L. & NAUGHTON, J. W. 1999. The thin oil film equation.

BRüCKER, C., BAUER, D. & CHAVES, H. 2007. Dynamic response of micro-pillar sensors measuring fluctuating wall-shear-stress. *Experiments in fluids,* 42**,** 737-749.

CANNY, J. 1986. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence***,** 679-698.

CANTWELL, B. 1979. Coherent turbulent structures as critical points in unsteady flow. *Archives of Mechanics,* 31**,** 707-721.

CHEN, D., GAO, X., XU, C., CHEN, S., FANG, J., WANG, Z. & WANG, Z. FlowGAN: A conditional generative adversarial network for flow prediction in various conditions. 2020 IEEE 32nd international conference on tools with artificial intelligence (ICTAI), 2020. IEEE, 315-322.

CHEN, J., VIQUERAT, J. & HACHEM, E. 2019a. U-net architectures for fast prediction of incompressible laminar flows. *arXiv preprint arXiv:1910.13532*.

CHEN, L.-C., PAPANDREOU, G., SCHROFF, F. & ADAM, H. 2017. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.

CHEN, L., SUZUKI, T., NONOMURA, T. & ASAI, K. 2019b. Characterization of luminescent mini-tufts in quantitative flow visualization experiments: surface flow analysis and modelization. *Experimental Thermal and Fluid Science,* 103**,** 406-417.

CHONG, M. S., PERRY, A. E. & CANTWELL, B. J. 1990. A general classification of three-dimensional flow fields. *Physics of Fluids A: Fluid Dynamics,* 2**,** 765-777.

CIREGAN, D., MEIER, U. & SCHMIDHUBER, J. Multi-column deep neural networks for image classification. 2012 IEEE conference on computer vision and pattern recognition, 2012. IEEE, 3642-3649.

CIRESAN, D., GIUSTI, A., GAMBARDELLA, L. & SCHMIDHUBER, J. 2012. Deep neural networks segment neuronal membranes in electron microscopy images. *Advances in neural information processing systems,* 25.

CIRESAN, D. C., MEIER, U., MASCI, J., GAMBARDELLA, L. M. & SCHMIDHUBER, J. Flexible, high performance convolutional neural networks for image classification. Twenty-second international joint conference on artificial intelligence, 2011. Citeseer.

DALLMANN, U. Topological structures of three-dimensional vortex flow separation. 16th Fluid and Plasmadynamics Conference, 1983. 1735.

DHAWAN, S. 1953. Direct measurements of skin friction.

DIXON, S. L. & HALL, C. 2013. *Fluid mechanics and thermodynamics of turbomachinery*, Butterworth-Heinemann.

DUTA, I. C., GEORGESCU, M. I. & IONESCU, R. T. Contextual convolutional neural networks. Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021. 403-412.

ECKERLE, W. A. 1985. *HORSESHOE VORTEX FORMATION AROUND A CYLINDER.* Ph.D., University of Connecticut.

ECKERLE, W. A. & LANGSTON, L. 1987. Horseshoe vortex formation around a cylinder.

ETMINAN, A., MUZYCHKA, Y. S., POPE, K. & NYANTEKYI-KWAKYE, B. 2022. Flow visualization: state-of-the-art development of micro-particle image velocimetry. *Measurement Science and Technology,* 33**,** 092002.

FONOV, S., JONES, G., CRAFTON, J., FONOV, V. & GOSS, L. 2006. The development of optical techniques for the measurement of pressure and skin friction. *Measurement Science and Technology,* 17**,** 1261.

FREYMUTH, P. 1993. Flow visualization in fluid mechanics. *Review of scientific instruments,* 64**,** 1-18.

FRID-ADAR, M., DIAMANT, I., KLANG, E., AMITAI, M., GOLDBERGER, J. & GREENSPAN, H. 2018. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing,* 321**,** 321-331.

FUKAGATA, K., IWAMOTO, K. & KASAGI, N. 2002. Contribution of Reynolds stress distribution to the skin friction in wall-bounded flows. *Physics of fluids,* 14**,** L73-L76.

GATYS, L. A., ECKER, A. S. & BETHGE, M. Image style transfer using convolutional neural networks. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016. 2414-2423.

GENDRICH, C., KOOCHESFAHANI, M. & NOCERA, D. 1997. Molecular tagging velocimetry and other novel applications of a new phosphorescent supramolecule. *Experiments in fluids,* 23**,** 361-372.

GOLDSTEIN, R. 2017. Fluid mechanics measurements.

GOMEZ, T., FLUTET, V. & SAGAUT, P. 2009. Contribution of Reynolds stress distribution to the skin friction in compressible turbulent channel flows. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics,* 79**,** 035301.

GOODFELLOW, I. 2016. Deep learning. MIT press.

GOODFELLOW, I., POUGET-ABADIE, J., MIRZA, M., XU, B., WARDE-FARLEY, D., OZAIR, S., COURVILLE, A. & BENGIO, Y. 2014. Generative adversarial nets. *Advances in neural information processing systems,* 27.

GRAVES, C. P. 1985. *Secondary flows and losses in gas turbines.* Durham University.

GUI, J., SUN, Z., WEN, Y., TAO, D. & YE, J. 2023. A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications. *IEEE Transactions on Knowledge and Data Engineering,* 35**,** 3313-3332.

HAN, J., TAO, J., ZHENG, H., GUO, H., CHEN, D. Z. & WANG, C. 2019. Flow Field Reduction Via Reconstructing Vector Data From 3-D Streamlines Using Deep Learning. *IEEE Computer Graphics and Applications,* 39**,** 54-67.

HARIHARAN, P. 2010. *Basics of interferometry,* Elsevier.

HARRIS, C. R., MILLMAN, K. J., VAN DER WALT, S. J., GOMMERS, R., VIRTANEN, P., COURNAPEAU, D., WIESER, E., TAYLOR, J., BERG, S. & SMITH, N. J. 2020. Array programming with NumPy. *Nature,* 585**,** 357-362.

HARTLEY, R. & ZISSERMAN, A. 2003. *Multiple view geometry in computer vision,* Cambridge university press.

HE, J., ZHANG, S., YANG, M., SHAN, Y. & HUANG, T. Bi-directional cascade network for perceptual edge detection. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019. 3828-3837.

HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B. & SALESIN, D. H. 2001. Image analogies. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques.* Association for Computing Machinery.

HORN, B. K. P. & SCHUNCK, B. G. 1981. Determining optical flow. *Artificial Intelligence,* 17**,** 185-203.

HUNT, J. C., WRAY, A. A. & MOIN, P. 1988. Eddies, streams, and convergence zones in turbulent flows. *Studying turbulence using numerical simulation databases, 2. Proceedings of the 1988 summer program.*

HUNTER, J. D. 2007. Matplotlib: A 2D graphics environment. *Computing in science & engineering,* 9**,** 90-95.

HUSEN, N. M. 2017. *Skin friction measurements using luminescent oil films.* Purdue University.

HUSEN, N. M., LIU, T. & SULLIVAN, J. P. 2018a. Luminescent oil film flow tagging skin friction meter applied to FAITH hill. *AIAA Journal,* 56**,** 3875-3886.

HUSEN, N. M., LIU, T. & SULLIVAN, J. P. 2018b. The ratioed image film thickness meter. *Measurement Science and Technology,* 29**,** 065301.

HUSSAIN, F. & JEONG, J. 1995. On the identification of a vortex. *J. Fluid Mech,* 285**,** 69-94.

INGRAM, G. 2009. *Basic concepts in turbomachinery,* Bookboon.

INGRAM, G., GREGORY-SMITH, D. & HARVEY, N. 2005. Investigation of a novel secondary flow feature in a turbine cascade with end wall profiling. *J. Turbomach.,* 127**,** 209-214.

ISIKDOGAN, F., BOVIK, A. & PASSALACQUA, P. 2017. RivaMap: An automated river analysis and mapping engine. *Remote Sensing of Environment,* 202**,** 88-97.

ISOLA, P., ZHU, J.-Y., ZHOU, T. & EFROS, A. A. Image-to-image translation with conditional adversarial networks. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017. 1125-1134.

JEONG, J. & HUSSAIN, F. 1995. On the identification of a vortex. *Journal of fluid mechanics,* 285**,** 69-94.

KAMRAVA, S., TAHMASEBI, P. & SAHIMI, M. 2021. Physics-and image-based prediction of fluid flow and transport in complex porous membranes and materials by deep learning. *Journal of Membrane Science,* 622**,** 119050.

KASTNER, P. & DOGAN, T. 2023. A GAN-based surrogate model for instantaneous urban wind flow prediction. *Building and Environment,* 242**,** 110384.

KHAN, Z. Y., NIU, Z., SANDIWARNO, S. & PRINCE, R. 2021. Deep learning techniques for rating prediction: a survey of the state-of-the-art. *Artificial Intelligence Review,* 54**,** 95-135.

KIM, B. & GüNTHER, T. Robust reference frame extraction from unsteady 2D vector fields with convolutional neural networks. Computer Graphics Forum, 2019. Wiley Online Library, 285-295.

KOLáŘ, V. 2007. Vortex identification: New requirements and limitations. *International journal of heat and fluid flow,* 28**,** 638-652.

KOZOMORA, V., ŽIVKOV, A., VUKIĆ, M. & OLUŠKI, N. 2024. Application of flow visualization methods for determining drag force around a projectile. *Journal on Processing and Energy in Agriculture,* 28**,** 31-37.

KRIZHEVSKY, A., SUTSKEVER, I. & HINTON, G. E. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems,* 25.

LAKSHMINARAYANA, B. 1995. Cascade Inviscid Flows. *Fluid Dynamics and Heat Transfer of Turbomachinery.*

LAUNDER, B. E. & SHARMA, B. I. 1974. Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc. *Letters in heat and mass transfer,* 1**,** 131-137.

LAUNDER, B. E. & SPALDING, D. B. 1983. The numerical computation of turbulent flows. *Numerical prediction of flow, heat transfer, turbulence and combustion.* Elsevier.

LEE, T., NONOMURA, T., ASAI, K. & LIU, T. 2018. Linear least-squares method for global luminescent oil film skin friction field analysis. *Review of Scientific Instruments,* 89.

LEER, M. & KEMPF, A. 2021. Fast flow field estimation for various applications with a universally applicable machine learning concept. *Flow, Turbulence and Combustion,* 107**,** 175-200.

LI, M., LIN, Z., MECH, R., YUMER, E. & RAMANAN, D. Photo-sketching: Inferring contour drawings from images. 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), 2019a. IEEE, 1403-1412.

LI, W., FAN, Y., MODESTI, D. & CHENG, C. 2019b. Decomposition of the mean skin-friction drag in compressible turbulent channel flows. *Journal of Fluid Mechanics,* 875**,** 101-123.

LIEN, F.-S. Low-Reynolds-number eddy-viscosity modelling based on non-linear stress-strain/vorticity relations. Proc. 3rd Symposium On Engineering Turbulence Modelling and Measurements, 1996. 1-10.

LIGHTHILL, M. 1963. Attachment and separation in three-dimensional flow. *Laminar boundary layers***,** 72-82.

LIU, T. 2013. Extraction of skin-friction fields from surface flow visualizations as an inverse problem. *Measurement Science and Technology,* 24**,** 124004.

LIU, T. 2019. Global skin friction measurements and interpretation. *Progress in Aerospace Sciences,* 111**,** 100584.

LIU, T., MONTEFORT, J., WOODIGA, S., MERATI, P. & SHEN, L. 2008. Global luminescent oil-film skin-friction meter. *AIAA journal,* 46**,** 476-485.

LIU, T. & SHEN, L. 2008. Fluid flow and optical flow. *Journal of Fluid Mechanics,* 614**,** 253-291.

LIU, T., WOODIGA, S., GREGORY, J. & SULLIVAN, J. 2014. Global skin-friction diagnostics based on surface mass-transfer visualizations. *AIAA Journal,* 52**,** 2369-2383.

LIU, T., WOODIGA, S. & MA, T. 2011. Skin friction topology in a region enclosed by penetrable boundary. *Experiments in fluids,* 51**,** 1549-1562.

LIU, X., INGRAM, G., SIMS-WILLIAMS, D. & BRECKON, T. P. 2024. Extracting Quantitative Streamline Information from Surface Flow Visualization Images in a Linear Cascade using Convolutional Neural Networks. *Proceedings of Global Power and Propulsion Society*. Chania.

LIU, Y., CHENG, M.-M., HU, X., WANG, K. & BAI, X. Richer convolutional features for edge detection. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017. 3000-3009.

LU, F. 2010. Surface oil flow visualization. *The European Physical Journal Special Topics,* 182**,** 51-63.

MALTBY, R. 1962. Flow visualization in wind tunnels using indicators. Advisory group for aeronautical research and development paris (France).

MARCHAL, P. & SIEVERDING, C. 1977. Secondary flows within turbomachinery bladings. *AGARD Secondary Flows in Turbomachines 20 p(SEE N 78-11083 02-07)*.

MARTINEZ-CASTRO, P. A. 2022. *Entropy Generation Rate for Profiled Endwall Design in Turbines.* Doctor, Durham University.

MCINTOSH, J., MACPHERSON, R., INGRAM, G. & HOGG, S. Profiled endwall design using genetic algorithms with different objective functions. Turbo Expo: Power for Land, Sea, and Air, 2011. 2393-2405.

MEHDI, F., JOHANSSON, T. G., WHITE, C. M. & NAUGHTON, J. W. 2014. On determining wall shear stress in spatially developing two-dimensional wall-bounded flows. *Experiments in fluids,* 55**,** 1-9.

MEHDI, F. & WHITE, C. M. 2011. Integral form of the skin friction coefficient suitable for experimental data. *Experiments in Fluids,* 50**,** 43-51.

MéLY, D. A., KIM, J., MCGILL, M., GUO, Y. & SERRE, T. 2016. A systematic comparison between visual cues for boundary detection. *Vision research,* 120**,** 93-107.

MENTER, F. R. 1994. Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA journal,* 32**,** 1598-1605.

MENTER, F. R., KUNTZ, M. & LANGTRY, R. 2003. Ten years of industrial experience with the SST turbulence model. *Turbulence, heat and mass transfer,* 4**,** 625-632.

MERZKIRCH, W. 1987. Techniques of flow visualization. Advisory group for aerospace research and development neuilly-sur-seine (France).

MERZKIRCH, W. 2012. *Flow visualization*, Elsevier.

MIRZA, M. & OSINDERO, S. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

NAIR, V. & HINTON, G. E. Rectified linear units improve restricted boltzmann machines. Proceedings of the 27th international conference on machine learning (ICML-10), 2010. 807-814.

NAKAYAMA, Y. 2018. *Introduction to fluid mechanics*, Butterworth-Heinemann.

NAUGHTON, J. W. & SHEPLAK, M. 2002. Modern developments in shear-stress measurement. *Progress in Aerospace Sciences,* 38**,** 515-570.

ONGSULEE, P. Artificial intelligence, machine learning and deep learning. 2017 15th International Conference on ICT and Knowledge Engineering (ICT&KE), 2017. IEEE, 1-6.

PANCHIGAR, D., KAR, K., SHUKLA, S., MATHEW, R. M., CHADHA, U. & SELVARAJ, S. K. 2022. Machine learning-based CFD simulations: a review, models, open threats, and future tactics. *Neural Computing and Applications,* 34**,** 21677-21700.

PARK, T., LIU, M.-Y., WANG, T.-C. & ZHU, J.-Y. 2019. Gaugan: semantic image synthesis with spatially adaptive normalization. *ACM SIGGRAPH 2019 Real-Time Live!*

PASZKE, A., GROSS, S., MASSA, F., LERER, A., BRADBURY, J., CHANAN, G., KILLEEN, T., LIN, Z., GIMELSHEIN, N. & ANTIGA, L. 2019. Pytorch: An

imperative style, high-performance deep learning library. *Advances in neural information processing systems,* 32.

PERRY, A. & CHONG, M. 1994. Topology of flow patterns in vortex motions and turbulence. *Applied Scientific Research,* 53**,** 357-374.

PERRY, A. & CHONG, M. 2012. Interpretation of flow visualization. *Flow Visualization: Techniques and Examples.* World Scientific.

POMA, X. S., RIBA, E. & SAPPA, A. Dense extreme inception network: Towards a robust cnn model for edge detection. Proceedings of the IEEE/CVF winter conference on applications of computer vision, 2020. 1923-1932.

POST, F. H. & VAN WALSUM, T. 1993. Fluid Flow Visualization. *In:* HAGEN, H., MüLLER, H. & NIELSON, G. M. (eds.) *Focus on Scientific Visualization.* Berlin, Heidelberg: Springer Berlin Heidelberg.

POURREZA, R., ZHUGE, Y., NING, H. & MILLER, R. Brain tumor segmentation in MRI scans using deeply-supervised neural networks. Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: Third International Workshop, BrainLes 2017, Held in Conjunction with MICCAI 2017, Quebec City, QC, Canada, September 14, 2017, Revised Selected Papers 3, 2018. Springer, 320-331.

RAISSI, M., YAZDANI, A. & KARNIADAKIS, G. E. 2020. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science,* 367**,** 1026-1030.

RAO, A. R. & SCHUNCK, B. G. 1991. Computing oriented texture fields. *CVGIP: Graphical Models and Image Processing,* 53**,** 157-185.

REDA, D. C., WILDER, M. C., FARINA, D. J. & ZILLIAC, G. 1997. New methodology for the measurement of surface shear stress vector distributions. *Aiaa Journal,* 35**,** 608-614.

RENARD, N. & DECK, S. 2016. A theoretical decomposition of mean skin friction generation into physical phenomena across the boundary layer. *Journal of Fluid Mechanics,* 790**,** 339-367.

ROBINSON, S. 1991. Coherent motions in the turbulent boundary layer. *Annual review of fluid mechanics,* 23**,** 601-639.

RONNEBERGER, O., FISCHER, P. & BROX, T. U-net: Convolutional networks for biomedical image segmentation. International Conference on Medical image computing and computer-assisted intervention, 2015. Springer, 234-241.

SAMUEL, A. L. 1959. Machine learning. *The Technology Review,* 62**,** 42-45.

SCHüLEIN, E. Development and Application of the Thin Oil Film Technique for Skin Friction Measurements in the Short-Duration Hypersonic Wind Tunnel. *In:* BREITSAMTER, C., LASCHKA, B., HEINEMANN, H.-J. & HILBIG, R., eds. New Results in Numerical and Experimental Fluid Mechanics IV, 2004// 2004 Berlin, Heidelberg. Springer Berlin Heidelberg, 407-414.

SETHI, D., BHARTI, S. & PRAKASH, C. 2022. A comprehensive survey on gait analysis: History, parameters, approaches, pose estimation, and future work. *Artificial Intelligence in Medicine,* 129**,** 102314.

SETTLES, G. S. 2001. *Schlieren and shadowgraph techniques: visualizing phenomena in transparent media,* Springer Science & Business Media.

SHIH, T.-H. 1993. *A realizable Reynolds stress algebraic equation model,* National Aeronautics and Space Administration.

SHIH, T.-H., LIOU, W. W., SHABBIR, A., YANG, Z. & ZHU, J. 1995. A new k-ϵ eddy viscosity model for high reynolds number turbulent flows. *Computers & fluids,* 24**,** 227-238.

SHOTTON, J., BLAKE, A. & CIPOLLA, R. 2008. Multiscale categorical object recognition using contour fragments. *IEEE transactions on pattern analysis and machine intelligence,* 30**,** 1270-1281.

SHOURANGIZ-HAGHIGHI, A., HAGHNEGAHDAR, M. A., WANG, L., MUSSETTA, M., KOLIOS, A. & LANDER, M. 2020. State of the art in the optimisation of wind turbine performance using CFD. *Archives of Computational Methods in Engineering,* 27**,** 413-431.

SIMS-WILLIAMS, D. 2001. *Self-excited aerodynamic unsteadiness associated with passenger cars.* Durham University.

SOBEL, I. E. 1970. *Camera models and machine perception*, stanford university.

SONG, H.-S., MUGABI, J. & JEONG, J.-H. 2023. Pix2Pix and deep neural network-based deep learning technology for predicting vortical flow fields and aerodynamic performance of airfoils. *Applied Sciences,* 13**,** 1019.

SORIA, J. & CANTWELL, B. J. 1994. Topological visualisation of focal structures in free shear flows. *Applied scientific research,* 53**,** 375-386.

SORIA, X., POMBOZA-JUNEZ, G. & SAPPA, A. D. 2022. Ldc: Lightweight dense cnn for edge detection. *IEEE Access,* 10**,** 68281-68290.

SVOBODA, J., MONTI, F. & BRONSTEIN, M. M. Generative convolutional networks for latent fingerprint reconstruction. 2017 IEEE International joint conference on biometrics (IJCB), 2017. IEEE, 429-436.

THOMPSON, J. F., WARSI, Z. U. & MASTIN, C. W. 1985. *Numerical grid generation: foundations and applications,* Elsevier North-Holland, Inc.

TOMASI, C. & MANDUCHI, R. Bilateral filtering for gray and color images. Sixth international conference on computer vision (IEEE Cat. No. 98CH36271), 1998. IEEE, 839-846.

TRAN, T. H., AMBO, T., LEE, T., CHEN, L., NONOMURA, T. & ASAI, K. 2018. Effect of boattail angles on the flow pattern on an axisymmetric afterbody surface at low speed. *Experimental Thermal and Fluid Science,* 99**,** 324-335.

TRAN, T. H., AMBO, T., LEE, T., OZAWA, Y., CHEN, L., NONOMURA, T. & ASAI, K. 2019. Effect of Reynolds number on flow behavior and pressure drag of axisymmetric conical boattails at low speeds. *Experiments in Fluids,* 60**,** 1-19.

TRAN, T. H. & CHEN, L. 2021. Wall shear-stress extraction by an optical flow algorithm with a sub-grid formulation. *Acta Mechanica Sinica,* 37**,** 65-79.

TREASTER, A. & STINEBRING, D. 1980. The Use of Fluorescent Mini-Tufts for Hydrodynamic Flow Visualization. PENNSYLVANIA STATE UNIV UNIVERSITY PARK APPLIED RESEARCH LAB.

TRITTON, D. J. 2012. *Physical fluid dynamics*, Springer Science & Business Media.

TRUESDELL, C. 1954. *The kinematics of vorticity*, Indiana University Press.

TUKEY, J. W. 1977. Exploratory data analysis. *Reading/Addison-Wesley*.

VAN DYKE, M. & VAN DYKE, M. 1982. *An album of fluid motion*, Parabolic Press Stanford.

VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, Ł. & POLOSUKHIN, I. Attention is all you need. Advances in neural information processing systems, 2017. 5998-6008.

VIRTANEN, P., GOMMERS, R., OLIPHANT, T. E., HABERLAND, M., REDDY, T., COURNAPEAU, D., BUROVSKI, E., PETERSON, P., WECKESSER, W. & BRIGHT, J. 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods,* 17**,** 261-272.

VOLLMERS, H., KREPLIN, H. & MEIER, H. 1983. Separation and vortical-type flow around a prolate spheroid: evaluation of relevant parameters. *AGARD Aerodyn. of Vortical Type Flows in Three Dimensions 14 p(SEE N 84-12099 03-02)*.

VON KARMAN, T. 1934. Turbulence and skin friction. *Journal of the Aeronautical Sciences,* 1**,** 1-20.

WALKER, J., GUPTA, A. & HEBERT, M. Dense optical flow prediction from a static image. Proceedings of the IEEE International Conference on Computer Vision, 2015. 2443-2451.

WALTRUP, P. & SCHETZ, J. 1973. Supersonic turbulent boundary layer subjected to adverse pressure gradients. *AIAA Journal,* 11**,** 50-57.

WATT, D. & VEST, C. 1987. Digital interferometry for flow visualization. *Experiments in fluids,* 5**,** 401-406.

WILCOX, D. 1998. Turbulence Modeling for CFD. *DCW industries, La Canada*.

WINTER, K. 1979. An outline of the techniques available for the measurement of skin friction in turbulent boundary layers. *Progress in aerospace sciences,* 18**,** 1-57.

WOODIGA, S. & LIU, T. 2009. Skin friction fields on delta wings. *Experiments in Fluids,* 47**,** 897-911.

WOODIGA, S., LIU, T., RAMASAMY, R. V. & KODE, S. K. 2016. Effects of pitch, yaw, and roll on delta wing skin friction topology. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering,* 230**,** 639-652.

WOODIGA, S., SALAZAR, D. M., WEWENGKANG, P., MONTEFORT, J. & LIU, T. 2018. Skin-friction topology on tail plate for tractor—trailer truck drag reduction. *Journal of Visualization,* 21**,** 1017-1029.

WU, P., PAN, K., JI, L., GONG, S., FENG, W., YUAN, W. & PAIN, C. 2022. Navier–stokes Generative Adversarial Network: a physics-informed deep learning model for fluid flow generation. *Neural Computing and Applications,* 34**,** 11539-11552.

XIA, Z., ZHANG, P. & YANG, X. I. 2021. On skin friction in wall-bounded turbulence. *Acta Mechanica Sinica,* 37**,** 589-598.

XIE, S. & TU, Z. Holistically-nested edge detection. Proceedings of the IEEE international conference on computer vision, 2015. 1395-1403.

XU, D., WANG, J. & CHEN, S. 2022. Skin-friction and heat-transfer decompositions in hypersonic transitional and turbulent boundary layers. *Journal of Fluid Mechanics,* 941**,** A4.

YAKHOT, V., ORSZAG, S. A., THANGAM, S., GATSKI, T. & SPEZIALE, C. 1992. Development of turbulence models for shear flows by a double expansion technique. *Physics of Fluids A: Fluid Dynamics,* 4**,** 1510-1520.

YAMAGUCHI, K., SAKAMOTO, K., AKABANE, T. & FUJIMOTO, Y. A neural network for speaker-independent isolated word recognition. ICSLP, 1990. 1077-1080.

YANG, M.-H., KRIEGMAN, D. & AHUJA, N. 2002. Detecting faces in images: A survey. IEEE Transactions on Pattern Analysis & Machine Intelligence.

YE, S., ZHANG, Z., SONG, X., WANG, Y., CHEN, Y. & HUANG, C. 2020. A flow feature detection method for modeling pressure distribution around a cylinder in non-uniform flows by using a convolutional neural network. *Scientific reports,* 10**,** 1-10.

YU, F. & KOLTUN, V. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.

ZANOUN, E.-S., EGBERS, C., NAGIB, H., DURST, F., BELLANI, G. & TALAMELLI, A. 2021. Wall friction relations in wall-bounded shear flows. *European Journal of Mechanics-B/Fluids,* 89**,** 171-179.

ZHANG, D. Comparison of various turbulence models for unsteady flow around a finite circular cylinder at Re= 20000. Journal of Physics: Conference Series, 2017. IOP Publishing, 012027.

ZHANG, K., ZHANG, L., LAM, K.-M. & ZHANG, D. 2015. A level set approach to image segmentation with intensity inhomogeneity. *IEEE transactions on cybernetics,* 46**,** 546-557.

ZHANG, Z. 2000. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence,* 22**,** 1330-1334.

ZHONG, H., WOODIGA, S., WANG, P., SHANG, J., CUI, X., WANG, J. & LIU, T. 2015. Skin-friction topology of wing–body junction flows. *European Journal of Mechanics-B/Fluids,* 53**,** 55-67.

ZHU, J.-Y., PARK, T., ISOLA, P. & EFROS, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. Proceedings of the IEEE international conference on computer vision, 2017. 2223-2232.

ZIERKE, W., FARRELL, K. & STRAKA, W. Measurements of the tip clearance flow for a high Reynolds number axial-flow rotor: Part 1—Flow visualization. Turbo Expo: Power for Land, Sea, and Air, 1994. American Society of Mechanical Engineers, V001T01A140.

ZIOU, D. & TABBONE, S. 1998. Edge detection techniques-an overview. *Распознавание образов и анализ изображен/Pattern Recognition and Image Analysis: Advances in Mathematical Theory and Applications,* 8**,** 537-559.